VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

HARDWAROVĚ AKCELEROVANÉ ZAŘÍZENÍ PRO OCHRANU PŘED DOS ÚTOKY

HARDWARE-ACCELERATED DEVICE FOR PROTECTION AGAINST DOS ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MÁRIO KUKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN KUČERA**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19924/2016/xkukam00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Kuka Mário**
Obor: Informační technologie
Téma: **Hardwarově akcelerované zařízení pro ochranu před DoS útoky**
Hardware-Accelerated Device for Protection Against DoS Attacks
Kategorie: Návrh číslicových systémů

Pokyny:

1. Seznamte se s problematikou volumetrických útoků typu odepření služby a s hardwarovými akcelerátory rodiny COMBO.
2. Navrhněte firmware pro FPGA na kartě COMBO-100G pro ochranu sítě před popsanými útoky.
3. Proveďte implementaci firmware s využitím vývojové platformy NetCOPE.
4. Otestujte vytvořené zařízení v laboratorních podmínkách.
5. Proveďte vyhodnocení a navrhněte pokračování práce.

Literatura:

- Dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kučera Jan, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá vývojom firmvéru pre hardvérovo akcelerované zariadenie pre ochranu pred amplifikačnými (D)DoS útokmi. V dnešnej dobe sú útoky typu (D)DoS veľmi rozšírené a zapríčiňujú nemalé finančné škody. Cieľom práce je preto vytvoriť cenovo dostupné a ľahko nasaditeľné centralizované zariadenie zamerané na problematiku amplifikačných (D)DoS útokov. K dosiahnutiu tohto cieľu využíva zariadenie hardvérový akcelerátor, ktorý umožňuje spracovávať vysoké dátové prenosy prostredníctvom jedného, bežne dostupného serveru. Návrh a implementácia firmvéru je uskutočnená s ohľadom na použitie zariadenia v sieťach s rýchlosťami 100 Gbps. Celý systém prešiel funkčnou verifikáciou a v rámci laboratórneho testovania bola overená jeho reálna priepustnosť. Vytvorené zariadenie bolo už v dobe písania bakalárskej práce experimentálne nasadené v sieťovej infraštruktúre CESNET a testované sieťovými administrátormi. Na základe spätnej väzby bude na vytvorenom zariadený pokračovať ďalší vývoj v spolupráci združenia CESNET, ktorý bude zameraný na rozširovanie detekcie o ďalšie typy útokov.

Abstract

This thesis deals with the development of a firmware for hardware-accelerated device used as a protection against amplification (D)DoS attacks. In the today's world, (D)DoS attacks are very common and cause significant financial damages. Therefore the goal is to create affordable and easy to deploy centralized device that would resolve this issue. To reach this goal, a hardware accelerator is being used for the high-volume data transfer processing through a single commonly used server. Design and implementation of the firmware had been done considering the fact that this device will be used in the networks with 100 Gbps speed. The whole system had undergone functional verification and its real throughput was verified within the laboratory testing as well. Created device has been already deployed into the CESNET network infrastructure during the time of the writing of this thesis and it has been tested by the network administrators. Based on the received feedback, the development will continue focusing on expanding of the detection of more types of attacks.

Kľúčové slová

DCPro, (D)DoS Protector, COMBO, CESNET, Hardvérová Akcelerácia, FPGA, vysokorýchlostné siete, 100 Gbps, spracovanie sietových dát

Keywords

DCPro, (D)DoS Protector, COMBO, CESNET, Hardware Acceleration, FPGA, High-speed Networks, 100 Gbps, Network Traffic Processing

Citácia

KUKA, Mário. *Hardwarově akcelerované zařízení pro ochranu před DoS útoky.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kučera Jan.

Hardwarově akcelerované zařízení pro ochranu před DoS útoky

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Jana Kučery. Ďalšie informácie mi taktiež poskytli ľudia z projektu Liberouter. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Mário Kuka
17. mája 2017

Poďakovanie

V prvom rade by som chcel venovať veľké poďakovanie vedúcemu práce Ing. Janu Kučerovi za odbornú pomoc, ochotu a trpezlivosť nielen pri tvorbe tejto práce ale taktiež za všetkú ostanú pomoc pri tvorbe ďalších materiálov prezentujúcich výsledky tejto práce. Ďalej by som chcel poďakovať pracovníkom z oddelenia nástrojov pre monitorovanie a konfiguráciu so združením CESNET za všetku poskytnutú odbornú pomoc a podporu. Následne by som chcel venovať poďakovanie mojej rodine za ich podporu počas celej doby môjho bakalárskeho štúdia.

Obsah

Kapitola 1

Úvod

So súčasným rozvojom moderných technológií a ich snahou čo najefektívnejšie prepojiť celú spoločnosť neustále narastajú požiadavky na zefektívňovanie a skvalitňovanie komunikačných prostriedkov. Jedným z hlavných prostriedkov, ktoré nám dneska umožňujú komunikovať, je rozvoj osobných počítačov a s tým spojený rozvoj počítačových sietí. Prostredníctvom týchto sietí si tak môžu užívatelia vymieňať správy, realizovať hovory, zdieľať súbory, prípadne využívať sieťové prostriedky ako sú sieťové tlačiarne, súborové servery a ďalšie množstvo služieb, ktoré sú poskytované prostredníctvom počítačových sietí. Neustále zvyšujúce sa nároky na množstvo prenášaných informácií má za následok neustáli rozvoj počítačových sietí. V súčasnosti je tak trendom neustále zvyšovanie rýchlostí počítačových sietí a využívanie nových dostupných technológií ako je 100 GbE na hlavných sieťových linkách a serverových farmách. Tento neustály rozvoj sa však premianta aj do oblastí bezpečnosti a monitorovania počítačových sietí, ktorých úlohou je práve udržovať tieto siete vo funkčnom stave a chrániť ich pred neustále hroziacim nebezpečím. Jedným typom z týchto hrozieb sú práve útoky určené na odoprenie služby DoS (Distributed Denial of Service), ktorých úlohou je práve zamedzenie prístupu legitímnych užívateľov k poskytovaným službám prostredníctvom počítačových sietí. Cieľom niektorých (D)DoS útokov (napr. SYN flood) býva samotný server s cieľom vyčerpať jeho zdroje (pamäť, CPU), čím dôjde k jeho nedostupnosti pre legitímnych užívateľov. Naproti tomu cieľom volumetrických útokov typu (D)DoS býva samotná sieťová infraštruktúra (počítačová sieť organizácie) a nie iba jedno koncové zariadenie. Ukazovateľom takýchto útokov sú typicky obrovský počet paketov nezvyčajne veľkej dĺžky, na zraniteľných, verejných službách ako je DNS alebo NTP. Pred týmto útokom nie je možné sa brániť na koncových sieťach, pretože linky a vyhradené prenosové pásmo, prostredníctvom ktorého sú koncové siete pripojené, sú zahltené. Preto je potrebné tento problém riešiť už na úrovni počítačovej siete nadradenej organizácie. Tieto siete typicky pracujú na vysokých prenosových rýchlostiach 100Gbps. Bežne používané prostriedky v súčasnosti neposkytujú dostatočný výkon pre zložité operácie nad sieťovým prenosom pri takýchto vysokých rýchlostiach. Riešením tohto problému je práve využívanie hardvérovej akcelerácie na realizáciu týchto operácií.

Hardvérová akcelerácia poskytuje vysokú rýchlosť spracovania a preto je typicky používaná pre rýchle výpočty konkrétnej úlohy tam, kde nedostačujú obecné procesory. Pri spracovávaní a monitorovaní na vysokorýchlostných sieťach sa preto využíva kombinácia oboch prístupov, kde menej náročné úlohy sú spracovávané softvérovo na obecných procesoroch a pre náročné operácie sa využíva hardvérová akcelerácia. Uvedený princíp využíva navrhované hardvérovo akcelerované zariadenie pre ochranu pred (D)DoS útokmi. Základné časti tohto zariadenia sú softvér, ktorý bude vyhodnocovať získané informácie a riadiť čin-

nosť zariadenia, a hardvérová akcelerácia, ktorá bude spracovávať prijímané sieťové dáta a realizovať operácie určené softvérom.

Cieľom bakalárskej práce je vytvoriť hardvérovú akceleráciu (firmvér) pre toto zariadenie. Pri návrhu firmvéru je dávaný dôraz predovšetkým na dosiahnutie požadovanej dátovej priepustnosti pre možnosť nasadania zariadenia na vysokorýchlostných sieťach pri rýchlosti 100 Gbps. Implementácia firmvéru je realizovaná pre FPGA akceleračnú kartu s využitím vývojovej platformy NetCOPE. Okrem samotnej implementácie firmvéru a základných simulácií je nad rámec zadania bakalárskej vytvorené prostredie funkčnej verifikácie pre dôkladné overenie správnej funkcionality celého systému. Firmvér je ďalej otestovaný v laboratórnych podmienkach na reálnom hardvéri prostredníctvom hardvérového testeru, ktorý umožňuje generovanie a monitorovanie sieťového prenosu. Pomocou týchto testov je overené správne fungovanie vytvoreného firmvéru a taktiež je overená reálna priepustnosť 100 Gbps.

Práca je logicky rozdelená do niekoľkých kapitol, ktoré postupne opisujú jednotlivé časti riešenia daného problému. Kapitola 2 sa zaoberá teoretickým rozborom problematiky pre praktickú časť práce a zameriava sa na dnešné počítačové siete, a taktiež na bezpečnosť týchto sietí so zameraním na útoky typu (D)DoS. Kapitola sa taktiež venuje hardvérovým akcelerátorom rodiny COMBO, ktoré využívajú technológiu FPGA, a tiež vývojovej platforme NetCOPE poskytovanej k týmto kartám, ktorá je následne použitá pri tvorbe firmvéru hardvérovej akcelerácie. Kapitola 3 je venovaná popisu konkrétneho problému vznikajúceho pri amplifikačných (D)DoS útokoch, na ktorú je zariadenie primárne zamerianované. Následne sa kapitola venuje konceptuálnemu návrhu zariadenia určeného na riešenie danej problematiky. Kapitola 4 popisuje požiadavky a vlastnosti vychádzajúce z návrhu zariadenia z predošlej kapitoly. Ďalej sa kapitola venuje podrobnému popisu architektúry navrhovaného firmvéru a následne je venovaný priestor podrobnému popisu funkčného návrhu firmvéru. Kapitola 5 popisuje priebeh implementácie navrhnutého firmvéru a taktiež sa venuje popisu funkčnej verifikácie, ktorá bola realizovaná nad rámec zadania práce. Dosiahnuté výsledky z jednotlivých častí implementácie, a tiež namerané a dosiahnuté výsledky získané pri testovaní v laboratórnych podmienkach na reálnom hardvéry, sú popísané v kapitole 6. Posledná kapitola 7 zhrnuje celkovo dosiahnuté výsledky práce.

Kapitola 2

Teoretický rozbor

V tejto kapitole sú stručne popísané vedomosti, ktoré sú základom praktickej časti práce popísané v nasledujúcich kapitolách. Začiatok rozboru sa zaoberá základným popisom princípu a modelu počítačových sietí. Následne sa kapitola venuje popisu sieťovej bezpečnosti so zameraním na problematiku útokov typu odoprenie služby, ktorými sa táto práca primárne zaoberá. Posledná časť je venovaná popisu technológiám FPGA a hardwarovým akcelerátorom rodiny COMBO, ktoré sú použite pri realizácii praktickej časti tejto práce.

2.1 Princípy počítačových sietí

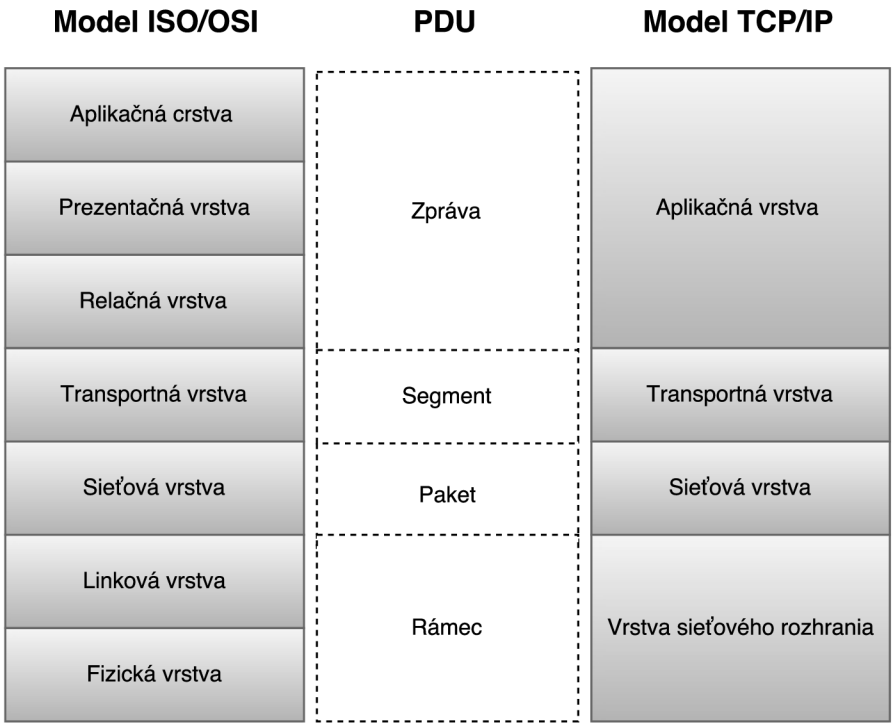
Pre uskutočnenie ľubovoľnej komunikácie je potrebné, aby existovali minimálne dva prvky, ktoré si chcú medzi sebou vymieňať informácie. Minimálne sa vždy jedná o prvok, ktorý je zdrojom informácie (správy), a prvok, ktorému je táto správa určená (príjemca). V prípade komunikácie v počítačových sieťach sa tieto prvky (zariadenia) označujú ako koncové zariadenia. V počítačových sieťach sa okrem odosielateľa a príjemcu typicky nachádzajú aj zariadenia označované ako sprostredkovateľské (intermediary devices). Tieto zariadenia sa v počítačových sieťach starajú predovšetkým o bezporuchové prenosy správ a ich správne nasmerovanie s čo najväčším úsilím o najrýchlejšie doručenie príjemcovi. Ak nie je uvedené inak, text celej sekcie vychádza z poznatkov uvedených v literatúre [12, 18].

Ďalším dôležitým prvkom pre uskutočnenie sieťovej komunikácie je mať prenosové médium, prostredníctvom ktorého sa prenášajú informácie (správy). V sieťach sa ako prenosové médium typicky používajú káble (metalické káble, optické káble) alebo bezdrôtové technológie (wifi). Prenos dát sa realizuje formou binárnej reprezentácie (logických úrovní 0/1), ktorá je prevádzaná na určitú fyzikálnu veličinu (veľkosť elektrického napätia, vlnovú dĺžku elektromagnetického žiarenia). Týmto spôsobom môžu byt zdieľané aplikačné programy, súbory, prípadne zariadenia ako tlačiarne. Tieto zariadenia alebo aplikačné programy komunikujú prostredníctvom sady štandardizovaných protokolov (sady pravidiel pre sieťovú komunikáciu).

Už na začiatku rozvoja počítačových sietí bol kladený dôraz na to, aby sa odlišovali tri základné časti komunikačného systému – technológia pre prenos signálu, vrstva pre zaistenie spoľahlivého prenosu a aplikačná vrstva ktorá poskytuje služby užívateľovi. Na základe tohto rozdelenia bol zavedení takzvaný princíp vrstvového riadenia dátovej komunikácie. Tento princíp spočíva v rozdelení sieťovej komunikácie do niekoľkých úrovní (vrstiev), kde každá z týchto vrstiev vykonáva určitú, špecifickú úlohu. Daná vrstva vždy využíva služby

nižšej vrstvy bez toho, aby musela vedieť o tom, akým spôsobom a za pomoci akých protokolov sú implementované jej funkcie.

Pre základný popis sieťovej architektúry sa používa referenčný model OSI (Open Systems Interconnect Reference Model), ktorý definuje medzinárodná organizácia ISO (International Standards Organization). Väčšina dnešných protokolov vychádza práve z tohto modelu a preto dnes môžeme považovať model OSI za primárnu architektúru pre komunikáciu v počítačových sieťach. Model OSI je referenčným modelom a slúži hlavne k pochopeniu procesov, ktoré sú využívané v sieťovej komunikácii. V vzhľadom na komplexnosť modelu OSI sa v praxi implementovala iba časť tohto modelu.



Obr. 2.1: Porovnanie referenčného modelu ISO/OSI a modelu TCP/IP

Dnes najrozšírenejším používaným modelom v počítačových sieťach, a zároveň aj v najväčšej globálnej sieti internet, je model TCP/IP. Tento model, ako už vypovedá jeho názov, je založený na protokoloch TCP (Transmission Control Protocol) a IP (Internet Protocol). Model TCP/IP je oproti modelu ISO/OSI výrazne jednoduchší a obsahuje iba 4 vrstvy na rozdiel od pôvodných siedmych vrstiev modelu ISO/OSI. Model TCP/IP spojuje služby aplikačnej, prezentačnej a relačnej vrstvy do jednej aplikačnej vrstvy modelu TCP/IP. Taktiež na úrovni fyzického rozhrania spojuje služby fyzickej a linkovej vrstvy do jednej vrstvy sieťového rozhrania. Porovnanie vrstiev oboch modelov je znázornené na obrázku 2.1. Každá z týchto vrstiev definuje dátovú jednotku pre prenos informácii (PDU, Process Data Unit), vykonáva určitú úlohu pri komunikácii a využíva určití spôsob identifikácie a adresovania účastníkov komunikácie. Pri prenose dát dochádza na strane odosielateľa k zapúzdrovaniu dát z vyššej vrstvy do dátovej jednotky (PDU) nižšej vrstvy. Na strane príjemcu dochádza k opačnému procesu, kde sú dáta postupne spracovávané (rozbaľované) a následne predávané vyšším vrstvám. Od najnižšej po najvyššiu vrstvu TCP/IP sú to tieto:

Vrstva sieťového rozhrania reprezentuje najnižšiu vrstvu modelu TCP/IP, ktorá zabezpečuje komunikáciu a riadenie hardvérových zariadení (sieťové karty), prostredníctvom ktorých sa pristupuje k prenosovému médiu. Vrstva taktiež popisuje štandardy, ktoré určujú vlastnosti linky (typ média, rýchlosť prenosu, kódovanie logických úrovní). Najrozšírenejšie protokoly používané na realizáciu vrstvy sieťového rozhrania sú Ethernet (na adresovanie sa používa MAC adresa) prípadne Token Ring. Prenášané dátové jednotky sa nazývajú rámce.

Sieťová vrstva zodpovedá za správne doručovanie dátových blokov prostredníctvom sieťových zariadený až na miesto určenia. Pri doručovaní dát sa usiluje o nájdenie tej najvhodnejšej cesty medzi komunikujúcimi zariadeniami. Tento proces za nazýva smerovanie, ktorý však nezaručuje spoľahlivosť doručenia dátových blokov. Sieťová vrstva nezabezpečuje kontrolu správnosti presunu dát ani ich opravu. O kontrolu a opravu dát sa musia postarať vyššie vrstvy modelu TCP/IP. O základnú funkčnosť sieťovej vrstvi sa stará protokol IP, ktorý pre adresovanie využíva IP adresu. Správna funkcionalita protokolu IP je podporovaná ďalšími pomocnými protokolmi, medzi ktoré patria napríklad: ICMP (Internet ControlMessage Protocol), ktorý sa využíva na posielanie informácii o výnimočných stavoch ako sú nedostupnosť cieľovej aplikácie alebo nedostupnosť koncového zariadenia. RIP (Routing Information Protocol) na získavanie potrebných informácii pre smerovanie dát. IGMP(Internet Group Management Protocol) pre riadenie multicastových skupín. ARP(Address Resolution Protocol) na preklad IP adries na MAC adresy (využívané v nižších vrstvách).

Transportná vrstva zabezpečuje prenos dát medzi koncovými zariadeniami. Vytvára takzvané logické spojenia medzi aplikáciami (procesmi), ktoré sú na týchto zariadeniach spustené. Na tejto vrstve pracujú protokoly TCP a UDP. TCP je protokolom, ktorý garantuje spoľahlivý prenos dát. Spojenie medzi komunikujúcimi aplikáciami sa vytvára ešte predtým, než sú odosielané užívateľské dáta. Dáta sú rozdelené do menších segmentov, ktoré sú označované prostredníctvom sekvenčných čísel. O úspešnom doručený dátových segmentov je odosielacia strana informovaná prostredníctvom potvrdení (ACK pakety) odosielaných zo strany príjemcu. Protokol UDP, na rozdiel od protokolu TCP, nevytvára spojenie pred zahájením prenosu dát a negarantuje spoľahlivosť doručenia. Vzhľadom na túto skutočnosť je výhodou protokolu UDP oproti protokolu TCP nižšia réžia na prenos dát.

Aplikačná vrstva je najvyššou vrstvou modelu TCP/IP, ktorej úlohou je posielanie správ a interakcia medzi komunikujúcimi stranami (aplikáciami). Vrstva zabezpečuje reprezentáciu dát, ich kódovanie a definuje pravidlá určené pre riadenie dialógu medzi komunikujúcimi aplikáciami. Jednotlivé protokoly sieťových aplikácii pracujú na tejto vrstve. Medzi aplikačné protokoly patrí napríklad protokol pre prenos súborov FTP (File Transfer Protocol), protokol pre prenos webových stránok HTTP (Hypertext Transfer Protocol) a protokol na zaistenie prenosu elektronickej posty SMTP (Simple Mail Transfer Protocol).

2.2 Bezpečnosť počítačových sietí

S rozvojom počítačových sietí na globálnu úroveň sa objavila nová problematika zoberajúca sa bezpečnosťou počítačových sietí, ktorou sa bolo nutné zažať vážne zaoberať. Problematika bezpečnosti a ochrany sa stala neoddeliteľnou súčasťou počítačových sietí. V dnešnej dobe rozmachu počítačovej kriminality je potrebné chrániť počítačové siete a systémy využívajúce túto technológiu pred mnohými spôsobmi zneužitia prístupu neoprávnenými osobami. Cieľom rôznych útokov, ktorých účelom je získať takýto neoprávnený prístup, môže byť ukradnutie, zneužitie a modifikácia citlivých údajov alebo zapríčinenie nekorektného správania, prípadne úplne vyradenie činnosti daného systému a sieťovej služby. V nasledujúcej sekcii sa buďme zaoberať útokmi typu odoprenie služby, ktorých účelom je práve zneprístupnenie alebo vyradenie daného systému, služby z prevádzky. Ak nie je uvedené inak, text nasledujúcich sekcií vychádza prevažne z [15, 13, 8].

Účelom útokov typu odoprenie služby (Denial of Service, DoS) je vyvolať nedostupnosť sieťovej služby pre ostatných legitímnych užívateľov, prípadne úplný výpadok celej služby. Na základe spôsobu akým je útok vedení môžeme DoS útoky rozdeliť na dva základné typy: aplikačné útoky a volumetrické útoky. Aplikačné útoky využívajú zraniteľnosti a chyby cieľovej služby. Priebeh útoku spočíva v tom, že útočník na základe odhalenej zraniteľnosti, pošle na cieľovú službu špeciálne upravené správy, ktoré vyvolajú na strane cieľovej služby nekorektné chovanie. Tým zapríčiní jej nedostupnosť pre ostatných užívateľov, prípadne úplnú nefunkčnosť danej služby. Účelom volumetrických útokov je vyčerpanie (zahltenie) zdrojov sieťovej služby, prípadne celkových zdrojov sieťovej infraštruktúry za použitia hrubej sily. Cieľom týchto útokov typicky býva preťaženie sieťovej linky (šírky pásma), prostredníctvom ktorej je daná služba pripojená, prípadne vyčerpanie pamäte alebo iných prostriedkov daného systému a zariadení zapojených v sieťovej infraštruktúre. Tento typ útoku prebieha tak, že útočník posiela na cieľovú službu obrovské množstvo vygenerovaných správ, čím dôjde k už spomínaným problémom. Následkom je že legitímny užívatelia k tejto službe majú výrazne obmedzený prístup, prípadne je pre nich služba úplne nedostupná.

Distribuované útoky typu odoprenie služby (Distributed Denial of Service, DDoS) sú založené sa tých istých princípoch ako útoky typu DoS ale prevažná väčšina z nich je cielená ako volumetrické útoky. Hlavným rozdielom je že útočník na generovanie útoku využíva väčšie množstvo počítačových staníc ako je tomu pri útokoch typu DoS, kde útočník typicky používa iba jedno zariadenie (počítačovú stanicu, mobilné zariadenie). Týmto spôsobom je možné vygenerovať útok s niekoľko násobne vyššiu intenzitou a väčším, negatívnym dopadom. Zariadenia, ktoré sa účastnia takéhoto útoku môžeme rozdeliť do dvoch skupín. Do prvej skupiny môžeme zaradiť tie zariadenia, ku ktorým má útočník oprávnený prístup a cielene ich využíva na generovanie DDoS útoku. Do druhej skupiny môžeme zaradiť zariadenia, ktoré sa takéhoto útoku účastia nedobrovoľne, respektíve zariadenia, u ktorých útočník využíva neoprávnený prístup. K získaniu tohoto prístupu využíva útočník známe zraniteľnosti a chyby sieťových služieb. Najrozšírenejším spôsobom, prostredníctvom ktorého sa dá získať prístup k takýmto zariadeniam, je pomocou škodlivého programu označovaného taktiež ako bot. Účelom takéhoto škodlivého programu je sprístupniť útočníkovi zdroje zariadenia, ktoré následne útočník môže využívať k svojim účelom. Najčastejším prostriedkom na šírenie takéhoto programu sú rôzne webové stránky, elektronická pošta, prípadne priamym nainštalovaním na zariadenie. Takto infikované zariadenia sa taktiež prezívajú ako bot, slovensky taktiež ako zombie. Pomenovaním bot označujeme taktiež počítačových robotov, ktorých účelom je vykonávať autonómne nejakú činnosť. Infikované zariadenia, boti, následne kontaktujú server nazývaný ako príkaz a kontrola (C&C), ktorý

ovláda útočník. Prostredníctvom tohto serveru útočník dáva príkazy jednotlivým botom a môže tak vytvárať rozsiahle útoky typu DDoS. Skupina takto infikovaných zariadení, ktoré sú navzájom prepojené a pod kontrolou jedného alebo skupiny útočníkov, nazývame Botnet.

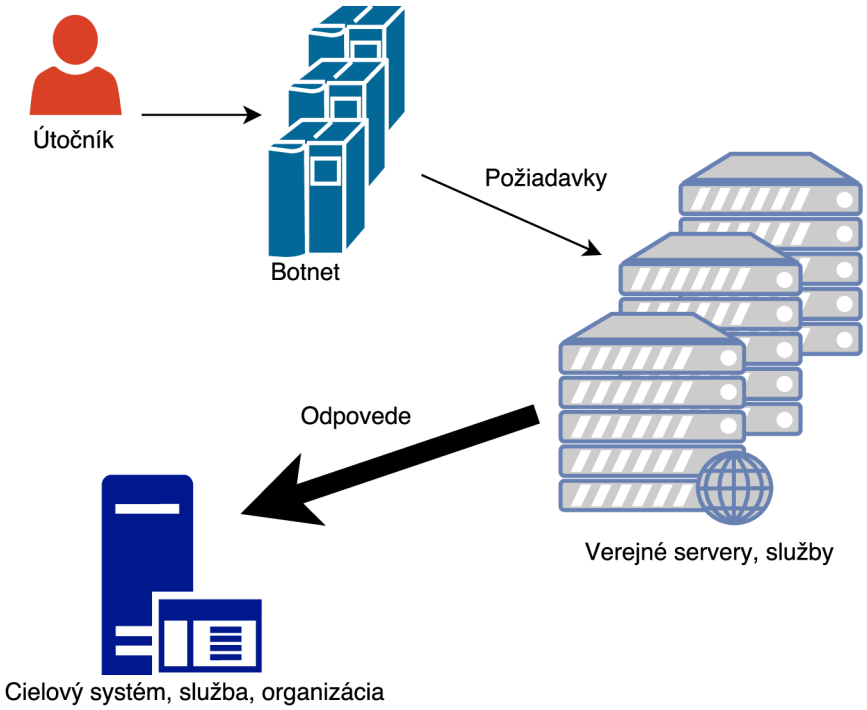
V dnešnej dobe existuje veľké množstvo metód používaných pri útokoch typu DoS a DDoS. Pre potrebu tejto práce si popíšeme tie najpoužívanejšie metódy v súčasnosti a metódy, na ktoré sa táto práca v praktickej časti primárne zameriava.

SYN Flood útok je metóda, ktorá zneužíva protokol TCP. TCP protokol je spojovo orientovaná služba a pred zasielaním dát je nutné najprv ustanoviť spojenie. K ustanoveniu spojenia sa používa takzvaný „3-way handshake“. Ustanovenie tohto spojenia prebieha nasledovne: Klient pošle na server paket SYN, čím dáva najavo že chce ustanoviť spojenie. Server po prijatí tohto paketu alokuje potrebné zdroje na vytvorenie tohto spojenia, nastaví časovač a odpovie klientovi paketom SYN-ACK. Časovač určuje dobu po ktorú bude server čakať na potvrdenie od klienta a bude tým držať alokované zdroje pre toto spojenie. Následne klient po obdŕžaní paketu SYN-ACK odošle na server paket ACK, čím potvrdí vytvorenie tohto spojenia a môže sa začať realizovať prenos užívateľských dát. Útočník pri tomto spôsobe útoku posiela veľké množstvo paketov SYN na všetky porty cieľového zariadenia. Zariadenie na všetky tieto žiadosti odpovedá paketmi SYN-ACK a na určitú dobu pre každú žiadosť alokuje potrebné zdroje pre toto spojenie. Server následne čaká na potvrdenie paketom ACK. Útočník už ale tieto pakety neposiela, a tým je server nútení držať alokované zdroje po stanovenú dobu v časovači. Počas tejto doby útočník stále posiela nové pakety SYN, čím núti server vytvárať nové spojenia a alokovať si čím ďalej tím viacej zdrojov. Tým sa server dostane do stavu, kedy vyčerpá všetky svoje dostupné zdroje, a už nie je schopný vytvárať nové spojenia ani pre legitímnych užívateľov. Útočník taktiež môže pri generovaní paketov SYN podvrhnúť falošnú IP adresu, čo má za následok že mu nebude doručená žiadna odpoveď od serveru ale odpovede budú posielané na podvrhnutú IP adresu [6].

UDP Flood útok využíva bezstavový transportní protokol UDP (User Datagram Protocol). U protokolu UDP nie je nutné ustanovovať spojenia ako je to u protokolu TCP. UDP taktiež nezaručuje spoľahlivé doručenie paketov ani ich doručenie v správnom poradí. Útočník pri tomto type útoku posiela na cieľový server UDP pakety na náhodné alebo konkrétne porty. Server sa po prijatí paketu UDP snaží nájsť aplikáciu, ktorá na danom porte očakáva príjem paketov. Ak sa nenájde žiadna aplikácia, ktorá by na danom porte očakávala príjem paketov, server odošle naspäť informatívny paket so správou o nedostupnosti danej služby (Destination Unreachable). Pri tomto útoku sa taktiež využíva podvrhnutie falošnej IP adresy, čím útočník docieli aby odpovede od napadnutého serveru neboli smerované naspäť ku zdroju útoku. Cieľom útoku UDP Flood je zahltiť sieťovú linku, prenosové pásmo napadnutého serveru. Tento ciel útočník dosiahne posielaním mohutného objemu UDP paketov v kombinácii posielaných odpovedí o nedostupnosti z napadnutého zariadenia. Z pohľadu legitímnych užívateľov sú služby pracujúce na takto napadnutom zariadení výrazne spomalené, prípadne úplne nedostupné [7].

Amplifikačné, reflektované útoky typu DDoS pri ktorých je cieľom útočníka zosilniť svoj útok. Toto zosilnenie útočník dosiahne vygenerovaním malého množstva dotazov na typicky verejné dostupné systémy, ktoré disponujú veľkou prenosovou kapacitou.

Tieto systémy následne na tieto dotazy generujú veľké odpovede, ktoré sú následne preposielané na cieľový systém, službu. K tomuto účelu sa typicky zneužívajú verejne dostupné sieťové služby ako je napríklad systém doménových mien (Domain Name System, DNS) alebo systém pre synchronizáciu času (Network Time Protocol, NTP). Aby útočník docielil požadovaného efektu, musí splniť minimálne nasledujúce podmienky. Aby sa dosiahlo zosilnenie útoku DDoS je potrebné vygenerovať taký typ dotazov, na ktoré bude daný systém generovať podstatne väčšie odpovede. Ďalej je potrebné, aby útočník zabezpečil, že tieto odpovede budú smerované na cieľový systém útoku. To sa dosiahne sfalšovaním, podvrhnutím zdrojovej IP adresy za IP adresu systému, kam má byť útok smerovaný. Útoky využívajúce tento princíp sa taktiež prezívajú ako útoky s odrazom, prípadne reflektované útoky. Princíp takéhoto útoku je znázornený na obrázku 2.2.



Obr. 2.2: Znázornenie priebehu amplifikačného útoku DDoS

DNS Amplifikačný útok je spôsob pri ktorom sa zneužíva systém doménových mien DNS. Samotný útok prebieha tak, že útočník posiela na DNS servery požiadavky na kompletné DNS záznamy určitej doménovej adresy. Tieto záznamy sú následne posielané na systém, ktorý je cieľom útoku. Pred zahájením útoku si útočník vyberie takú doménovú adresu, pre ktorú existuje veľké množstvo DNS záznamov. Tým sa dosiahne že pre pomerne malý dotaz, na tieto záznamy, vygeneruje DNS server mnohonásobne väčšiu odpoveď. Odpovede od DNS serverov v niektorých prípadoch môžu byť tak veľké, že musia byť rozdelené do niekoľkých paketov. Ako príklad môžeme uviesť dotaz o veľkosti 64 bajtov, ktorý viedol na odpoveď DNS serveru o veľkosti

3223 bajtov, čo je takmer 51 násobné zosilnenie [16]. Pre dosiahnutie toho, aby DNS servery tieto odpovede posielali na systém, ktorý má byť cieľom útoku, útočník v dotazoch pre DNS servery sfalšuje svoju zdrojovú IP adresu za adresu cieľového systému. Týmto spôsobom je cieľový systém zahlcovaný veľkým množstvom mohutných odpovedí od systému DNS. To má za následok, že cieľový systém útoku pod touto záťažou už nie je schopný spracovávať požiadavky od legitímnych užívateľov a služby bežiace v tomto systéme sa stávajú nedostupné. Priemerné zosilnenie DNS amplifikačného útoku, v pomere veľkosti dotazu a odpovede DNS serveru, sa typicky pohybuje v 70:1 čo predstavuje sedemdesiat krát väčšie zosilnenie útoku [5].

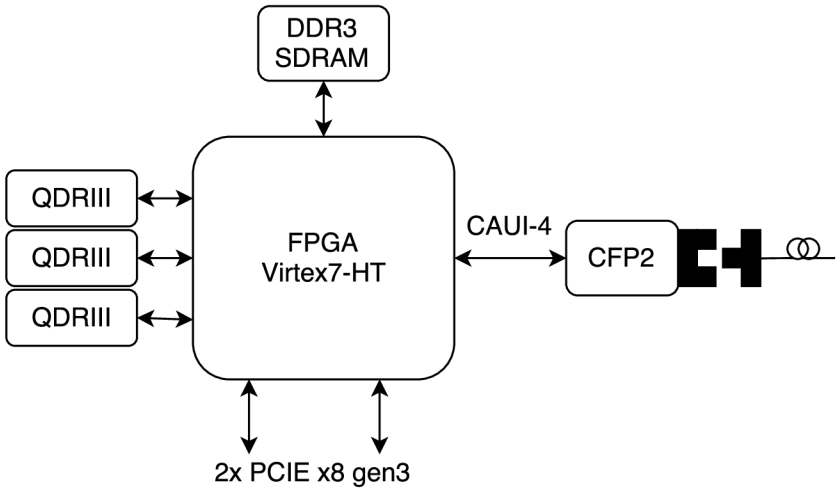
NTP Amplifikačný útok zneužíva systém NTP. Pri tomto type amplifikačného útoku posiela útočník na servery NTP dotazy, ktoré obsahujú príkaz monlist. Tento príkaz slúži na monitorovanie činnosti na NTP serveri. Pomocou tohto príkazu sa môžeme serverov NTP dotazovať na zoznam posledných pripojených adries k tomuto serveru. Tento zoznam môže obsahovať až 600 záznamov IP adries, čo má za následok vygenerovanie niekoľko násobne väčšej odpovede ako bol samotný dotaz. Aj tu sa využíva princíp podvrhnutia falošnej IP adresy pre presmerovanie odpovedi zo systému NTP na cieľový systém. Podobne ako je to u DNS amplifikačného útoku aj tu je cieľový systém zahltený veľkým množstvom odpovedí od systému NTP, čo má za následok zhoršenie dostupnosti daného systému, prípadne úplnú nedostupnosť z pohľadu legitímnych užívateľov. Tento typ útoku je v súčasnosti možné uskutočniť iba prostredníctvom NTP serverov ktorých programové vybavenie je v nižšej verzii ako je 4.2.8. NTP servery aktualizované na vyššiu verziu už príkaz monlist nepodporujú. Zosilnenie NTP amplifikačného útoku sa typicky nachádza v rozmedzí pomeru medzi veľkosťou dotazu a odpovede od serveru NPT v 20:1 až 200:1. To reprezentuje možné zosilnenie útoku až dvesto krát [5].

SNMP amplifikačný útok k uskutočneniu útoku využíva systém SNMP (Simple Network Management Protocol), ktorý je určení na zber štatistík, testovanie stavu a manažment sieťových zariadení. Útočník pri tomto spôsobe útoku podvrhne IP adresu v hlavičke paketu, ktorý obsahuje príkaz SNMP GET. Takto upravený paket sa bude na koncovom zariadení javiť ako legitímny dotaz a všetky odpovede sú následne smerované na cieľový systém. Dopad tohto útoku na daný cieľový systém je taký istý ako u predošlých dvoch popísaných, amplifikačných útokov. Tento spôsob útoku je možné realizovať iba na protokol SNMP vo verzii jedna a dva. Od tretej verzie SNMP už tento princíp útoku nie je možný, pretože u tejto verzia sa vyžaduje overenie prostredníctvom mena/hesla a taktiež sa používa šifrované spojenie.

2.3 Hardvérové akcelerátory rodiny COMBO

Hardwarové akcelerátory rodiny COMBO sú zariadenia vyvíjané v rámci vedeckovýskumnej skupiny Liberouter, ktorá patrí pod združenie CESNET, v spolupráci s firmou NetCOPE Technologies. CESNET je združenie vysokých škôl a akadémie vied českej republiky, ktorá spravuje a rozvíja národnú e-infraštruktúru pre vedu, výskum a vzdelanie zahrnujúce počítačovú sieť, dátové úložisko, prostredie pre spoluprácu a poskytuje široké množstvo služieb [1]. Praktická časť tejto práce je realizovaná pre hardvérový akcelerátor s označením COMBO-100G1. Blokové schéma karty je znázornené na obrázku 2.3. Jedná sa o rýchlu sieťovú kartu, ktorá disponuje obvodom FPGA Xilinx Virtex-7HT pre možnosť pokročilého

spracovania sieťovej komunikácie pri rýchlosti 100 Gbps. Firmvér realizujúci hardvérovú akceleráciu je v tejto práci vyvíjaný primárne pre tento čip FPGA. Ďalšie komponenty nachádzajúce sa na tejto karte sú optické sieťové rozhrania CFP2 pre príjem a odosielanie sieťových dát, rozhranie systémovej zbernice PCI-Express Gen3 pre komunikáciu so softvérovým rozhraním a prístup do operačnej pamäte, moduly statických pamätí QDRIIIE a dynamických pamätí DDR3 SDRAM [20].

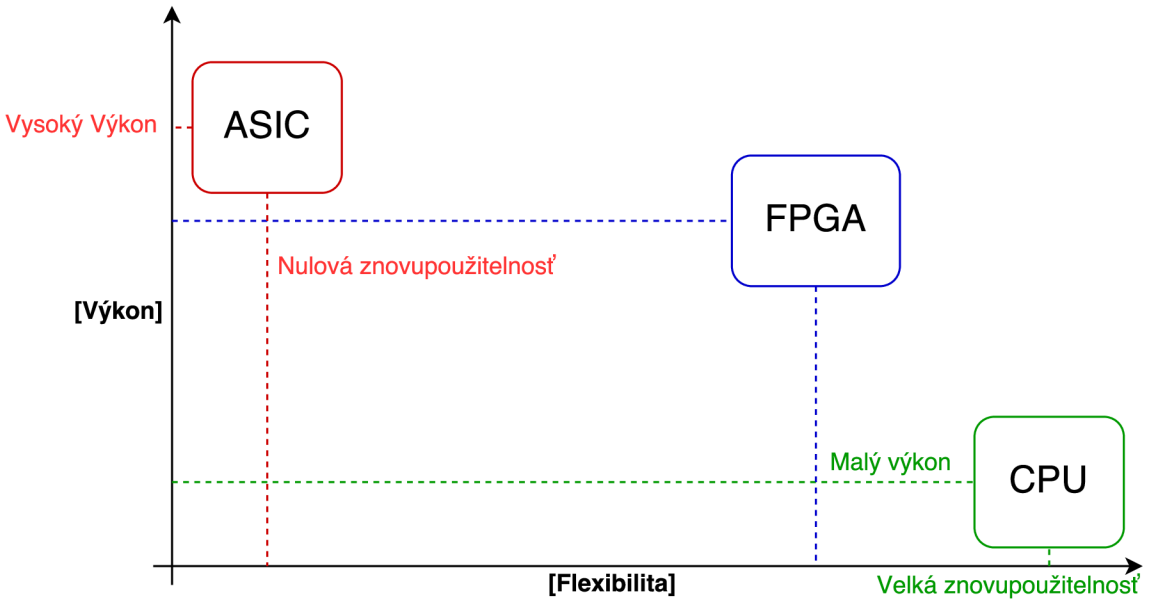


Obr. 2.3: Blokové schéma karty COMBO [20]

Technológia programovateľných logických hradiel FPGA (Field Programmable Gate Array) je kompromisom medzi aplikačne špecifikovanými integrovanými obvodmi ASIC (Application Specific Integrated Circuits) a základnými výpočetnými jednotkami dnešných počítačov CPU (Central Procesing Unit). Pred samotným popisom princípu a východ technológie FPGA je potrebné najprv sa oboznámiť už so spomenutými technológiami ASIC a CPU. Porovnanie technológií FPGA, ASIC a CPU je znázornené na obrázku 2.4, kde na osi X je znázornená dostupná znovu použiteľnosť (flexibilita) a na osi Y je znázornený výkon jednotlivých technológií.

CPU, slovensky pomenované ako procesory, sú hardvérové čipy, ktoré sú určené na vykonávanie strojových inštrukcií, z ktorých sú tvorené počítačové aplikácie (programy). Tieto procesory tak nie sú obmedzené na vykonávanie jednej konkrétnej špecifickej aplikácie (úlohy). Týmto spôsobom je možné vytvoriť ľubovoľnú aplikáciu (úlohu, program) popísanú pomocou týchto základných inštrukcií, ktorá bude následne vykonaná prostredníctvom procesoru. Výhodou takéhoto riešenia je veľká flexibilita. Nevýhodou je však relatívne pomalá rýchlosť spracovania takýchto aplikácii oproti hardvérovým riešeniam špecializovaných presne pre danú aplikáciu, úlohu.

Technológia ASIC je pravým opakom procesorov. Označenie ASIC reprezentuje hardvérový obvod, čip, ktorý je špecializovaný na vykonávanie presne danej aplikácie, úlohy. Výhodou tejto technológie je veľká rýchlosť a efektivita s akou dokážu vykonávať danú aplikáciu, úlohu. Veľkou nevýhodou je práve nulová flexibilita. Pri akejkoľvek zmene, prípadne oprave navrhnutej aplikácie, je nutné vyrobiť nový hardvérový čip ASIC, pretože do štruktúry už vyrobeného čipu nie je možné zasahovať. Vývoj aplikácií pre túto technológiu je oproti softwarovému programovaniu pre procesory výrazne náročnejší, a tým aj nákladnejší z ekonomického hľadiska. Zároveň pri návrhu a tvorbe takejto aplikácie znamenajú akékoľvek chyby značné predraženie vývoja vzhľadom na fakt, že je nutné vyrobiť



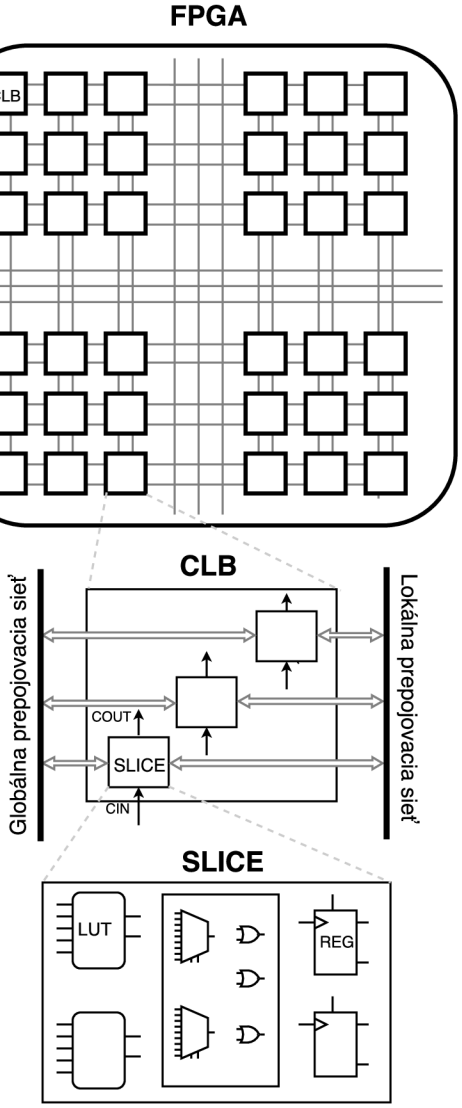
Obr. 2.4: Porovnanie technológií FPGA, ASIC a CPU

úplne nové hardvérové čipy. Táto technológia je preto určená pre aplikácie, ktoré vyžadujú veľkú rýchlosť spracovania, nízku spotrebu a malé rozmery výsledného obvodu.

Programovateľné logické hradlá FPGA sú hardvérové obvody, čipy, ktoré umožňujú svoju rekonfiguráciu. Tieto čipy môžu byť naprogramované tak, aby realizovali ľubovoľnú užívateľskú aplikáciu prostredníctvom kombinačnej a sekvenčnej logiky. Hlavnou výhodou oproti ASIC je jednoduchší, rýchlejší vývoj aplikácií, s čím súvisí aj rýchlejšie uvedenie daného produktu na trh. Prípadná oprava chýb a rozširovanie vlastností aplikácie je ďalšou veľkou výhodou, pretože hardvérové čipy nie je nutné vymieňať za nové, ale stačí ich preprogramovať. Oproti ASIC poskytujú menšiu výpočetnú rýchlosť, avšak s dnešným neustálim vývojom v oblasti technológií FPGA sa tento rozdiel neustále zmenšuje. Táto rýchlosť sa udáva v počte cyklov hodinového signálu za jednotku času, pomenovaná taktiež ako taktovacia frekvencia. Naopak oproti procesorom poskytuje FPGA oveľa vyššiu rýchlosť vykonávania konkrétnych aplikácií, napríklad pri spracovávaní a filtrovaní paketov sieťovej komunikácie. Zároveň si zachováva čiastočnú flexibilitu vďaka možnej rekonfigurácii. Aktuálny trend rozvoja vývojových nástrojov pre technológiu FPGA prináša stále nové, rýchlejšie a jednoduchšie spôsoby implementácie užívateľských aplikácií, čím sa tieto nástroje postupne dostávajú na úroveň vyjadrovacej schopnosti dnešných moderných programovacích jazykov používaných na vývoj aplikácii pre obecné procesory. Takýmto jazykom je napríklad jazyk C alebo Pascal. Najrozšírenejšie programovacie jazyky pre vývoj aplikácií pre technológiu FPGA sú jazyk VHDL (VHSIC Hardware Description Language) alebo Systém Verilog.

Nasledujúci text, popisujúci základný princíp technológie FPGA, vychádza predovšetkým z vedomostí uvedených v [17]. Štruktúra obvodu FPGA je znázornená na obrázku 2.5. FPGA obvody (vrchná časť obrázku) sú tvorené maticou konfigurovateľných logických blokov CLB (Configurable Logic Block). CLB bloky (prostredná časť obrázku) sú navzájom poprepájané prostredníctvom prepojovacej siete, matice PSM (Programmable Switch Matrix). Bloky CLB sú ďalej rozdelené na menšie bloky nazývané ako slice (spodná časť ob-

rázku). CLB boky môže vývojár aplikácie konfigurovať podľa svojich potrieb. Každý blok



Obr. 2.5: Štruktúra obvodu FPGA

slice obsahuje funkcionálne generátory, multiplexory, registre, prípadne ďalšie komponenty s možnosťou konfigurácie. Funkcionálne generátory LUT (Look-Up Table) sú realizované ako vyhľadávacie tabuľky s n-vstupmi. Pre vyjadrenie funkcií obsahujúce ľubovoľný počet premenných sú LUT spojované do stromovej štruktúry a tvoria tak základnú jednotku pre implementáciu kombinačnej logiky. Registre môžu byt nakonfigurované ako preklápacie obvody typu FF (flip-plop), reagujúce na hladinu hodinového signálu, alebo ako záchytné preklápacie obvody nazývané ako latch, reagujúce na hladinu nejakého signálu. Multiplexor sa správa ako prepínač a umožňuje na jeden výstup nastaviť hodnotu z niekoľkých pripojených vstupov na základe výberového, adresovacieho vstupu. Funkcionalitu mutiplexoru je taktiež možné realizovať prostredníctvom funkcionálnych generátorov LUT.

Prepojovanie blokov CLB je možne realizovať prostredníctvom lokálnej alebo globálnej prepojovacej siete. Vzhľadom na veľkú kombináciu možných prepojení sú tieto siete veľmi rozsiahle a zaberajú väčšinu plochy obvodov FPGA. Lokálna prepojovacia sieť je určená

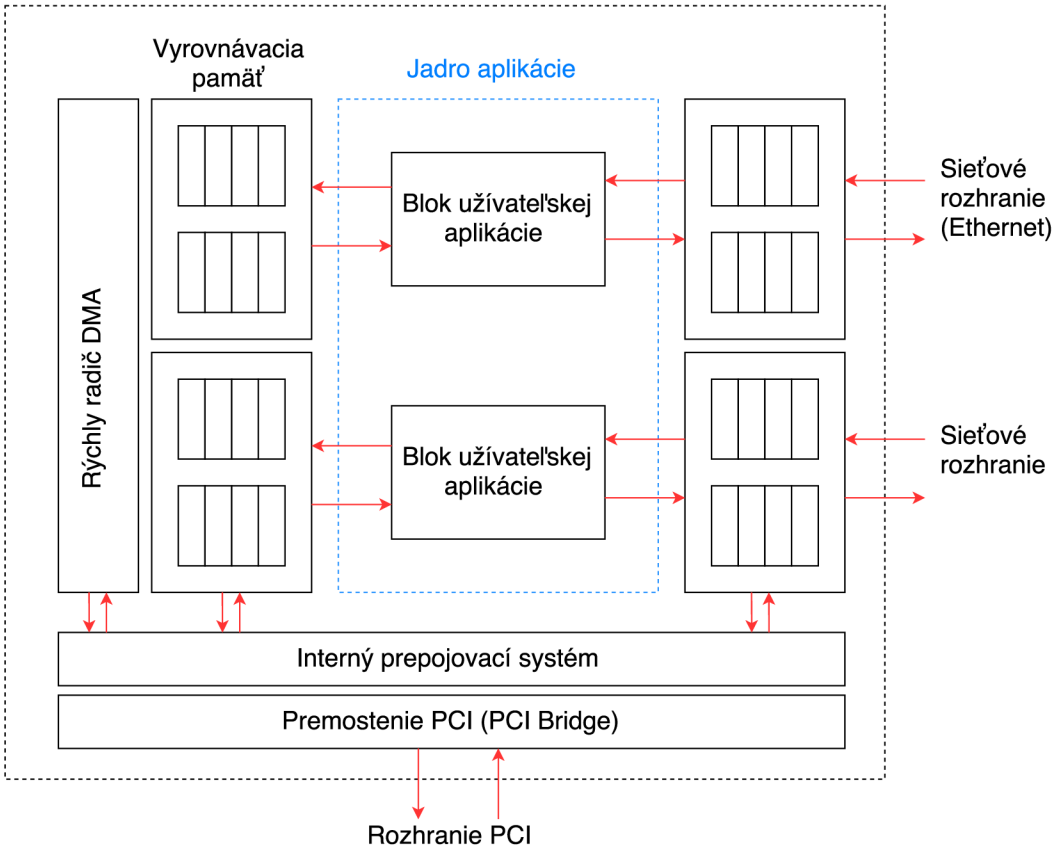
na prepojovanie susediacich CLB blokov a slúži na vytváranie rýchlych prepojení. Dlhé prepojovacie cesty medzi logickými obvodmi majú za následok znižovanie maximálnej možnej taktovacej frekvencie. Pri tvorbe obvodu, prepojení (na základe navrhnutej aplikácie) je preto snaha o čo nekratšie prepojenia medzi logickými jednotkami. Zdroje prepojovacej logiky majú svoj limit a neje zaručené že sa pre akúkoľvek ľubovoľnú aplikáciu podarí takýto obvod vytvoriť, prípadne že sa podarí vytvoriť obvod splňujúci minimálnu požadovanú taktovaciu frekvenciu.

Okrem samotných logických blokov CLB a prepojovacej siete sa na obvode FPGA tiež nachádzajú špeciálne obvody pre rozvod hodinového signálu a taktiež resetovacieho signálu pre sekvenčné obvody. Okrem základných blokov CLB sa na obvode FPGA typicky vyskytujú blokové pamäte BRAM (block RAM) pre uloženie väčšieho množstva dát, prípadne na realizáciu iných pomocných jednotiek ako je FIFO. Ďalej obvody FPGA obsahujú komponenty pre spracovanie signálov DSP (Digital Signal Processing), ktoré obsahujú rýchle aritmetické logické jednotky ALU, prípadne rýchle porovnávacie obvody a ďalšie optimalizované, matematické jednotky pre vstupy so širokým dátovým rozhraním. Taktiež tieto obvody obsahujú komponenty realizujúce pripojenie externých zariadení ako sú napríklad rýchle sériové spoje (Ethernet, PCI-Express) alebo externé pamäte.

Medzi hlavných výrobcov FPGA v dnešnej dobe patria firmy Xilinx a Intel. Jednotlivé typy a verzie obvodov FPGA sa typicky líšia počtom blokov CLB, BRAM, DSP, maximálnou možnou taktovacou frekvenciou, podporovaných rozhraní a použitou technologickou výrobou. Štruktúra a princíp fungovania, ktorý už bol popísaný, je však u všetkých takýchto obvodov FPGA rovnaký.

Pri návrhu zložitých aplikácií ako je napríklad spracovanie sieťového prenosu je možnosť využiť takzvané IP jadrá (Intellectual Property Cores). Tieto IP jadrá poskytujú samotný výrobcovia FPGA alebo iné spoločnosti zaoberajúce sa vývojom aplikácii pre technológiu FPGA. Praktická časť tejto práce využíva prostriedky poskytované IP jadrom známeho ako platforma NetCOPE [11]. NetCOPE je modulárne vývojové prostredie, ktoré poskytuje infraštruktúru, sadu komponent a softvérových nástrojov, pre podporu rýchleho vývoja sieťových aplikácií na hardwarových akcelerátoroch používajúce FPGA, ktorá bola pôvodne vyvinutá združením CESNET pre akcelerátory rodiny COMBO. Účelom platformy NetCOPE je uľahčiť prácu vývojárovi samotnej aplikácie a zbaviť ho tak nízko úrovňovej problematiky, ktorá sa zaoberá prácou s jednotlivými komponentami konkrétnej sieťovej karty pre ktorú je platforma NetCOPE poskytovaná. NetCOPE poskytuje jednotné komunikačné rozhranie pre prenos dát medzi samotnou aplikáciou a infraštruktúrou NetCOPE. To umožňuje pomerne ľahký prenos danej aplikácie medzi jednotlivými, podporovanými, akceleračnými kartami. Štruktúra platformy NetCOPE je znázornená na obrázku 2.6.

Hlavné komponenty, ktoré platforma NetCOPE poskytuje, sú vstupné a výstupné komponenty a vyrovnávacie pamäte pre prístup ku komunikácii prostredníctvom sieťového rozhrania (pravá časť obrázku). Taktiež zaisťuje komunikáciu a prenos dát prostredníctvom zbernice PCI-Express (spodná časť obrázku), Kruhové vyrovnávacie pamäte a radič priameho prístupu do operačnej pamäte DMA (Direct Memory Access) (ľavá časť obrázku). Súčasťou sú aj softvérové knihovny a nástroje pre ich správu. Pri vývoji samotnej aplikácie pre platformu NetCOPE sú poskytované už vytvorené, základné komponenty ako je napríklad implementácia pamätí FIFO (First in First out), pamätí CAM (Content-addressable Memory) a pomocné prepojovacie komponenty medzi jednotlivými blokmi užívateľskej aplikácie. Súčasťou sú taktiež komponenty pre prácu s komunikačnými protokolmi medzi infraštruktúrou NetCOPE a samotnou aplikáciou, prípadne komponenty umožňujúce ko-



Obr. 2.6: Architektúra platformy NetCOPE

munikáciu prostredníctvom protokolov externých prvkov nachádzajúcich sa na akceleračnej karte mimo samotného FPGA ako sú napríklad externé pamäte QDR (Quad Data Rate).

Kapitola 3

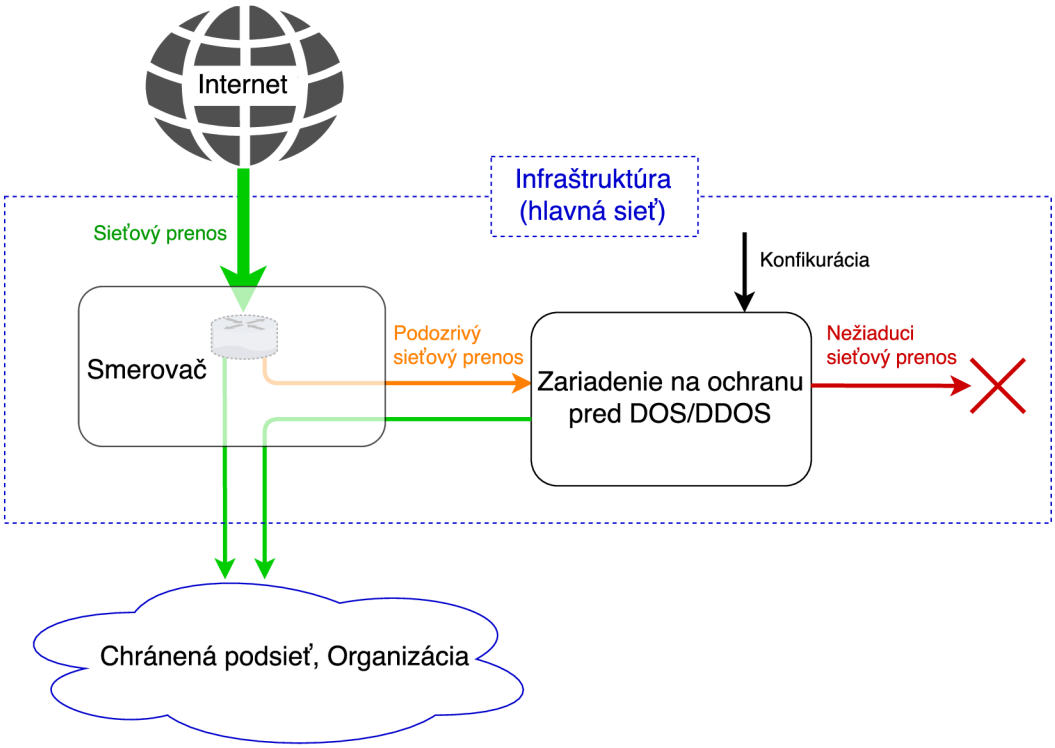
Konceptuálny návrh zariadenia

Táto kapitola sa venuje návrhu a popisu zariadenia na ochranu pred útokmi typu DoS a DDoS. Prvá časť je venovaná popisu problematike, ktorou sa zariadenie zaoberá. Druhá časť je venovaná popisu navrhovaného zariadenia určeného na riešenie danej problematiky.

3.1 Problematika amplifikačných útokov

Účelom amplifikačných, reflektovaných útokov typu DoS/DDoS typicky nie je priamo cieľové koncové zaradenie (služba), ale ich cieľom býva samotná sieťová infraštruktúra. Účelom týchto útokov je zahltiť sieťové linky a vyčerpať všetky dostupné prostriedky sieťových zariadení tvoriace infraštruktúru. Cieľom typicky býva celá počítačová sieť (organizácia) a nie iba jedno koncové zariadenie (služba). Pred týmto útokom tak nie je možné sa brániť na koncových zariadeniach, pretože linky a vyhradené prenosové pásmo, prostredníctvom ktorého je koncové zariadenie pripojené, sú zahltené. Preto je potrebné tento problém riešiť už na úrovni počítačovej siete. Ukázateľnom takýchto útokov sú typicky obrovský počet paketov, nezvyčajne veľkej dĺžky, na zraniteľných verejných službách ako je DNS alebo NTP. Sieťové zariadenia (Routery) takúto zložitú detekciu, pri vysokých prenosových rýchlostiach 100 Gbps, nie sú schopné realizovať. Distribuované riešenia pre takýto typ útokov sú z hľadiska ekonomickej a správcovskej strany veľmi nákladné.

Navrhovaním riešením tohto problému je vytvorenie centralizovaného zariadenia využívajúce hardvérovú akceleráciu, realizovanú prostredníctvom technológie FPGA, za účelom dosiahnutia vysokých nárokov na rýchlosť spracovania a dátovej priepustnosti. Spôsob nasadenia zariadenia je znázornený na obrázku 3.1. Princíp centralizovaného zariadenia spočíva v tom že v danej infraštruktúre (modré ohraničenie na obrázku) bude exitovať iba jedno takéto zariadenie (pravá časť obrázku). Následne bude na toto zaradenie presmerovaná sieťová prevádzka (určená pre danú podsieť, organizáciu), prostredníctvom smerovacích zariadení danej infraštruktúry (ľavá, prostredná časť obrázku), ktorú zaradenie očistí od nežiaducich dát (paketov). Takto vyčistená sieťová prevádzka bude preposlaná naspäť do sieťovej infraštruktúry a následne do danej podsiete, organizácii (spodná časť obrázku).



Obr. 3.1: Spôsob nasadenia zariadenia na ochranu pred DoS/DDoS

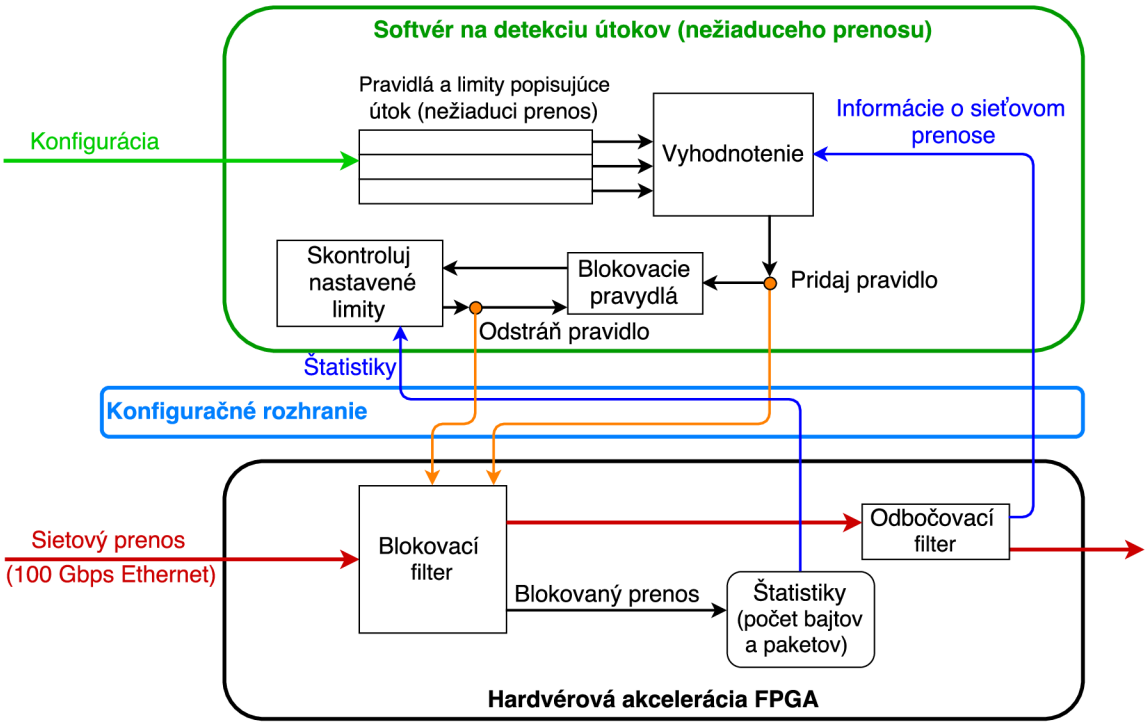
3.2 Princíp zariadenia

Zariadenie je vyobrazené na obrázku 3.2. Zariadenie je tvorené troma časťami a to softvérom na detekciu útokov a nežiaduce prenosu (Horná, zelená časť obrázku), hardvérovým akcelerátorom FPGA (čierna, spodná časť obrázku) na spracovanie sieťového prenosu a komunikačným (riadiacim) rozhraním (prostredná, modrá časť obrázku) určeného pre konfiguráciu hardvérového akcelerátoru a vzájomnú výmenu dát medzi softvérovou a hardvérovou častou zariadenia.

Aby bolo možné vykonávať úspešnú detekciu a blokovanie nežiaduceho prenosu (útoku) je potrebné zariadenie vhodne nakonfigurovať. K tomuto účelu slúži možnosť špecifikácie pravidiel. Pravidlá sú tvorené kombináciou hľadaných špecifických hodnôt a informácií o sieťovom prenose ako sú rozsahy zdrojových a cieľových adries IP a portov, transportní protokol, dĺžka paketov, fragmentácia paketov a hodnoty protokolu TCP. Okrem tohto popisu sieťového prenosu obsahujú pravidlá informácie o limitoch, ktoré špecifikujú maximálne povolené množstvo prenesených dát daného sieťového prenosu, ktoré môže byť prepustené a nebude tak vyhodnotené ako nežiaduci sieťový prenos (útok). Limity sú udávané v počte paketov, prípadne bajtov za jednotku času. Po nakonfigurovaní týchto pravidiel prebieha kontrola a vyhodnocovanie pravidiel (pravá, horná zelená časť obrázku). K tejto činnosti sú využívané získavané informácie o sieťovom prenose z hardvérovej akcelerácie prostredníctvom odbočovacieho filtru (pravá, horná časť čiernej časti obrázku).

K tomuto účelu nie je potrebné, aby softvérová časť zariadenia pracovala s celými sieťovými dátami tak ako boli prijaté na sieťovom rozhraní akceleračnej karty. Odbočovací filter tak neposiela celé pakety sieťového prenosu ale vytvára takzvané hlavičky UH, ktoré sú poskytované softvéru namiesto sieťových paketov. Hlavičky UH obsahujú informácie

o jednotlivých prijatých paketoch ako sú adresy IP, porty, protokol, dĺžka a ďalšie potrebné informácie v jednotnom formáte, ktoré sú vyžadované softvérovou častou zariadenia. Týmto spôsob sa softvér na detekciu nežiaduceho sieťového prenosu už nemusí zaoberať získavaním informácii z paketov a ich filtrovaním, pretože táto činnosť je realizovaná hardvérovým akcelerátorom. Touto funkcionalitou je tak možné dosiahnuť softvérové spracovanie a detekciu pre požadovanú prenosovú rýchlosť 100 Gbps.



Obr. 3.2: Zariadenie na ochranu pred DoS/DDoS

Pri zistený prekročenia limitu maximálneho povoleného sieťového prenosu pre niektorú zo špecifikovaných podmienok, sú na základe zdrojovej adresy IP vybraný najväčší prispievatelia do tohto sieťového prenosu tak, aby po ich zablokovaní už nebol prekročený maximálny povolený limit pre pravidlo popisujúce sieťový prenos. Po výbere zdrojových IP adries, s príslušným popisom konkrétneho sieťového prenosu na základe pravidiel (porty, protokol, TCP flags, fragmentácia, veľkosť), sú pre tabuľku vygenerované pravidlá popisujúce blokovaný prenos. Tieto pravidlá sú tvorené jednotlivými, vybratými, zdrojovými adresami IP a príslušným popisom sieťového prenosu (porty, protokol, TCP flags, fragmentácia, veľkosť, limit), na základe ktorých budú prijímané pakety blokované. Súčasne sú tieto pravidlá nahrané do blokovacieho filtru hardvérovej akcelerácie, ktorá uskutočňuje samotné blokovanie sieťového prenosu (ľavá, spodná časť čiernej časti obrázku). Hardvér si k takto zablokovanému prenosu uchováva štatistiky o počte zablokovaných bajtov a paketov k jednotlivým blokovacím pravidlám. Softvér následne tieto štatistiky v periodických intervaloch vyčíta (ľavá, spodná časť zelenej časti obrázku) a kontroluje či ešte stále dochádza pri blokovaných sieťových prenosoch k prekračovaniu nastavených limitov v pravidlách. V prípade zistenia, že už nedochádza k tomuto prekračovaniu limitu, je pravidlo na blokovanie daného, sieťového prenosu odstránené z tabuľky pravidiel a súčasne taktiež z blokovacieho filtru v hardvérovej akcelerácii.

Popísaný konceptuálny návrh zariadenia bol vytvorený v rámci spolupráce so združením CESNET. Obsah a zameranie tejto práce je ďalej venovaný návrhu a implementácii hardvérovej časti zariadenia (spodná, čierna časť obrázku) a softvérového konfiguračného rozhrania (prostredná, modrá časť obrázku).

Kapitola 4

Návrh firmvéru

Kapitola popisuje podrobný návrh firmvérovej časti zariadenia pre hardvérový akcelerátor COMBO-100G1 s využitím IP coru NetCOPE. Prvá časť je venovaná popisu požiadavkám a vlastnostiam vychádzajúcich z návrhu zariadenia na ochranu pred DoS a DDoS útokom, ktoré tvoria základ pri tvorbe návrhu firmvéru. Druhá časť sa venuje popisu architektúry navrhovaného firmvéru pre FPGA Virtex-7 nachládajúcom sa na hardvérovom akcelerátore COMBO-100G1 s využitím vlastností a funkcionality NetCOPE. Tretia časť sa zaoberá podrobnému popisu a funkčného návrhu firmvérovej časti.

4.1 Požiadavky pre implementáciu firmvéru

Na základe miesta a spôsobu nasadenia zariadenia v sieťovej infraštruktúre vyplývajú na firmvér základné požiadavky a vlastnosti potrebné na správnu a bezproblémovú funkcionalitu celého zariadenia.

Vysoká dátová priepustnosť je vzhľadom na daný spôsob zapojenia zariadenia nevyhnutná. Pre správnu funkcionalitu je potrebné, aby zariadenie pracovalo na najvyššej úrovni sieťovej infraštruktúry (hlavnej sieti), prostredníctvom ktorej sú jednotlivé podsiete (organizácie) navzájom prepojované a pripojované do globálnej siete internetu. Tieto hlavné siete sú typicky dimenzované na vysoké prenosové rýchlosti, aby bola zaručená bezproblémová sieťová komunikácia medzi pripojenými organizáciami pracujúcich typicky na nižších prenosových rýchlostiach. Ako príklad môžeme uviesť sieťovú infraštruktúru CESNET, kde v súčasnosti pracuje hlavná sieť na rýchlosti 100 Gbps (Gigabyťs per second), kde pripojované organizácie (napr. univerzity, výskumné organizácie) pracujú na prenosovej rýchlosti 1 až 10 Gbps [8]. Z tohto dôvodu je potrebné, aby firmvérová časť bola schopná pracovať na vysokých prenosových rýchlostiach. Návrh firmvérovej časti zariadenia, popísaný v nasledujúcich častiach kapitoly, je vzhľadom na túto skutočnosť navrhovaný pre plnú sieťovú priepustnosť na rýchlosti 100 Gbps. Prostredníctvom vstupného a výstupného rozhrania NetCOPE sú sieťové dáta prenášané v dátových slovách po 512 bitovej zbernici. Aby bola zaručená sieťová priepustnosť 100 Gbps, musí implementácia firmvéru dosahovať minimálnu taktovaciu frekvenciu 200 MHz.

Blokovanie nežiaduceho prenosu (útoku). Na základe špecifikácie amplifikačných, reflektovaných útokov typu DDoS je pre tento typ útokov typické generovanie obrovského

množstva sieťových paketov z veľkého počtu zdrojových adries IP. Tieto pakety, vzhľadom na zneužívanie zraniteľných verejných sieťových služieb, majú rovnakú charakteristiku ako sú čísla portov, protokol, fragmentácia, veľkosť a nastavenie hlavičky TCP protokolu. V prípade týchto DDoS amplifikačných útokov, kedy je útok generovaný z veľkého počtu zariadení, je potrebné, aby blokovací filter bol schopný pracovať rádovo s tisícmi až desaťtisícimi pravidlami. Súčasne je potrebné, aby pridávanie a odoberanie týchto pravidiel mohlo byť realizovateľné dynamicky za chodu, bez zastavenia spracovávania sieťového prenosu.

Detekcia konca útoku. Aby bolo možné túto činnosť realizovať, je potrebné, aby si firmvér uchovával štatistiky o sieťovom prenose k jednotlivým filtračným pravidlám blokovacieho filtru, určujúce nežiaduci sieťový prenos. Tieto štatistiky búdu obsahovať informácie ako je počet zahodených paketov, počet zahodených bajtov. Štatistiky sú následne prístupné softvérovej časti, ktorá si tieto údaje bude periodicky vyčítať, a na základe týchto informácií a pravidiel popisujúcich koniec daného útoku, odstráni alebo ponechá filtračné pravidlo blokujúce tento sieťový prenos vo firmvéri.

Modifikácia sieťového prenosu za účelom poskytnutia jednoduchšieho, nenáročného spôsobu zapojenia zariadenia do samotnej infraštruktúry. Táto požiadavka vyplýva na základe komunikácie so sieťovými administrátormi, ktorých úlohou je spravovať sieťové infraštruktúry. Požadovaná modifikácia sieťového prenosu zahrnuje zásah do údajov slúžiacich pre samotné smerovanie a preposielanie prenosu na sieťových zariadeniach (routeroch) infraštruktúry. Zariadenie by malo poskytovať možnosť modifikácie položiek sieťových paketov, na základe zvolenej konfigurácie, ako sú VLAN (Local Area Network), adresy sieťových rozhraní MAC (Media Access Control) a kontrolu (modifikáciu) položky reprezentujúcej dobu platnosti dát paketu TTL (Time to Live). Účelom poskytnutia týchto prostriedkov je tak umožniť podstatne rýchlejšieho nasadenia a konfiguráciu samotného zariadenia. Bez týchto možností by bolo nutné pri zapojovaní zariadenia využívať výrazne náročnejšiu techniku smerovania (napríklad Policy based routing) namiesto jednoduchších prostriedkov ako je technológia VRF, prostredníctvom ktorej je možne realizovať smerovanie paketov na základe hodnôt VLAN.

Ďalšie požiadavky a vlastnosti firmvérovej časti zariadenia, ktoré vyplynuli z nasledujúceho popisu návrhu firmvéru a použitej technológií, budú popísané v ďalšej časti textu.

4.2 Architektúra firmvéru

Architektúra firmvéru je znázornená na obrázku 4.1 na ktorú sa nasledujúci text odkazuje. Podrobné schéma navrhovaného firmvéru je znázornené v prílohe A.

Za účelom dosiahnutia požadovanej dátovej priepustnosti je použitý princíp hlbokého zreťazeného spracovania. Firmvér je rozdelený na jednotlivé samostatné bloky (moduly), ktoré vykonávajú jednotlivé operácie (úkony). Moduly znázornené modrou farbou reprezentujú komponenty implementované, poskytované štruktúrou NetCOPE. Moduly znázornené čiernou a zelenou farbou reprezentujú navrhovanú firmvérovú časť zariadenia, realizujúcu požadovanú hardvérovú akceleráciu. Štruktúra firmvéru je rozdelená do troch hlavných častí (ciest) a to na dátovú cestu (znázornená červenou farbou), riadiacu cestu (znázornená čiernou farbou) a cestu určenú na konfiguráciu firmvéru prostredníctvom softvérového ro-

zhrania (znázornená zelenou farbou). K tomuto účelu sú využívané nasledovné rozhrania poskytované architektúrou NetCOPE.

Vstupné sieťové rozhranie poskytuje dáta sieťového rozhrania vo forme ethernetových rámcov. Rozhranie taktiež poskytuje základnú kontrolu dát prijímaných zo sieťového rozhrania ako je overenie správnosti kontrolného súčtu jednotlivých rámcov a maximálny povolený limit dĺžky rámcov. Ethernetové rámce sú zbavené nepotrebnej časti pre ďalšie spracovanie ako je preambula a oddeľovač ethernetových rámcov. Rámce sú taktiež zbavené kontrolného súčtu ktorý je opätovne prepočítavaný a pridávaný na výstupnom rozhraní platformy NetCOPE. Okrem samotného poskytovania sieťových dát, poskytuje rozhranie tiež doprovodné informácie k jednotlivým rámcom ako je presná časová značka príchodu daného rámcu, veľkosť prijímaných dát a identifikácia sieťového rozhrania z ktorého dáta pochádzajú. K prenosu dát využíva toto rozhranie zbernicu a veľkosti 512 bitov.

Výstupné sieťové rozhranie je určené na odosielanie ethernetových rámcov. Platforma NetCOPE následne prijaté rámce podľa potreby rozširuje na požadovanú minimálnu dĺžku. Ďalej je k rámcom pridávaná preambula a oddeľovač začiatku rámca. Taktiež je prepočítaný nový kontrolný súčet a pridaný na koniec rámcov. Pri odosielaný rámcov na sieťové rozhrania sa vytvára požadovaná medzera medzi jednotlivými rámcami, ktorá vychádza na základe špecifikácie popisujúcej tento prenos.

Rozhranie pre komunikáciu pomocou DMA slúži pre posielanie príslušných dát prostredníctvom radiča DMA a zbernice PCI-Express do operačnej pamäte počítača. Okrem samotných dát je potrebné špecifikovať číslo kanálu pre prenos dát, presnú veľkosť dát a formát popisujúci prenášané dáta. Tieto údaje sú pridávané v radiči DMA pred samotné dáta pomocou vkladanej hlavičky. Tieto údaje sú následné využívané pre spracovanie prenesených dát v operačnej pamäti prostredníctvom softvérových nástrojov.

Konfiguračné a riadiace rozhranie. Prostredníctvom tohto rozhrania, poskytovaného platformou NetCOPE, je možné zapisovať a konfigurovať riadiace a stavové registre jednotlivých komponent platformy. Rozhranie je taktiež dostupné pre komponenty užívateľskej aplikácie a môže byť využité na konfiguráciu jednotlivých častí, komponent navrhovaného firmvéru prostredníctvom radiča slúžiaceho na prístup k jednotlivým registrom pomocou softvérových nástrojov.

Úlohou dátovej cesty je poskytovať a prenášať dáta sieťovej komunikácie (pakety) prostredníctvom vstupného a výstupného rozhrania platformy NetCOPE medzi jednotlivými modulmi (komponentami) pracujúcich s týmto dátami. Komponenty pracujúce s týmito dátami, nachádzajúce sa v architektúre navrhovaného firmvéru, môžeme zaradiť do dvoch skupín. Prvá skupina reprezentuje komponenty ktoré využívajú prijaté dáta (pakety) na získanie požadovaných informácii o jednotlivých paketoch ale nevykonávajú žiadnu modifikáciu a zmenu samotných dát. Do tejto skupiny patrí komponenta určená na získanie požadovaných informácii z paketov HFE (Header Field Extractor) a vyrovnávacia pamäť slúžiaca na uchovávanie paketov určených na ďalšie spracovanie (ľavá strana obrázku). Do druhej skupiny patria komponenty, ktoré na základe prijatých inštrukcii (úkonov) z riadiacej cesty vykonávajú modifikáciu a úpravu samotných paketov v takom poradí ako je to znázornené na obrázku. Sem sa riadia komponenty určené na zmenu položiek VLAN, kontrolu a úpravu

položky TTL a jednotku, ktorá preposiela dáta na požadované rozhranie. Na základe rozhodnutia tejto jednotky pakety pokračujú spracovaním v obecnom editore a editore MAC adries alebo sú spracovávané v generátore hlavičiek UH a v skartovači paketov (pravá, spodná časť obrázku).

Úlohou riadiacej cesty je na základe získaných informácii zo vstupných dát a nastavených riadiacich, stavových registrov prostredníctvom softvérovej konfigurácie určiť akcie (úkony), ktoré majú byť vykonané na prijatých sieťových dátach. Po vyhodnotený a určený požadovaných operácii sú následne vygenerované príkazy poslané modulom vykonávajúce zmeny na prijatých dátach. Riadiaca cesta je tvorená modulmi HFE pre získanie potrebných dát o jednotlivých paketoch, odbočovací filter, IP filter, modul s tabuľkou špecifických podmienok, modul na počítanie štatistík zahodených paketov a rozhodovací modul na vyhodnocovanie priorít jednotlivých operácií.

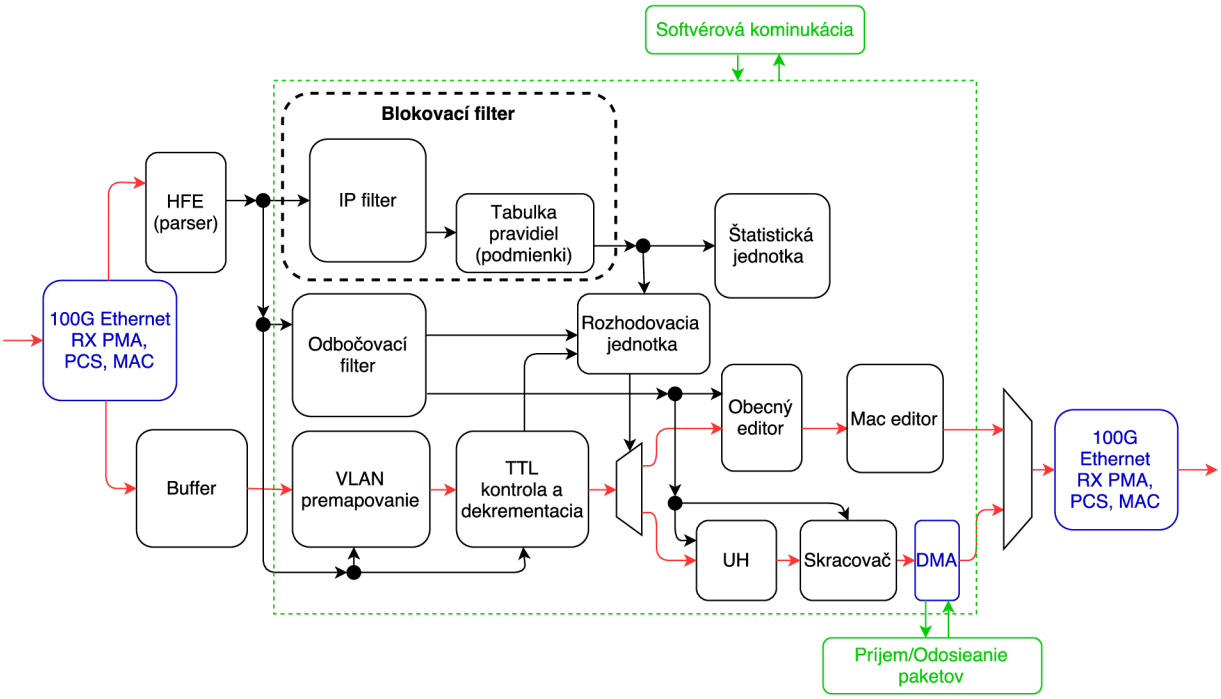
Cesta pre konfiguráciu firmvéru je určená na riadenie a nastavenie činnosti, správania jednotlivých modulov prostredníctvom softvérového rozhrania, podľa požadovaného spôsobu chovania. K tomuto účelu je využívané dostupné konfiguračné a riadiace rozhranie platformy NetCOPE. Softvérové rozhranie, ktoré je súčasťou navrhovaného firmvéru, je popísané v kapitole 5.

4.3 Podrobný implementačný návrh

Táto časť popisuje fungovanie firmvéru a návrh jednotlivých modulov, komponentou uskutočňujúcich požadované operácie, ktoré svojou spoločnou činnosťou tvoria požadovanú funkcionalitu navrhovaného firmvéru. Pre dosiahnutie požadovanej dátovej priepustnosti sú dáta medzi jednotlivými modulmi predávané prostredníctvom vyrovnávacích pamätí FIFO. Implementácia modulov HFE, odbočovacieho filtru a jednotky schopnej fitltrovať sieťový prenos na základe vyhľadávacieho kľúča (IP filter), sú prevzaté z existujúcej hardvérovej akcelerácie Hanic [3] a upravené pre potreby navrhovaného firmvéru. Nasledujúci text popisuje princíp fungovania firmvéru na obrázku 4.1.

Vyhodnocovanie sieťových dát začína v komponente HFE ktorej účelom je získať potrebné informácie o rámcoch (paketoch) z jednotlivých záhlaví protokolov, v ktorých sú zapuzdrené prenášané dáta (pravá, horná časť obrázku). Tieto informácie sú následne poskytované odbočovaciemu filtru, ktorý na základe týchto informácii a nakonfigurovaných filtračných pravidlách, určí akcie, ktoré budú vykonané s príslušnými sieťovými dátami (paketmi). Tento filter rozhoduje o tom, kam môžu byť dáta preposlané (sieťové rozhranie, kanál DMA), prípadne či majú byť dáta zahodené.

Súčasne prebieha vyhodnocovanie v druhom IP filtri. Úlohou tohto filtru je označiť príslušné dáta, pakety na základe zdrojovej IP adresy, ktoré budú následne ďalej vyhodnocované modulom obsahujúcim tabuľku so špecifickejšími podmienkami popisujúce konkrétne hľadané sieťové dáta. Podmienka, ktorá bude následne použitá pri vyhodnocovaný príslušného paketu, je určená na základe zhody v IP filtri. Na základe zhody alebo nezhody získaných informácií o pakete a vybranej podmienky z tejto tabuľky, sa rozhoduje o zahodení alebo prepustený paketu k ďalšiemu spracovaniu. Táto akcia má väčšiu prioritu ako akcia vybraná prostredníctvom odbočovacieho filtru. Funkcionalita blokovacieho filtru (popísaná v nasledujúcej sekcii) je tak tvorená činnosťou IP filtru a tabuľkou špecifických podmienok. Pakety, ktoré splnili danú podmienku a budú tým pádom zahodené, sú následne spracovávané v štatistickej jednotke, ktorá zaznamená požadované údaje v niektorom zo štatistických čítačov. Konkrétny štatistický čítač, ktorý bude použití pre príslušné pakety,



Obr. 4.1: Firmvér hardvérového akcelerátoru

je určený na základe zhody v blokovacom filtri (čiarkované čierne ohraničenie v hornej časti obrázku).

Vybrané akcie, na základne vyhodnotenia odbočovacieho filtru a blokovacieho filtru, sú ďalej poslané rozhodovaciemu modulu, ktorý na základe daných priorít určí výslednú akciu a vygeneruje príkazy pre jednotlivé modli realizujúce tieto úkony. Tieto príkazy určia či budú dáta preposlané na sieťové rozhranie a stým spojené ďalšie sparovanie, či budú preposlané do operačnej pamäte prostredníctvom prenosov DMA, prípadne či budú dáta preposlané na oba tieto smery.

Samotné sieťové dáta sú po dobu počas ktorej bude vyhodnotené ich spracovanie uložené vo vyrovnávacej pamäti typu FIFO. Z tejto pamäte sa dáta dostavajú do jednotky, ktorá slúži na výmenu položiek VLAN podľa zvolenej konfigurácie. Následne sú dáta preposlané na spracovanie do modulu pre kontrolu TTL. V tejto jednotke dochádza ku samotnej kontrole položky TTL a prípadnému prepočtu novej hodnoty. Pri nesplnení požadovanej podmienky pre hodnotu TTL je vygenerovaná akcia, ktorá požaduje zahodenie daného paketu, ktorá je ďalej spracovaná v rozhodovacom module. Dáta pokračujú na spracovanie do modulu, ktorý na základe určených príkazov z rozhodovacieho modulu vykoná zahodenie alebo prípadné preposlanie dát na sieťové rozhranie alebo operačnej pamäte.

Pri preposlaní na sieťové rozhranie dáta prechádzajú ďalším spracovaním v obecnom editore. Obecný editor na základe svojej konfigurácie vykoná vloženie alebo modifikáciu štvôr bajtového bloku na požadovanom mieste v pakete. Následne sú dáta ďalej spracovávané v editore MAC adries. Tu dochádza k nahradeniu zdrojovej a cieľovej MAC adresy alebo ich vzájomnej výmene. Následne sú pakety predané výstupnému sieťovému rozhraniu platformy NetCOPE a poslané na príslušné sieťové rozhranie.

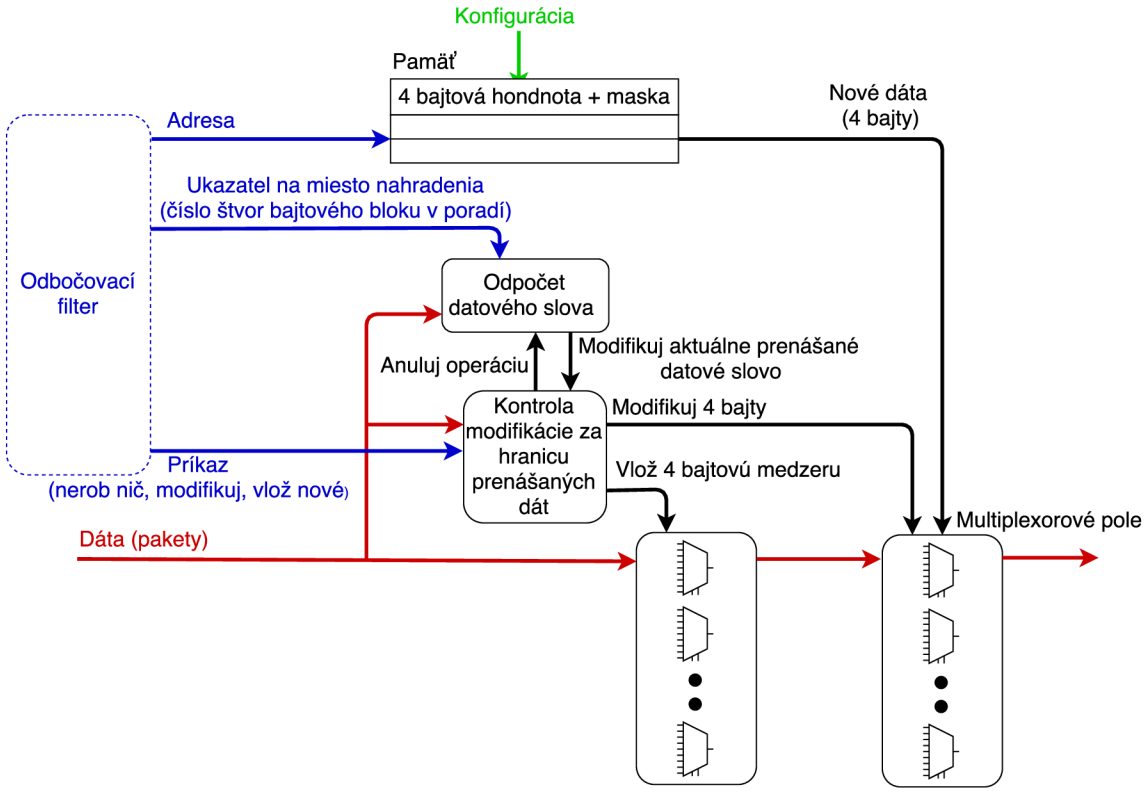
Pri preposlaní dát do operačnej pamäte sú následne spracovávané v generátore hlavičiek UH. Na základe zvolenej konfigurácie a výsledku odbočovacieho filtru sú ďalej posie-

lané buďto samotné sieťové dáta (pakety) alebo vygenerované hlavičky UH, ktoré obsahujú spracované požadované informácie o danom pakete. Nasleduje spracovanie v module, ktorý v prípade preposielania celých paketov do operačnej pamäte, vykoná skrátenie paketu na požadovanú maximálnu veľkosť. Pakety alebo hlavičky UH sú ďalej poslané na rozhranie radiču DMA, ktorý vykoná prenos dát do operačnej pamäte.

Parser HFE (Header Field Extractor) je komponenta ktorej úlohou je získanie údajov z hlavičiek jednotlivých sieťových protokolov, v ktorých sú postupne zapuzdrované prenášané dáta. Táto komponenta tak musí byť schopná postupne extrahovať informácie z jednotlivých úrovní zapuzdrenia protokolov. Získané informácie sú následne poskytované prostredníctvom rozhrania komponenty ďalším modulom k spracovaniu. Poskytované informácie sú napríklad IP adresy, MAC adresy, číslo protokolu, číslo portov, dĺžka rámcu, dĺžka paketu IP a ďalšie položky obsiahnuté v hlavičkách jednotlivých protokolov do ktorých boli dáta zapuzdrené. Okrem samotných dát sú k vybraným položkám poskytované informácie o ich polohe v samotných prenášaných dátach. To je realizované prostredníctvom ofsetu (adresy), ktorý obsahuje informácie o poradovom čísle dátového slova a číslo konkrétneho bajtu kde sa daná informácia nachádza. Týmto spôsob je možné získať informáciu o polohe začiatku hlavičky protokolu IP v prenášaných dátach, ktorá je následne využívaná v moduloch uskutočňujúcich modifikáciu dát.

Vyrovnávacia pamäť (Buffer) je modul, ktorého úlohou je uchovávať prijaté dáta po dobu, kým nie je vyhodnotené aké operácie budú s príslušnými dátami vykonané. Po určení príslušných operácii, sú dáta na vyžiadanie poskytnuté nasledujúcemu modulu pre spracovanie. Komponenta je realizovaná ako pamäť typu FIFO. To znamená že dáta, ktoré boli prijaté (uložené) na vstupnom rozhraní ako prvé, sú poskytnuté na výstupnom rozhraní taktiež ako prvé v poradí. Veľkosť pamäte je optimálne určená na základe maximálnej možnej dĺžky zreťazeného spracovania takým spôsobom, aby na vstupnom sieťovom rozhraní nedochádzalo k zbytočnému, nežiaducemu zahadzovaniu prijímaných dát a zároveň, aby sa neplytvalo dostupnými zdrojmi FPGA pre zbytočne obrovskú pamäť.

Obecný editor realizuje vkladanie alebo výmenu štvôr bajtového bloku dát v prenášaných paketoch. Principiálne schéma modulu je znázornené na obrázku 4.2. Hodnoty ktoré môžu byť vložené alebo vymenené sú uložené v pamäti zloženej s BRAM pamätí (horná časť). K uložením, štvôr bajtovým, dátam sú taktiež ukladané masky, ktorými sa označujú jednotlivé bajty, ktoré majú alebo nemajú byť prípadne vymenené. Pamäť je možné konfigurovať prostredníctvom softvérového rozhrania (zelená časť). To ktoré údaje z pamäte budú použité je určené na základe zhody a výberu akcie z odbočovacieho filtru (modrá časť). Filter taktiež poskytuje informáciu o tom, ktorý štvôr bajtový blok v paketoch má byť vymenený, prípadne kam majú byť dáta vložené. Na základe tejto hodnoty a postupne prijímaných dátových slov, z ktorých je daný paket tvorený, prebieha postupný prepočet a následné určenie dátového slova kde má byť modifikácia vykonaná (prostredná časť). Súčasne prebieha kontrola či sa príkaz na vykonanie modifikácie nepokúša túto akciu vykonať mimo dáta aktuálne prijímaného paketu. Po zistení tohto stavuje je operácia modifikácie paketu zrušená (anulovaná) a paket je ďalej preposlaný bez vykonania danej modifikácie (prostredná, čierna časť). Obecný editor umožňuje operácie nahradenia štvôr bajtového bloku alebo vloženie nového štvôr bajtového bloku do paketu. Po určený dátového slova a vý-

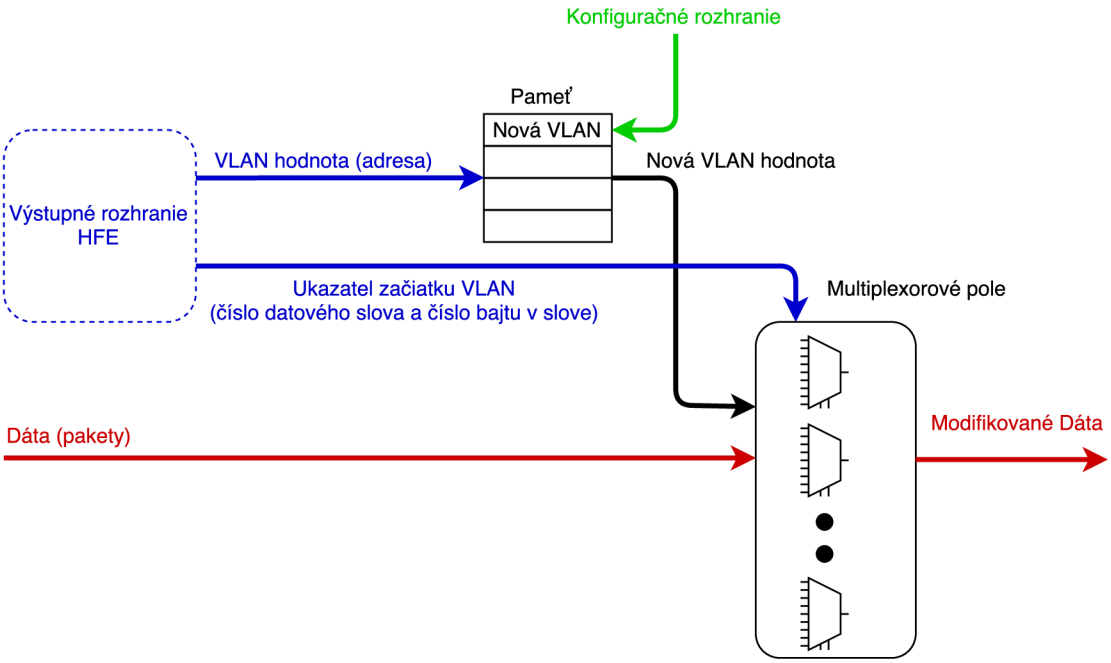


Obr. 4.2: Princíp fungovania obecného editoru

bere dát z pamäte je uskutočnená modifikácia prostredníctvom multiplexorových polí (spodná časť). Táto akcia je taktiež určená odbočovacím filtrom. Editor je možné využiť na modifikáciu IP adries, portov a ďalších položiek. Pre modifikáciu položiek VLAN, MAC adresy a TTL sú vyhradené samostatné modifikačné jednotky popísané v nasledujúcom texte.

Modul pre výmenu VLAN slúži k výmene starej položky VLAN nachládajúcej sa v prijatých dátach za novú hodnotu podľa zvolenej konfigurácie. Principiálne schéma modulu je znázornené na obrázku 4.3. Modul konkrétne realizuje výmenu dvanásť bitovej položky VLAN ID, ktorá je jednou z hodnôt z ktorých je tvorená celá položka VLAN. Hodnota VLAN je získavaná z komponenty HFE (modrá farba). Modul obsahuje pamäť vyskladanú z blokov BRAM (horná, prostredná časť). Táto pamäť má nastavenú veľkosť presne na hodnotu 4096, ktorá zároveň reprezentuje hodnotu všetkých možných kombinácii hodnôt, ktoré je možné nastaviť prostredníctvom dvanástich bitov. Týmto spôsobom je možné použiť starú hodnotu VLAN ID ako adresu do tejto pamäte. Hodnota VLAN ID (adresa) je získaná z modulu HFE. Na príslušných adresách sú následne uložené nové hodnoty VLAN ID, ktoré nahradia starú hodnotu VLAN ID v dátach. Pamäť je taktiež pripojená ku konfiguračnému rozhraniu, ktoré umožňuje nastaviť pamäť na požadovanú konfiguráciu. Nová hodnota VLAN ID je následne poskytnutá multiplexorovému poli, ktorý na základe ofsetu (adresy) začiatku položky VLAN z komponenty HFE, uskutoční požadovanú výmenu (spodná, pravá časť).

Modul pre kontrolu TTL. Principiálne schéma je znázornené na obrázku 4.4. K tomuto účelu modul využíva hodnotu TTL a adresu začiatku hlavičky IP získanú z kompo-

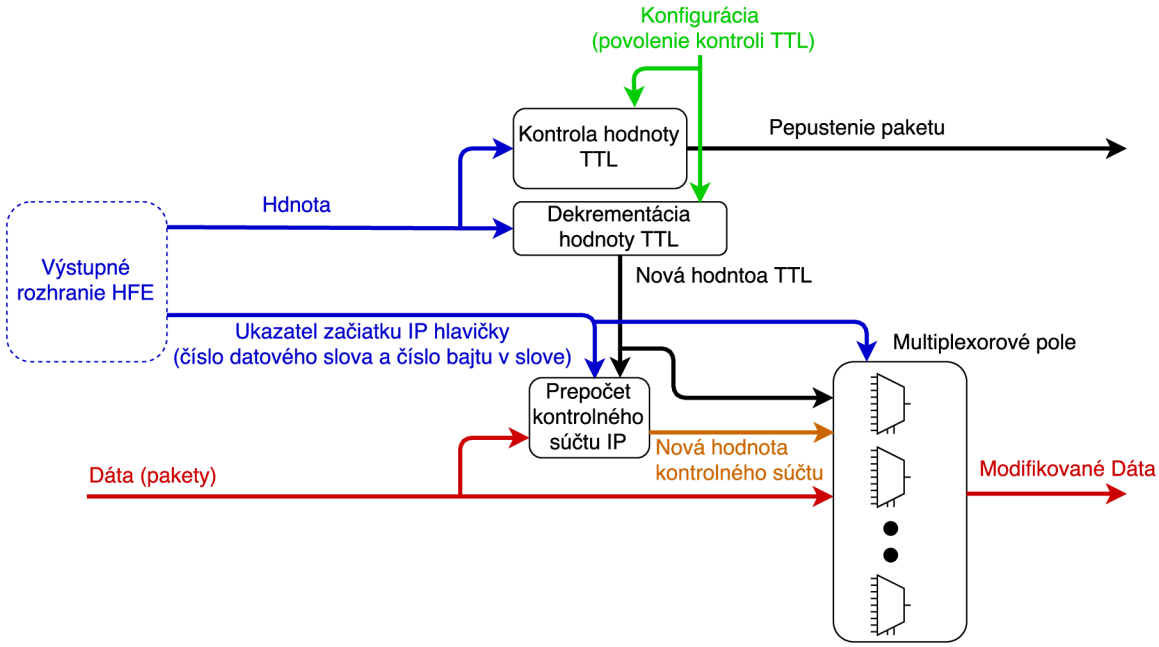


Obr. 4.3: Princíp fungovania modulu na výmenu VLAN

nenty HFE (modrá časť). Hodnota TTL je porovnávaná na hodnotu jedna. V prípade splnenia tejto podmienky je následne vygenerovaný príkaz na zahodenie spracovávaných dát (rámca, paketu) rozhodovacej jednotke (prostredná horná časť). Súčasne je hodnota TTL znížená o hodnotu jedna a poskytnutá multiplexorovému poli na realizáciu nahradenia (prostredná časť). S nahradením hodnoty TTL za novú je nutné taktiež prepočítať novú hodnotu kontrolného súčtu v IP hlavičke. K tomuto účelu je využívaný mechanizmus popísaný v [10] (prostredná, spodná časť). Nová hodnota položky TTL a kontrolného súčtu je spolu s adresou začiatku hlavičky protokolu IP poskytnutá multiplexorovému poli, ktoré zrealizuje dané nahradenie (pravá, spodná časť). Kontrolu hodnoty TTL je možné prostredníctvom konfiguračného rozhrania povoľovať alebo zakazovať (zelená časť).

MAC editor vykonáva nahradenie adries MAC za nové alebo uskutočňuje ich vzájomnú výmenu. Principiálne schéma modulu je znázornené na obrázku 4.5. Zdrojová a cieľová MAC adresa sú získavané z modulu HFE (modrá časť). Nové adresy MAC a akcia, ktorá má byť vykonaná, sú uložené v stavových registroch (horná časť). Tieto registre je možné konfigurovať prostredníctvom softvérového rozhrania (zelená časť). Na základe tejto konfigurácie sú nastavené hodnoty MAC adries (prostredná časť) pre príslušné multiplexorové polia, ktoré uskutočnia samotnú výmenu dát (spodná časť). MAC adresy sa nachádzajú vždy na začiatku prijímaných paketov, preto nie je potrebné k vykonaniu požadovanej modifikácie poznať ofset (adresu) začiatku týchto údajov.

Odbočovací filter je určený predovšetkým na špecifikovanie sieťového prenosu, ktorý chceme aby bol presmerovaný do operačnej pamäte počítača. Okrem samotného presmerovania je možné nastavovať, podľa potrebných požiadavkou, prípadné zahadzovanie sieťového prenosu, určiť aké údaje budú do operačnej pamäte poposielané (celé



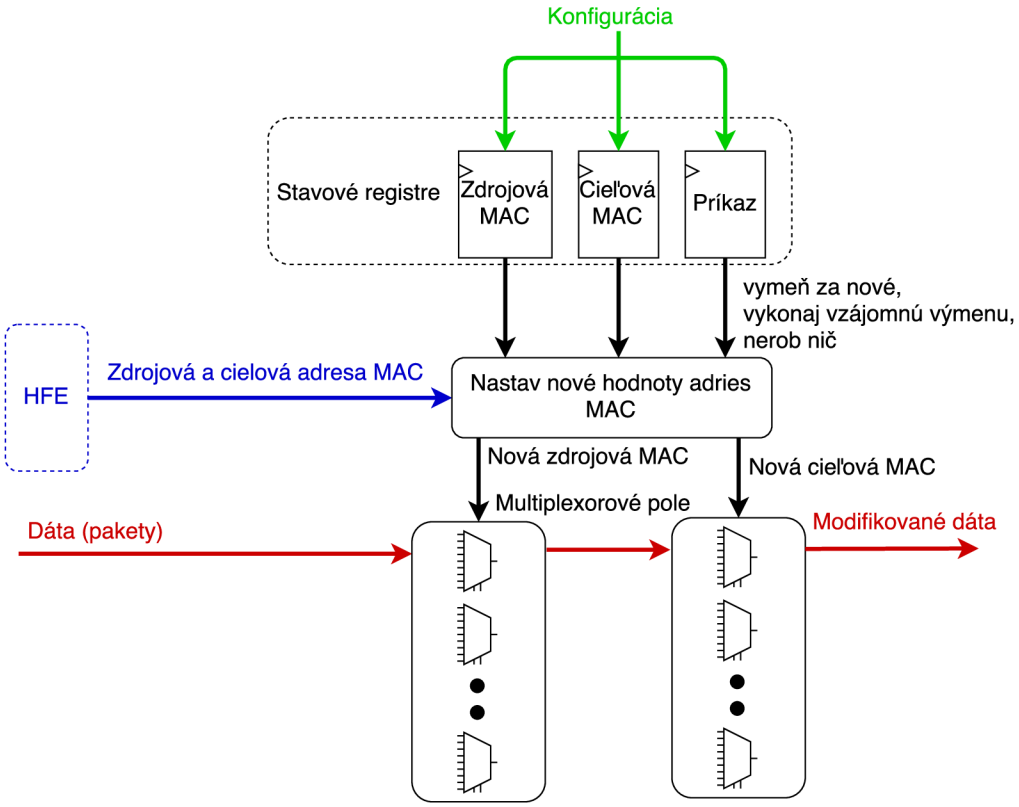
Obr. 4.4: Princíp fungovania modulu na kontrolu hodnoty TTL

pakety, hlavičky UH), či majú byť pakety preposielané do operačnej pamäte skracované a či má byť vykonaná nejaká editácia paketov prostredníctvom obecného editoru.

Skracovač vykonáva skracovanie preposielaných paketov do operačnej pamäte na maximálnu zvolenú dĺžku v prípade výberu preposielania celých sieťových paketov. K vyhodnoteniu a spracovaniu sieťového prenosu typicky nie sú potrebné zabalené prenášané dáta ale doležíte sú predovšetkým hlavičky jednotlivých protokolov v ktorých sú dáta zapuzdrené. Preto je možné pakety obsahujúce veľké množstvo prenášaných dát skrátiť na minimálnu požadovanú veľkosť potrebnú k úspešnému spracovaniu a tým tak šetriť dostupné zdroje ako je operačná pamäť a urýchliť tak softvérové spracovanie.

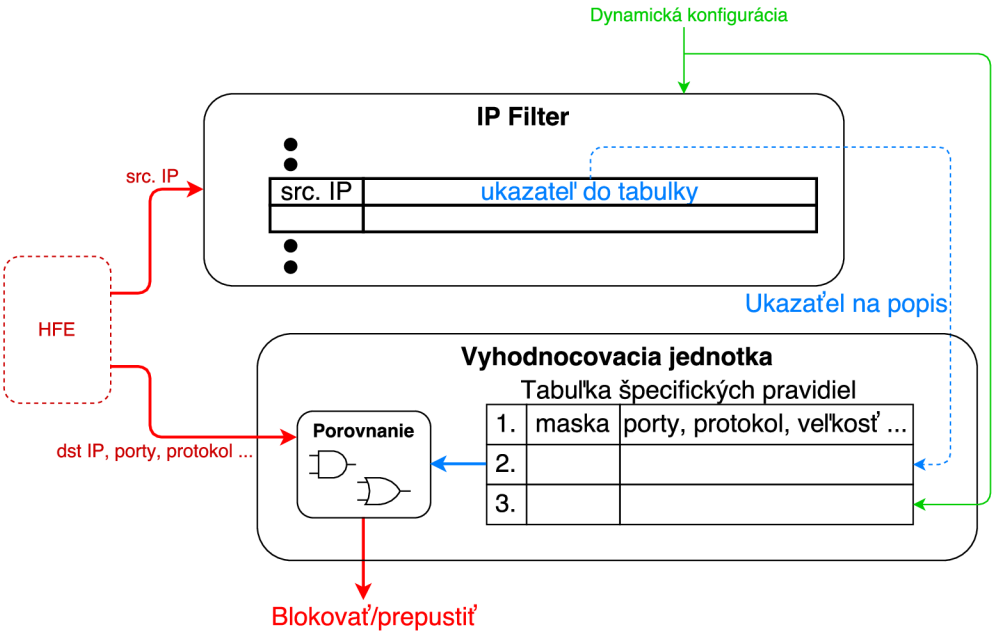
Generátor UH v prípade preposielania sieťového prenosu do operačnej pamäte počítača vytvára takzvané hlavičky UH. Tieto hlavičky sú vytvárané na základe informácii z modulu HFE. Obsahujú spracované informácie o jednotlivých paketoch ako sú hodnoty IP adries, číslo portov a protokolu, informácie o fragmentácii a ďalšie požadované informácie softvérovou časťou zariadenia v jednotnom formáte. Takto vygenerované hlavačky môžu byť preposielané do operačnej pamäte namiesto celých paketov. Týmto spôsobom je tak možné výrazne uľahčiť prácu s dátami o sieťovom prenose pri softvérovom spracovávaní, vzhľadom na to , že softvér už nemusí spracovávať a prípadne filtrovať prijaté pakety ale rovno môže využiť už spracované informácie potrebné k ďalšej činnosti. Formát a typ informácií v hlavičkách UH je možné ľubovoľne špecifikovať, prispôsobovať a upravovať na základe potrieb a rozširovania funkcionality softvérového spracovania. Formát hlavičky UH je popísaný v prílohe B. Preposielanie celých paketov alebo hlavičiek UH je volené na základe zhody v odbočovacom filtri a následne vybratej akcie.

Blokovancí filter. Podľa popisu amplifikačných útokov DoS a DDoS je typické generovanie nežiaduceho prenosu z veľkého počtu zdrojových IP adries. Súčasne tieto pakety



Obr. 4.5: Princíp fungovania modulu na výmenu MAC adries

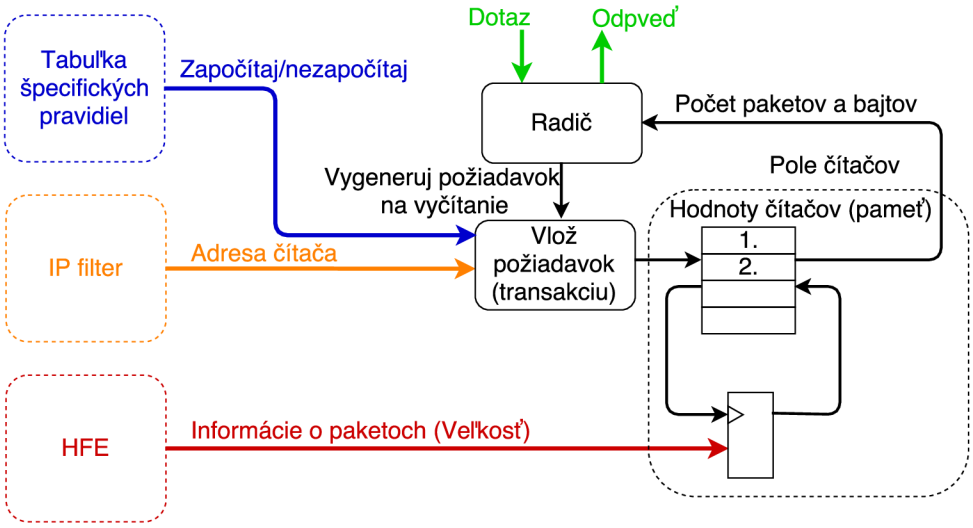
mávajú typicky spoločné vlastnosti a charakteristiky. Pri blokovaní takéhoto útoku tak nie je potrebné, pri každej blokovanej IP adrese, uchovávať aj celý komplikovaný popis sieťového prenosu (ktoré tvoria jednotlivé pravidlá) a vytvárať tak zbytočne duplikované informácie v pamäti. Pre dosiahnutie požadovaného efektu bude postačovať keď takto blokované IP adresy, ktoré majú rovnakú charakteristiku, sa budú odkazovať na jedno príslušné miesto v pamäti, ktoré obsahuje popis daného sieťového prenosu. Týmto spôsobom tak nie je nutné uchovávať veľký počet duplicitných informácii o popise sieťového prenosu u každej blokovanej IP adresy, čím sa zároveň šetria pamäťové prostriedky potrebné na realizáciu tohto procesu. Tento princíp uloženia jednotlivých pravidiel taktiež umožňuje nerovnomerne prerozdeliť pamäťové zdroje medzi pamäťou pre uloženie blokovaných adries IP a pamäťou pre uloženie popisov jednotlivých sieťových prenosov. Pre blokovanie útokov typu DDoS, s veľkým počtom zdrojových adries, tak typicky budeme potrebovať uložiť rádovo tisíce adries IP a pre uloženie popisov jednotlivých sieťových prenosov bude dostačovať kapacita rádovo stovky položiek. Na základe tohto spôsobu blokovania nežiaduce sieťového prenosu bude blokovanie realizované dvoma vyhodnocovacími jednotkami. Princíp blokovania je znázornený na obrázku 4.6. Prvá jednotka v poradí je IP filter (horná časť obrázku), ktorý na základe zdrojových IP adries a svojej konfigurácie zo softvérovej časti zariadenia, vyberie k príslušným IP adresám ukazateľ na popis konkrétneho sieťového prenosu (modrá šípka). Vyhodnotenie pokračuje do jednotky, ktorá na základe vybraného sieťového popisu uskutoční vyhodnotenie a rozhodne o prepustený alebo zablokovaní daného paketu (spodná časť obrázku). Táto jednotka je tvorená tabuľkou, ktorá obsahuje



Obr. 4.6: Princíp blokovacieho filtru

pravidlá, ktoré špecifikujú jednotlivé sieťové prenosy, ktoré sú následne porovnávane na zhodu z aktuálne spracovávaným paketom. Pravidlo, ktoré bude použité na zistenie zhody, prípadne nezhody, je vybrané na základe zhody zdrojovej adresy IP v IP filtri (horná časť). Pravidlo je tvorené rozsahom cieľových IP adries, portmi, protokolom, rozsahom veľkosti paketu, informácie o fragmentácii a hodnotami protokolu TCP. K týmto údajom je taktiež možné špecifikovať masku, prostredníctvom ktorej je možné špecifikovať, ktorá z týchto hodnôt má byť braná do úvahy pri vyhodnotený. Vyhodnocovanie realizujú komparátory s podporou maskovania. Vybraná podmienka je porovnávaná s údajmi získanými z modulu HFE (červená časť). O výsledku porovnania sú informované ďalšie moduly ako je rozhodovacia jednotka a štatistická jednotka. Obidve jednotky je možné dynamicky konfigurovať bez nutnosti zastavenia spracovávania sieťového prenosu prostredníctvom softvérového rozhrania.

Štatistická jednotka má za úlohu počítať počet bajtov a paketov blokovaných sieťových prenosov. Principiálne schéma modulu je znázornené na obrázku 4.7. Pripočítavanie hodnôt k jednotlivým čítačom je realizované vytváraním transakcií (prostredná časť) pre pole čítačov (pravá časť). Transakcia je tvorená adresou konkrétneho čítača získaného z IP filtru na základe zhody zdrojovej IP (oranžová časť) a akciou (započítaj, nezapočítaj), ktorá je určená zhodou v tabuľke špecifických pravidiel (modrá časť). Na základe tejto transakcie dôjde v poli čítačov k výberu konkrétnej položky z pamäte a vykonaniu danej operácie. Hodnoty, ktoré majú byť pripočítavané, sú získané z komponenty HFE (červená časť). Hodnoty jednotlivých čítačov je možné softvérovo vyčítať (zelená časť). K tomu slúži radič. Radič na základe požiadavku softvérovej časti zariadenia, vygeneruje špeciálnu transakciu pre vyčítanie, ktorú následne pošle poli čítačov na spracovanie. Tieto transakcie majú prednosť pred bežnými transakciami. Reakcia poľa čítačov na túto transakciu je získanie hodnoty z príslušného miesta v pamäti a následné predanie tejto hodnoty radiču. Radič následne realizuje prenos údajov do softvérovej časti zariadenia.



Obr. 4.7: Princíp fungovania modulu na počítanie štatistík zahodených paketov

Rozhodovacia jednotka určuje výslednú akciu, ktorá bude uskutočnená na základe vyhodnotenia a spracovania sieťového prenosu v obecnom filtri, blokovacom filtri a kontroly hodnôt TTL. Vyhodnotenie začína kontrolou výsledku porovnania hodnoty TTL. Ak tento výsledok oznamuje blokovanie prenášaného paketu, je paket zablokovaný bez ohľadu na výsledky vyhodnotenia ostatných jednotiek. Ak je paket prepustený, tak je ďalej vyhodnocovaný výsledok blokovacieho filtru. Ak je výsledok blokovacieho filtru zahodenie daného paketu, paket je zahodený. V prípade že paket prejde úspešne vyhodnotením na prepustenie cez kontrolu TTL aj blokovacieho filtru, tak dochádza k posednému vyhodnoteniu na základe výsledku odbočovacieho filtru. Odbočovací filter následne rozhodne o tom, či bude paket preposlaný na sieťové rozhranie, do operačnej pamäte alebo na oba tieto smery.

Kapitola 5

Implementácia

5.1 Implementácia firmvéru

Implementácia firmvéru bola realizovaná v jazyku VHDL. Prvý krok realizácie firmvéru spočíval v postupnej implementácii jednotlivých navrhnutých modulov popísaných v kapitole 4. Pri implementácii firmvéru boli využité základné generické komponenty poskytované platformou NetCOPE ako sú pamäte FIFO, aritmetické jednotky ALU, komponenty slúžiace na komunikáciu s jednotlivými rozhraniami platformy NetCOPE a ďalšie komponenty umožňujúce prepojovanie jednotlivých logických celkov (modulov).

Základné implementácie modulov HFE, obecného filtru a blokovacieho filtru, ktoré boli následne prispôsobené pre potreby navrhnutého firmvéru, sú prevzaté z už existujúcej hardvérovej akcelerácie Hanic [3]. Všetky implementované moduly majú jednotnú štruktúru, ktorá je zložená z entity *názov_modulu_ENT.vhd* (rozhranie pripojovaného modulu) a architektúry *názov_modulu_ARCH.vhd*, ktorá obsahuje implementáciu danej funkcionality. Jednotlivé entity väčšiny modulov obsahujú, okrem samotných signálov tvoriace komunikačné rozhrania, takzvané generické parametre. Prostredníctvom nich je možné špecifikovať parametre ako sú šírka dátovej zbernice, kapacita filtračných pamäťových jednotiek a možnosť zapínať prídavné registre pre zvýšenie taktovacej frekvencie. Väčšina modulov je kvôli čiteľnosti a prehľadnosti kódu realizovaná z ďalších (menších) implementovaných podkomponent.

Niektoré moduly ako sú kontrola TTL, obecný editor, MAC editor, výmena VLAN a štatistická jednotka, boli vzhľadom na svoju obecnú funkcionalitu vyčlenené a zaradené do platformy NetCOPE pre možnosť využitia ich funkcionality v ostatných projektoch. Tieto moduly taktiež obsahujú špeciálny parameter, prostredníctvom ktorého sa dá vypnúť ich funkcionalita. V praxi to znamená, že ak vo výslednom firmvéri nepotrebujem funkcionalitu niektorého z modulov (napríklad editáciu MAC adries), môžem vypnúť podporu daného modulu použitím jednoduchého parametru bez nutnosti zasahovania do zdrojového kódu. Týmto spôsobom sa tak šetria zdroje čipu FPGA, ktoré sa dajú následne použiť napríklad pre rozšírenie veľkosti pamätí pre filtračné moduly. Vďaka tejto možnosti taktiež nie je nutné udržovať rôzne verzie firmvéru (viacej zdrojových kódov), ktoré obsahujú rozličnú kombináciu zapojenia modulov.

Druhá časť implementácie spočívala vo vytvorení výslednej architektúry firmvéru zapojením jednotlivých implementovaných modulov spolu s platformou NetCOPE. V dobe písania bakalárskej práce je výsledný firmvér podporovaný a otestovaný na akceleračných kartách COMBO-100G, COMBO-100G2 a COMBO-100G2Q [2]. Podpora výsledného firm-

véru je pre jednotlivé karty zaisťovaná dostupnosťou a následným použitím príslušnej verzie platformy NetCOPE. Jadro implementovaného firmvéru ostáva vždy rovnaké.

5.2 Verifikácia firmvéru

Pri implementácii rozsiahlych a zložitých návrhov obvodov (firmvéru) sa taktiež zvyšuje výskyt chýb, ktoré pri tomto procese vznikajú. Následné odstraňovanie týchto chýb pri testovaní vytvorenej implementácie výrazne komplikuje a predlžuje samotný proces vývoja.

Cieľom je tak odhaliť a odstrániť tieto chyby čo najskôr za pomoci simulačných a verifikačných prostriedkov. Preto sa tento proces stáva neoddeliteľnou súčasťou implementácie. Použité prostriedky na odhaľovanie takýchto chýb sú popísané v nasledujúcom texte. Tento text vychádza z poznatkov uvedených v [19].

Simulácia je spôsob overovania správnosti fungovania vytvoreného systému pri ktorom je na vstupné rozhranie tohto systému postupne predávaná, dopredu vytvorená, množina vstupných hodnôt. Popis a priebeh simulácie je vytváraný v jazyku VHDL. Tento popis obsahuje aj samotnú množinu vstupných hodnôt, ktoré budú postupne predávané na vstupné rozhranie vytvoreného obvodu, ktorý chceme otestovať. Výsledkom tejto simulácie, po uskutočnení simulácie špecializovaným simulačným nástrojom (ako je Modelsim [14]), je časový diagram obsahujúci vybraté, vypočítané signálny testovaného obvodu. Návrhár následne realizuje vizuálnu kontrolou správnosti týchto signálov. Simulácia je teda proces, ktorým je návrhár schopný odhaliť chyby v návrhu a overiť správnu funkcionalitu daného obvodu. Simulácia však nezaručuje odhalenie všetkých chýb navrhnutého obvodu, pretože nedochádza k overovaniu všetkých možností a kombinácií vstupných parametrov obvodu.

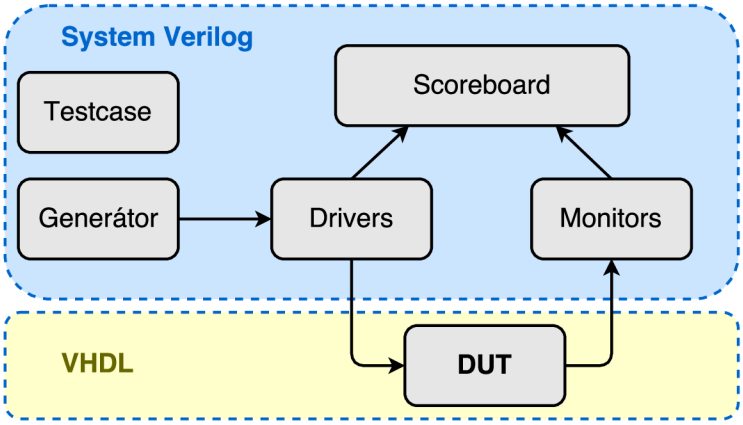
Funkčná verifikácia je metóda pri ktorej, na rozdiel od klasickej simulácie, dochádza k automatickému náhodnému generovaniu vstupných hodnôt a následne k ich automatickému vyhodnoteniu (kontrole). Pre funkčné verifikácie hardvérových obvodov sa využívajú jazyky rodiny HVL (Hardware Verification Language), ktoré poskytujú už dopredu vytvorené konštrukcie pre automatického generovania náhodných vstupných hodnôt a konštrukcie k automatickej kontrole výsledkov. Funkčnou verifikáciu je tak možné otestovať podstatne väčšie množstvo vstupných kombinácií ako je tomu u klasickej simulácie, avšak vytvorenie takejto verifikácie je z pohľadu prípravy a realizácie náročnejšie ako klasická simulácia. Funkčnou verifikáciou však nejde preukázať bezchybnosť testovaného obvodu ako je to napríklad u formálnej verifikácie. Vytvorenie formálnej verifikácie je však mnohonásobne zložitejšie a tiež sú potrebné poznatky z oblasti matematického dokazovania.

Navrhnutý firmvér bol najprv otestovaný prostredníctvom simulácie. Nad rámec zadania bakalárskej práce boli vytvorené aj verifikačné prostredia. Najprv boli vytvorené jednotlivé verifikačné prostredia pre samostatné moduly z ktorých sa skladá celé firmvérové jadro ako sú obecný editor, editor MAC adries, štatistická jednotka, jednotka na kontrolu TTL a jednotka na výmenu VLAN. Po úspešnom odstránení nájdených chýb u týchto jednotiek došlo k realizácii funkčnej verifikácie celého firmvérového jadra. Cieľom tejto verifikácie bolo odhaliť ďalšie chyby, ktoré mohli vzniknúť pri integrácii jednotlivých modulov do jedného celku. Vytvorené verifikácie však nepokrývajú všetky možné kombinácie vstupných parametrov a preto sa niektoré chyby prejavia až pri testovaní v laboratórnych podmienkach

na reálnom hardvéry. Takto odhalené chyby sa následne dajú spätne navodiť vo verifikačnom prostredí. K vytvoreniu funkčných verifikácií bol použitý jazyk SystemVerilog. SystemVerilog je objektovo orientovaný jazyk, ktorý obsahuje prvky jazyka verilog a je tak vhodný na popis hardvérových obvodov. Tento jazyk je zároveň jeden zo zástupcov jazykov HVL a je tak vhodný na tvorbu verifikačných prostredí. Tento jazyk je taktiež používaný na verifikačné prostredia pre komponenty vývojovej platformy NetCOPE.

Základnou jednotkou s ktorou verifikácia pracuje je takzvaná transakcia. Transakciou môžeme označiť sled požiadavkou alebo skupinu vstupných, navzájom závislých, signálov, ktoré sa nachádzajú na komunikačnom rozhraní. Tieto transakcie sú následne automaticky porovnávane na zhodu so získanými výsledkami (hodnotami) z testovacej komponenty. Pri vytváraní vstupných transakcií pre testovanú jednotku je potrebné, aby boli dodržané komunikačné protokoly na vstupnom rozhraní komponenty. K tomuto účelu je možné použiť metódu zvanú constrained-random generovanie. Pri tejto metóde je možné špecifikovať obmedzujúce podmienky, ktoré musí každá vygenerovaná transakcia spĺňať. Týmto podmienkami je tak možné docieliť aj vygenerovania konkrétnych situácií, ktoré chceme otestovať.

Navrhnuté verifikačné prostredie zodpovedná štandardnému modelu, ktoré je znázornené na obrázku 5.1.



Obr. 5.1: Schéma verifikačného prostredia

Pri tvorbe prostredia boli použité dostupné prostriedky platformy NetCOPE určené pre tvorbu verifikačného prostredia. Časti obrázku ohraničené modrou farbou boli implementované v jazyku SystemVerilog a časti ohraničené žltou farbou boli implementované v jazyku VHDL. Ďalšie časti navrhnutého verifikačného prostredia sú popísané v nasledujúcom texte.

Testcase je jednotka, ktorá obsahuje konfiguračné parametre jednotlivých častí verifikačného prostredia (generátory, driveri a monitory), ktorými je možné nastavovať hodnoty ako je počet generovaných transakcií, rozsah veľkosti generovaných sieťových dát (paketov) alebo pravdepodobnosť pre vkladanie čakacích stavov na vstupné a výstupné rozhrania testovanej komponenty. Touto konfiguráciou je tak možné vytvoriť viacero testovacích scenárov so zameraním na konkrétne situácie alebo na konkrétnu funkcionalitu testovanej komponenty. Pre verifikačné prostredia jednotlivých modulov, z kotrých je zložené výsledné firmvérové jadro, bola vytvorená iba jedna testovacia sada zameraná na testovanie správnej funkcionality daného modulu. Pri verifikačnom prostredí celého firmvérového jadra boli výtvorené dve testovacie sady. Prvá sada bola

zameraná na otestovanie správnej funkcionality jednotlivých modulov a ich vzájomnej komunikácie. Druhá sada bola zameraná konkrétne na filtračné jednotky (obecný a blokovací filter) a kontrolu správneho počítania zahodených paketov v štatistickej jednotke.

Generátor je jednotka, ktorá zaisťuje generovanie náhodných vstupných transakcií. Tieto transakcie sú následne ďalšími jednotkami prevádzané a posielané na rozhranie testovacej komponenty. Jedná sa o generovanie sieťových paketov a doprovodných informácií ako je čas príchodu paketu. Taktiež sú náhodne generované ďalšie radiace signáli pre jednotlivé testované jednotky pre určenie akcie (činnosti), ktorá má byť nad danými sieťovými paketmi vykonaná. Pre všetky verifikačné prostredia (jednotlivých modulov aj celého jadra) boli vytvárané rovnaké transakcie v podobe sieťových dát. Jednotlivé transakcie sa líšia iba v konkrétnych riadiacich signáloch.

Drivers sú objekty, ktoré prijímajú vygenerované transakcie od generátoru a prevádzajú ich na konkrétne hodnoty vstupných signálov pre komunikačné rozhranie pripojenej jednotky. Tieto objekty taktiež umožňujú vkladanie čakacích stavov podľa zvolenej konfigurácie. Transakcie sú taktiež posielané do scoreboardu k ďalšiemu vyhodnocovaniu. Vzhľadom na rozdielnosť signálov, na jednotlivých komunikačných rozhraniach všetkých verifikovaných komponent, bolo nutné tento objekt implementovať pre každé prostredie zvlášť.

Monitors sú objekty, ktoré prijímajú výstupné signáli z testovacej komponenty a následne ich prevádzajú na transakcie, ktoré sú ďalej posielané do scoreboardu k ďalšiemu spracovaniu. Tieto objekty taktiež umožňujú vkladanie čakacích stavov na výstupnom rozhraní komponenty podľa zvolenej konfigurácie. Podobne ako pri objektoch Drivers aj tu jednotlivé verifikačné prostredia modulov obsahujú vlastnú implementáciu realizujúcu prevod.

DUT (Design Under Test) je obálka popísaná v jazyku VHDL, ktorá obsahuje zapojenú instanciu testovanej komponenty (jednotlivé moduly alebo jadro firmvéru). Súčasťou obálky sú jednotlivé parametre, ktoré definujú jednotlivé signáli komunikačného rozhrania (datová šírka, smer komunikácie) a umožňuje tak ich prevod do SystemVerilogu.

Scoreboard je jednotka, ktorá prijíma vstupné transakcie z driverov a taktiež výstupné transakcie, ktoré boli vytvorené v monitoroch. Táto časť verifikačného prostredia je najzložitejšia, pretože pre získanie očakávaného výsledku musí obsahovať kompletný model testovanej komponenty. Tento model už je popísaný v abstrakčnejšom jazyku (SystemVerilog) a jeho vytvorenie je tým pádom podstatne jednoduchšie ako vytvorenie popisu samotnej komponenty (vo VHDL). Všetky verifikačné prostredia jednotlivých komponent obsahujú príslušný model danej komponenty. V scorebordu je následne do tohto modelu poslaná vtupná transakcia, obdobne ako do testovanej komponenty. Výstupná transakcia tohto modelu je následne porovnávaná s výstupnou transakciou z komponenty. V prípade nezhody je vygenerovaná chyba a celá verifikácia je následkom tejto chyby úplne prerušená. Táto vzniknutá chyba oznamuje že očakávaný výsledok transakcie poslanej na vstupné rozhranie komponenty sa nezhoduje s výsledkom transakcie z vytvoreného modelu. V prípade verifikačného prostredia celého firmvérového jadra bol model zložený zo všetkých existujúcich modelov z verifikácií jednotlivých komponent. K tomuto modelu boli ďalej doimplementované

ďalšie časti funkcionalít jadra firmvéru, pre ktoré nebola realizovaná samostatná funkcionálna verifikácia.

5.3 Implementácia softvérového rozhrania

Softvérové rozhranie je určené na konfiguráciu a získavanie informácii o aktuálnom stave firmvéru. Rozhranie je poskytované formou knižnice jazyka C a taktiež prostredníctvom softvérových nástrojov, pomocou ktorých je možné konfigurovať firmvér z príkazovej riadky. K realizácii samotného prenosu dát medzi softvérovým rozhraním a firmvérom je využité konfiguračné a riadiace rozhranie poskytované platformou NetCOPE označované taktiež ako MI32. Prostredníctvom rozhrania MI32 je možné prenášať 32 bitové slová medzi pamäťou softvéru a stavovými registrami (prípadne pamäťami) firmvéru.

Rozhranie je rozdelené na jednotlivé moduly (obdobne ako firmvérová časť), ktoré sú určené na konfiguráciu jednotlivých modulov firmvérovej časti ako sú – obecný filter, blokovací filter, štatistická jednotka, výmena VLAN, kontrola TTL, obecný editor, editor MAC adries a generátor hlavičiek UH. Moduly softvérového rozhrania sú tvorené troma hlavnými typmi funkcií. Príklad rozhrania takéhoto modulu je znázornený v prílohe C.

Prvú skupinu tvori inicializačná funkcia, ktorá alokuje potrebné zdroje a vyčíta potrebné údaje z príslušného firmvérového modulu ako sú dostupné pamäťové prostriedky, adresový priestor, verzia modulu a prípadne ďalšie údaje nevyhnuté k následnej konfigurácií daného modulu (príklad funkcie: *dcpro_load_MODUL(...)*). Druhú skupinu tvori uvoľňovacia funkcia, ktorej úlohou je uvoľniť alokované zdroje vytvorené inicializačnou funkciou a korektne ukončiť komunikáciu s príslušným firmvérovým modulom (*dcpro_free_MODUL(...)*). Tretiu skupinu tvoria funkcie, ktoré slúžia na samotnú konfiguráciu jednotlivých firmvérových modulov (*dcpro_MODUL_AKCIA(...)*).

Kapitola 6

Dosiahnuté výsledky

Kapitola popisuje výsledky dosiahnuté pri realizácii navrhnutého firmvéru v jednotlivých častiach vývoja až po dosiahnuté výsledky získané laboratórnym testovaním na reálnom hardvéry. Prvá časť je venovaná výsledkom dosiahnutým syntézou implementovaného firmvérového jadra, ktorý prevádza popis navrhnutého firmvéru z VHDL na prvky cieľovej technológie. Ďalej sa kapitola venuje dosiahnutým výsledkom z procesu implementácie celého firmvéru (firmvérové jadro + platforma NetCOPE), kedy dochádza k mapovaniu prvkov obvodu na konkrétnu cieľovú architektúru (čip FPGA). Nakoniec sa kapitola venuje výsledkom merania dátovej priepustnosti celého systému, ktoré boli získané počas laboratórneho testovania vytvoreného firmvéru na reálnom hardvéry v laboratórnych podmienkach.

6.1 Výsledky syntezy jadra firmvéru

Syntéza je proces, pri ktorom dochádza k prevodu popísaného obvodu v jazyku VHDL na ekvivalentné vyjadrenie obvodu prostredníctvom obecných logických prvkov. Výsledkom syntézy je výsledný obvod zložený z logických prvkov cieľovej technológie nazývaný taktiež ako netlist. Výsledkom je taktiež odhad spotreby využitých zdrojov na cieľový čip FPGA. Syntéza bola realizovaná softvérovým nástrojom Vivado vo verzii 2016.2 pre konkrétny čip FPGA Virtex-7 H580T nachádzajúci sa na akceleračných kartách rodiny COMBO-100G. Výsledky odhadu spotrebovaných zdrojov sú znázornené v tabuľke 6.1.

Tabuľka obsahuje, pre jednotlivé moduly tvoriace jadro firmvéru, spotrebu základných elementov ako sú LUT, registre a spotrebu blokových pamätí BRAM. Na konci tabuľky je následne znázornená spotreba zdrojov celého firmvérového jadra. V zátvorkách je ďalej udávaná percentuálna spotreba jednotlivých zdrojov na čipe FPGA. Vzhľadom na uskutočňované optimalizácie softvérovým nástrojom Vivado, pri syntéze celého jadra firmvéru, sa súčet spotreby zdrojov jednotlivých modulov nemusí rovnať výslednej spotrebe celého jadra firmvéru.

Najviac zdrojov zaberá jednotka HFE. Použitá konfigurácia podporuje spracovanie až dvoch záhlaví protokolu IPv6 a taktiež podporuje spracovanie až štyroch záhlaví ethernetu VLAN a MPLS. Ďalšou najväčšou komponentnou s ohľadom na spotrebované zdroje je obecný filter. V danej konfigurácií rozhoduje filter o následnom spracovaní prijatých paketov na základe filtračných pravidiel, ktoré môžu byť tvorené kombináciou zdrojových a cieľových IP prefixov, portov, protokolov a číslom vstupného sieťového rozhrania. Ďalšou jednotkou, ktorá zaberá značné množstvo zdrojov je obecný editor. Vzhľadom na to že jednotka umožňuje vloženie alebo editáciu ľubovoľného štvôr bajtového bloku v pakete, je značná časť

Tabuľka 6.1: Využité zdroje čipu FPGA Virtex-7 H580T

Moduly	**Počet LUT**	**Počet registrov**	**Počet BRAM**
HFE	9 417 (2.6%)	16 442 (2.3%)	8 (0.9%)
Obecný filter	4 074 (1.1%)	5 708 (0.8%)	7 (0.7%)
IP Filter	2 169 (0.6%)	1 127 (0.2%)	164 (17.4%)
Tabulka pravidiel	1 153 (0.3%)	426 (0.1%)	8 (0.9%)
Štatistická jednotka	1 206 (0.3%)	630 (0.1%)	96 (10.2%)
Premapovanie VLAN	541 (0.2%)	1 776 (0.2%)	2 (0.2%)
Kontrola TTL	906 (0.3%)	1 870 (0.3%)	0 (0.0%)
Obecný editor	2 076 (0.6%)	4 735 (0.7%)	1 (0.1%)
Editor MAC adries	397 (0.1%)	162 (0.1%)	0 (0.0%)
Generátor UH	33 (0.1%)	36 (0.1%)	0 (0.0%)
Skracovač	21 (0.1%)	18 (0.1%)	0 (0.0%)
Celé jadro firmvéru	21308 (5.9%)	35251 (4.9%)	289 (30.7%)

týchto zdrojov použitá na realizáciu riadiaceho mechanizmu. Tento mechanizmus vyhodnocuje a určuje konkrétne miesto v dátových slovách, kde má byť realizovaná modifikácia a taktiež rieši rôzne kolízie a situácie, najme pri rýchlom prijímaní krátkych paketov. S ohľadom na najväčšie množstvo spotrebovaných blokových pamätí BRAM je najväčšia jednotka IP filter. V použitej variante je filter schopný vykonávať rozhodovanie na základe až 32 768 zdrojových IP adries s podporov oboch protokolov IPv4 aj IPv6. Druhov jednotkou, ktorá zaberá značné množstvo pamätí BRAM je štatistická jednotka. V danej konfigurácii obsahuje jednotka 32 768 štatistických čítačov a je tak schopná počítať a uchovávať štatistické údaje pre každú IP adresu z IP filtru aj pri maximálnej konfigurácii. Z tabuľky môžeme taktiež vidieť, že zabrané zdroje elementov LUT a registrov celého jadra firmvéru zaberajú iba mále množstvo prostriedkov dostupných na čipe. V prípade pamätí BRAM zaberá jadro približne jednu tretinu.

6.2 Výsledky implementácie firmvéru

Implementácia je proces, pri ktorom dochádza k mapovaniu logických celkov výsledného obvodu z procesu syntézy na jednotlivé dostupné prvky cieľovej architektúry, k ich presnému rozmiestneniu a taktiež k ich vhodnému prepojeniu na čipe FPGA. Do tohto procesu taktiež vstupujú podmienky špecifikované užívateľom ako sú perióda hodinového signálu alebo pridelenie portov z popisu entity na konkrétne pini čipu FPGA. Výsledkom tohto procesu sú informácie o spotrebovaných zdrojoch čipu FPGA, informácia či sa podarilo dosiahnuť požadovanú taktovaciu frekvenciu a bitstream, prostredníctvom ktorého je následne konfigurovaný čip FPGA.

K procesu implementácie bol taktiež použitý softvérový nástroj Vivado vo verzii 2016.2. Implementácia bola uskutočnená vrátane použitej platformy NetCOPE. Výsledky pre podporované akceleračné karty, v dobe písania bakalárskej práce, sú znázornené v tabuľke 6.2.

Väčšinu zdrojov zaberá práve použitá platforma NetCOPE. V prípade prvkov LUT a registrov pripadá samotnému jadru firmvéru iba zlomok z celkových zabratých zdrojov. v prípade pamätí BRAM je podiel zabraných zdrojov opačný a jadru firmvéru pripadá približne 2/3 zo všetkých využitých pamätí BRAM. To je spôsobené práve modulmi ako

Tabuľka 6.2: Využité zdroje čipu FPGA pre jednotilvé karty

Karta	LUT	Registre	BRAM	Frekvencia
COMBO-100G1	107 168 (29.5%)	106 840 (14.7%)	451 (44.2%)	> 200 MHz
COMBO-100G2	81 150 (22.4%)	93 242 (12.9%)	292 (31.1%)	> 200 MHz
COMBO-100G2Q	108 234 (39.8%)	110 721 (15.3%)	309 (32.9%)	> 200 MHz

je blokovací filter a štatistická jednotka, ktoré tvoria hlavnú časť jadra firmvéru a majú veľké pamäťové nároky. U akceleračných kariet COMBO-100G2 a COMBO-100G2Q boli pamäťové nároky, pre tieto dva moduly, zmenšené na polovicu z dôvodu dosiahnutia požadovanej frekvencie 200 MHz.

V poslednom stĺpci je uvedená dosiahnutá taktovacia frekvencia výsledku implementácie. U všetkých variant sa podarilo dosiahnuť frekvenciu 200 MHz. Pri použitej 512 bitovej zbernice je k dosiahnutiu plnej priepustnosti, aj pri najkratších paketoch, nutná práve minimálna taktovacia frekvencia 200 MHz. Frekvencia 200 MHz nie je konečnou maximálnou hodnotou, na ktorej by mohol vytvorený firmvér pracovať, avšak pre potreby dátovej priepustnosti 100 Gbps je táto frekvencia dostatočná a preto nebola realizovaná ani implementácia s požiadavkou na vyššiu taktovaciu frekvenciu.

Jedným z najťažších úloh pri návrhu obvodu pre čip FPGA je práve dosiahnutie splnenia požiadavku pre taktovaciu frekvenciu. Riešením tohto problému je takzvané zmenšovanie kritickej cesty. Kritická cesta je časť obvodu, kde dochádza k najdlhšiemu oneskorovaniu signálnu kvôli veľkému množstvu, za sobov idúcich, prvkov kombinačnej logiky. Jednou z techník odstraňovania týchto kritických ciest je vkladanie registrov do tejto cesty, čím dôjde k jej rozdeleniu a tým jej skráteniu. Vkladanie registrov do obvodu spôsobuje zvyšovanie latencie, čo v niektorých častiach obvodu, kde je závislosť na tejto latencii (kvôli využívaniu spätnej väzby), zapríčiní nefunkčnosť a je tak potrebné danú časť obvodu vhodne upraviť prípadne prepracovať.

6.3 Funkčné testovanie

V laboratórnom prostrední boli prostredníctvom hardvérového testeru Spirent TestCenter realizované testy na overenie správnej funkcionality jednotlivých častí vytvoreného firmvéru. Realizované boli tieto konkrétne testy:

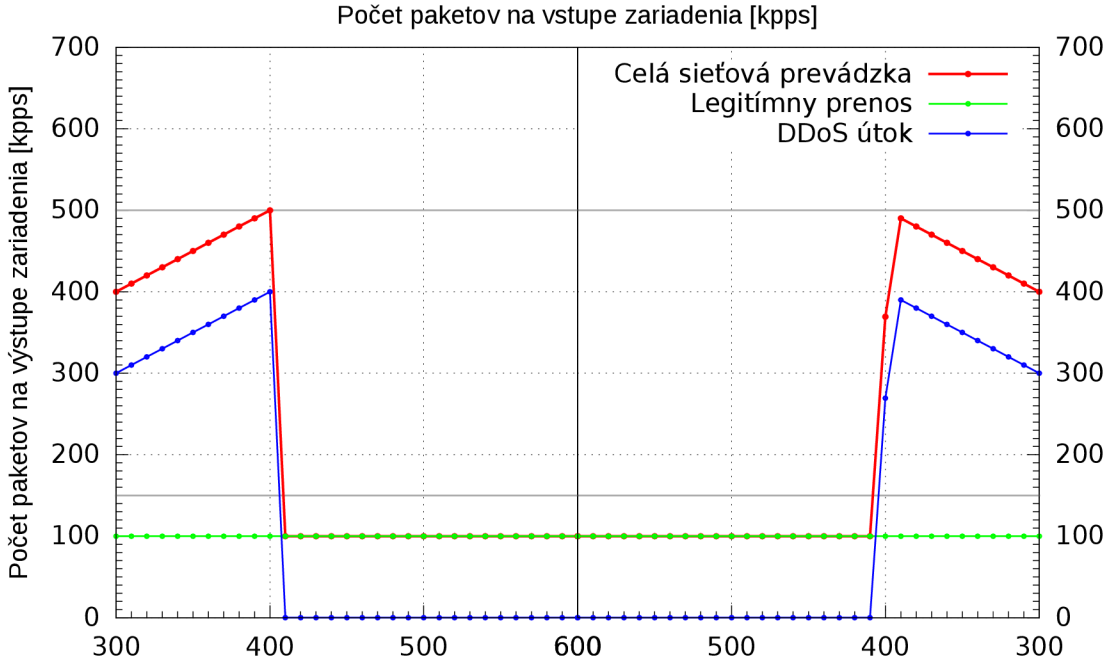
Test MAC editoru. Pri tomto teste bolo overované správne fungovanie nahrádzania cieľových a zdrojových MAC adries v paketoch. Taktiež bola testovaná funkcionalita vzájomnej výmeny týchto adries. Pri tomto teste boli na pripojené zariadenie (akceleračnú kartu obsahujúci vytvorený firmvér) postupne posielané náhodne vygenerované sieťové pakety hardvérovým testerom. Následne boli na tomto hardvérovom testeri monitorované prijímané (preposielané) pakety z pripojenej akceleračnej karty. U týchto paketov bola následne uskutočnená kontrola MAC adries, či testovaný firmvér správne vykonal úpravu paketov podľa zvolenej konfigurácie. Konkrétne boli testované konfigurácie – keď nemala byť vykonaná žiadna modifikácia MAC, keď mala byť vykonaná výmena zdrojovej MAC, výmena cieľovej MAC, výmena oboch MAC a vzájomná výmena cieľovej a zdrojovej MAC adresy.

Test kontroly TTL bol realizovaný obdobne, prostredníctvom generovania a monitorovania paketov hardvérovým testerom, ako test pre MAC editor. Pre tento modul boli

realizované dva konkrétne testy. Pri prvom teste boli posielané pakety s nastavenou hodnotou TTL na 1. Pri tomto teste bolo overované že firmvér uskutočnil zahodenie týchto paketov. U druhého testu boli posielané pakety s náhodnou, nenulovou TTL hodnotou. Pri tomto teste bolo kontrolované znižovanie hodnoty TTL o jedna a zároveň to, že pakety neboli zahadzované ale opätovne preposielané.

Test výmeny VLAN overoval správne fungovanie nahrádzania položiek VLAN ID podľa zvolenej konfigurácie. Prvý realizovaný test prebiehal pri východiskovej konfigurácii testovaného firmvéru, kedy nemalo dochádzať k žiadnej modifikácii položiek VLAN ID. Pre druhý test bola vygenerovaná konfiguračná tabuľka, ktorá obsahovala informácie o nových hodnotách VLAN ID, ktoré majú nahradiť (vymeniť) príslušné hodnoty VLAN ID v paketoch. Táto konfiguračná tabuľka bola nakonfigurovaná náhodnými vygenerovanými hodnotami. Následne boli cez firmvér posielané náhodné pakety. Pri opätovnom prijatí týchto paketov z firmvéru bola uskutočnená kontrola správnej výmeny hodnoty VLAN ID za novú podľa konfiguračnej tabuľky.

Po integrácii firmvérovej časti a softvérovej časti (ktorá nebola predmetom tejto práce) do výsledného zariadenia bol vykonaný test celého zariadenia. Cieľom tohto testu bolo overiť správne fungovanie detekcie nežiaduceho sieťového prenosu a tým aj správne fungovanie celého zariadenia (firmvéru aj softvéru, ktorý využíva vytvorené konfiguračné rozhranie). Pri tomto teste boli posielané dva typy sieťovej prevádzky. Prvý typ reprezentoval sieťovú komunikáciu legitímnych užívateľov a simuloval prevádzku pri reálnej sieťovej komunikácii. Druhý typ sieťovej komunikácie predstavoval nežiaduci sieťový prenos, útok typu (D)DoS, ktorý chceme zablokovať. Výsledky tohto testu sú znázornené na grafe 6.1.



Obr. 6.1: Výsledky testovania detekcie (D)DoS útoku

Graf znázorňuje pomer medzi počtom generovaných paketov na zariadenie a počtom primaných paketov zo zariadenia. Na osi x je znázornená sieťová prevádzka v počte paketov,

ktorá je posielaná na vstupné sieťové rozhranie zariadenia. Na osi y je znázornená sieťová prevádzka prijímaných paketov z výstupného sieťového rozhrania zariadenia. Červená krivka znázorňuje pomer medzi všetkými prijímanými a odosielanými sieťovými paketmi.

Modrá krivka znázorňuje pomer medzi počtom paketov generovaného útoku (D)DoS. Pri tomto teste je zariadenie nakonfigurované tak, aby sieťový prenos začal blokovať po prekročení limitu 500 kpps na vstupnom rozhraní. Z grafu môžeme vidieť, že pri prekročený tohto limitu začalo zariadenie blokovať konkrétny sieťový prenos, ktorý prispieval najväčším počtom sieťových dát. Zablokovaný sieťový prenos bol práve generovaný (D)DoS. Následne môžeme pozorovať, že keď generovaný útok (D)DoS začal zoslabovať a stanovený limit 500 kpps už nebol prekračovaný, došlo k opätovnému prepusteniu celého sieťového prenosu.

Zelená krivka znázorňuje pomer počtu paketov legitímneho sieťového prenosu. Na grafe môžeme pozorovať, že pri úspešnom zablokovaní útoku (D)DoS nebol tento legitímny prenos nijak obmedzený.

Týmto testom tak bolo úspešne overené správne fungovanie vytvoreného zariadenia. Tým sa úspešne otestovala funkcionalita firmvérových jednotiek ako sú obecný filter, blokovací filter a štatistická jednotka.

6.4 Dátová priepustnosť

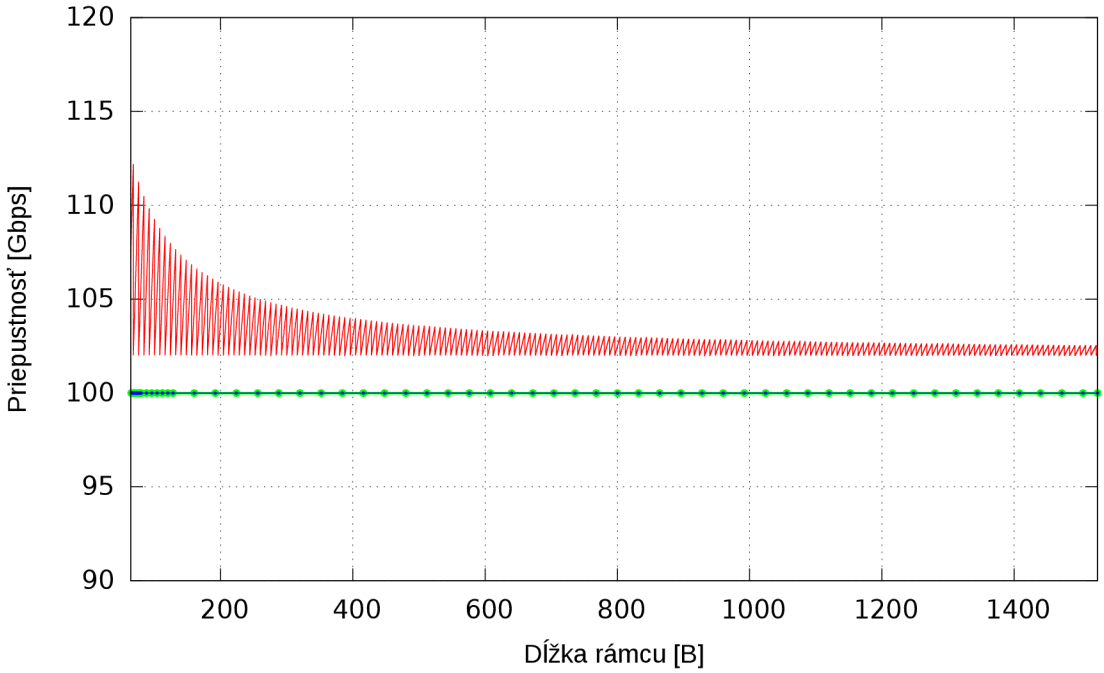
Meranie priepustnosti vytvoreného firmvéru bolo uskutočnené v laboratórnych podmienkach. Testovanie prebehlo na hardvérovej akceleračnej karte COMBO-100G1 vo verzii jedného, 100G, ethernetového portu. Generovanie testovacích sieťových dát bolo realizované hardvérovým generátorom Spirent TestCenter. Pre účely tohto testovania boli ethernetové rámce generované so záhlavím protokolov IPv4 aj IPv6 s náhodnými hodnotami a s náhodnými vygenerovanými dátami bez transportného protokolu. Jadro firmvéru bolo nastavené tak, aby prijímané sieťové pakety prechádzali spracovaním cez všetky vyhodnocovacie a riadiace moduly, z ktorých je jadro firmvéru tvorené.

Meranie prebiehalo takým spôsobom že na vstupné ethernetové rozhranie boli posielané ethernetové rámce v takom množstve, aby bola zaručená plná saturácia pripojenej 100G linky. Následne boli v pravidelných intervaloch vyčítané čítače rámcov zo vstupných sieťových blokov firmvéru, ktoré sú poskytované v rámci použitej platformy NetCOPE. Výsledná dátová priepustnosť bola následne odvodená na základe pomeru medzi počtom všetkých prijatých a spracovaných rámcov firmvérom, a počtom všetkých prichádzajúcich rámcov na pripojené rozhranie. Meranie bolo vykonané pre rôzne dĺžky ethernetových rámcov v rozsahu 64–1526 bajtov. Výsledky merania sú znázornené na grafe 6.2. Graf znázorňuje dátovú priepustnosť v Gbps v závislosti na dĺžke prijímaných ethernetových rámcoch.

Na grafe sú znázornené výsledky pre čisto hardvérové spracovanie, kedy nedochádza k žiadnemu prenosu dát do operačnej pamäte, a taktiež výsledky so zapnutým prenosom do operačnej pamäte prostredníctvom DMA. Pri zapnutom prenose dát do operačnej pamäte sa na výslednej priepustnosti možu prejaviť obmedzenia spojené s dátovou priepustnosťou na systémovej zbernici PCI-Express.

Červená krivka znázorňuje maximálnu priepustnosť hardvérového spracovania pri dosiahnutej taktovacej frekvencii 200 MHz. Táto priepustnosť je odvodená na základe znalosti šírky dátovej zbernice a spôsobu zarovnávania dátových slov. Oscilácia krivky je spôsobená práve realizovaným zarovnávaním, kedy dochádza k zarovnávaniu dátových slov na násobky 8 bajtov.

Zelená krivka znázorňuje nameranú priepustnosť hardvérového spracovania kedy nie je realizovaný prenos do operačnej pamäte prostredníctvom radiču DMA (Dáta sú pred radičom



Obr. 6.2: Priepustnosť systému v závislosti na dĺžke rámcu (v Gbps)

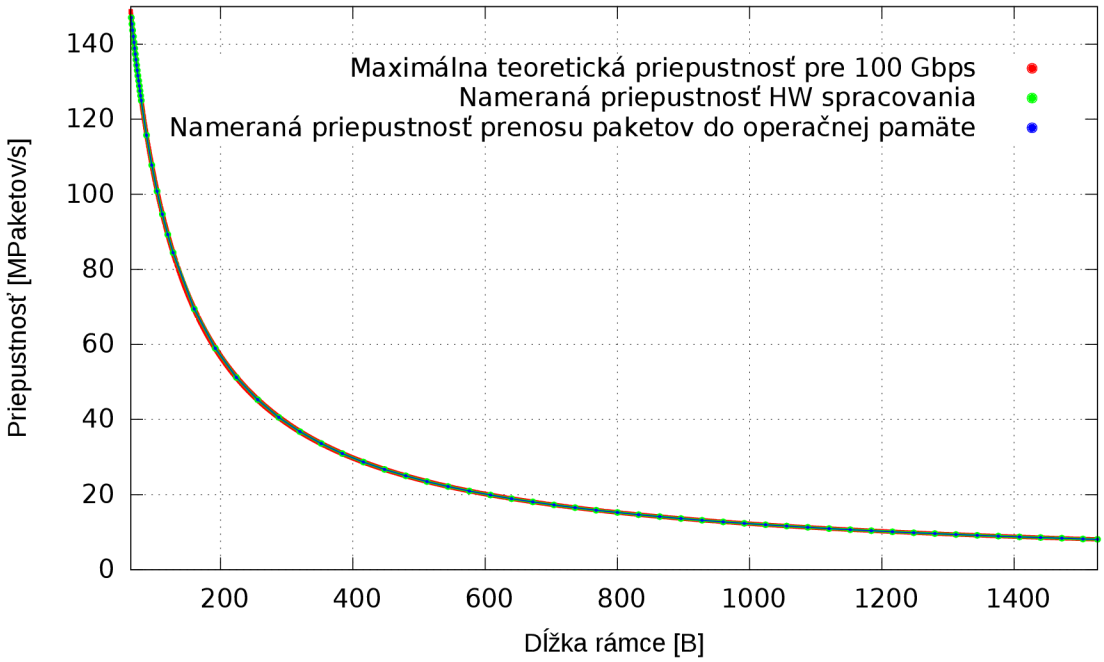
DMA zahadzované). Pri tomto teste prechádzajú prijaté dáta spracovaným vo všetkých moduloch jadra firmvéru ako sú kontrola hodnoty TTL, výmena VLAN, obecný editor, MAC editor, generátor UH, skracovač paketov, obecný filter, blokovací filter a štatistická jednotka. Z grafu je vidieť, že bola dosiahnutá plná dátová priepustnosť 100 Gbps aj pri najkratších paketoch (64 B), kedy dochádza k najväčšiemu vyťažovaniu dátovej zbernice.

Modrá krivka reprezentuje nameranú dátovú priepustnosť pri hardvérovom spracovávaní sieťového prenosu so zapnutým prenosom sieťových dát do operačnej pamäte prostredníctvom DMA. Pri tomto teste prechádzajú dáta taktiež spracovaním vo všetkých moduloch firmvérového jadra a zároveň spracovaním v radiči DMA, ktorý následne realizuje prenos sieťových dát do operačnej pamäte. Pri tomto teste bol spustení softvérový nástroj sze2multiread, ktorý na testovacom stroji následne vyčítal všetky prenesené dáta do operačnej pamäte prostredníctvom 8 dostupných kanálov DMA (8 paralelných vlákien). Do operačnej pamäte boli prenášané celé sieťové (neskracované) pakety. Z grafu je vidieť, že aj pri tomto teste bola dosiahnutá plná priepustnosť 100 Gbps. Na túto priepustnosť nemal vplyv ani negatívny dopad použitej zbernice PCI-Express.

S grafu je možné vidieť, že hardvérové spracovanie by bolo pravdepodobne schopné spracovávať dáta aj pri vyššej prenosovej rýchlosti ako 100 Gbps, ale vzhľadom na spôsob realizácie merania nebolo možné túto skutočnosť potvrdiť.

Na grafe 6.3 je znázornený ďalší pohľad na dosiahnuté výsledky z testovania dátovej priepustnosti. Tento graf znázorňuje dátovú priepustnosť v počne prenesených paketov za jednu sekundu pri ethernetových rámcoch v rozsahu 64-1526 bajtov.

Červená krivka znázorňuje priepustnosť 100 Gbps prepočítanú na počet prenesených rámcov danej dĺžky za jednu sekundu. Význam ostatných kriviek ostáva rovnaký ako u predošlého grafu. Rozdielny je iba spôsob vyjadrenia dátovej priepustnosti. Na grafe môžeme pozorovať, že priebeh všetkých kriviek sa prekrýva. To znamená, že nameraná dosiahnutá



Obr. 6.3: Priepustnosť systému v závislosti na dĺžke rámcu (v miliónoch paketoch/s)

priepustnosť dosahuje u oboch uskutočnených meraní (čisto hardvérové spracovanie a hardvérové spracovanie s prenosom do operačnej pamäte) vypočítaného teoretického maxima znázorneného červenou krivkou.

Kapitola 7

Záver

Cieľom bakalárskej práce bolo vytvoriť firmvér pre hardvérovo akcelerované zariadenie pre ochranu pred (D)DoS útokmi s využitím technológie FPGA. Pri návrhu a následnej realizácii firmvéru bol kladený dôraz na dosiahnutie požadovanej dátovej priepustnosti 100 Gbps a taktiež na splnenie všetkých špecifikácií, ktoré vyplynuli na základe analýzy danej problematiky. Súčasťou návrhu firmvéru bola tiež realizácia softvérového a konfiguračného rozhrania, ktoré umožňuje konfiguráciu vytvoreného firmvéru.

V rámci riešenia bakalárskej práce som sa najprv dôkladne oboznámil s architektúrou a základnými princípmi fungovania dnešných počítačových sietí. Následne som sa zameral na problematiku bezpečnosti počítačových sietí so zameraním na amplifikačné útoky typu (D)DoS. Ďalej som sa podrobne oboznámil s technológiami používanými pre hardvérové spracovávanie dát na vysokorýchlostných sieťach využívajúce technológiu programovateľných hradiel FPGA. S tohto dôvodu som sa ďalej venoval technológii hardvérových akceleračných kariet rodiny COMBO využívajúce čipy FPGA Xilinx Virtex-7, na návrhový jazyk VHDL používaný pre návrhy hardvérových obvodov a na prostriedky poskytované vývojovou platformou NetCOPE s možnosťou ich využitia.

Po získaní vedomostí som realizoval podrobný implementačný návrh firmvéru pre hardvérovo akcelerované zariadenie pre ochranu pred (D)DoS útokmi. Následne som na základe realizovaného návrhu uskutočnil implementáciu firmvérového jadra v jazyku VHDL. Pri implementácii firmvéru som využil dostupné prostriedky poskytované vývojovou platformou NetCOPE. Hotový firmvér som následne podrobil testovaniu prostredníctvom základných simulácií. Nad rámec bakalárskej práce som realizoval podrobné funkčné verifikácie jednotlivých samostatných modulov firmvéru a následne verifikáciu celého firmvérového jadra. Súčasťou realizácie funkčných verifikácií bol ich samotný návrh, implementácia jednotlivých verifikačných prostredí a modelov.

Následne som integroval vytvorené firmvérového jadro s vývojovou platformou NetCOPE a preložil výsledný firmvér pre konkrétny čip FPGA Virtex-7 H580T nachádzajúci sa na akceleračných kartách rodiny COMBO. Ďalej som realizoval ručné optimalizácie navrhnutej (vytvorenej) architektúry výsledného firmvéru, čím sa mi podarilo dosiahnuť vysokej taktovacej frekvencie 200 MHz pre požadovanú dátovú priepustnosť celého systému na rýchlosti 100 Gbps.

Správnu funkcionalitu vytvoreného firmvéru som následne overil prostredníctvom laboratórneho testovania na akceleračnej karte COMBO-100G1 s využitím hardvérového testovacieho zariadenia Spirent TestCenter, ktorý umožňuje generovanie a monitorovanie sieťového prenosu pri rýchlostiach 100 Gbps. V tomto laboratórnom prostredí som taktiež otestoval a overil reálnu priepustnosť celého systému na rýchlosti 100 Gbps.

V dobe písania bakalárskej práce bolo celé hardvérovo akcelerované zariadenie pre ochranu pred (D)DoS útokmi nasadené na hlavnej linke akademickej siete CESNET, ktorá pracuje na prenosových rýchlostiach 100 Gbps. Toto zariadenie majú k dispozícii sieťový administrátori, ktorý na zariadení realizujú vlastné testovanie funkcionality a experimenty. K tomuto účelu je cez zariadenie presmerovaná reálna sieťová prevádzka Libereckej univerzity. Ďalším pokračovaním tejto práce bude rozširovanie vlastností už exitujúceho zariadenia o ďalšie možnosti pre detekciu nových útokov.

Výsledky tejto práce a výsledky z nasadenia celého zariadenia boli prezentované formou článku na študentskej konferencii inovácií, technológií a vied v IT s názvom Excel@FIT 2017 [9]. V rámci tejto konferencie článok získal ocenenie odborným panelom, ktoré udeľovala komisia akademických pracovníkov. Článok taktiež získal ocenenie komerčným partnerom, ktoré bolo udelené zástupcom firmy Edhouse [4] za praktický prínos navrhnutého riešenia v oblasti bezpečnosti a ochrany počítačových sietí.

Literatúra

[1] CESNET: CESNET, zájmové sdružení právnických osob, [online]. [cit. 11-03-2017].
URL http://www.cesnet.cz/

[2] CESNET: Cards. Liberouter / CESNET TMC group, [online]. [cit. 05-04-2017].
URL https://www.liberouter.org/technologies/cards/

[3] CESNET: Hanic. Liberouter / CESNET TMC group. Hanic, [online]. [cit. 18-03-2017].
URL https://www.liberouter.org/?page_id=821

[4] Edhouse: Edhouse s.r.o. [online]. [cit. 05-05-2017].
URL http://www.edhouse.cz/cz/

[5] Imperva Incapsula: NTP Amplification | DDoS Attack Glossary. DDoS Knowledge Center – Latest Threats and Mitigation Methods, [online]. [cit. 08-03-2017].
URL
https://www.incapsula.com/ddos/attack-glossary/ntp-amplification.html

[6] Juniper Networks: Understanding SYN Flood Attacks. JUNOS Software Security Configuration Guide, [online]. [cit. 08-03-2017].
URL http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-syn-flood-attacks.html#id-3412834128

[7] Juniper Networks: Understanding UDP Flood Attacks. JUNOS Software Security Configuration Guide, [online]. [cit. 08-03-2017].
URL http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-udp-flood-attacks.html#id-60351

[8] Kolouch, J.: Kybernetické útoky. CESNET, zájmové sdružení právnických osob, [online]. [cit. 08-03-2017].
URL https://csirt.cesnet.cz/_media/cs/documents/kyberneticke_utoky.pdf

[9] Kuka, M.: Hardvérovo akcelerované zariadenie pre ochranu pred (D)DoS útokmi. Excel@FIT: Studentská Konference Inovací, Technologií a Vědy v IT, [online]. [cit. 05-05-2017].
URL http://excel.fit.vutbr.cz/submissions/2017/032/32.pdf

[10] Mallory, T.; Kullberg, A.: Incremental updating of the Internet checksum. RFC 1141, RFC Editor, January 1990.

[11] Martinek, T.; Kosek, M.: *NetCOPE: Platform for Rapid Development of Network Applications. Proceedings of the 11th IEEE Workshop on Design and Diagnostics of*

Electronic Circuits and Systems. DDECS 2008. Bratislava, SK, 2008, ISBN 978-1-4244-2277-7, 6 s.

[12] Matoušek, P.: *Síťové služby a jejich architektura*. Publishing house of Brno University of Technology VUTIUM, 2014, ISBN 978-80-214-3766-1, 396 s.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10567

[13] McClure, S.: *Hacking bez tajemství. 3. aktualiz. vyd.* Computer Press, 2003, ISBN 80-7226-948-8, 612 s.

[14] Mentor a Siemens Business: ModelSim. [online]. [cit. 14-05-2017].
URL
https://www.mentor.com/products/fpga/verification-simulation/modelsim/

[15] Mirkovic, J.: *Internet denial of service : attack and defense mechanisms*. Upper Saddle River, N.J: Prentice Hall Professional Technical Reference, 2005, ISBN 978-0131475731, 400 s.

[16] Prince, M.: Deep Inside a DNS Amplification DDoS Attack. CloudfFlare – The web performance & security company, [online]. [cit. 08-03-2017].
URL https:
//blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/

[17] Sekanina, L.; Vašíček, Z.; Růžička, R.; aj.: *Evoluční hardware: Od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. Edice Gerstner, Academia, 2009, ISBN 978-80-200-1729-1, 328 s.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=9123

[18] Shinder, D. L.: *Počítačové sítě: nepostradatelná příručka k pochopení síťové teorie, implementace a vnitřních funkcí*. Praha: SoftPress, 2003, ISBN 80-86497-55-0, 752 s.

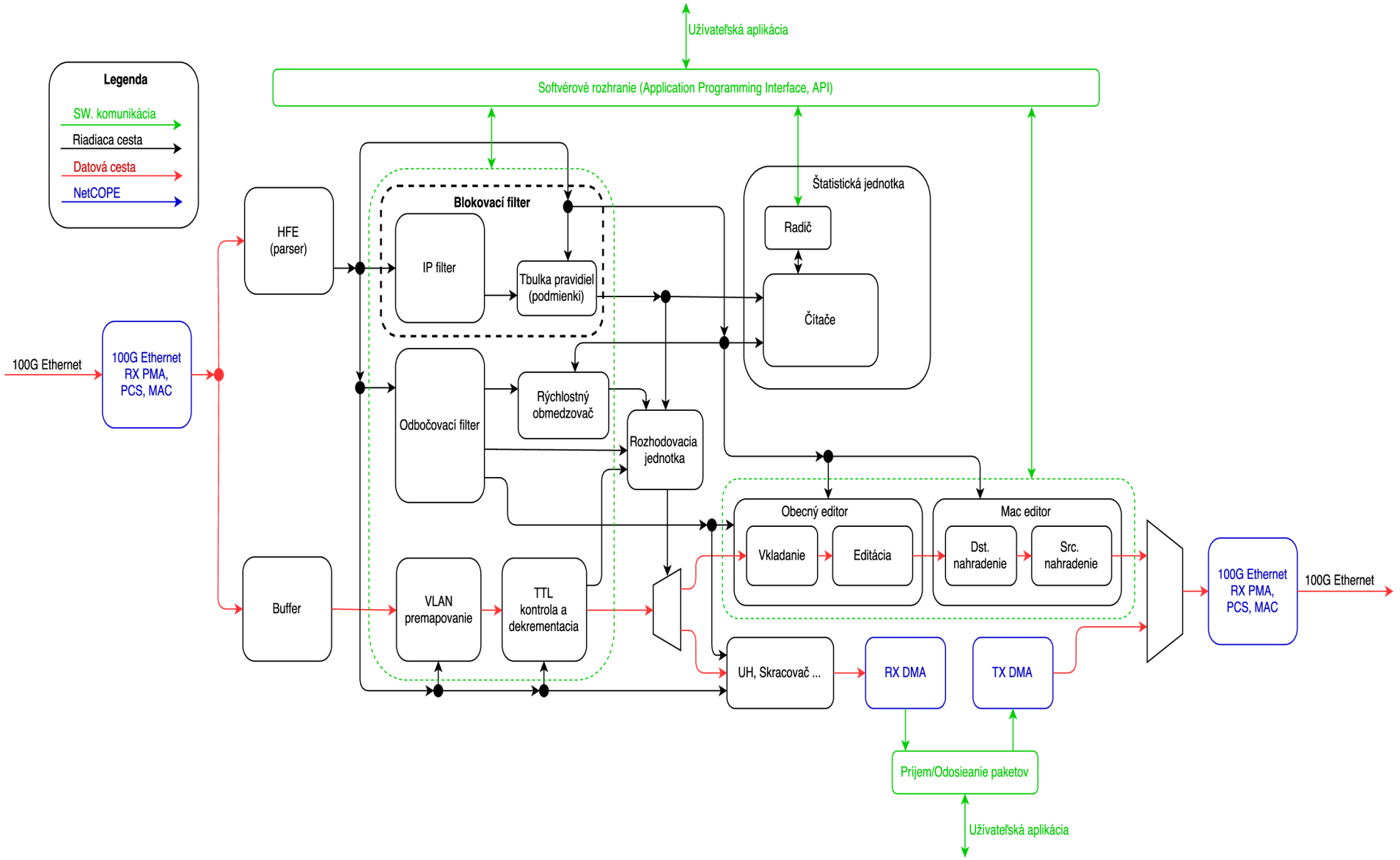
[19] Šimková, M.: *Hardwarově akcelerovaná funkční verifikace*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2011.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=9900

[20] Štěpán, F.; Puš, V.; Matoušek, J.; aj.: CESNET Technical Report 7/2013 Designing a Card for 100 Gb/s Network Monitoring. CESNET, zájmové sdružení právnických osob, [online]. [cit. 18-03-2017].
URL https://www.cesnet.cz/wp-content/uploads/2014/02/card.pdf

Prílohy

Príloha A

Struktúra firmvéru



Obr. A.1: Struktúra firmvéru

Príloha B

Formát hlavičky UH

Bajty	Velkosť	Pole	Popis
00-15	16B	SourceIP	Zdrojová IP adresa v network byte orderu. (IPv6 alebo IPv4, nuly pokial paket nie je IP). (IPv4 0-7B 0x0, 8-11B ipv4, 12-15B 0xffffffff)
16-31	16B	DestinationIP	Cielová IP adresa v network byte orderu (IPv6 alebo IPv4, nuly pokial paket nie je IP). (IPv4 0-7B 0x0, 8-11B ipv4, 12-15B 0xffffffff)
32-33	2B	SourcePort	Číslo zdrojového TCP, UDP, SCTP portu (nuly v prípade fragmentovaného paketu).
34-35	2B	DestinationPort	Číslo cielového TCP, UDP, SCTP portu (nuly v prípade fragmentovaného paketu).
36-37	2B	Octets	Pôvodná velkosť paketu od začiatku IP hlavičky do konca bez CRC. Pokial paket nebol IP (pozná sa podla dalších položiek hlavičky) tak velkosť Ethernetového rámca od MAC do konca bez CRC.
38-38	1B	Protocol	L4 protokol (6 = TCP, 17 = UDP, ...)

39-39	1B	IPVer / Flags / Fragmentace	Bity 0-1: Typ následujúceho pola UH: 0 = následujúce pole je nevyužité, 1 = následujúce pole obsahuje MPLS, 2 = následujúce pole obsahuje jednu VLAN TCI, 3 = následujúce pole obsahuje dve VLAN TCI.
			Bit 2: Typ posledného pola UH: 0 = posledné pole UH je nevyužité, 1 = posledné pole UH obsahuje MPLS
			Bit 3: Verzia protokolu IP (0 = IPv4, 1 = IPv6)
			Bit 4: Validita IP (0 = neIP paket, 1 = IP paket)
			Bit 5: Validita fragmentácie (0 = následujúcích bitov nie je validných, 1 = následujúci bit je validný)
			Bity 6-7: Fragmentácia (význam bitov: prvý/posledný fragment; tzn. 11 = nefragmentované, 10 = prvý fragment, 00 = prostredný fragment, 01 = posledný fragment)
40-43	4B	Inner MPLS / VLAN	Najvnútornešia (posledná) MPLS label alebo spodné 2B najvnútornešej (poslednej) VLAN TCI a horné 2B najvrchnejšej (prvej) VLAN TCI (alebo nuly). ak je iba jedna VLAN, budú vyplnené horné 2B. ak sú 2 a viac MPLS a nejaká VLAN, má VLAN v tomto poli prednosť.
44-47	1B	Tcp flags	tcp príznaky FIN (0), SYN (1), RST (2), PSH (3), ACK (4), URG (5), ECE (6), CWR (7)

Príloha C

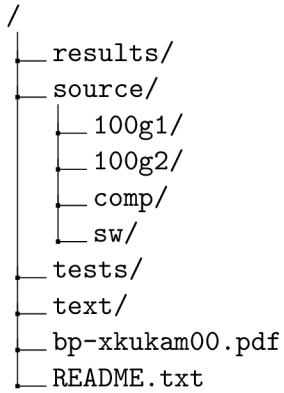
Softvérové rozhrnanie štatistického modulu

```
//#####################################################################
//##### STATS MODULE
//#####################################################################

/**
 * stats type
 */
typedef struct dcpro_stats_struct {
    cs_device_t *dev;
    dcprostats_component_t comp;
} dcpro_stats_t;

/**
 * result from stats unit
 */
struct stats_data {
   unsigned long bytes;
   unsigned long packets;
};

/**
 * stats unit configuration
 */
struct dcpro_stats_conf {
   uint32_t addr_width;
   uint32_t bytes_width;
   uint32_t packets_width;
};

/**
* Load stats unit from device and allocate memory.
* @param stats stats unit data.
* @param Path to device file to use. (set "" to use default patch)
* @param Path to design.xml file. (set "" to use default patch)
* @return SUCCESS|FAIL_ATTACH_NOEX|FAIL_DESIGN_USR
   |FAIL_DESIGN_DEF|FAIL_FIND|FAIL_SUPPORT;
*/
int
dcpro_load_stats(const char *device_file,
```

```
   const char *design_xml_file, dcpro_stats_t *stats);

/**
 * Free all allocated memory
 * @param stats stats unit data.
 */
void
dcpro_free_stats(dcpro_stats_t *stats);

/**
* Read statistic data from counter in stats unit
* @param address counters address in stats unit
* @param control_addr 1 -> enable, 0 -> disable to control
*        maximal value of address
* @param result result from stats unit
* @param stats stats unit data.
* @return SUCCESS|FAIL_ADRESS_OVERFLOW;
*/
int
dcpro_stats_read(unsigned address, int control_addr,
   struct stats_data *result, dcpro_stats_t *stats);

/**
 * Return number of counters from stats unit
 * @param stats stats unit data.
 * @return number of counters.
 */
unsigned
dcpro_stats_cnts_num(dcpro_stats_t *stats);

/**
 * Get configuration from stats unit
 * @param strules strules data
 */
void
dcpro_stats_get_conf(struct dcpro_stats_conf *conf,
   dcpro_stats_t *stats);
```

Príloha D

Obsah CD



Adresár `results/` obsahuje vytvorený bitstream pre FPGA pre akceleračnú kartu COMBO-100G1. Súčasťou sú taktiež súbory s nameranými hodnotami pri hardvérovom testovaní.

Adresár `source/` obsahuje zdrojové kódy implementovaných VHDL komponent, vrátane vytvorených simulačných a verifikačných prostredí a potrebných častí pre opätovné prevedenie syntézy. Jednotlivé adresáre `100g1/ 100g2/ comp/ sw/` obsahujú zdrojové súbory firmvéru, softvérového rozhrania a softvérových nástrojov.

Adresár `text/` obsahuje zdrojové súbory textu bakalárskej práce pre možnosť úpravy a opätovného prekladu systémom LATEX, vrátane zdrojových súborov použitých obrázkov.

Adresár `tests/` obsahuje konfiguračné a zdrojové súbory nástrojov a skriptov vytvorených alebo použitých pre účely testovania a automatického merania.

Súbor `bp-xkukam00.pdf` obsahuje text bakalárskej práce vo formáte PDF.

Súbor `README.txt` obsahuje informácie o adresárovej štruktúre a obsahu priloženého CD. Ďalej obsahuje pokyny pre umiestnenie implementovaných komponent do adresárovej štruktúry platformy NetCOPE a inštrukcie pre využitie prekladového systému platformy pre spustenie procesu syntézy.