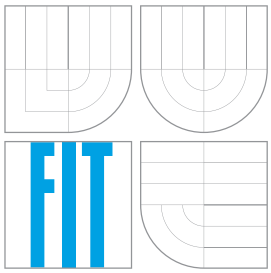
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

APLIKAČNĚ SPECIFICKÝ PROCESOR PRO STAVOVÉ ZPRACOVÁNÍ SÍŤOVÝCH DAT

APPLICATION SPECIFIC PROCESSOR FOR STATEFUL NETWORK TRAFFIC PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN KUČERA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ KEKELY

BRNO 2014

Zadání bakalářské práce/16018/2013/xkucer73

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2013/2014

Zadání bakalářské práce

Řešitel: **Kučera Jan**
Obor: Informační technologie
Téma: **Aplikačně specifický procesor pro stavové zpracování síťových dat**
Application Specific Processor for Stateful Network Traffic Processing
Kategorie: Návrh číslicových systémů

Pokyny:

1. Seznamte se s technologií FPGA Xilinx Virtex-7 a konceptem softwarově definovaného měření.
2. Navrhněte procesor, který bude schopen provádět vysokorychlostní stavové měření toků na síti.
3. Implementujte navržený procesor v jazyce VHDL.
4. Ověřte funkčnost procesoru v simulaci a ve fyzické implementaci.
5. Zjistěte výkonové parametry procesoru (propustnost, latence), identifikujte slabá místa v návrhu a navrhněte zlepšení.

Literatura:

- http://www.fit.vutbr.cz/study/DP/all.php?id=15490
- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

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: **Kekely Lukáš, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2013
Datum odevzdání: 21. května 2014

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

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Bakalářská práce se zabývá návrhem a implementací aplikačně specifického procesoru pro vysokorychlostní stavové měření síťových toků. Hlavním cílem je vytvoření komplexního systému pro akceleraci různých aplikací z oblasti monitorování a bezpečnosti počítačových sítí. Aplikačně specifický procesor tvoří hardwarovou část systému implementovanou v FPGA na akcelerační síťové kartě. Návrh procesoru je proveden s ohledem na nasazení na sítích o rychlostech 100 Gb/s a je založen na unikátní kombinaci rychlosti hardwarového zpracování a flexibility softwarového řízení vycházející z konceptu softwarově definovaného monitorování (SDM). Vytvořený systém prošel funkční verifikací a v rámci hardwarového testování byla ověřena jeho reálná propustnost a další výkonové parametry.

Abstract

This bachelor's thesis deals with the design and implementation of an application-specific processor for high-speed network traffic processing. The main goal is to provide complex system for hardware acceleration of various network security and monitoring applications. The application-specific processor (hardware part of the system) is implemented on an FPGA card and has been designed with respect to be used in 100 Gbps networks. The design is based on the unique combination of high-speed hardware processing and flexible software control using a new concept called Software Defined Monitoring (SDM). The performance and throughput of the proposed system has been verified and measured.

Klíčová slova

FPGA, VHDL, vysokorychlostní sítě, 100 Gb/s, hardwarová akcelerace, aplikačně specifický procesor, zpracování síťových dat, SDM, softwarově definované monitorování.

Keywords

FPGA, VHDL, High-speed Networks, 100 Gbps, Hardware Acceleration, Application-specific Processor, Network Traffic Processing, SDM, Software Defined Monitoring.

Citace

Jan Kučera: Aplikačně specifický procesor pro stavové zpracování síťových dat, bakalářská práce, Brno, FIT VUT v Brně, 2014

Aplikačně specifický procesor pro stavové zpracování síťových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Lukáše Kekelyho a že jsem uvedl všechny literární prameny a publikace, z nichž jsem čerpal.

.....
Jan Kučera
21. května 2014

Poděkování

V první řadě patří velké poděkování vedoucímu práce Ing. Lukáši Kekelymu za odborné vedení, veškerou další pomoc, ochotu a trpělivost při konzultacích. Následně bych chtěl poděkovat také pracovníkům z Oddělení nástrojů pro monitoring a konfiguraci ze sdružení CESNET, obzvláště pak Ing. Viktoru Pušovi, Ph.D. za cenné rady. Poděkování patří také mým blízkým za pomoc při finální korekci textu a mým rodičům za jejich podporu po celou dobu mého bakalářského studia.

Obsah

Kapitola 1

Úvod

Rozvoj počítačových sítí je úzce spjat s příchodem osobních počítačů umožňujících uživatelům sdílet soubory, vyměňovat si zprávy nebo přistupovat ke sdíleným zdrojům, jako jsou síťové tiskárny nebo souborové servery. Dnešní paketově orientované a vysokorychlostní propojovací sítě dosáhly značného rozšíření. Pracují na vysokých rychlostech a poskytují dostatečnou šířku přenosového pásma pro aplikace, jakými jsou přenos multimediálních dat a videokonference ve vysoké kvalitě. Současný trend vývoje počítačových sítí vede k přechodu na stále vyšší rychlosti komunikace a nasazování nových technologií 40 Gb/s a 100 Gb/s na páteřních linkách a v datových centrech. Vývoj komunikačních sítí se však musí promítnout i do oblasti bezpečnosti a monitorování těchto sítí. Aktuálně používané postupy totiž nedosahují požadované výkonnosti pro tuto novou generaci vysokorychlostních sítí. Řešení uvedeného problému se přitom neobejde bez podpory hardwarové akcelerace časově kritických operací.

Hardwarová akcelerace nabízí vysoký výkon a rychlost zpracování, avšak výsledný systém je špatně rozšiřitelný. Naopak běžné obecné výpočetní procesory poskytují dostatečnou flexibilitu, ale nepokrývají výkonnostní požadavky aplikace. Východiskem z této situace je vhodné rozložení činností mezi hardwarovou a softwarovou vrstvu a jejich úzké provázání, které poskytne dostatečnou výkonnost i flexibilitu. Na uvedenou problematiku se zaměřuje koncept softwarově definovaného monitorování (Software Defined Monitoring, SDM) využitelný pro flexibilní monitorování sítí, které je založené na sledování síťových toků. Základem SDM je aplikování myšlenky spojení inteligentního a propracovaného softwarového řadiče s relativně jednoduchým avšak dostatečně výkonným hardwarový zařízením v oblasti bezpečnosti a monitorování vysokorychlostních sítí.

Bakalářská práce se zabývá návrhem architektury a implementací aplikačně specifického procesoru (firmwaru) hardwarově akcelerovaného systému pro vysokorychlostní stavové měření síťových toků založeného na principech konceptu SDM. Při návrh firmwaru je kladen důraz především na dosažení požadované propustnosti systému s ohledem na možnost jeho budoucího nasazení na síti při rychlostech 100 Gb/s. Implementace je provedena pro FPGA akcelerační síťovou kartu s využitím vývojové platformy NetCOPE. Kromě samotné realizace aplikačně specifického procesoru a provedení základních simulací je nad rámec zadání vytvořeno verifikační prostředí a model systému za účelem spolehlivějšího ověření funkčnosti provedené implementace. Funkčnost procesoru je ověřena také ve fyzické implementaci s využitím hardwarového testeru umožňujícího generování a analýzu síťového provozu. Nad rámec zadání práce je také navržena a vytvořena jednotka vstupního síťového bloku rozšiřující funkčnost platformy NetCOPE o dosud chybějící podporu příjmu rámců na rychlosti 100 Gb/s. Dále je provedeno měření reálné propustnosti a dalších výkonových

parametrů celého systému. Jeho nasazení v praxi na reálné produkční síti je dalším logickým pokračováním této bakalářské práce.

Práce je rozdělena do několika kapitol shrnujících jednotlivé etapy řešení zadaného problému. Kapitola 2 poskytuje teoretický úvod k praktické části práce a zaměřuje se na popis principů fungování počítačových sítí, zabývá se oblastí jejich správy, monitorováním a rozborem technologie FPGA s možností jejího využití k hardwarové akceleraci zpracování síťových dat. Dále je věnován prostor konceptu SDM pro seznámení se s myšlenkami a principy jeho použití. V kapitole 3 je představena architektura a implementační návrh firmwaru akceleračního systému. Následující kapitola 4 nabízí podstatné informace týkající se výsledné implementace aplikačně specifického procesoru. Popis verifikačního prostředí navrženého nad rámec zadání práce za účelem spolehlivého ověření funkčnosti systému je předmětem kapitoly 5. Výsledky měření propustnosti a dalších výkonových parametrů spolu s diskuzí nad slabými místy návrhu a možnými vylepšeními jsou k dispozici v kapitole 6. Závěrečná kapitola 7 slouží jako shrnutí celkově dosažených výsledků práce a nabízí pohled na její další možné budoucí směřování.

Kapitola 2

Teoretický rozbor

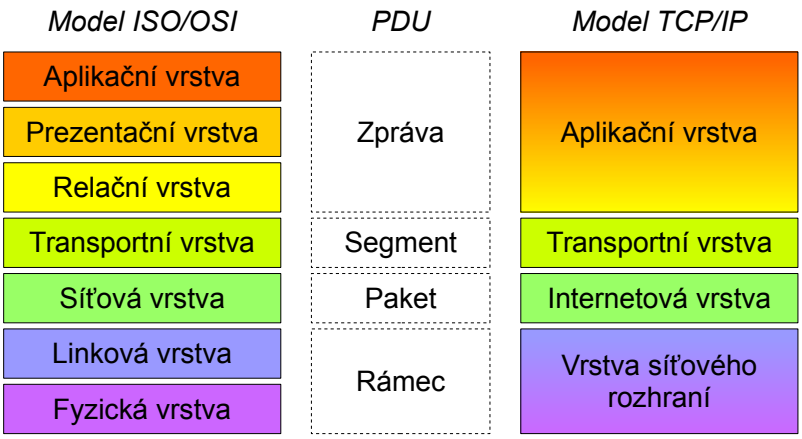
Kapitola shrnuje veškeré potřebné znalosti, které tvoří teoretický základ a úvod k praktické části bakalářské práce. Na počátku je představena architektura počítačových sítí a problematika jejich správy se zaměřením na monitorování. Z důvodu dalšího zaměření práce je dále věnován prostor technologii FPGA a možnostem jejího využití k hardwarové akceleraci zpracování a analýzy síťových dat na vysokorychlostních sítích. Poslední blok se pak zabývá konceptem softwarově definovaného monitorování sítí (Software Defined Monitoring, SDM), který je založený na sledování síťových toků a je využitelný pro flexibilní monitorování sítí.

2.1 Principy počítačových sítí

Není-li uvedeno jinak popis základů počítačových sítí vychází z poznatků uvedených v literatuře [24, 9, 16]. Počítačová síť je tvořena dvěma či více zařízeními navzájem propojenými za účelem sdílení informací a zdrojů. Tato zařízení se označují jako koncová. Jejich propojení může být uskutečněno pomocí kabelu (metalické kabeláže, optického kabelu) nebo bezdrátových technologií (Wi-Fi). Přenos dat probíhá ve formě binární reprezentace (logických úrovní 0/1) převedených na určitou fyzikální veličinu (velikost elektrického napětí, vlnovou délku elektromagnetického záření). Takto sdíleny mohou být datové soubory, aplikační programy či hardwarová zařízení jako jsou tiskárny a datová úložiště. Zařízení či počítačové aplikace spolu komunikují pomocí sady standardizovaných protokolů – sady určitých pravidel pro komunikaci.

Již na počátku rozvoje počítačových sítí byl kladen důraz na oddělení tří základních částí komunikačního procesu – technologie pro přenos signálu, zajištění spolehlivého přenosu a poskytnutí služeb uživateli počítačové sítě. Tento přístup vyústil v zavedení principu tzv. vrstvového řízení datové komunikace spočívající v rozdělení činností při datové komunikaci na vrstvy, kde každá z nich reprezentuje určitý krok v síťové komunikaci a vykonává specifickou úlohu. Při probíhající komunikaci mezi aplikacemi dochází k přenosu uživatelských dat (zpráv) mezi komunikujícími programy. Tyto programy se však nestarají o navázání spojení, vyhledání cesty pro přenos dat ani adresování, to je úkol nižších vrstev. Daná vrstva vždy využívá služeb nižší vrstvy, aniž by musela znát jak či pomocí jakých protokolů jsou její funkce implementovány.

Základním modelem pro popis síťové architektury je referenční sedmivrstvý model OSI (Open System Interconnect Reference Model) definovaný Mezinárodní standardizační organizací ISO (International Standards Organization). ISO/OSI je referenčním modelem, nikoliv protokolovým. Jeho úlohou je pochopení procesů využívaných v síťové komunikaci



Obrázek 2.1: Srovnání referenčního modelu ISO/OSI a modelu TCP/IP

a definice funkcí, nikoliv specifikace implementace protokolů jednotlivých vrstev. Z důvodu značné komplikovanosti struktury vrstev byla v praxi implementována pouze část tohoto modelu.

Dalším z běžně používaných modelů je model TCP/IP. Jak jeho označení napovídá, model je založen na protokolech TCP (Transmission Control Protocol) a IP (Internet Protocol), využívaných jako přenosová vrstva drtivé většiny dnešních počítačových sítí a také nejrozšířenější sítě Internet. Model vznikl z iniciativy amerického ministerstva obrany (Department of Defense, DoD) v rámci realizace koncepce projektu ARPAnet (Advanced Research Projects Agency Network). TCP/IP spojuje služby prezentační a relační vrstvy do jediné aplikační vrstvy. Na úrovni fyzického přenosu dat spojuje vrstvy fyzickou a linkovou do vrstvy síťového rozhraní, která je obvykle realizována na síťové kartě. Srovnání funkcionality vrstev obou modelů zobrazuje obrázek 2.1. Architektura TCP/IP je výrazně jednodušší. Narozdíl od ISO/OSI obsahuje pouze čtyři funkční vrstvy. Každá z vrstev definuje základní datovou jednotku pro přenos informací (PDU, Process Data Unit), zajišťuje určitou úlohu při komunikaci a využívá určitý způsob adresování a identifikace komunikujících stran. Od nejnižší z nich jsou to následující vrstvy:

Vrstva síťového rozhraní — nejnižší vrstva modelu zajišťuje ovládání síťových karet poskytujících přístup k přenosovému médiu (metalická, optická kabeláž, rádiové vlny) a realizujících samotný fyzický přenos dat. Vrstva dále definuje a popisuje standardy určující vlastnosti linky (typ média, napěťové charakteristiky elektrických signálů, způsob kódování logických úrovní, rychlost přenosu). Operuje na ní jeden z nejrozšířenějších protokolů Ethernet. Z dalších jsou to například Frame Relay nebo Token Ring. Přenášené datové jednotky se nazývají rámce a k adresování zařízení se používá fyzická nebo také hardwarová adresa. Na ethernetových sítích jde o 48-bitové binární číslo zapisované v hexadecimálním tvaru identifikující síťovou kartu a označované jako MAC (Media Access Control) adresa.

Internetová vrstva — vytváří logické spojení mezi zařízeními (koncovými uzly) a zodpovídá za nalezení nejvhodnější cesty k doručení dat a jejich správné směrování na místo určení. V případě ztráty dat například z důvodu výpadku linky či zaplnění vyrovnávacích front na prvcích infrastruktury je vysílající uzel o situaci informován a musí

sám zajistit opětovné odeslání informací. Základní funkčnost internetové vrstvy zajišťuje protokol IP (Internet Protokol) a k identifikaci zařízení na ní slouží IP adresa. Součástí jsou však další podpůrné protokoly pro předávání zpráv o výjimečných stavech (Internet Control Message Protocol, ICMP), informací potřebných pro směrování (Routing Information Protocol, RIP) nebo pro správu multicastových skupin (Internet Group Management Protocol, IGMP). Nezbytná je také podpora pro překlad logických IP adres na MAC adresy zajištěná protokolem ARP (Address Resolution Protocol). Tento překlad je naprosto nezbytný, jelikož nižší vrstva je schopná interpretovat pouze MAC adresy. Přenášené PDU se nazývají pakety. Protokol IP provádí jejich doručování s největším úsilím – tzv. best-effort delivery. Přenos probíhá bez záruky, že data budou bez poškození a ve správném pořadí skutečně doručena. Internetová vrstva neprovádí žádné kontroly správnosti přenosu ani opravu dat. Kontrolu a opravu chyb musí zajistit vyšší vrstvy modelu TCP/IP.

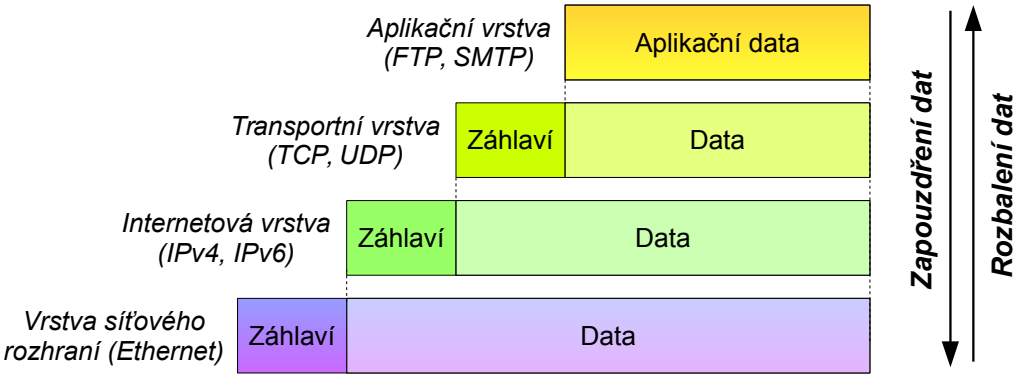
Transportní vrstva — vytváří logické spojení mezi aplikacemi nacházejícími se na koncových uzlech. Základní prováděnou činností je rozdělování aplikačních dat na menší jednotky tzv. segmenty a jejich odesílání. Na transportní vrstvě operují protokoly TCP (Transmission Control Protocol) a UDP (User Datagram Protocol). TCP je spojovaným protokolem garantujícím spolehlivý přenos. Před samotným odesláním dat je nejprve ustaveno spojení, odesílaná data jsou označena sekvenčními čísly a správnost jejich doručení je signalizována příchodem potvrzení. Protokol dále poskytuje mechanismy řízení toku a předcházení zahlcení. UDP je protokol bez spojení, má nižší režii, avšak neposkytuje prostředky pro kontrolu správnosti přenosu. K adresaci komunikujících aplikací slouží 16-bitová čísla přenášená v záhlaví protokolů TCP a UDP označovaná jako zdrojový a cílový port.

Aplikační vrstva — nejvyšší vrstva modelu zajišťuje interakci mezi aplikačními programy a předávání zpráv. Protokoly pracující na aplikační vrstvě provádějí úkoly jako jsou přenos souborů (File Transfer Protokol, FTP), sdílení souborů a tiskáren (služba Samba a síťový souborový systém NFS), služby doručování emailových zpráv (Simple Mail Transfer Protocol, SMTP), terminálové služby (Secure Shell, SSH) a další. Úlohou aplikační vrstvy je vhodná reprezentace dat, jejich kódování, vytváření a udržování relací (sezení). Pokrývá tak hned tři vrstvy referenčního modelu OSI – aplikační, prezentační a relační. Způsob adresování při datové komunikaci na aplikační úrovni závisí na konkrétní aplikaci či službě.

Při přenosu v síti dochází k zapouzdření dat vyšších vrstev do PDU nižších vrstev na straně odesilatele a jejich rozbalení na straně příjemce, tak jako to ilustruje obrázek 2.2. Uživatelskou aplikací odeslaná data jsou rozdělena na TCP či UDP segmenty a předána síťové vrstvě k doručení. Síťová vrstva zapouzdří segmenty do IP paketů, označí je síťovou adresou a předá je síťovému rozhraní – síťové kartě. Ta je převede na rámce, jimž přidá fyzickou adresu a zapíše je na přenosové médium. Po doručení je proveden opačný postup, kdy jsou postupně odstraněna záhlaví přenosových protokolů a data jsou předána cílové aplikaci.

2.2 Popis významných síťových protokolů

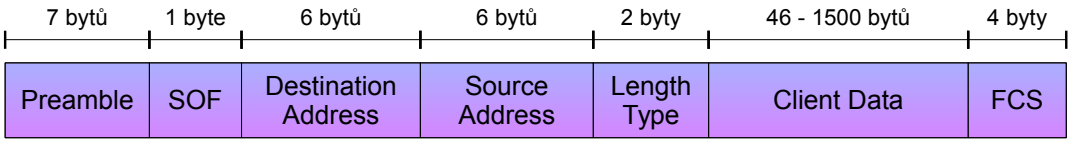
Nezbytným základem pro zpracování síťových dat a provádění stavového měření na síti je detailní znalost a pochopení principů při komunikaci používaných síťových protokolů.



Obrázek 2.2: Zapouzdření dat uživatelské aplikace v modelu TCP/IP

V textu se zaměříme na běžně používané komunikační protokoly tří ze čtyř vrstev síťového modelu TCP/IP. Popořadě to bude protokol Ethernet (vrstva síťového rozhraní), protokol IP (internetová vrstva) ve verzi 4 a verzi 6 a dvojice transportních protokolů TCP a UDP. Text kapitoly popisující vybrané komunikační protokoly vychází z příručky [9] a dokumentů RFC a standardů IEEE odkazovaných přímo v textu.

Ethernet je nejrozšířenější technologií používanou v lokálních LAN (Local Area Network) sítích a pokrývá oblast spadající do vrstvy v TCP/IP souhrnně označované jako vrstva síťového rozhraní. Současně používaná specifikace IEEE 802.3 [7] a její další rozšíření byla odvozena od původní technologie Ethernetu navržené firmou Xerox Corporation. Po fyzické stránce jsou standardem definovány typy a požadované vlastnosti propojovacích vodičů (kroucená dvojlinka, optické vlákno, koaxiální kabel), možnosti jejich propojování nebo způsob zakódování přenášených dat. Z linkových vlastností popisuje standard metodu pro správu řízení přístupu k médiu či maximální přenosovou rychlost. Dřívější standardy definovaly přenosové rychlosti 10 Mb/s a 100 Mb/s. Později byl představen tzv. Gigabitový Ethernet a dnešní standardy dokonce dovolují přenášet data rychlostí 40 až 100 Gb/s. Technologie Ethernet je tak nasazována nejen v lokálních sítích, ale také na vysokorychlostních páteřních linkách a v datových centrech. Formát ethernetového rámce je zobrazen na obrázku 2.3 a význam jednotlivých polí je vysvětlen v následujícím seznamu (číselné hodnoty v obrázku udávají velikosti příslušných polí v bytech):



Obrázek 2.3: Schéma struktury rámce Ethernet

Preamble, SOF (Start of Frame): úvodní část rámce slouží především k synchronizaci obvodů přijímače a vysílače a signalizuje vlastní začátek obsahu rámce.

Source/Destination Address: dvě šestibytová pole obsahují MAC adresy zdroje a příjemce. Adresa příjemce určuje, komu má být rámec zaslán. Může být jak individuální (unicast), tak skupinová (broadcast, multicast). Adresa zdroje je vždy typu unicast.

Length/Type: dvoubytové pole může mít dva významy. Udává velikost obsažených dat v poli *Client Data* nebo určuje typ protokolu vyšší vrstvy, který je v poli *Client Data* zapouzdřen. Identifikace použitého významu se provádí na základě obsažené hodnoty.

Client Data: pole obsahuje zapouzdřená data vyšších vrstev. Jeho velikost je proměnná, vždy ale musí být dodrženy požadavky na minimální a maximální povolenou délku rámce. Velikost pole se musí pohybovat v rozsahu 46 až 1500 bytů. Není-li dosažena minimální délka 46 bytů, musí být zbývající prostor vyplněn libovolnými daty.

FCS (Frame Check Sequence): obsahuje kontrolní součet (Cyclic Redundancy Check, CRC) délky 4 byty určený na základě obsahu předchozích polí a sloužící ke zjištění chyb při přenosu rámce.

Omezení minimální velikostí rámce je dáno historickým vývojem technologie, kdy byl přístup k přenosovému médiu řízen metodou CSMA/CD (Carrier Sense Multiple Access with Collision Detection), aby komunikující stanice byly schopny správně detekovat kolizi při vysílání.

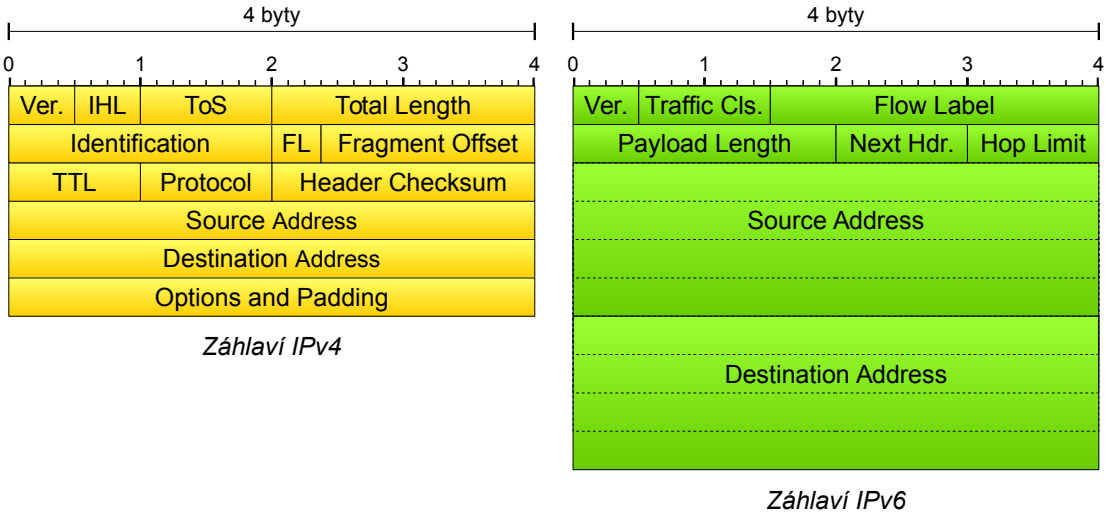
Dalším významným protokolem je internetový protokol (Internet Protocol, IP) definovaný v RFC 791 [19]. Jde o protokol síťové, respektive internetové, vrstvy zajišťující adresování koncových zařízení a směrování dat mezi nimi. Neobsahuje však mechanismy zaručující spolehlivé doručení. To musí být zajištěno protokoly vyšších vrstev. Charakteristickým rysem je nespojovaný typ komunikace a také doručování s nejlepším úsilím (best-effort delivery), kdy je snahou odeslat data co nejvhodnější cestou k příjemci. Protokol existuje ve dvou běžně rozšířených a používaných verzích – v původní verzi IPv4 a novější IPv6 [4]. Hlavním důvodem k přechodu na novou verzi je postupné vyčerpání adresového prostoru IPv4 adres. K identifikaci síťových zařízení se v IPv4 používá 32-bitová adresa. Teoreticky tak poskytuje více jak 4 miliardy adresovatelných zařízení. V dnešní době však není tento počet plně dostačující. IPv6 využívá k adresaci 128-bitové adresy a poskytuje tak dostatečně (alespoň v blízké budoucnosti) velký adresový prostor. Mimoto se IPv6 zaměřuje na snížení zátěže směrovačů, podporu kvality služeb, mobilních zařízení (tzv. multihoming), možnost automatické konfigurace adres a přidává řadu nových vlastností.

Struktura záhlaví protokolů obou verzí IPv4 a IPv6 se vzájemně liší a pro porovnání je zobrazena na obrázku 2.4 (číselná osa v záhlaví slouží k odečtení délky jednotlivých polí, je uvedena v bytech). Některé typy původních údajů zůstaly v nové verzi protokolu zachovány, ať již ve stejné či nové podobě. Jiné zmizely úplně a byly nahrazeny zcela novými položkami. Detailnější popis a vysvětlení funkcí významných políček je uvedeno v následujícím seznamu:

Version: pole značící verzi použitého protokolu IP. Pro IPv4 by pole mělo obsahovat binární vyjádření hodnoty 4, pro IPv6 pak hodnotu 6.

Internet Header Length (IHL): určuje délku záhlaví IP paketu v násobcích 32-bitových slov. Velikost záhlaví IPv4 se může měnit z důvodu proměnné délky pole *Options and Padding*. Minimální hodnota pole platná pro IPv4 je 5 (velikost záhlaví 20 bytů). IPv6 používá pevnou velikost záhlaví 40 bytů, proto není obdobné pole přítomno.

Type of Service (ToS)/Traffic Class: specifikuje způsob zpracování IPv4 paketu protokoly vyšších vrstev a přiřazuje mu různé úrovně důležitosti. Hodnotu pole lze chápat jako prioritu paketu a umožňuje přednostní zpracování některých paketů a zajištění požadavků na kvalitu služby (Quality of Service, QoS). Obdobné pole v IPv6 určuje



Obrázek 2.4: Schéma struktury záhlaví protokolů IPv4 a IPv6

třídu provozu či prioritu paketu. Jeho význam je podobný jako hodnota *ToS* v záhlaví IPv4 a slouží pro potřeby zajištění kvality služeb.

Total Length/Payload Length: určuje celkovou délku IPv4 paketu v bytech včetně záhlaví. Obdobné pole v IPv6 však udává délku pouze vlastních dat paketu (bez jeho záhlaví) v bytech. Maximální velikost celého IP paketu ,respektive jeho datové části, je $2^{16} - 1 = 65535$ bytů (64 kB). Na nižší vrstvě (Ethernetu) je však délka dat v rámci omezena na 1500 bytů. Větší délky IP paketů se proto nepoužívají, docházelo by totiž k fragmentaci na lokálních sítích.

Flags (FL): skládá se z pole jednobitových příznaků ovlivňujících fragmentaci IPv4 paketů. Jeden z bitů specifikuje, zda mohou být pakety fragmentovány. Druhý bit určuje je-li přijatý paket posledním fragmentem v řadě a třetí z bitů není využit.

Time to Live (TTL)/Hop Limit: vyjadřuje životnost paketu nebo také počet skoků. Hodnota je dekrementována každým směrovačem. Dosažení nuly způsobí zahození paketu. Tímto způsobem je řešen problém možného zacyklení paketů mezi směrovači a jejich nekonečného zpracovávání.

Protocol/Next Header: značí protokol vyšší vrstvy, jehož PDU je přenášeno ve vlastních datech paketu. Typicky jde o identifikátor některého z transportních protokolů (hodnota 6 pro TCP nebo hodnota 17 pro UDP). V případě IPv6 určuje také typ volitelného záhlaví. Rozšiřující záhlaví přidávají podporu fragmentace či šifrování obsahu IPv6 paketů a lze je za sebou řetězit.

Source Address/Destination Address: pole obsahující IPv4, respektive IPv6, adresu identifikující odesilatele, respektive příjemce, paketu. Délka každé IPv4 adresy k adresaci koncového zařízení je 32 bitů. IPv6 adresy mají čtyřnásobnou délku 128 bitů.

Ačkoliv specifikace protokolu IPv6 byla představena již před více než 15 lety a současné moderní operační systémy ji také plně podporují, rozšířenější verzí je stále IPv4. I když podíl využití IPv6 neustále roste, překážkou bránící ve vyšším rozšíření IPv6 mohou být nově

vzniklé bezpečnostní problémy související s jejím zaváděním nebo také jednoduše neochota poskytovatelů internetového připojení investovat nemalé prostředky do modernizace vlastní infrastruktury.

Další dvojicí významných protokolů, kterou nelze v tomto přehledu opomenout, jsou transportní protokoly TCP (Transmission Control Protocol) a UDP (User Datagram Protocol) definované v dokumentech RFC 793 [20] a RFC 768 [18]. Protokoly vytvářejí logické spojení mezi aplikacemi. K identifikaci aplikací se používají 16-bitová bezznaménková čísla z rozsahu 0 až 65 535, takzvaná čísla portů. Pro odeslání dat je třeba znát číslo portu cílové aplikace. Pro užívání portů je zavedeno jejich následující rozdělení:

Well Known Ports (0 – 1023) — známé porty používané pro serverové části aplikací a rezervované pro běžně používané standardní služby, jako je FTP, SMTP, SSH a jiné. V případě použití nestandardního portu jej většinou musí specifikovat uživatel klientské aplikace. Rezervované porty přiděluje organizace IANA (Internet Assigned Numbers Authority).

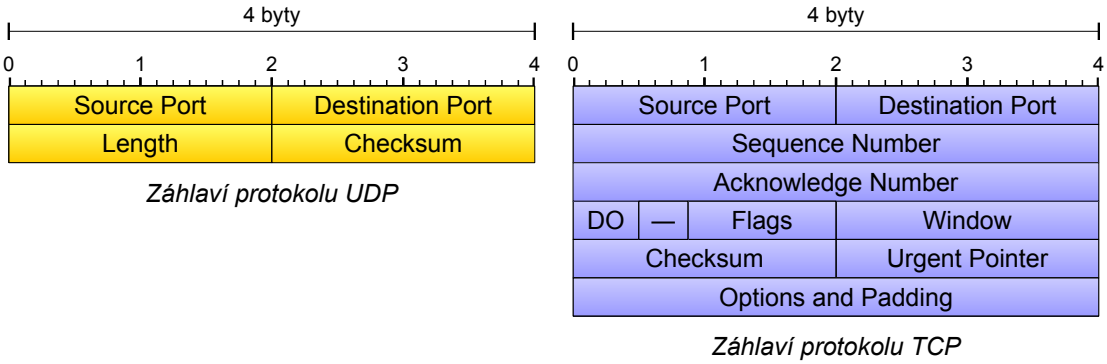
Registered Ports (1024 – 49 151) — registrované porty mají využití pro nestandardní uživatelské služby a procesy. Typicky jsou sem zahrnuty proprietární aplikační protokoly. Tyto porty nejsou žádným způsobem přidělovány jednotlivým organizacím. IANA pouze vede evidenci jejich využití.

Dynamic and Private Ports (49 152 – 65 535) — dynamické porty jsou určeny k volnému, krátkodobému použití. Využívají se k identifikaci především klientských aplikací, kdy jsou čísla portů generována právě z tohoto rozsahu. Jejich využití pro serverové aplikace není zcela běžné.

Transportní protokol UDP je velmi jednoduchý a poskytuje pouze nespojovanou formu služby bez záruky spolehlivého doručení či zachování pořadí paketů. Využití najde všude tam, kde není nutné použití složitého mechanismu protokolu TCP zajišťující vysokou spolehlivost. Používá jej pro komunikaci například síťový souborový systém NFS.

Protokol TCP je na druhou stranu daleko složitější. Za cenu vyšší režie však poskytuje zajištění spolehlivého přenosu bez omezení maximální velikosti přenášených dat. Data předávaná protokolu TCP nemusí být nijak speciálně strukturována. Jde o prostý proud bytů, což umožňuje aplikacím přenášet souvislé bloky dat bez ohledu na to, jak bude přenos zajištěn nižšími vrstvami. Implementace protokolu obstarává úlohů členění aplikačních dat do bloků a jejich předávání internetové vrstvě. TCP je spojově orientovanou službou určenou pouze pro dvojici komunikujících stran. Služba vyžaduje počáteční vytvoření a inicializaci parametrů spojení. Plně duplexní režim dále poskytuje možnost obousměrného přenosu dat v rámci jednoho ustaveného spojení. Pro zajištění spolehlivosti používá techniku PAR (Positive Acknowledgment and Retransmission) založenou na principu odesílání potvrzujících zpráv o přijetí dat příjemcem. Po odeslání paketu ze zdroje je spuštěn časový odpočet. Není-li potvrzení od příjemce o přijetí dat doručeno před vypršením časové lhůty, je paket opětovně odeslán. Použitý zřetězený přenos dat umožňuje zasílání více paketů najednou, bez nutnosti čekání na jejich potvrzení, čímž je zvyšována efektivita přenosu. Takto odesílaným segmentům jsou přiřazována sekvenční čísla. Jejich číslování má význam pro účely detekce duplicitně přijatých a identifikaci chybějících, ještě nedoručených segmentů. Potvrzení pak obsahují informaci pro protistranu, že předchozí odesílané segmenty s určitým sekvenčním číslem byly úspěšně doručeny. TCP implementuje také mechanismus Sliding

Window, který sloužící k řízení toku dat a předchází zahlcení jejich příjemce. V rámci protokolu předávaná hodnota velikosti okénka signalizuje zdrojové aplikaci dostupnou kapacitu vyrovnávací paměti na straně příjemce. Velikost okénka se vyjadřuje v bytech a zároveň značí počet bytů, které může odesilatel odeslat, aniž by musel dostat potvrzení o úspěšném doručení předcházejícího přenášeného segmentu. Nulová hodnota představuje požadavek na pozastavení vysílání.



Obrázek 2.5: Schéma struktury záhlaví transportních protokolů TCP a UDP

Na obrázku 2.5 je uvedena struktura záhlaví segmentů obou transportních protokolů. Číselná osa v horní části slouží k odvození délek jednotlivých polí záhlaví. Hodnoty na stupnici jsou uvedeny v bytech. Popis významných polí je předmětem následujícího textu. Některá z polí jsou stejná nebo mají přinejmenším podobný význam v obou protokolech:

Source/Destination Port: zdrojový a cílový port slouží k adresaci a identifikaci přijímající a odesílající aplikace na zdrojové a cílové stanici.

Length/Data Offset (DO): pole určuje délku celého UDP segmentu (včetně záhlaví a vlastních dat) v bytech. V případě TCP obsahuje pouze délku záhlaví segmentu, uvedenou v násobcích 32-bitových slov. Uvedená hodnota se projevuje proměnnou délkou pole *Options and Padding*.

TCP kromě polí významově společných s UDP přidává další položky související především s procesem ustavení spojení, potvrzováním přijetí segmentů, obecně zajištěním spolehlivého přenosu dat či mechanismem řízení zahlcení. Pro potřeby monitorovacích a bezpečnostních aplikací je významným z nich pole *Flags* tvořené jednobitovými řídícími příznaky signalizujícími stav spojení a souvisejících s přenášeným segmentem dat. Jednotlivé bity mají následující význam:

URG — hodnota v poli *Urgent Pointer* označující data datové části segmentu je platná. Bit je použit v případě, kdy je nutné mimořádně doručit data segmentu mimo pořadí. Takový segment je na straně příjemce zpracován přednostně, bez ohledu na stav zpracování ostatních příchozích data.

ACK — číslo v poli potvrzení *Acknowledge Number* je platné. Značí segment potvrzující správné přijetí dat příjemcem.

PSH — značí segment, jehož data mají být předána aplikaci bez ohledu na zaplnění vyrovnávacích pamětí. Na straně odesilatele značí, že data musí být bezodkladně

odeslána, na straně příjemce, že musí být okamžitě předána aplikačnímu programu. Příznak se uplaněje v aplikacích komunikujících v reálném čase.

RST — signalizuje požadavek na uvedení spojení do počátečního stavu – resetování spojení. Není standardním prostředkem pro ukončení komunikace.

SYN — požadavek na ustavení nového spojení. Spojení je navázáno tzv. třícestným mechanismem (three-way handshaking), kdy dojde k synchronizaci sekvenčních čísel příjemce a odesilatele.

FIN — požadavek na standardní ukončení spojení. Ukončení se provádí modifikovaným třícestným mechanismem. Protistrana potvrzuje uzavření spojení odesláním segmentu s příznakem *ACK*.

ECE, CWR, NS — tři příznaky související s rozšířením ECN (Explicit Congestion Notification) definovaným v dokumentech RFC 3168 [22] a RFC 3540 [25]. Přetížený směrovač podporující uvedené rozšíření označí tímto způsobem přenášené segmenty. Komunikující strany na základě jejich příjmu pak mohou snížit přenosovou rychlost a zmírnit zatížení směrovače dříve, než dojde ke ztrátám segmentů z důvodu jeho přetížení, čímž se přispívá ke zvýšení efektivity celého přenosu.

2.3 Správa a monitorování počítačových sítí

Do oblasti správy počítačových sítí nespadá pouze vhodné propojení a konfigurace funkčních zařízení síťové infrastruktury zabezpečujících správné směrování a přenos dat od zdroje k cíli. Nedílnou součástí správy sítí a činností každého síťového administrátora je také sledovaní stavu sítě, aktivit a komunikace na ní probíhajících. Takto získané informace přispívají k zajištění efektivního a bezproblémového fungování sítě. Údaje jsou důležité například pro plánování rozvoje sítě. K úspěšné správě počítačové sítě potřebujeme informace o její topologii (jaká zařízení jsou do sítě připojena, jak jsou nastaveny jejich síťové adresy), o aktuálním stavu těchto zařízení (přehled aktivních rozhraní, spuštěných služeb) a statistické údaje o probíhajících přenosech (množství přenesených paketů, přenesených bytů). V některých případech je však nutné znát detailnější informace o probíhající komunikaci či přímo obsah přenášených dat. Mezi základní fáze zjišťování informací o síti patří monitorování sítě a analýza síťových dat [16].

Monitorování sítě představuje zjišťování stavu uzlů sítě, síťových služeb a stavu probíhající komunikace. Monitorování přitom dále dělíme na aktivní a pasivní.

Aktivní monitorování využívá techniky pravidelného aktivního zasílání zpráv testujících či dotazujících se na dostupnost linek, uzlů a služeb. Přenos zpráv je realizován např. protokoly ICMP či SNMP. K dotazování na dostupnost aplikací lze pak využít třeba službu `telnet`. Běžnými nástroji používanými pro tento typ měření jsou programy `ping` a `traceroute`. Využívají se k určování propustnosti, zpoždění či ztrátovosti paketů při přenosu dat. Neposkytují však žádnou informaci o aktuálně probíhající komunikaci na síti.

Pasivní monitorování naopak není zdrojem žádné přídavné komunikace. Získávání veškerých informací o stavu sítě a zařízeních je založeno na sledování již probíhající komunikace. Pasivní monitorování zahrnuje také sledování logovacích informací aplikací

a služeb, sbíraných např. s pomocí nástroje `syslog` nebo asynchronních zpráv, hlášek či upozornění aplikací doručovaných protokolem SNMP případně formou elektronické pošty nebo informací ve formě NetFlow dat [16].

Proces analýzy síťových dat se zabývá podrobnějším vyhodnocením a zkoumáním přenášených dat. Analýze jsou podrobeny nejen záhlaví linkové, síťové, případně transportní vrstvy, ale také obsah paketů na aplikační úrovni. Datový analyzátor obvykle pracuje na sdíleném médiu nebo je pro potřeby analyzátoru zkoumaný provoz replikován z jiného portu. Jeho činnost lze rozdělit do dvou fází – snímání dat a provádění samotné analýzy. Mezi používané softwarové nástroje k zachytávání provozu patří `Wireshark` či `tcpdump`. Uvedené nástroje umožňují sledování dat v reálném čase. Jsou založené na síťové knihovně `libpcap`, která podporuje přímé čtení dat ze síťového rozhraní. Mimo jiné je jejich součástí i analyzátor odchycených dat, který kromě interpretace položek IP záhlaví či transportních protokolů umožňuje i analýzu některých aplikačních protokolů. Sledování přenášených dat a jejich analýza v reálném čase je velmi náročná, jak na výpočetní výkon (interpretace záhlaví), tak na kapacitu úložiště v případě zachycení dat.

Pro potřeby správy sítě není důležitý pouze aktuální obsah dat a stav současně probíhající komunikace, ale také dlouhodobé souhrnné statistiky získané analýzou provozu, umožňující sledování tendencí a vývoje síťového provozu. Mezi typicky sledované statistické údaje patří např. vytíženost jednotlivých linek, jak z hlediska časového, tak aplikačního (jaké typy požadavků převažují), objem přenášených dat, směr největších toků dat, žebříček nejvíce komunikujících uzlů, dále pak zastoupení různých komunikačních protokolů při přenosu dat či úroveň zabezpečení komunikace (šifrování spojení) [16].

Hlavním problémem při monitorování a zjišťování stavu probíhající komunikace je obrovský objem zpracovávaných dat. Na páteřních sítích jde o velké množství paketů přenášejících data v řádu desítek gigabitů za sekundu. Sledování a ukládání celých paketů nebo i pouze dílčích informací o každém z nich je prakticky nemožné. Za cenu ztráty některých informací o paketech, lze zpracovávaný objem dat výrazně snížit ukládáním pouze statistických či agregovaných informací o nich. Vhodným způsobem agregace je sdružení paketů do tzv. síťových toků. Ukládány jsou pak pouze souhrnné informace o těchto tocích nikoliv o každém jednotlivém paketu.

Síťový tok lze chápat jako posloupnost paketů, které procházejí monitorovaným bodem sítě za určitý časový interval a mají společné vlastnosti, jež jsou typicky odvozené z obsahu protokolových záhlaví paketů. Síťový tok bývá nejčastěji identifikován položkami zdrojové a cílové IP adresy paketu, zdrojovým a cílovým portem, typem protokolu transportní vrstvy, položkou označující typ služby a identifikátorem vstupního rozhraní monitorovacího prvku. O každém toku je uchovávána řada informací. Mezi typické z nich patří především časová značka začátku a konce sledovaného síťového toku, počet přenesených bytů, počet přenesených paketů či logický součet polí ze záhlaví paketů s TCP příznaky [16].

Architektura monitorovacích systémů založených na sledování síťových toků vychází z RFC 3954 [2] popisující systém NetFlow, který byl vyvinut společností Cisco pro účely monitorování provozu. Systém je nejčastěji tvořen následujícími základními prvky:

Exportér. Exportér, někdy nazývaný také jako sonda, je síťové zařízení, které monitoruje provoz procházející sledovaným bodem sítě a vytváří záznamy o tocích. Na základě příchozích paketů jsou vytvářeny zcela nové nebo aktualizovány starší záznamy v paměti aktivních (probíhajících) toků, tzv. flow cache. Export záznamu o toku nemusí probíhat bezprostředně po jeho ukončení, ale může být pozdržen tak, aby bylo za

účelem vyšší efektivity exportováno více záznamů současně. Existuje několik případů, kdy lze síťový tok považovat za ukončený a lze jej vyřadit z paměti flow cache:

Detekce konce toku — detekce paketu označující konec toku, např. příchod paketu s nastaveným příznakem *FIN* či *RST* na úrovni transportního protokolu TCP.

Neaktivita toku — neaktivní timeout, dojde k překročení doby, po kterou nepřišel žádný paket náležící danému toku. Jde o dobu v řádu sekund.

Příliš dlouhý tok — aktivní timeout, dojde k překročení maximální doby trvání toku. Daný tok je považován za ukončený i přes neustálý příchod paketů. Doba je volena v řádu minut.

Zaplnění paměti flow cache — paměť flow cache je zaplněna a je nutné uvolnit místo čerstvě zakládanému záznamu nového toku. Některé toky tak mohou být uměle ukončeny za účelem uvolnění kapacity paměti.

Exportér může být samostatným síťovým prvkem nebo jako (softwarová) součást jiného aktivního síťového zařízení. Pro zajištění kvalitního sběru dat však musí poskytovat dostatečný výpočetní výkon, aby byl schopen zpracovat všechny pakety procházejících sledovaným bodem sítě. Za účelem snížení zátěže exportéru bývá někdy využito přístupu vzorkování nebo filtrování provozu, či jejich kombinace. Sledovány pak mohou být ale pouze vybrané síťové toky. Exportér se tak z důvodu požadavku na vysoký výkon jeví jako vhodný prvek systému pro využití hardwarové akcelerace.

Kolektor. Kolektor je zařízení, které přijímá záznamy exportované jednou, či více sondami. Záznamy ukládá na disk či do databáze a dále je poskytuje ke zpracování. Na kolektoru mohou být nad daty prováděny další operace např. další úroveň agregace.

Komunikační protokol. Exportované záznamy jsou ze sondy na kolektor přenášeny v určitém formátu. K tomu slouží komunikační protokol. Protokol NetFlow byl původně jako proprietální vyvinut firmou Cisco, existují však i další jeho implementace v několika verzích. Další verze nabízejí rozšíření přenášených položek o podporu IP verze 6 nebo možnost definování uživatelských položek. Přenos záznamů na kolektor probíhá většinou formou nespojované komunikace UDP.

Další nástroje. Mezi další prvky architektury patří nástroje pro analýzu, zpřístupnění a další zpracování dat uložených na kolektoru, jejichž součástí může být uživatelské rozhraní umožňující vhodnou vizualizaci ve formě grafů či dotazování nad sesbíranými daty.

Principy a architektura NetFlow navrženého společností Cisco jsou natolik obecné, že jsou využity i u jiných monitorovacích systémů. Jako příklad uveďme systémy založené na protokolu IPFIX (IP Flow Information Export) [3], jehož vývoj zastřešuje organizace IETF (Internet Engineering Task Force) a podílejí se na něm i původní tvůrci NetFlow. Kromě monitorování sítě za účelem bezpečnostní analýzy a detekce útoků lze monitorování na bázi toků využít v souvislosti s účtováním poplatků za služby dle přenesených dat či využití šířky pásma nebo při plánování budoucího rozvoje sítě. Dále také poskytuje vhodné řešení pro ukládání informací o přenesených datech. [16].

Monitorování vysokorychlostních sítí je specifické svými požadavky na propustnost a výkon monitorovacího systému. Dostatečná rychlost zpracování dat je velmi důležitá. Nekontrolovaná ztráta dat může výrazně ovlivnit přesnost měření a může třeba zamezit správné

detekci útoku. Pro zajištění kvalitního sběru dat je nutné zpracovávat každý příchozí paket. Doba pro zpracování jediného paketu je přitom na linkách s rychlostí 100 Gb/s značně omezena a pohybuje se v řádech jednotek nanosekund. I přes využití nejlepších známých algoritmů, podpory více procesorových jader je realizace čistě softwarového řešení zpracování paketů na komoditních procesorech pro uvedené rychlosti nemožná. Východiskem z této situace může být hardwarová akcelerace časově kritických operací s využitím principu zřetězeného zpracování. V oblasti počítačových sítí a zpracování síťových dat lze přitom identifikovat řadu časově kritických operací:

Filtrace paketů — vybrání pravidla či množiny pravidel odpovídající přijatému paketu s ohledem na možnost využití pokročilých např. kontextově závislých pravidel, různých typů porovnávání položek s informacemi v záhlaví paketu, přesné porovnání, intervalové porovnání, vyhledání nejdelšího shodného prefixu.

Analýza paketů — analýza záhlaví paketů, přesné určení umístění položek v záhlaví a jejich efektivní extrakce, identifikace aplikačního protokolu na základě obsahu příchozích dat.

Stavové zpracování síťového provozu — zpracování velkého množství záznamů o síťových tocích, jejich ukládání a uchování ve vhodném formátu na disku nebo v databázi, vhodný způsob komprimace, možnost dotazování nad daty, zajištění vyhledání záznamu v konstantním čase.

Hledání útoků — hledání vzorů a regulárních výrazů v databázi s velkým množstvím záznamů o síťových tocích, použití heuristické analýza pro odhalení neznámých útoků.

Hardwarová akcelerace umožňuje zvýšení rychlosti zpracování dat na základě využití specializovaných hardwarových jednotek navržených pro konkrétní úlohu. Vyšší rychlosti je dosaženo díky možnosti paralelního či zřetězeného zpracování dat a přizpůsobení výpočetní jednotky použitému algoritmu. Trendem se stává využití rekonfigurovatelného čipu FPGA vedle klasického procesorového jádra. Procesor počítače tak může souběžně provádět jinou operaci a od akcelerační jednotky pouze přebírá již předzpracovaná data, respektive finální výsledek operace. Hardwarová akcelerace poskytuje mimo jiné nižší spotřebu díky možnosti snížení pracovní frekvence či dynamické správy napájení. Z hlediska vysokorychlostních sítí pak například přináší možnost zpracování dat v reálném čase bez nutnosti jejich vzorkování.

Komplexní systém pro spolehlivé monitorování a bezpečnost vysokorychlostních počítačových sítí se neobejde bez několika klíčových prvků:

Dostatečná rychlost zpracování síťového provozu. Požadovaná propustnost a výkon lze zajistit hardwarovou akcelerací algoritmů a architektury například s využitím technologie FPGA. Optimálním řešením poskytujícím i určitou míru flexibility je použití klasické serverové platformy s několika více-jádrovými procesory v kombinaci s FPGA akcelerační kartou.

Řízení zpracování paketů v hardware. Hardwarová akcelerace nabízí vysoký výkon a rychlost zpracování, avšak výsledný systém je špatně rozšiřitelný. Východiskem je úzké provázání se softwarovým řízením a vůbec vhodné rozložení činností mezi hardwarovou a softwarovou vrstvu, které poskytne dostatečnou flexibilitu. Touto problematikou se zabývá koncept softwarově definovaného monitorování (SDM, Software Defined Monitoring) využitelný pro flexibilní monitorování sítí, které je založeno na

sledování síťových toků, a umožňující softwarově řízenou ztrátu dat na základě analýzy síťového provozu.

Aplikace pro detekci bezpečnostních incidentů. Aplikace provádějící zpracování aplikačních protokolů a behaviorální analýzu provozu za účelem detekce útoků a anomálií na síti s možností modelování důvěryhodnosti, identifikace botnetů či hlášení detekovaných incidentů do systémů včasného varovaní. Známé útoky jsou odhaleny na základě předem definovaných vzorů, ty neznámé na základě detekce anomálií. Cílem je poskytnutí zpětné vazby systému zpracování paketů pro účely filtrace provozu (zmírnění DDoS útoků) nebo sběru detailnějších informací pro podrobnější analýzu incidentu.

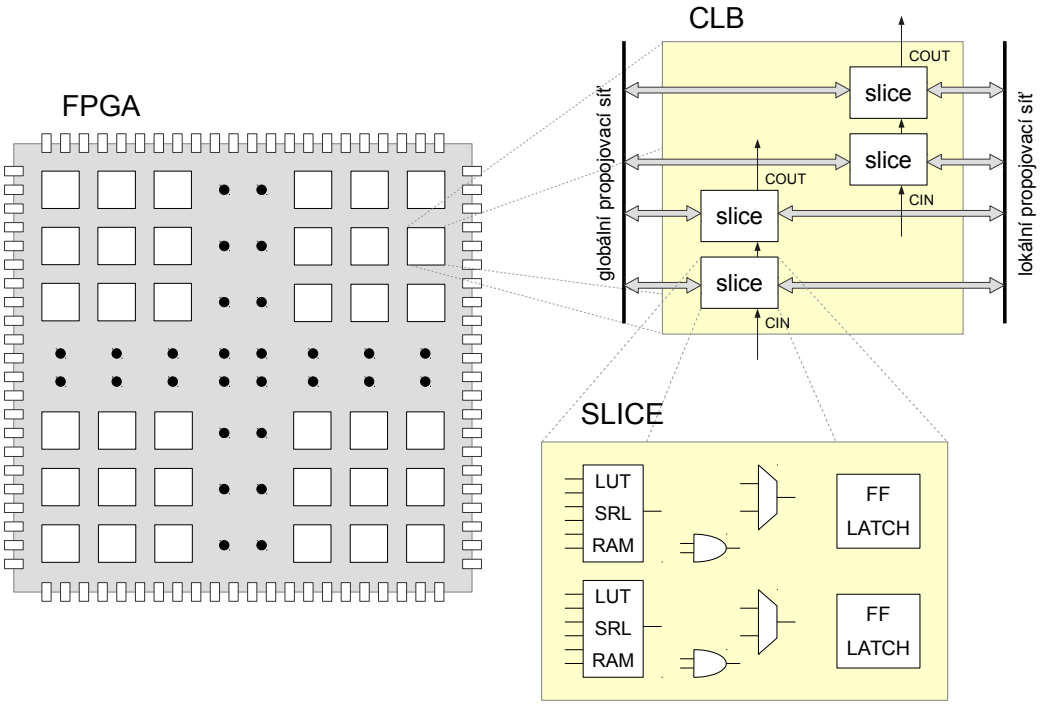
Technologii FPGA využitelné k hardwarové akceleraci exportéru pro stavového měření toků na síti a možnostem implementace algoritmů na této hardwarové architektuře je dále podrobněji věnována následující část 2.4. Sekce 2.5 se pak detailněji zabývá zmíněným konceptem softwarově definovaného monitorování.

2.4 Technologie FPGA

Aplikačně specifické integrované obvody (Application specific integrated circuits, ASIC) jsou schopny provádět určité výpočty a operace mnohem rychleji než obdobná softwarová řešení na obecném procesoru. Bohužel takové specifické výpočetní obvody jsou velmi drahé a vyžadují velké úsilí v době vývoje. Cena návrhu masky ASIC obvodů se s příchodem nových technologií navíc neustále zvyšuje. ASIC obvody se tak vyplatí vyrábět pouze ve velkých sériích. Jsou využívány zejména pro náročné aplikace vyžadující rychlost, malé rozměry a nízkou spotřebu. Poté, co je takovýto integrovaný obvod vyroben, již není možné provádět změny jeho struktury. Výroba ASIC obvodů je tak velmi náchylná na chyby v návrhu. Z tohoto důvodu je vývoj dlouhý a počáteční náklady velmi vysoké. Tento výrazný nedostatek odstraňují programovatelná hradlová pole (Field Programmable Gate Arrays, FPGA), která nabízejí střední cestu mezi rychlostí a výkonem ASIC obvodů a flexibilitou softwarového řešení.

Programovatelná hradlová pole jsou prefabrikované hardwarové čipy s možností rekonfigurace, které mohou být naprogramovány tak, aby realizovaly různé uživatelsky definované kombinační a sekvenční logické obvody. Ve srovnání s ASIC obvody je tato realizace daleko levnější a nabízí vyšší flexibilitu díky rekonfigurovatelnosti. Hlavní výhodou je jednodušší návrh aplikace, rychlejší vývoj a s tím související rychlejší uvedení cílového produktu na trh. Rekonfigurace přináší rychlou možnost opravy chyb návrhu a přidávání nových vlastností aplikace. Z tohoto důvodu je technologie FPGA využívána při vývoji prototypů a umožňuje rychlé ověření funkčnosti návrhu dané aplikace. Dále se také snižuje počet aplikací, které je nutné z důvodu výkonnostních požadavků řešit pomocí ASIC obvodů, neboť díky novým výrobním technologiím se zvyšuje rychlost FPGA čipů. Rychlost práce čipu (procesoru počítače, ale i FPGA) se vyjadřuje v počtu cyklů hodinového signálu za sekundu a je označována jako pracovní nebo taktovací frekvence. Popis uvedený v následujícím textu vysvětlující základní strukturu FPGA a návrh číslicových obvodů pro ně vychází z [23].

Obvody FPGA jsou tvořeny maticí konfigurovatelných logických bloků (Configurable Logic Block, CLB), které jsou propojeny programovatelnou propojovací maticí (Programmable Switch Matrix, PSM). Funkci CLB lze modifikovat podle potřeb návrháře. Tyto bloky jsou dále děleny na menší buňky označované jako *slice*. Každá tato buňka obsahuje funkční



Obrázek 2.6: Hierarchická struktura FPGA

generátory, hradla, multiplexory, registry, případně další konfigurovatelné součásti. Funkční generátory jsou provedeny jako n-vstupé vyhledávací tabulky (Look-Up Table, LUT), které jsou základním stavebním kamenem při implementaci kombinační logiky. LUT může realizovat libovolnou funkci n proměnných. Spojováním LUT do stromové hierarchie lze dosáhnout funkcí více proměnných. Fyzicky jsou LUT realizovány pamětí SRAM. Tuto paměť lze v rámci buňky slice využít také jako posuvný registr či jako tzv. distribuovanou paměť. Registr je základní stavební prvek pro realizaci sekvenční logiky a lze jej nastavit jako záchytný klopný obvod (latch) – reagující na hladinu signálu nebo jako klopný obvod typu FF (flip-flop) – reagující na hranu hodinového signálu. Základní struktura FPGA a bloků CLB je znázorněna na obrázku 2.6. Šedá část vlevo na obrázku představuje čip FPGA, který je tvořen bloky CLB (bílé čtverečky). Detailnější pohled na blok CLB poskytuje obrázek vpravo nahoře. CLB má přístup k lokální, respektive globální, propojovací síti a skládá se z buněk slice, jejichž struktura je zobrazena vpravo dole. Základem buňky slice jsou funkční generátory LUT, klopné obvody FF a další pomocná logika.

Propojení CLB bloků lze realizovat pomocí globální či lokální propojovací sítě. Lokální síť poskytuje propojení pouze v oblasti sousedních bloků a zajišťuje rychlé lokální linky pro podporu skládání LUT nebo speciální propoje pro konstrukci sčítaček. Při realizaci propojů je snahou minimalizovat délku cest z důvodu jejich zpozdění. Dlouhé propojovací cesty mohou mít negativní vliv na dosaženou maximální pracovní frekvenci výsledného obvodu. Propojení a rozvody zabírají většinu FPGA čipu a i přesto není možné zaručit jakékoliv propojení libovolných bloků z důvodů velkého množství CLB. V některých případech tak může realizace obvodu selhat právě v důsledku nedostatečného množství propojů.

Kromě základních programovatelných logických bloků obsahují FPGA čipy obvody pro řízení hodinového signálu včetně speciálních hodinových rozvodů a další vestavěné komponenty. Typicky jsou obsaženy blokové paměti (Block RAM, BRAM) vhodné pro uložení

Tabulka 2.1: Přehled parametrů vybraných rodin čipů FPGA Xilinx (hodnoty jsou uvedeny pro maximální možnou konfiguraci v rámci dané rodiny); [27]

	Spartan-6	**Artix-7**	**Kintex-7**	**Virtex-7**
Počet bloků CLB	147 443	215 360	477 760	1 954 560
Velikost BlockRAM	4,8 Mb	13 Mb	34 Mb	68 Mb
Počet DSP bloků	180	740	1 920	3 600
Počet transceiverů	8	16	32	96
Rychlost transceiverů	3,2 Gb/s	6,6 Gb/s	12,5 Gb/s	28,05 Gb/s
Paměťové rozhraní	DDR3-800	DDR3-1066	DDR3-1866	DDR3-1866
Rozhraní PCI-Express	x1 Gen1	x4 Gen2	x8 Gen2	x8 Gen3
Počet I/O pinů	576	500	500	1 200

většího objemu dat či realizaci FIFO paměti, bloky pro podporu zpracování signálů (Digital Signal Processing, DSP), obvody pro připojení k externím zařízením pomocí rychlých sériových spojů (Ethernet, PCI-Express), rozhraní pro externí paměti nebo rovnou vestavěná procesorová jádra.

Mezi hlavní výrobce FPGA patří firmy Xilinx a Altera. Aktuálně dostupné rodiny čipů FPGA se liší především cílovými požadavky vyvíjené aplikaci – požadovaným výkonem, počtem programovatelných CLB bloků a dostupností vestavěných komponent a podporovaných rozhraní. Základní struktura představená dříve však zůstává zachována. Pro srovnání je uvedena tabulka 2.1 zobrazující přehled základních parametrů rodin čipů FPGA Xilinx. Všechny uvedené technologie využívají 6-vstupé tabulky LUT. Dle hodnot v tabulce poskytuje technologie Virtex-7 nejvyšší počet programovatelných logických bloků CLB a výpočetních bloků DSP, největší kapacitu blokových pamětí BRAM a maximální propustnost díky přítomným komunikačním rozhraním. Jako jediná nabízí také nejnovější rozhraní PCI-Express třetí generace pro připojení k systémové sběrnici počítače. Řada Virtex-7 je určena pro obzvláště náročné aplikace. Řady Spartan-6 a Artix-7 se naopak zaměřují na nenáročné aplikace s udržením nízké spotřeby a nízké ceny a Kintex-7 nabízí nejlepší poměr ceny a výkonu. V době psaní práce se objevila čestvá informace o vývoji nových nástupců Kintex UltraScale a Virtex UltraScale využívajících 20 nm a 16 nm výrobní technologii pro dosažení ještě větší úrovně integrace a vyššího výkonu [27].

Z důvodu požadavků na vysokou propustnost a výkon při zpracování síťových dat na vysokorychlostních sítích cílí praktická část bakalářské práce při návrhu aplikačně specifického procesoru na technologii FPGA Xilinx Virtex-7. Podrobným popisem jeho návrhu se zabývá následující kapitola 3. Postupy používané při návrhu číslicových obvodů prošly v průběhu let značným vývojem. Dříve se k návrhu využívala především grafická reprezentace obvodu pomocí logického schématu. Avšak se zvyšující se složitostí navrhovaných číslicových zařízení se tento způsob stal neúnosným, což vedlo ke vzniku specializovaných jazyků pro popis hardware (Hardware Description Language, HDL). Návrhář místo logického schématu popisuje funkci obvodu pomocí jazyka. Výhodou tohoto přístupu je, že takto popsané zařízení je možné modelovat a provádět jeho simulace a rozsáhlejší verifikace funkcionality. Syntéza (proces podobný kompilaci u jiných programovacích jazyků) umožňuje transformaci HDL popisu zařízení do prvků cílové technologie (FPGA, ASIC). V praxi běžně využívanými HDL jazyky jsou jazyky Verilog a VHDL. Verilog dominuje v USA, VHDL je používán především v Evropě. V současnosti jsou dále vyvíjeny techniky pro popis zařízení na vyšší úrovni abstrakce a jsou zaváděny další jazyky např. SystemC, CatapultC vycházející z jazyka C.

Jazyk VHDL (Very High Speed Integrated Circuit Hardware Description Language) byl původně vyvinut pro vojenské účely, později byl standardizován standardem IEEE 1076 [6] pro účely specifikace číslicových systémů. Číslicové obvody, či jejich části jsou ve VHDL popsány pomocí komponent. Každá komponenta je složena z entity a z jedné či více architektur. Entita slouží k definici rozhraní komponenty a generických parametrů. Rozhraní komponenty pro komunikaci mezi komponentou a jejím okolím je definováno pomocí signálů rozhraní – tzv. portů. Architektura popisuje funkci komponenty a je vždy svázána s její entitou.

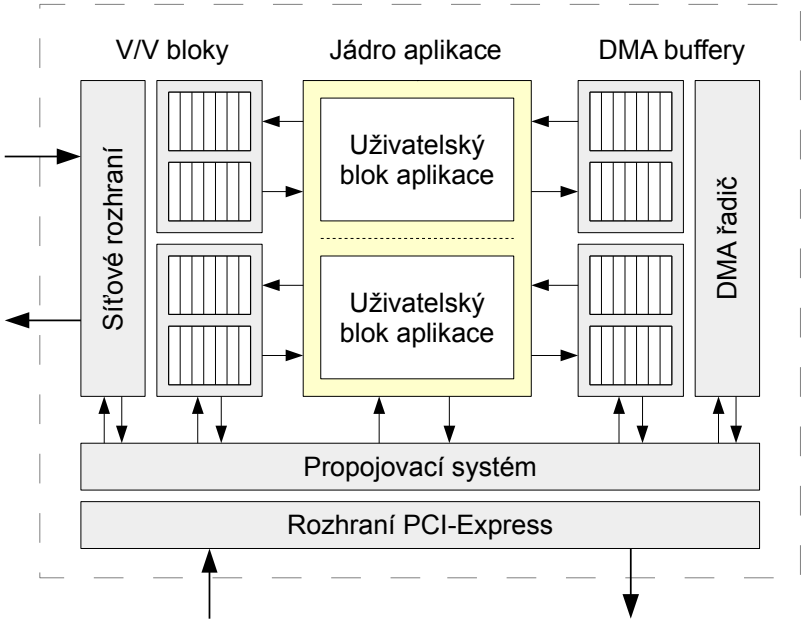
Pro popis funkce komponenty poskytuje VHDL tři základní úrovně – behaviorální, strukturní a tzv. dataflow popis. Behaviorální popis popisuje chování komponenty sadou sekvenčních příkazů s cílem vyjádřit, jak se mění výstupní signály na základě vstupních signálů a vnitřních proměnných uchovávajících stav komponenty. Strukturní popis určuje, z jakých podkomponent je komponenta složena. Porty dílčích komponenty jsou pak spojovány pomocí signálů. Dataflow popisuje datové závislosti a umožňuje zápis chování zkráceně pomocí paralelních příkazů.

Návrh číslicového obvodu končí vytvořením konfigurace FPGA. Na základě popisu obvodu v programovacím jazyce je při procesu syntézy a implementace vytvořena konfigurace (nazývána též jako bitstream), kterou je možné nahrát do FPGA a to pak realizuje obvod popsaný HDL jazykem. Bitstream určuje funkci každého CLB, jejich vzájemné propojení a výchozí stavy paměťových prvků. Konfigurace je uložena v paměti SRAM a po odpojení napájení je tak nutné ji znovu nahrát. Konfigurace je nejčasněji prováděna z externí paměti hned po připojení napájecího napětí FPGA čipu skrze sériové rozhraní např. SPI nebo JTAG.

Jednou z možností zjednodušení návrhového procesu složitých číslicových obvodů je využití principu znovupoužitelnosti. V oblasti návrhu hardware se nabízí možnost využití tzv. IP jader (Intellectual Property Cores). Jde o obdobu softwarových knihoven. Tato jádra jsou poskytována samotnými výrobci čipů FPGA, ale také jinými společnostmi třeba v podobě otevřeného zdrojového kódu. Příkladem nechť je platforma NetCOPE (Network COMBO Pipe) [15]. NetCOPE je modulární vývojové prostředí poskytující infrastrukturu, řadu knihovních komponent a softwarových nástrojů pro podporu a urychlení vývoje síťových aplikací na akceleračních kartách využívajících technologii FPGA. Původně bylo navrženo sdružením CESNET [1] pro rodinu akceleračních FPGA karet COMBO. Platforma byla však přenesena také na akcelerační kartu NetFPGA-10G vyvinutou na Standford University [12]. Sdružení CESNET je poskytovatelem síťové infrastruktury pro akademické instituce v ČR. Mimo jiné se rovněž podílí na vývoji akceleračních karet s FPGA pro vysokorychlostní sítě a vývoji představené platformy [14].

NetCOPE si klade za cíl odstínit návrháře uživatelské aplikace od nízkoúrovňové problematiky práce s jednotlivými hardwarovými prvky konkrétní síťové karty pro kterou je aplikace vyvíjena. Návrhář uživatelské aplikace se tak nemusí soustředit na komplikovanou implementaci řady komunikačních rozhraní. Platforma navíc využívá vždy stejné komunikační rozhraní pro přenos dat mezi uživatelskou aplikací a infrastrukturou NetCOPE na různých akceleračních kartách, takže umožňuje snadný přenos uživatelské aplikace.

Základní firmwarové prvky modulární architektury NetCOPE jsou zobrazeny na obrázku 2.7. Mezi klíčové prvky architektury patří vstupní a výstupní síťové bloky pro podporu příjmu a vysílání dat na síťových rozhraních Ethernet (vlevo na obrázku). Dále sem patří propojovací systém sběrnic pro přenos dat mezi uživatelskou aplikací a infrastrukturou platformy NetCOPE a zajištění komunikace po sběrnici PCI-Express (v dolní části obrázku). Nedílnou součástí jsou kruhové vyrovnávací buffery a řadič přímého přístupu do



Obrázek 2.7: Blokové schéma modulární architektury platformy NetCOPE

operační paměti (Direct Memory Access, DMA) umožňující rychlé přenosy dat do paměti počítače (vpravo na obrázku) spolu s řadou softwarových nástrojů a knihoven pro jejich správu.

Dále jsou poskytovány základní komponenty pro tvorbu uživatelské aplikace např. implementace pamětí FIFO (First In First Out), pamětí CAM (Content-addressable Memory), komponent pro práci s komunikačními protokoly infrastruktury pro přenos dat a paměťovým rozhraním, bloků pro zpracování síťového provozu (extrakce záhlaví paketů, výpočet kontrolního součtu CRC) nebo komponent k řízení prvků mimo čip FPGA např. pro komunikaci s modulem pro příjem přesných časových značek ze systému GPS nebo komunikaci s externí pamětí.

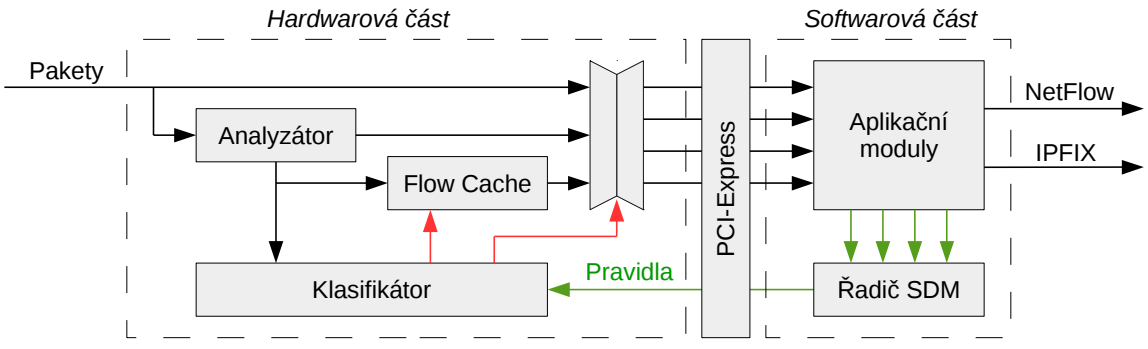
2.5 Koncept SDM

Text následující sekce je založen na diplomové práci [10] a poznatcích publikovaných na konferenci INFOCOM [11]. Model standardně používaný a rozšířený pro monitorování sítí s přenosovými rychlostmi do 10 Gb/s využívá prostředků klasické serverové platformy. Dále se používá běžná síťová karta, která provádí zachycení celých síťových paketů, jejich přenos do operační paměti hostitelského počítače a distribuci na procesorová jádra, kde následně probíhá jejich plně softwarové zpracování. Uvedený přístup však není aplikovatelný na sítě vyšších rychlostí z důvodu dvou významných výkonnostních omezení. Limitujícím faktorem je jak propustnost systémové sběrnice PCI-Express, tak nedostatečný výpočetní výkon obecného procesoru a softwarového zpracování. Nový koncept softwarově definovaného monitorování (Software Defined Monitoring, SDM) přichází s myšlenkou základního softwarem řízeného hardwarového předzpracování síťových dat na hardwarové akcelerační síťové kartě. Softwarově jsou pak prováděny až související pokročilé a komplexnější úlohy, které lze v hardwaru realizovat jen velmi obtížně nebo vůbec.

Princip SDM vychází z poznatku, že ve většině případů jen malá část dat přenášených pakety obsahuje informace nezbytné pro potřeby aplikací, které zajišťují bezpečnost monitorovaných sítí. Zpravidla jde o údaje obsažené v záhlavích paketů. Je tedy možné provést jejich extrakci přímo na hardwarové akcelerační síťové kartě a přes systémovou sběrnici přenášet daleko menší množství (pouze užitečných) dat. Zároveň lze tímto způsobem výrazně snížit zátěž procesoru spojenou s analýzou paketů a extrakcí položek z jejich záhlaví. Dalším logickým krokem může být provádění agregace a výpočtu např. NetFlow statistik k celému síťovému toku přímo v hardwaru. Na druhé straně přímá analýza vlastních dat zachycených paketů je základem řady bezpečnostních aplikací. Z tohoto důvodu je vhodné zajistit určitou specifickou úroveň předzpracování dat pro různé případy síťových toků na základě konkrétních požadavků monitorovacích aplikací.

Koncepce SDM nabízí softwarem řízenou na úrovni síťových toků kontrolovanou ztrátu příchozích informací. Celý systém pracuje tak, že několik prvních paketů nového, neznámého síťového toku je ve výchozím stavu odesláno k softwarovému zpracování. Softwarový řadič dále na základě požadavků běžící bezpečnostní aplikace zpracující příchozí data rozhodne, jaký způsob hardwarového předzpracování bude zvolen pro následující pakety tohoto síťového toku a zavede do hardwaru příslušné pravidlo. Následující příchozí pakety uvedeného toku jsou pak do softwaru odesílány již patřičně předzpracované – ve formě extrahovaných informací ze záhlaví paketů nebo agregovaných např. NetFlow statistik k celému toku. V případě nutnosti detailnější analýzy příchozích síťových dat bezpečnostní aplikací lze zvolit i nadále odesílání celých paketů k plně softwarovému zpracování.

Spolehlivost uvedeného mechanismu silně závisí na době trvání rozhodnutí o způsobu zpracování daného síťového toku, která je určena časem příchodu prvního paketu toku a časem následné aktivace pravidla v hardware. Dalším omezujícím faktorem může být omezená kapacita hardwarové paměti pravidel a paměti flow cache. Podle analýzy síťových toků na reálné síti uvedené v odkazované literatuře bylo ověřeno, že i rozhodnutí v řádu desítek milisekund nemá výrazný negativní dopad na výkonnost systému a lze tímto přístupem pokrýt a v hardwaru tak zpracovat 80 % až 90 % všech příchozích paketů. Měření dále ukazují, že síťový provoz vykazuje tzv. heavy-tailed rozložení velikosti toků. To znamená, že i nepatrný podíl nejtěžších toků (toků přenášejících největší objemy dat) obsahuje vysoké množství všech přijímaných paketů. Promile největších toků pokrývá přes polovinu všech paketů, jedno procento toků pak pokrývá až 85 % z nich. Ani omezená kapacita hardwarové paměti tak nemusí mít významný dopad na účinnost použitého mechanismu.



Obrázek 2.8: Základní konceptuální pohled na systém založený na SDM

Základní pohled na celý systém založený na konceptu SDM nabízí obrázek 2.8. Jak je vidět na obrázku, systém se skládá ze dvou částí – hardwarové a softwarové, komunikujících

spolu prostřednictvím sběrnice PCI-Express. Datové cesty jsou zakresleny černými šipkami, řídící jsou provedeny barevně. Zpracování všech příchozích paketů začíná jejich analýzou a extrakcí (pro bezpečnostní aplikace zajímavých) informací z jejich záhlaví (blok *Analyzátor*). Extrahovaná data slouží ke klasifikaci paketu na základě softwarem definovaných pravidel (blok *Klasifikátor*). Každé pravidlo určuje, jaká operace má být provedena s příchozími pakety daného toku. Udává, jak způsob hardwarového předzpracování dat, tak určuje některý z nezávislých logických kanálů pro jejich přenos do softwaru. Hardware zajišťuje odesílání paketů do softwaru beze změny nebo ve formě již extrahovaných dat. Umožňuje také jejich agregaci do podoby záznamů o tocích (blok *Flow Cache*) a jejich pravidelný export do software. Data do softwaru jsou přenášena metodou přímého přístupu do paměti (Direct Memory Access, DMA). Uživatelské aplikace (blok *Aplikační moduly*) mají přístup k doručeným datům, ať už přišla ve formě celých paketů, extrahovaných dat ze záhlaví nebo jako agregované záznamy o tocích a vykonávají uživatelem definované úlohy (exportování NetFlow, IPFIX). Směrem k řadiči dávají najevo svůj zájem, respektive nezájem, o přijímaná data jednotlivých toků. Úkolem řadiče je tyto požadavky vhodně vyhodnotit a vytvořit a instalovat na jejich základě pravidla pro předzpracování toků v hardware s cílem dosažení co největší redukce objemu dat ze sítě a urychlení jejich softwarového zpracování.

Kapitola 3

Návrh firmwaru

Hardwarová část systému vycházejícího z konceptu SDM bude realizována akcelerační kartou s čipem FPGA Xilinx Virtex-7 a s externí pamětí QDR (Quad Data Rate) pro uložení klasifikačních pravidel a záznamů flow cache. Návrh a následná implementace jejího firmwaru tvoří stěžejní část bakalářské práce. Z důvodu usnadnění celého vývoje a zajištění určité míry přenostitelnosti počítá návrh s využitím už dříve představené platformy NetCOPE. Jádro firmwaru pro FPGA pak bude tvořeno vlastní implementací aplikačně specifického procesoru. Jeho návrh je zaměřen na identifikaci požadavků na systém, vhodné rozdělení do jednotlivých funkčních jednotek a způsob jejich zapojení do celé architektury vyvíjeného systému a platformy NetCOPE. Architektura procesoru vychází ze základního návrhu firmwarové části akceleračního systému uvedeného v [10]. Uvedený návrh je v rámci řešení bakalářské práce dále upřesněn a rozšířen.

3.1 Požadavky na realizaci firmwaru

Pro navrhovaný systém pro monitorování vysokorychlostních sítí založený na konceptu SDM lze identifikovat dva základní a velmi klíčové požadavky na cílovou realizaci:

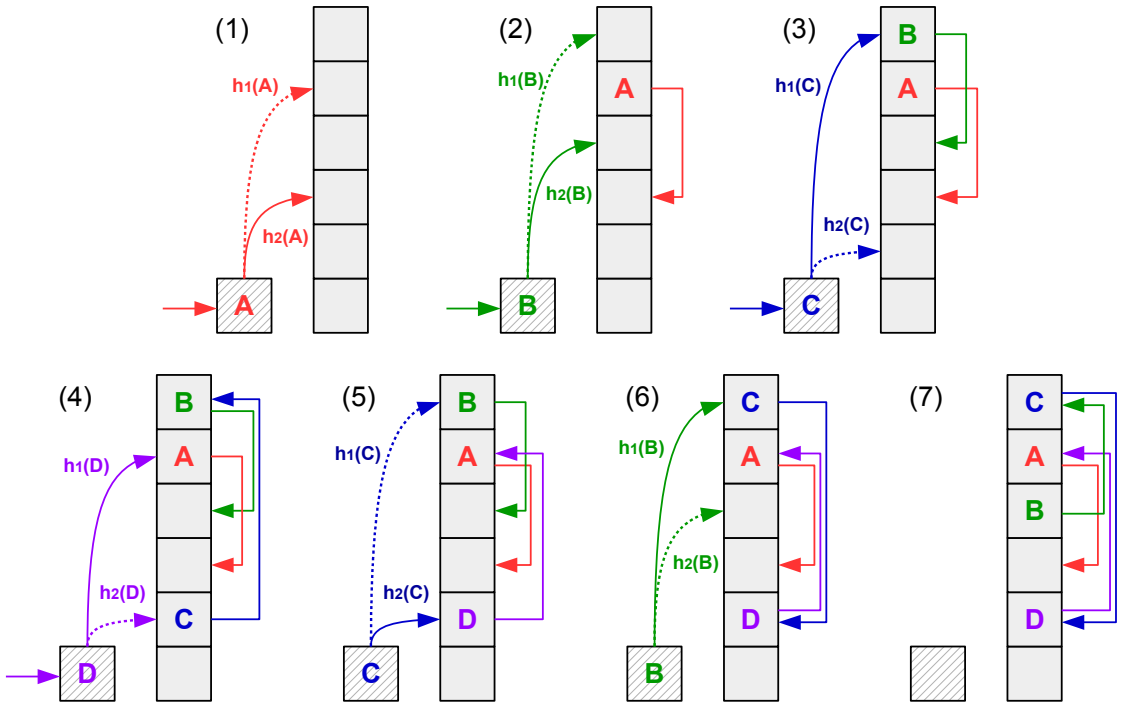
Vysoká propustnost. Vysokou propustností se rozumí podpora zpracování příchozích síťových dat na plné rychlosti připojených linek a to až do celkové rychlosti 100 Gb/s. Pro její dosažení je velmi důležitý návrh datových cest. Z důvodu dodržení vhodné (nízké) pracovní frekvence designu FPGA (100–200 MHz) je tak nutné volit velmi široké datové cesty (512–1024 bitů).

Správa a přístup k paměti. Při klasifikaci paketů se pracuje s velkým množstvím dynamicky se měnících pravidel pro síťové toky. Správa a přístup k paměti patří mezi kritické části návrhu, ať už z důvodu potřeby vysoké kapacity paměti (a nutnosti využití externí paměti), ale také počtu a frekvence změn a prováděných vyhledání. Zpracovávání nejkratších ethernetových rámců s velikostí 64 bytů při uvedené šířce datové cesty znamená příchod paketu v každém taktu hodinového signálu a s tím také zahájení nové klasifikace s každým hodinovým taktem. Současně musí být možné provádět atomické přidávání a odebírání pravidel bez výrazného omezení propustnosti procesu souběžně probíhající klasifikace.

Na efektivitu využití datové sběrnice má výrazný vliv způsob zarovnání dat. Vhodnou metodou pro její zvýšení je využití přenosového protokolu a technik umožňující sdílení datových slov sběrnice a částečného zarovnání začátku přenášených dat. Platforma NetCOPE

standardně využívá obou technik v rámci protokolů interní komunikační infrastruktury pro přenos dat a poskytuje také řadu komponent pro práci s nimi.

Řešením problematiky správy paměti klasifikačních pravidel může být použití určité formy hašovací tabulky minimalizující při vyhledávání počet potřebných přístupů do paměti. Nevýhodou hašovacích tabulek je však neefektivní využití kapacity paměti z důvodu vzájemných kolizí. Jistým způsobem řešení tohoto problému může být použití metody tzv. kukaččího hašování [17], kdy se do paměti přistupuje přímo na pozice dané hašovacími funkcemi a kolize při vkládání položek jsou řešeny vhodnou částečnou reorganizací položek vyhledávací tabulky. V případě využití tohoto řešení by ukládané záznamy v tabulce měly tvar dvojice – klíč (identifikátor síťového toku), hodnota (pravidlo udávající způsob předzpracování toku). K identifikaci toku se však nejčastěji používá klasická pětice obsahující mimo jiné také zdrojovou a cílovou IP adresu. S nezbytnou podporou IPv6 adres je to délka klíče téměř 300 bitů, což představuje značné nároky na kapacitu paměti. Není však nutné ukládat do paměti celý klíč, postačí pouze jeho otisk. Tento přístup v kombinaci s kukaččím hašováním ale vyžaduje kromě otisku klíče a pravidla uložení také adresy alternativního umístění položky v paměti z důvodu potřeby reorganizace tabulky při vkládání položek.



Obrázek 3.1: Postup při vkládání položek do vyhledávací tabulky

Navržený postup vkládání položek do vyhledávací tabulky využitelný pro klasifikaci znázorňuje obrázek 3.1. Kromě vlastní paměti pro uložení položek (šedě vyplněná oblast) je použit pomocný registr (označen šrafováním), který se využívá k odkládání položek při reorganizaci paměti. Proces je rozdělen do fází (1)–(7) a ukazuje postupné vložení položek A, B, C a D do paměti:

1. Vkládání záznamu A začíná jeho umístěním do pomocného registru. Záznam je možné vložit na místa v paměti určená hodnotou hašovacích funkcí $h_1(A)$ a $h_2(A)$. U žádné

z možností nedochází ke kolizi. Záznam je v tomto případě úspěšně vložen na pozici danou hodnotou $h_1(A)$ (označeno přerušovaně).

2. Příklad pokračuje vkládáním záznamu B. Ani v tomto případě nedochází ke kolizi a záznam je možné úspěšně vložit na pozici $h_1(B)$. Důležité je uvědomit si, že dříve vložený záznam A v paměti obsahuje adresu možného alternativního umístění určeného druhou hašovací funkcí (znázorněno červenou šipkou vpravo).

3. Při vkládání záznamu C dochází ke kolizi na pozici $h_1(C)$, nicméně druhá z pozic je volná a vložení je tak opět úspěšné.

4. Při vkládání záznamu D však vzniká kolize na obou pozicích, kterou je možné vyřešit změnou uspořádání položek v tabulce. Reorganizace záznamů je zahájena výběrem oběti – položky na pozici $h_1(D)$ nebo $h_2(D)$. V tomto příkladě je zvolen záznam C na pozici $h_2(D)$ (označeno přerušovaně). Záznam C je z tabulky vyjmut a umístěn do pomocného registru a na uvolněné místo je vložen nový záznam D.

5. Reorganizace tabulky pokračuje snahou o umístění vyňatého záznamu C (obdobně jako při vkládání nové položky). Záznam je možné umístit na pozice $h_1(C)$ a $h_2(C)$. Je nutné si uvědomit, že hodnoty hašovacích funkcí nemohly být určeny na základě výpočtu, neboť do tabulky se vždy ukládá pouze otisk původního klíče. Hodnota $h_1(C)$ byla součástí vyňatého záznamu a $h_2(C)$ je dána pozicí, kde se záznam v tabulce původně nacházel. Obě možné pozice pro umístění záznamu C jsou obsazeny a kolize se opět řeší změnou uspořádání tabulky. V tomto případě je potřeba zvolit jako oběť záznam B na pozici $h_1(C)$, neboť volba druhé z možností by nás vrátila do předchozí situace. Záznam B je umístěn do pomocného registru a na jeho místo je vložen záznam C (označeno přerušovaně).

6. Cílem je teď umístění záznamu B z pomocného registru do paměti. Součástí čerstvě přemístěného záznamu C je opět odkaz na jeho alternativní pozici, kde byl také původně umístěn (znázorněno modrou šipkou). Záznam B je vložen na volnou pozici $h_2(B)$. Ke kolizi nedošlo.

7. Reorganizace tabulky končí úspěšně. Situace znázorňuje stav paměti po vložení záznamů A, B, C a D. Pomocný registr je prázný a lze zahájit vkládání další položky zápisem do něj.

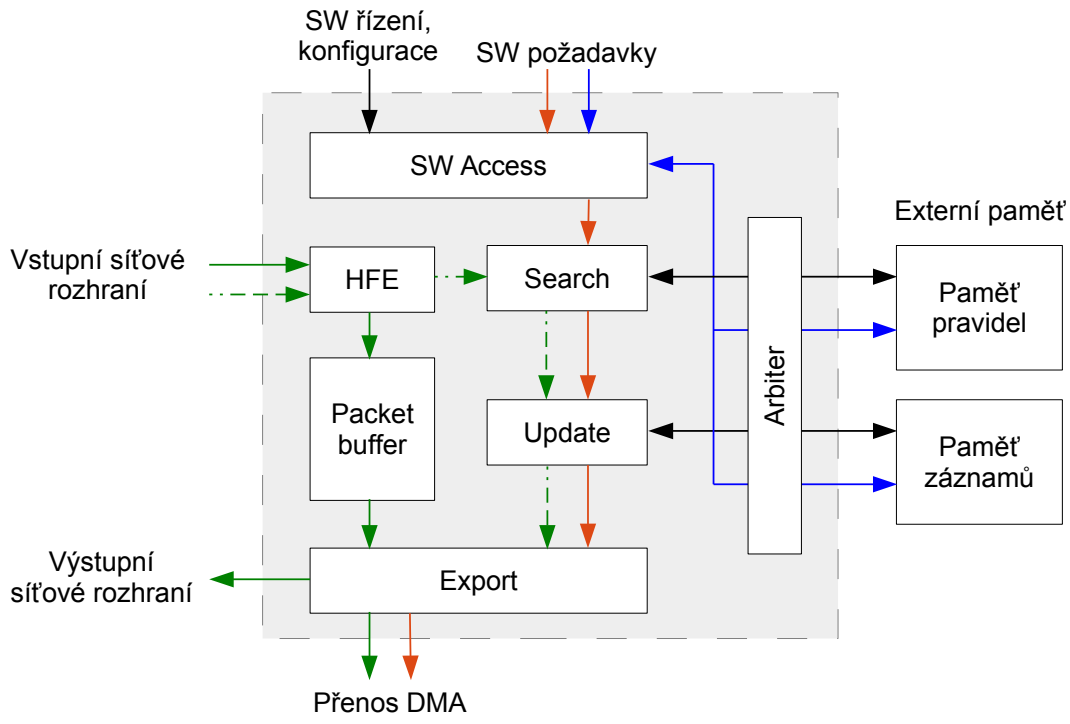
Situace z fáze (5) se může několikanásobně opakovat, než dojde k dosažení cílového stavu. Může nastat i situace, kdy reorganizace tabulky bude probíhat v cyklu, dokud nebude některý záznam z paměti odstraněn. Vyhledávání položky se pak provádí tak, že se přistoupí paralelně na obě možné pozice jejího umístění v paměti a probíhá porovnání na shodu s uloženými otisky klíče. Porovnání na shodu se realizuje i se záznamem umístěným v pomocném registru. Vkládání záznamů spolu s reorganizací tabulky a vyjmutím položky z paměti tak neomezuje souběžně probíhající proces vyhledávání. Položka se vždy nachází buď v paměti nebo pomocném registru. Existuje i obecná varianta využívající libovolný vyšší počet hašovacích funkcí a vykazující vyšší efektivitu využití paměti. Z hlediska hardwarové realizace je však jednodušší varianta se dvěma hašovacími funkcemi. Při jejich vyšším počtu je totiž nutné řešit souběžný vícenásobný přístup do paměti.

Mezi další méně významné požadavky realizace patří především ty na funkcionalitu firmwaru. Jde například o možnost dělení a distribuce vstupního datového toku mezi několik

softwarových rozhraní umožňující nezávislé zpracování na více jádrech procesoru hostitelského počítače. Řešení by mělo být konfigurovatelné a umožňovat rozdělování do softwarových kanálů i na základě vybraných položek záhlaví paketů, tak aby všechny pakety stejného síťového toky mohly být zpracovány jedním jádrem. Pro zajištění kvalitního měření síťových dat je také důležité přidělování přesných časových značek s přesností v řádu nanosekund. Jako vhodná se jeví také možnost konfigurace položek ze záhlaví paketů, které budou sloužit k identifikaci toku, či podpora směrování přijatého paketu nejen do jednoho ze softwarových kanálů (v původní nebo zkrácené verzi), ale také zpět na linku na některé z výstupních síťových rozhraní.

3.2 Architektura procesoru

Zpracování všech vstupních požadavků probíhá zřetězeně za účelem dosažení vysoké propustnosti. Činnost procesoru je řízena instrukcemi obsaženými v pravidelech uložených v externí paměti a příslušejících jednotlivým síťovým tokům. Instrukce určuje, jaká operace má být provedena s každým příchozím paketem daného síťového toku. Každé pravidlo tak udává způsob hardwarového předzpracování dat ze vstupního síťového rozhraní. Po provedení instrukce jsou data předávána softwarové vrstvě buď ve formě původního paketu, jeho zkrácené verze, extrahovaných informací ze záhlaví paketu nebo agregovaných statistik o daném síťovém toku. Schéma architektury a členění navrženého procesoru do sedmi základních bloků je zobrazeno na obrázku 3.2. Šedá ohraničená oblast znázorňuje já-



Obrázek 3.2: Schéma architektury navrženého procesoru

dro firmwaru FPGA – vlastní aplikačně specifický procesor zasazený do prostředí platformy NetCOPE. Procesor zpracovává dva druhy požadavků – požadavky vzniklé na základě příchodu paketu na vstupním síťovém rozhraní (zeleně znázorněná datová cesta) a požadavky

softwarového radiče (červeně a modře znázorněná datová cesta). Tato a další komunikace navrženého procesoru s prostředím NetCOPE probíhá pomocí řady rozhraní:

Vstupní síťové rozhraní. V rámci NetCOPE a vstupních síťových bloků je nad příchozími ethernetovými rámci prováděna řada kontrol (dodržení limitů velikosti rámce, správnost hodnoty pole FCS). Kromě vlastních dat přenášených příchozími rámci jsou na tomto rozhraní předávány přidružené přídavné informace o přijímaných datech (přesná časová značka příchodu rámce, identifikace fyzického vstupního rozhraní, velikost příchozích dat). Rozhraní tvoří primární zdroj požadavků pro zpracování procesorem.

Řídicí a konfigurační rozhraní platformy NetCOPE. Rozhraní je součástí propojovacího systému sběrnic platformy NetCOPE. Slouží ke čtení a zápisu hodnot konfiguračních a stavových registrů komponent v rámci platformy. Prostřednictvím tohoto rozhraní budou také přicházet požadavky softwarového řadiče systému.

Výstupní síťové rozhraní. Rozhraní slouží k odesílání ethernetových rámců do sítě. Samotná data jsou v rámci výstupního síťového bloku v režii NetCOPE v případě potřeby rozšířena na minimální potřebnou délku, opatřena kontrolním součtem FCS a odeslána na specifikované fyzické síťové rozhraní.

Rozhraní pro přenosy DMA. Jednotné rozhraní slouží k přenosu dat do operační paměti počítače prostřednictvím řadiče DMA a systémové sběrnice PCI-Express. Kromě určení DMA kanálu pro přenos a samotných dat je nutné specifikovat také jejich přesnou velikost a identifikaci formátu. Řadič DMA vkládá před vlastní data určitý typ záhlaví obsahující tyto informace a umožňující jejich odpovídající zpracování softwarovým nástrojem.

Rozhraní externí paměti QDR. Do externí paměti QDR budou ukládána pravidla pro zpracování síťového provozu a agregované záznamy a statistiky o síťových tocích. Jednotné rozhraní poskytuje několik paralelních čtecích a zápisových slotů a slouží pro přístup k těmto položkám v externí paměti. V rámci platformy NetCOPE poskytuje odstínění od řadiče konkrétní paměti umístěné na akcelerační kartě.

Počáteční zpracování všech příchozích paketů ze vstupního síťového rozhraní (zeleně znázorněná datová cesta) probíhá v jednotce *HFE (Header Field Extractor)*. Začíná analýzou a extrakcí jejich záhlaví a synchronizací s dodatečnými informacemi o přijímaných datech ze vstupních síťových bloků. Původní přijatý paket je dočasně odložen do FIFO paměti. Extrahovaná data jsou v jednotném formátu předána k dalšímu zpracování (přerušovaná, zeleně znázorněná datová cesta). Jednotka *Search* dále podle identifikace síťového toku vyhledá příslušné pravidlo v externí paměti. Je-li pravidlo nalezeno, je předáno spolu s extrahovanými daty jednotce *Update*, kde je provedena operace dle obsažené instrukce – aktualizace záznamu o příslušném síťovém toku v externí paměti (adresa záznamu je součástí obdrženého pravidla). Blok *Export* dále zajistí vyzvednutí přijatého paketu z vyrovnávací paměti, jeho další zpracování (zkrácení, zahození) a případně odeslání paketu nebo extrahovaných dat do softwaru současně s volitelným odesláním jeho původní nezkrácené verze na některé z výstupních síťových rozhraní (dle příslušného pravidla). V případě, že není nalezeno odpovídající pravidlo, se provede výchozí způsob zpracování (všechny neznámé příchozí pakety jsou ve výchozím stavu odesílány do softwaru k další analýze). Odesílání dat do softwaru se realizuje přímým přístupem do operační paměti (DMA).

Paměť pravidel je spravována softwarovým řadičem skrze jednotku *SW Access* (modře označená datová cesta). Softwarový přístup k paměti záznamů slouží především k počáteční inicializaci záznamu a pro ladicí účely. V případě požadavku na exportování záznamu o síťovém toku do softwaru nebo exportování záznamu se současným vynulováním záznamu či odstraněním pravidla (červeně znázorněná cesta) je tento požadavek předán jednotce *Search*, která na základě obsažených informací o síťovém toku vyhledá a případně odstraní pravidlo. Součástí pravidla je také adresa požadovaného záznamu, ta je předána jednotce *Update*, která zajistí načtení a případné vynulování záznamu. Záznam je následně odeslán jednotkou *Export* do kanálu DMA.

Detailnějšímu popisu vlastností, funkcí a shrnutí problematiky jednotlivých jednotek zobrazených na schématu z obrázku 3.2 se věnuje následující text:

SW (Software) Access. Jednotka realizuje veškeré přístupy ze softwarového řadiče. Ve vlastní režii provádí vkládání záznamů pravidel do externí paměti. Vkládání probíhá mechanismem kukaččího hašování. V případě reorganizace položek v paměti pravidel musí být přeskládávaná (z paměti vyjmutá) položka v pomocném registru dostupná pro jednotku *Search*, tak aby při vkládání záznamu nebyl omezen probíhající proces vyhledávání pravidel. Požadavky na exportování záznamů o tocích či smazání pravidel předává *SW Access* jednotce *Search*. Pro ladicí účely *SW Access* dále poskytuje možnost vyčtení nebo zápisu z/na libovolnou adresu externí paměti. Mimoto umožňuje také softwarový přístup ke stavovým a konfiguračním registrům celého procesoru.

HFE (Header Field Extractor). Jednotka zajišťuje analýzu a extrakci položek ze záhlaví protokolů vstupních síťových dat. Obzvláště významná je pak extrakce informací identifikujících síťový tok. Jednotka tak musí minimálně zajistit rozbalení dat z ethernetového rámce a být schopná zpracovat síťové protokoly IPv4 a IPv6 (pro extrakci IP adres a identifikace protokolu) a transportní protokoly TCP a UPD (pro určení portů). K identifikaci síťového toku slouží právě pětice tvořená zdrojovou a cílovou IP adresou, zdrojovým a cílovým portem a typem transportního protokolu doplněná o identifikátor vstupního síťového rozhraní. Takto extrahovaná data (obsahující i další údaje např. informace o fragmentaci či TCP příznacích) jsou v jednotném formátu spolu s přídavnými informacemi ze vstupních síťových bloků (časová značka, identifikace vstupního síťového rozhraní, velikost příchozích dat) předána k dalšímu zpracování. Původní přijatý paket je vždy vložen do vyrovnávací FIFO paměti v bloku *Packet Buffer*.

Search. Základní činností této jednotky je vyhledání pravidla v externí paměti příslušejícího k příchozímu paketu. Jednotka čte dva záznamy z paměti pravidel určené výpočtem hodnot hašovacích funkcí na základě extrahovaných položek předaných jednotkou *HFE*. Pro výpočet jsou vybrána jen určitá pole identifikující síťový tok dle aktuální konfigurace systému. Dále kontroluje stavový bit vyčtených záznamů a porovnává uložené otisky klíče s identifikátorem toku. Kromě toho musí také ověřit, zda hledané pravidlo není právě přesouváno jednotkou *SW Access* (a proto nebylo nalezeno v externí paměti). V případě shody předává příslušné vyčtené pravidlo spolu s extrahovanými informacemi z paketu jednotce *Update* k dalšímu zpracování. V případě, že odpovídající pravidlo není nalezeno je předána informace o výchozím způsobu zpracování paketu. Mimoto přijímá jednotka *Search* také požadavky od *SW Access* na odstranění pravidla nebo požadavky na exportování záznamů o tocích. Zneplatnění pravidla je umožněno vynulováním stavového bitu záznamu v paměti a je nutné jej

provádět tak, aby byla zajištěna atomicita této operace s ohledem na souběžně probíhající zpracování požadavků příchozích paketů. Požadavky na export záznamů jsou předávány jednotce *Update*.

Update. Úlohou jednotky *Update* je správa záznamů o tocích v externí paměti spočívající v aktualizaci hodnot příslušného záznamu na základě obdržených informací o příchozím paketu a k němu přiřazeného pravidla. Adresa záznamu je součástí obdrženého pravidla spolu s identifikací jeho typu a samotné operace nad ním. Návrh je proveden s ohledem na budoucí rozšiřitelnost, tak aby jednotka kromě základních NetFlow statistik umožňovala přidávání podpory dalších typů záznamů o tocích pro další (dosud neznámé) monitorovací metody. Dle typu záznamu se o jeho aktualizaci stará některý z aktualizačních modulů. Základní NetFlow modul bude například provádět zvýšení čítačů paketů a bytů, aktualizaci hodnoty počáteční a koncové časové značky a logického součtu TCP příznaků. Pro vytváření nových aktualizačních modulů lze do budoucna počítat s využitím techniky HLS (High Level Synthesis), která urychlí jejich vývoj díky možnosti popisu na vyšší úrovni abstrakce ve formě jazyka C/C++. V rámci této jednotky je třeba řešit také problematiku přístupu do paměti QDR. Jakoukoliv modifikaci dat je nutné provádět ve dvou fázích (čtení a následný zápis), přičemž každá operace přístupu do externí paměti má určitou latenci a požadavky jsou z důvodu vysoké propustnosti zpracovávány zřetězeně. Pro zajištění atomicity těchto operací musí jednotka podporovat mechanismus blokování, respektive rezervace, záznamů v externí paměti. V případě výchozího způsobu zpracování paketu se neprovádí počítání statistik o síťovém toku. Jednotka pak vůbec nepřistupuje k paměti a obdržené informace pouze předá další jednotce k exportování do softwaru. Kromě operací spojených s aktualizací záznamu podporuje *Update* také jeho exportování na základě požadavku softwarového řadiče předaného skrze jednotku *Search*. Součástí exportu je volitelný příznak, který způsobí zahození tohoto požadavku v případě, že čítače záznamu jsou nulové. Další z možností je naopak jejich vynulování v paměti po úspěšném exportování záznamu.

Packet Buffer. Jednotka je tvořena vyrovnávací FIFO pamětí. Slouží k odložení a uchování přijatého paketu do doby, než bude rozhodnuto o jeho následném způsobu zpracování. Z tohoto důvodu je velmi důležité, aby zůstalo zachováno pořadí zpracování požadavků vzniklých na základě příchodu paketu na vstupním síťovém rozhraní a procházejících procesní linkou tvořenou jednotkami *Search* a *Update*, neboť následně bude nutné provádět jejich opětovnou synchronizaci s původně přijatými pakety připravenými ke konkrétnímu způsobu zpracování.

Export. Jednotka přijímá od *Update* pravidlo obsahující akci určující způsob zpracování paketu a informace extrahované z jeho záhlaví. Zajišťuje jejich synchronizaci s původním paketem z vyrovnávací FIFO paměti a provádí různé způsoby jeho zpracování a odeslání dat do softwaru. Mezi uvažované možnosti patří odeslání celého nebo zkráceného paketu či extrahovaných dat ze záhlaví do softwaru nebo jeho úplné zahození (vhodné především v kombinaci s počítáním statistik o síťovém toku při plném hardwarově prováděném měření). Souběžně lze provést odeslání nezkrácené verze paketu na některé z výstupních síťových rozhraní. V případě exportování záznamu o tocích na základě požadavku softwarového řadiče nedochází k synchronizaci s paketem ve vyrovnávací paměti, ani není možné provést odeslání těchto dat na výstupní síťové rozhraní. Data jsou tak vždy odeslána do softwaru. Činnost jednotky je řízena na

základě údajů obsažených v pravidle k paketu, případně v požadavku softwarového řadiče. Jejich součástí je také informace o způsobu určení konkrétního DMA kanálu pro odeslání dat do softwaru. Mezi základní možnosti patří výpočet hodnoty hašovací funkce na základě polí identifikujících síťový tok nebo určení na základě principu round-robin. Jednotka umožňuje také další omezení distribuce na určitý rozsah DMA kanálů.

Arbiter. Cílem bloku je řízení a synchronizace souběžného přístupu více jednotek do externí paměti, konkrétně jak do paměti pravidel, tak do paměti záznamů o tocích. Kromě toho musí zajistit nezávislost a atomicitu jednotlivých operací s ohledem na zřetězené zpracování požadavků a nutnost provádění modifikace ve dvou krocích (nejprve čtení, pak zápis). Dalším z úkolů je také správné směrování vyčtených dat k jednotce, která o ně požádala.

3.3 Podrobný implementační návrh

Následující sekce dále upřesňuje a rozšiřuje koncept návrhu a architektury procesoru představený výše. Součástí podrobněji provedeného návrhu, který bude sloužit především pro potřeby implementace systému, je například specifikace adresového prostoru procesoru a registrů pro komunikaci se softwarovým řadičem nebo definice formátu klasifikačních pravidel a záznamů o síťových tocích ukládaných do externí paměti.

Adresový prostor procesoru je organizován do bloků po 32 bitech. Toto uspořádání vychází z použití řídicího a konfiguračního rozhraní platformy NetCOPE, které používá šířku datové sběrnice právě 32 bitů a umožňuje čtení a zápis po 32 bitových blocích. Adresy stavových a konfiguračních registrů adresového prostoru procesoru ukazuje tabulka 3.1. Přesné adresy registrů závisí na umístění procesoru v adresovém prostoru infrastruktury NetCOPE a na volbě jeho bázové adresy. Při přístupu ke konfiguraci procesoru ze softwarových nástrojů se tak konkrétní adresy mohou lišit. Tabulka udává kromě názvu a krátkého popisu každého registru také povolený mód přístupu – R (Read) pro čtení a W (Write) pro zápis.

Tabulka 3.1: Adresový prostor aplikačně specifického procesoru

Adresa	Mód	Registr	Popis
0x00	R	Version	verze implementace
0x04	R	Config	generická konfigurace použitá při syntéze
0x0C	R	State	signalizace aktuálního stavu jednotky
0x10	W	Cmd	registr pro zadávání příkazů ze softwarového řadiče
0x14	RW	Address	adresa pro do externí paměti (pro ladicí účely)
0x18	RW	Seed1	parametrizace první hašovací funkce
0x1C	RW	Seed2	parametrizace druhé hašovací funkce
0x20	RW	Seed3	parametrizace funkce pro výpočet otisku klíče
0x28	RW	Mask1	maska při výpočtu otisku klíče
0x2C	RW	Mask2	maska při výpočtu DMA kanálu
0x38	RW	Data	registr pro předávání parametrů příkazů

Registry *Version* a *Config* najdou využití především v budoucnosti, v případě dalšího vývoje systému, kdy mohou vzniknout jeho další verze. Budou sloužit pro potřeby softwarového řadiče pro identifikaci verze a přesného nastavení generických parametrů použitých

při syntéze firmwaru, se kterým řadič pracuje. Nejvýznamnějšími z pohledu řízení činnosti procesoru jsou registry *State* a *Cmd*. Registr *State* poskytuje informace o aktuálním stavu a zaneprázdněnosti jednotky. Obsahuje například také informaci, zda se podařilo zkalibrovat a inicializovat externí paměť QDR. Další příznaky signalizují například stav dokončení předcházející prováděné operace a schopnost jednotky přijmou a vykonat další příkazy. Příkazy ze softwarového řadiče jsou předávány prostřednictvím registru *Cmd*. Před samotným zápisem do registru *Cmd* je třeba vždy nejprve nastavit parametry konkrétní operace. K tomu jsou k dispozici registry *Data* a *Address*. Jejich význam je různý podle typu příkazu. Registr *Data* slouží například pro předání identifikace síťového toku a způsobu jeho předzpracování při vkládání klasifikačního pravidla nebo k návratu z paměti vyčítaných dat. *Address* pak určuje například adresu položky při přímém přístupu do externí paměti. Konfigurační registry *Seed1–3* jsou využity pro parametrizaci hašovacích funkcí pro přístup do tabulky klasifikačních pravidel nebo pro parametrizaci funkce provádějící výpočet otisku klíče identifikujícího síťový tok. Registry *Mask1* a *Mask2* pak slouží k nastavení, které z položek záhlaví paketů se budou podílet na identifikaci síťového toku.

Nezbytnou součástí podrobného implementačního návrhu je také specifikace formátu pravidel a záznamů ukládaných do externí paměti QDR. Nejkratší adresovatelná položka při přístupu do externí paměti má 144 bitů. Na základě této informace byl navrhnut formát klasifikačních pravidel (tabulka 3.2) a záznamů o síťových tocích (tabulka 3.3).

Tabulka 3.2: Formát položek klasifikačních pravidel ukládaných do externí paměti

Pozice	Význam
0	obsazenost pozice v paměti, platnost následujících polí
1 – 19	adresa alternativní pozice v paměti pro umístění záznamu
20 – 71	otisk klíče identifikujícího síťový tok
72 – 79	operační kód instrukce pro jednotku Update
80 – 111	parametry operace – adresa záznamu daného toku v externí paměti
112 – 143	akce – typ zpracování přijatého paketu jednotkou Export

Kromě polí souvisejících s použitou metodou kukaččího hašování (platnost záznamu, odkaz na alternativní pozici a otisk identifikace toku) obsahuje záznam klasifikačního pravidla operační kód instrukce pro jednotku *Update*, který určuje, který z aktualizačních modulů určujících typ záznamu bude použit. V prvotní implementaci se počítá s podporou hodnot `0x00` – neprováděj výpočet statistik o síťovém toku a `0x10` – výpočet NetFlow. Kromě samotné identifikace operace je k dispozici pole umožňující její další parametrizaci. V případě NetFlow obsahuje adresu záznamu o síťovém toku v externí paměti. Poslední z polí klasifikačního pravidla určuje akci, kterou s příchozím paketem vykoná jednotka *Export*. To zahrnuje mimo jiné typ odesílaných dat do softwaru (extrahovaná data, zkrácený/nezkrácený paket) nebo rozsah a způsob určení DMA kanálu pro odeslání.

Základní Netflow aktualizační modul jednotky *Update* bude u záznamu o síťovém toku provádět zvýšení čítačů paketů a bytů, aktualizaci hodnoty počáteční a koncové časové značky a logického součtu TCP příznaků. Přesný formát záznamu poskytuje tabulka 3.3. Především z důvodu použití 64 bitových časových značek s nanosekundovou přesností se pro NetFlow záznam počítá s vyhrazením dvou po sobě jdoucích 144 bitových pozic v externí paměti.

Významnou částí je také návrh obvodu, který umožňuje blokování, respektive rezervaci, jednotlivých záznamů síťových toků v externí paměti. Modifikaci dat v externí paměti

Tabulka 3.3: Formát záznamu NetFlow statistik o síťovém toku

Pozice	Význam
0 – 31	čítač počtu paketů síťového toku toků
32 – 71	čítač počtu bytů síťového toku
135 – 72	časová značka počátku síťového toku
136 – 199	časová značka konce síťového toku
200 – 207	logický součet TCP příznaků

QDR je totiž nutné provádět ve dvou fázích (čtení a následný zápis), přičemž každá operace přístupu do paměti má určitou latenci a při zřetězeném zpracování je tak třeba zajistit atomicitu operací provádějících modifikaci. Pro potřeby vysoké propustnosti provádění aktualizace záznamů o tocích musí být realizace rezervačního obvodu schopna v každém hodinovém cyklu zpracovávat více požadavků. Z důvodu velkého množství záznamů v externí paměti se pro tento účel nabízí také využití vestavěných blokových pamětí BRAM. Obvod musí být schopen souběžně v jednom hodinovém cyklu provádět jednu kontrolu (čtení), zda je záznam k dispozici pro úpravy, jednu rezervaci záznamu (zápis) a jedno uvolnění záznamu (zápis). Jelikož vestavěné BRAM mohou realizovat pouze dvouportovou, nikoliv tříportovou paměť, je nutné využít složitější schéma. Rezervace každého paměťového místa bude vyjádřena třemi bity, kde každý z nich bude uchován v jiné paměti BRAM. V počátečním stavu jsou všechny bity vynulovány. Při kontrole rezervace záznamu se všechny tři bity načtou a provede se výpočet jejich sudé parity. Pokud mají sudou paritu, paměťové místo je volné. V tomto případě se do první BRAM zapíše negace původní vyčtené hodnoty, čímž je sudá parita porušena a je provedena rezervace položky. Protože kontrola a následná rezervace probíhá se zpožděním jednoho hodinového cyklu, musí paměť BRAM pracovat v tzv. režimu „Write First“, kdy je v případě čtení a zápisu ze stejné adresy paměti vyčtena už nová (právě zapisovaná) položka. Při uvolňování záznamu je nutné paritu opět opravit. To se provede vyčtením hodnoty příslušného bitu a zápisem jeho negace. Protože tato operace zabere dva hodinové cykly, je k dispozici druhá a třetí BRAM, kde stačí provést inverzi bitu jen v jedné z nich. Při různém využití zdrojů BRAM pamětí, lze pak provádět rezervaci jak po jednotlivých paměťových pozicích, tak (ignorováním spodních bitů adresy záznamu) po blocích více po sobě jdoucích pozic.

Pro potřeby výpočtu hodnot hašovacích funkcí byla v rámci podrobného implementačního návrhu navrženo také rozhraní a speciální obecná komponenta, která v budoucnu umožní snadné nahrazení konkrétní hašovací funkce a procesu jejího výpočtu, bez ohledu na její využití jinými komponentami systému. Tuto komponentu budou využívat jednotky *SW Access* a *Search* při přístupu do paměti klasifikačních pravidel. Pro implementaci prvotní verze se předpokládá využití funkce CRC spolu s předřazenou konfigurovatelnou jednotkou pro permutaci vstupního klíče. Výpočet CRC lze totiž ve VHDL snadno vyjádřit a implementovat pomocí dataflow popisu. V rámci NetCOPE je k tomu k dispozici také softwarový nástroj pro vygenerování rovnic na základě příslušného generujícího polynomu.

Kapitola 4

Implementace firmwaru

Jádro firmwaru systému SDM, jehož návrh byl popsán v předchozí kapitole 3, bylo v rámci práce implementováno v jazyce VHDL. Prvním krokem byla implementace jednotlivých modulů procesoru podle provedeného návrhu. Při implementaci byly použity některé komponenty dostupné v rámci vývojové platformy NetCOPE. Jedná se především o implementaci FIFO pamětí a komponent pro práci s interními komunikačními protokoly platformy. Implementace jednotky *HFE* pro analýzu a extrakci záhlaví příchozích paketů je pak založena na analyzátoru publikovaném na odborné konferenci ANCS [21] a dostupném také v rámci platformy NetCOPE. Jednotlivé moduly jsou vždy tvořeny entitou a architekturou s označením `SDM_`*`jméno_modulu`*, např. `SDM_ARBITER` nebo `SDM_SEARCH`. V některých případech je implementace navíc z důvodu lepší přehlednosti a udržovatelnosti zdrojového kódu rozdělena do podkomponent a více VHDL souborů. Například pro oddělení implementace řídicího konečného automatu nebo jiných význačných částí.

V průběhu implementace se v řadě případů vyskytla potřeba navrhnout a implementovat další drobnější podpůrné komponenty pro provádění určitých činností. Navržené komponenty provádějí obecné činnosti a jsou tak využitelné i v jiných projektech. Bylo tedy vhodné je vyčlenit do samostatných bloků. Takto vznikl například blok pro výpočet operace modulo potřebný v rámci jednotky *Export* pro určení DMA kanálu k odeslání dat. Operace modulo byla implementována genericky s využitím blokových pamětí. Při procesu syntézy je paměť, určená pouze pro čtení, naplněna hodnotami výsledku operace modulo pro různé dvojice vstupních operandů. Při výpočtu se pak konkrétní operandy využijí k adresaci paměťového místa obsahujícího správný výsledek operace. Další z podobných jednotek je například generický blok umožňující zřetězené porovnávání dvojice hodnot ve více hodinových cyklech za účelem zvýšení propustnosti této operace. Blok je potom použit v jednotce *Search* pro porovnávání otisků klíčů identifikujících síťové toky. V druhé fázi implementace pak probíhala integrace jednotek, tvorba samotné architektury procesoru a sestavení firmwaru včetně infrastruktury platformy NetCOPE.

V současnosti je sestavení vytvořeného systému pro cílovou akcelerační kartu podporující 100G Ethernet závislé na její dostupnosti a připravenosti platformy NetCOPE pro tuto kartu. V době řešení bakalářské práce nebyla uvedená karta k dispozici. Veškeré výsledky jsou tak uvedeny pro obdobnou akcelerační FPGA kartu Fiberblaze FB8XG@V7690 [5]. Vývojová karta je shodně osazena čipem FPGA Virtex-7, avšak pouze jedním modulem paměti QDRII+ pracujícím na frekvenci 300 MHz. Dále poskytuje rozhraní PCI-Express 8x gen3. Síťová rozhraní umožňují příjem a odesílání dat s celkovou maximální rychlostí 80 Gb/s a to v různých konfiguracích počtu portů (8x10 Gb/s nebo 2x40 Gb/s). Samotné aplikační jádro firmwaru však zůstává nezměněno a pracuje se stále stejnou šířkou datové

sběrnice 512 bitů. Veškeré konverze spojené s různými typy síťových rozhraní zajišťuje síťový modul v rámci platformy NetCOPE. Drobné úpravy v podobě změny generických parametrů procesoru si žádá pouze nižší kapacita externí paměti. Parametry ovlivňují především rozsah hodnot generovaný hašovacími funkcemi pro přístup k vyhledávací tabulce a možný počet spravovaných flow záznamů.

4.1 Výsledky syntézy jádra firmwaru

Při syntéze dochází k převodu VHDL popisu obvodu na funkčně ekvivalentní vyjádření pomocí obecných logických bloků. Výsledkem procesu je propojení a realizace obvodu pomocí prvků cílové technologie, tzv. Netlist. Součástí je také odhad zabraných zdrojů na čipu FPGA výslednou implementací. Syntéza jádra firmwaru byla provedena pomocí nástroje Vivado ve verzi 2013.4 pro konkrétní variantu XC7VX690T čipu Virtex-7, který je osazený na vývojové akcelerační kartě. Získané hodnoty zabraných zdrojů na čipu poskytuje tabulka 4.1. Kromě počtů využitých základních elementů (LUT a registrů) konfigurovatelných bloků

Tabulka 4.1: Využité zdroje na čipu XC7VX690T podle jednotlivých jednotek firmwaru

Jednotka	Počet LUT	Počet registrů	BlockRAM
SW Access	6 729 (1,6 %)	2 585 (0,3 %)	0 (0,0 %)
HFE	8 921 (2,0 %)	5 624 (0,6 %)	0 (0,0 %)
Search	7 909 (1,8 %)	3 453 (0,4 %)	0 (0,0 %)
Update	4 236 (1,0 %)	3 980 (0,5 %)	48 (3,3 %)
Packet Buffer	20 (0,0 %)	6 (0,0 %)	10 (0,7 %)
Export	7 918 (1,8 %)	4 101 (0,5 %)	1 (0,1 %)
Arbiter	794 (0,2 %)	20 (0,0 %)	0 (0,0 %)
Cekem	35 749 (8,3 %)	19 712 (2,3 %)	59 (4,0 %)

CLB je tabulka doplněna i o využití blokových pamětí BlockRAM. Hodnoty jsou uvedeny zvlášť pro jednotlivé jednotky a pak v souhrnu pro celé jádro firmwaru. Celkové zdroje zabrané jádrem se však mohou lišit od součtu zdrojů jednotlivých jednotek. Důvodem jsou optimalizace prováděné syntézním nástrojem napříč jednotkami. Údaje uvedené v závorkách pak vyjadřují podíl z celkového počtu dostupných zdrojů na zvoleném čipu FPGA. Jak je vidět z hodnot v tabulce, jádro firmwaru zabírá nevýraznou část (jednotky procent) všech zdrojů čipu.

Nejvyšší podíl zdrojů zabírá jednotka *HFE*. V použité konfiguraci umožňuje kromě záhlaví Ethenetu analýzu také VLAN a MPLS a podporuje až dvě IPv6 rozšiřující záhlaví. Další (s ohledem na zdroje) náročnou operací je zkracování síťových paketů před odesláním do SW. Komponenta realizující tuto činnost představuje značné množství zdrojů jednotky *Export*. Přibližně polovina zdrojů jednotek *SW Access* a *Search* připadá na komponentu pro výpočet hašovacích funkcí a otisku klíče, kterou obě používají při přístupu k paměti. Pokud se zaměříme na využití blokových pamětí, jsou použity jednak v rámci *Packet Bufferu* pro realizaci paměti FIFO, daleko vyšší počet však v rámci jednotky *Update* slouží k realizaci rezervačního obvodu pro zamykání záznamů o tocích v externí paměti. Jedinou paměť BlockRAM pak využívá jednotka *Export* pro realizaci výpočtu operace modulo při distribuci dat do DMA kanálů.

4.2 Výsledky implementace firmwaru

Při následném procesu implementace, někdy označovaném také jako „place & route“, dochází k namapování prvků obvodu na dostupné elementy konkrétní cílové technologie, včetně jejich přesného rozmístění a vhodného propojení na čipu. Do procesu vstupují také uživatelem definované omezující podmínky určující především periodu hodinového signálu či přiřazení portů VHDL entity ke konkrétním vývodům pouzdra čipu. Výstupem je bitstream a informace, zda se podařilo splnit podmínky časování obvodu pro bezchybnou činnost na požadované pracovní frekvenci.

Implementace byla provedena včetně infrastruktury platformy NetCOPE. K tomuto kroku byl shodně použit nástroj Vivado ve verzi 2014.3. Celkové výsledky zabraných zdrojů na čipu pro různé konfigurace síťových rozhraní poskytuje tabulka 4.2. Většina zdrojů je

Tabulka 4.2: Využité zdroje na čipu XC7VX690T pro jednotlivých varianty firmwaru

Varianta	Počet LUT	Počet registrů	BlockRAM	Frekvence
8x10 Gb/s	222 745 (51,5 %)	184 084 (21,3 %)	525 (35,7 %)	160 MHz
2x40 Gb/s	178 984 (41,4 %)	134 172 (15,5 %)	306 (20,7 %)	100 MHz

přitom využita komponentami a infrastrukturou platformy NetCOPE. Samotné aplikační jádro zabírá jen malou část z nich. V posledním sloupci je pro konkrétní variantu doplněna také informace o aktuálně dosažené pracovní frekvenci jádra vytvořeného firmwaru. Pro zajištění plné propustnosti na vývojové kartě je přitom nutné při použité datové šířce 512 bitů dosáhnout minimální pracovní frekvence jádra 156,25 MHz. Pro cílovou akcelerační kartu a dosažení plné propustnosti na rychlosti 100 Gb/s pak bude nutná frekvence jádra téměř 200 MHz.

Splnění požadavků časování provedené implementace je jednou z nejobtížnějších úloh při návrhu obvodů pro FPGA. Jedním ze základních postupů pro zvýšení pracovní frekvence je úprava prvků, které tvoří největší zpoždění, odstraňování tzv. kritických cest. Toho lze dosáhnou například technikou pipeliningu provádějící rozdělení většího úseku kombinační logiky na menší části a vložení registrů mezi ně. Tento přístup však vede ke zvýšení latence daného obvodu a není možné jej použít vždy. Vložením registru vzniká zpoždění, které při použití v obvodech například se zpětnou vabou může ovlivnit funkci původního obvodu. Je také důležité, aby velikost logiky v jednotlivých stupních oddělených registry byla vyvážená. Maximální dosažená frekvence je totiž dána cestou v obvodu s největším zpožděním.

Detailnější časová analýza vytvořeného firmwaru ukázala, že výrazný podíl doby zpoždění, v některých případech 80–90 %, je dán způsobem propojení jednotlivých bloků na čipu FPGA provedeným při implementaci syntézním nástrojem. Tento jev může být způsoben nedostatkem rychlých lokálním propojovacích cest v daném místě FPGA a nutností využití propoje s vyšším zpožděním. Nástroj Vivado sice nabízí mnoho voleb a řadu strategií pro proces implementace, žádná z vyzkoušených možností však neměla výrazný vliv na zlepšení dosažené pracovní frekvence. Tento druh kritických cest se přitom vyskytuje nejen ve vlastní implementaci procesoru, ale také v rámci komponent infrastruktury platformy NetCOPE. Další postup při zvyšování pracovní frekvence a propustnosti firmwaru systému si tak žádá také nezbytné úpravy platformy, vyžadující však pokročilejší znalosti její architektury a úzkou spolupráci s jejími vývojáři. Pro potřeby ověření základní funkčnosti a otestování provedené implementace však dosažená frekvence nepředstavuje omezení.

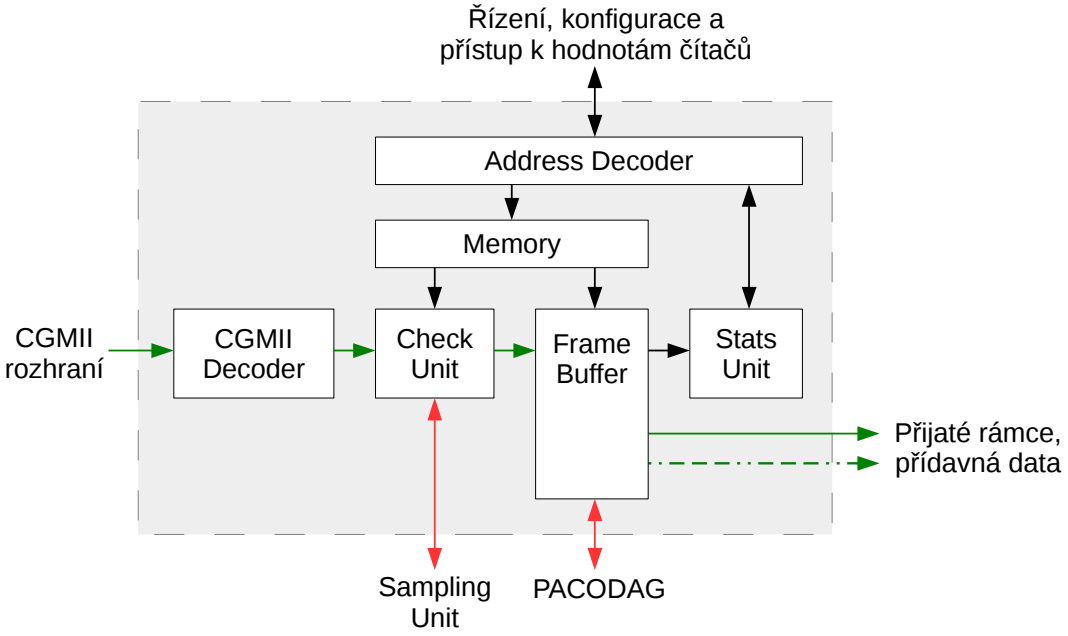
4.3 Vstupní síťový blok pro 100 Gb/s

Budoucí přenos firmwaru systému SDM na cílovou akcelerační kartu podporující 100G Ethernet závisí na dostupnosti platformy NetCOPE. Aktuálně dostupné síťové bloky podporují příjem a vysílání pouze pro nižší rychlosti (1, 10 a 40 Gb/s). Jednou ze zásadních a dosud chybějících součástí platformy je implementace vstupního síťového bloku pro příjem rámců na rychlosti 100 Gb/s. Nad rámec zadání bakalářské práce je proveden návrh architektury jednotky tohoto vstupního síťového bloku. Zároveň je podle návrhu vytvořena prvotní VHDL implementace.

Architektura technologie Ethernet je rozdělena do několika úrovní. V rámci fyzické vrstvy a podvrstev PCS (Physical Coding Sublayer) a PMA (Physical Medium Attachment) se provádí vhodné překódování dat a převod ze sériového typu přenosu, který je vhodný pro fyzické médium, na paralelní. Vstupní síťový blok pracuje na vyšší, takzvané MAC (Media Access Control) vrstvě, kde jsou data přenášena v podobě rámců, jejichž formát byl uveden již v úvodní sekci 2.2. Ke komunikaci s fyzickou vrstvou slouží nezávislé komunikační rozhraní MII (Media Independent Interface). Přesné označení rohraní MII se ve standardu pro různé přenosové rychlosti liší. Například XGMII odpovídá rozhraní pro přenosovou rychlost 10 Gb/s. Rozhraní pro 100 Gb/s je pak označeno jako CGMII.

Hlavním úkolem vstupního síťového bloku je kontrola příchozích rámců a transformace signálů použitých u rohraní CGMII na komunikační rozhraní interního protokolu pro přenos dat v rámci platformy NetCOPE. Blok umožňuje bufferování a zahazování příchozích rámců podle výsledků prováděných vstupních kontrol. Jsou k němu připojeny další pomocné komponenty. Komponenta *Sampling Unit* slouží k řízení případného vzorkování příchozího toku dat. Generování přídavných informací k příchozím rámců, jako jsou například přesné časové značky, zajišťuje komponenta *PACODAG* (Packet Control Data Generator). Mezi prováděné kontroly patří ověřování formátu příchozích dat, včetně detekce a ošetření případných chyb vstupního rozhraní, kontrola hodnoty pole *Destination Address* obsahujícího cílovou MAC Adresu, ověření hodnoty kontrolního součtu FCS a dodržení nastavených limitů pro velikost příchozího rámce. Kontrolu cílové adresy lze provádět v několika režimech – povolení všech MAC adres (tzv. promiskuitní mód) nebo povolení pouze adres uložených v paměti, respektive jejich kombinace se speciálními broadcastovými a multicastovými adresami. Výsledky kontrol ovlivňující, zda bude rámec zahozen či nikoliv, jsou maskovatelné, což umožňuje v určitém případě propouštění také chybných rámců k dalšímu zpracování. Konfigurace probíhá přes řídící a konfigurační rozhraní, které je pro tyto účely standardně součástí propojovacího systému sběrnic platformy NetCOPE. Toto rozhraní také poskytuje přístup ke statistickým informacím o činnosti vstupního síťového bloku v podobě čítačů rámců. Jeden z čítačů uchovává například počet všech příchozích rámců. Dále se rozlišuje počet korektně přijatých rámců (těch které úspěšně prošly vstupní kontrolou) a počet zahozených z důvodu chyby. Speciálně se pak sleduje informace o počtu zahozených rámců z důvodu přeplnění bufferů. Součástí činnosti vstupního síťového bloku je také odstraňování úvodní části záhlaví rámce obsahující preambuli a znak SOF a volitelné (na základě generického parametru) odstraňování pole FCS. Výstupem bloku jsou vlastní přenášená data rámce spolu s přídavnými informacemi vygenerovanými komponentou *PACODAG*.

Návrh architektury vychází z již existujících vstupních síťových bloků. Hlavní rozdíl je v rozšíření datových cest na 512 bitů a přidání podpory částečného zarovnání přenášených dat a možnosti sdílení datových slov. Architektura navrženého vstupního síťového bloku je zobrazena na obrázku 4.1. Bloky z obrázku provádí určité operace s příchozím datovým tokem. Následující seznam poskytuje přehled a detailnější popis funkcí zajišťovaných jednotlivými bloky.



Obrázek 4.1: Architektura vstupního síťového bloku pro 100 Gb/s

CGMII Decoder provádí dekódování formátu CGMII a převod do protokolu interně používaného v rámci platformy NetCOPE. Dalšími z činností jsou také detekce chyb na vstupním rozhraní a odstraňování úvodní části rámců.

Check Unit zodpovídá za provádění veškerých kontrol probíhajících nad příchozími rámci. Kontroly jsou ovlivněny konfiguračními registry v bloku *Memory*, kde jsou také uloženy povolené MAC adresy. Další funkcí komponenty je volitelné odstraňování pole FCS. Od komponenty *Sampling Unit* dále dostává informace o vzorkování a spolu s příchozími daty a výsledky provedených kontrol je předává další jednotce.

Frame Buffer provádí bufferování příchozích dat. Na základě obdržených výsledků kontrol, informací o vzorkování a aktuální konfigurace (nastavení maskování výsledků kontrol) umožňuje zahazování jednotlivých příchozích rámců. Současně předává informaci o zpracování rámce statistické jednotce. Spolu s údaji vygenerovanými externí komponentou *PACODAG* odesílá úspěšně přijaté příchozí rámce na výstup celého síťového bloku.

Memory obsahuje konfigurační a stavové registry vstupního síťového bloku. Nastavené hodnoty ovlivňují především činnost bloků *Check Unit* a *Frame Buffer*. Součástí jednotky je také CAM (Content-Addressable Memory) sloužící pro uložení a vyhledávání MAC adres.

Stats Unit zodpovídá za vedení statistik o příchozích rámcích na základě informací od jednotky *Frame Buffer*. Obsahuje a dále zpřístupňuje jednotlivé čítače přijatých a zahozených rámců.

Address Decoder zajišťuje propojení vstupního síťového bloku s řídicím a konfiguračním rozhraním platformy NetCOPE. Stará se o adresaci, vyčítání a zápis hodnot čítačů, registrů a přístup ke CAM paměti umístěných v jednotkách *Memory* a *Stats Unit*.

Podle uvedeného návrhu byl vstupní síťový blok implementován v jazyce VHDL a začleněn jako standardní součást platformy NetCOPE. Použití vytvořené implementace je daleko širší než jen pro potřeby realizace aplikačně specifického procesoru pro bezpečnostní monitorování vysokorychlostních sítí. Uplatnění najde i v jiných aplikacích provádějících zpracování síťových dat a využívajících technologii Ethernet. Implementace je syntetizovatelná a schopná pracovat na frekvenci nad 200 MHz, čímž je při použité šířce datových cest 512 bitů zajištěna plná propustnost 100 Gb/s. Vstupní síťový blok přitom zabírá jen 2 % LUT dostupných na čipu FPGA Virtex-7 XC7VX690T a ještě menší podíl registrů (< 1 %). Blok úspěšně prošel také funkční verifikací a je tak plně připraven na nasazení na nové generaci FPGA akceleračních karet podporujících technologii Ethernet s přenosovou rychlostí 100 Gb/s.

Kapitola 5

Verifikace jádra firmwaru

Se zvyšujícími se nároky a složitostí navrhovaného zařízení se zvětšuje i množství chyb, kterých se lze dopustit a které lze do implementace v průběhu práce zanést. Čím dříve se takové chyby podaří odhalit, tím méně kritické jsou následky jejich výskytu. Snahou různých přístupů testování a ověřování implementace je odhalení co největšího počtu chyb již v počátečních fázích vývoje. Systematický proces hledání a odstraňování chyb tak tvoří jeho nedílnou součást. Následující text poskytuje popis některých typicky používaných metod pro ověřování funkcionality implementace hardwarových obvodů. Získané poznatky vycházejí z [26].

Simulace je jednoduchý přístup, při kterém je základní funkčnost systému ověřována na předem připravené množině testovacích vektorů. Testovaná komponenta je zapojena do prostředí popsaného jazykem VHDL, takzvaného testbenche. V rámci VHDL popisu jsou také nastavovány její vstupní hodnoty. Výsledkem simulace, probíhající ve specializovaném softwarovém nástroji, je časový diagram obsahující vypočtené hodnoty vybraných signálů komponenty. Návrhář následně provede manuální kontrolu průběhů signálů a výstupních hodnot v časovém diagramu a rozhodne, zda komponenta daným testem v simulaci prošla či nikoliv. Tato metoda je schopná odhalit přítomnost chyby, nedokazuje a negarantuje však, že systém neobsahuje chyby další. Neprověřují se totiž všechny možné stavy a každé chování systému.

Formální verifikace naopak umožňuje prokázání správnosti a bezchybnosti dané implementace. Je založena na matematickém popisu systému a ověřování, zda je systém, jeho vlastnosti a funkčnost v souladu se specifikací. Postup je podobný provádění matematického dokazování. Není však možné jej vždy plně automatizovat. Jde o velmi složitý a časově náročný přístup používaný typicky pouze pro nejdůležitější komponenty navrhovaného systému. Výsledkem je buď důkaz, že neexistuje žádný vstup, který by způsobil porušení sledované podmínky nebo protipříklad – určitý vstupní vektor, který vede k jejímu porušení.

Funkční verifikace je přístup založený na simulaci implementace obvodu. Narozdíl od klasické simulace však využívá pokročilejších technik zvyšujících její efektivitu. Za tímto účelem vznikla nová rodina verifikačních jazyků (Hardware Verification Language, HVL). Jazyky HVL umožňují popis testbenche na vyšší úrovni abstrakce a poskytují speciální konstrukce pro generování náhodných vstupních hodnot a automatizovanou kontrolu výstupů. Při funkční verifikaci lze tak snáze otestovat větší množství možných stavů verifikované jednotky. Cílem je opět nalezení co možná největšího

množství chyb. Tímto přístupem však nelze prokázat bezchybnost, jako to umožňuje formální verifikace.

Provedená implementace navrženého jádra systému pro monitorování vysokorychlostních sítí byla otestována v rámci základních simulací. Před hardwarovým testováním a měřením reálných výkonových parametrů jsem navíc nad rámec zadání bakalářské práce provedl také podrobnější funkční verifikaci jádra systému. Díky funkční verifikaci se mně podařilo odhalit a opravit spoustu chyb, především takových, které vznikly při integraci jádra spojením jednotlivých funkčních jednotek. Přestože provedená verifikace nepokrývá všechny možné situace a stavy systému a řada chyb se projeví až při reálném nasazení, existující verifikační prostředí přináší nespornou výhodu. Situaci vedoucí k nalezené chybě při hardwarovém testování lze zpětně navodit ve verifikačním prostředí a chybu pohodlně odladit s použitím prostředků simulátoru. Popis navrženého a implementovaného verifikačního prostředí a verifikačního modelu jádra systému je předmětem následujícího textu.

5.1 Verifikační prostředí

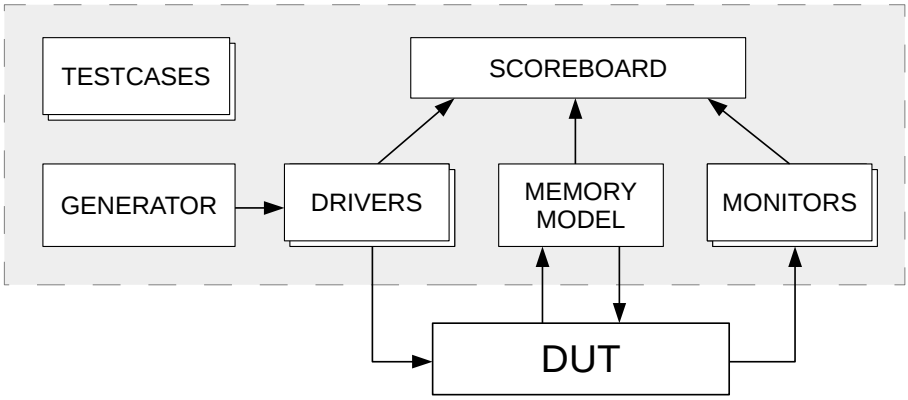
K implementaci verifikačního prostředí jádra systému byl zvolen jazyk SystemVerilog, který je používán také pro verifikaci komponent infrastruktury platformy NetCOPE. SystemVerilog [8] je objektově orientovaný jazyk speciálně určený pro tvorbu verifikačních prostředí. Vychází z prvků jazyka Verilog a lze jej tak využít i k implementaci hardwarových obvodů. Jako zástupce jazyků HVL však poskytuje primárně konstrukce pro podporu simulování a verifikace hardwarových obvodů.

Základní datovou jednotkou, se kterou se ve verifikačním prostředí pracuje je transakce. Lze ji chápat jako formu požadavku či paketu nebo jako sled souvisejících signálů na komunikačním rozhraní. V jazyce SystemVerilog je reprezentována objektem. Podstatou funkční verifikace je odesílání náhodně vygenerovaných transakcí na vstupní rozhraní testované jednotky a sledování odpovídajících výstupů. Automatizovaně je pak ověřováno, zda zachycené výstupy jednotky odpovídají jednotlivým vstupním transakcím. Při generování náhodných vstupních transakcí pro testovanou jednotku musí být zajištěna korektnost těchto vstupů a dodržení protokolu vstupního rozhraní jednotky. Používá se takzvané constrained-random generování transakcí definující formát vstupů pomocí omezujících podmínek, které umožňují také vytvoření určité konkrétní situace nebo stavu.

Kromě korektnosti výstupů testovaného systému lze během celé verifikace sledovat také takzvané pokrytí. Jde o statistiku shrnující, jaké stavy a vlastnosti systému byly při verifikaci prověřeny. Rozlišuje se několik typů – pokrytí funkcí a chování systému, pokrytí řádků zdrojového kódu a jednolivých konstrukcí implementace nebo pokrytí obsažených konečných automatů informující o navštívených stavech a přechodech mezi nimi.

Další z použitých technik jsou formální tvrzení, takzvané assertions – výrazy v temporální logice, které musí být ve verifikovaném systému vždy platné. Při jejich porušení je verifikace přerušena s chybou nebo je vypsáno varování a lze tak snáze odhalit zdroj problémů. Jejich použití je vhodné třeba pro kontrolu přechodů stavových automatů nebo protokolů na komunikačních rozhraních.

Pohled na navržené verifikační prostředí nabízí obrázek 5.1. Šedá ohraničená oblast znázorňuje části implementované v SystemVerilogu. Při implementaci byly využity prostředky platformy NetCOPE pro podporu provádění verifikací. Jedná se typicky o předpřipravené bázové třídy jednotlivých komponent, na jejichž základě byla založena vlastní implementace.



Obrázek 5.1: Verifikační prostředí jádra firmware

Následující seznam poskytuje přehled a detailnější popis jednotlivých komponent vytvořeného verifikačního prostředí.

DUT (Design Under Test) — instance verifikované jednotky popsané v jazyce VHDL a zapojená do verifikačního prostředí. Její součástí je také obálka definující jednotlivé signály rozhraní (šířku sběrnic, směr komunikace) a umožňující jejich převod na rozhraní jazyka SystemVerilog. V tomto případě je jako DUT zapojena VHDL implementace jádra systému – implementace aplikačně specifického procesoru.

Generator — prostředek zajišťující generování vstupních transakcí. Generování hodnot transakcí probíhá na základě definice omezení náhodných proměnných. Vygenerované transakce jsou předávány objektům driverů, zajišťujících jejich přeposlání a převod na rozhraní testované komponenty (DUT). V případě aplikačně specifického procesoru jde především o generování příchozích síťových dat (paketů), včetně přidružených přídavných informací o příchozích datech. Generovány jsou také řídicí požadavky, na jejichž základě se provádí vkládání klasifikačních pravidel do externí paměti a exportování záznamů o síťových tocích.

Drivers — objekty driverů přijímají obecné transakce vytvořené generátorem, provádí jejich konverzi a se správným časováním je předávají na vstupní rozhraní DUT, kde přímo nastavují konkrétní signály rozhraní a umožňují například vkládání čekacích stavů. Typicky bývá generování těchto stavů náhodné a nastavitelné v rámci konkrétního testu. Vstupní transakce jsou také odesílány do scoreboardu k dalšímu vyhodnocení. Pro verifikaci aplikačně specifického procesoru tato část zahrnuje drivery pro vstupní síťové rozhraní a rozhraní pro příjem softwarových požadavků.

Monitors — objekty monitorů naopak sledují hodnoty na výstupních rozhraních verifikované komponenty. Z příchozích dat sestavují transakce a ty odesílají k vyhodnocení do scoreboardu. Podobně jako drivery umožňují dle nastavení daného testu vkládání čekacích stavů na rozhraních (pokud to rozhraní umožňují). Ve vytvořeném verifikačním prostředí se jedná zejména o monitor výstupního síťového rozhraní a monitor rozhraní pro komunikaci s řadičem DMA a přenosy dat do operační paměti hostitelského počítače.

Scoreboard — prostředek provádějící automatické vyhodnocování průběhu verifikace. Zde probíhá ověřování, zda příchozí transakce od driverů odpovídají přijatým transakcím z monitorů. V případě nesrovnalostí je vypsáno varování nebo je verifikace rovnou ukončena s chybou a končí neúspěchem. Jde o nejsložitější část verifikačního prostředí, neboť musí prakticky a plně nezávisle implementovat veškerou funkcionalitu verifikované jednotky. Tuto část zajišťuje verifikační model jádra firmwaru, který je součástí scoreboardu. Jeho popis je však díky vyšší úrovni abstrakce daleko jednodušší než předcházející VHDL implementace firmwarového jádra. Bližším popisem verifikačního modelu se zabývá následující sekce 5.2. Model je totiž specifický tím, že jeho činnost je řízena na základě informací o probíhající komunikaci mezi verifikovanou jednotkou a modelem externí paměti.

Memory Model — komponenta navržená speciálně pro tuto verifikaci a ne úplně typická pro klasická verifikační prostředí. Verifikovaný systém zpracovává vstupní transakce (požadavky a síťové pakety) na základě stavu uloženého v externí paměti (dle klasifikačních pravidel). Tento stav se navíc v průběhu zpracování příchozích transakcí verifikovanou jednotkou může měnit. Model tak simuluje činnost externí paměti a o probíhající komunikaci s DUT informuje scoreboard. Scoreboard pak provádí detailní analýzu a kontrolu vkládání a odebírání klasifikačních pravidel nebo operací aktualizujících a exportujících záznamy o tocích a upozorňuje například na porušení atomicity nebo chybný formát ukládaných dat.

Testcases — blok představuje sadu prováděných testovacích případů a různých nastavení parametrů verifikačního prostředí. Jeho součástí jsou také konfigurační hodnoty jednotlivých prvků prostředí (generátoru, driverů, monitorů, modelu paměti) například pro nastavení počtu generovaných transakcí, latence modelu externí paměti nebo pravděpodobnosti vkládání čekacích stavů na rozhraních testované komponenty. Konkrétně pro verifikaci aplikačně specifického procesoru jsem sestavil sadu 16 testovacích případů, kde každý z nich se zaměřuje na určitou oblast funkcionality. Testy zahrnují ověření formátu komunikace na všech vnějších rozhraních a jejího řízení, včetně správné signalizace a interpretace čekacích stavů. Jsou testovány všechny úrovně předzpracování síťového provozu pro různé délky příchozích rámců s důrazem na minimální a maximální povolené hodnoty. Kontrole je podrobena také správa paměti pravidel a paměti záznamů o síťových tocích s ohledem na dodržení atomicity dílčích operací provádějících čtení a zápis do externí paměti. Speciální pozornost si zaslouží také mechanismus provádějící vkládání pravidel a reorganizaci položek na základě principu techniky kukaččího hašování. Ověřováno je i zpracování softwarových požadavků na exportování záznamů o tocích a odstraňování klasifikačních pravidel.

5.2 Verifikační model jádra

Verifikační model jádra firmwaru tvoří nejdůležitější část verifikačního prostředí. Je součástí scoreboardu a podílí se na automatizovaném vyhodnocování průběhu verifikace. V rámci modelu musí být implementován stejný algoritmus a funkcionalita, jaké provádí verifikovaná jednotka. Realizace modelu je však snažší díky vyšší úrovni abstrakce jeho popisu. Model provádí transformaci stejných vstupních požadavků na výstupy zcela nezávisle na VHDL implementaci verifikované jednotky. Průběžně je pak prováděna kontrola, zda výstupy z VHDL implementace souhlasí s těmi očekávanými a poskytnutými modelem. V opačném případě

je proces verifikace ukončen a je hlášena chyba. Kromě detekce chyb při provádění transformace vstupní transakce na požadovaný výstup se kontroluje také správné pořadí výstupních transakcí.

Implementaci modelu jsem provedl v jazyce SystemVerilog stejně jako celý zbytek verifikačního prostředí. Nic však nebrání využití jiných programovacích jazyků pro popis modelu nebo jeho částí a jejich začlenění do verifikačního prostředí. SystemVerilog nabízí speciální rozhraní (Direct Programming Interface, DPI) umožňující volání funkcí popsaných jinými programovacími jazyky. Stejně jako u verifikované jednotky musí být umožněno provedení paralelního zpracovávání požadavků přicházejících na různá vstupní rozhraní (příchozí pakety, softwarové požadavky, komunikace s pamětí). Implementace verifikačního modelu tak musí nezbytně využívat více procesů nebo vláken a zajistit také jejich správnou synchronizaci. Také tento fakt vedl k výběru jazyka SystemVeriflog, který nativně poskytuje řadu prostředků (semafory, schránky, asynchronní události) pro synchronizaci procesů. Implementace je rozdělena do několika tříd odpovídajících jednotlivým funkčním jednotkám z návrhu systému, viz sekce 3.2. Velmi důležitou částí je také sledování probíhající komunikace DUT s modelem externí paměti. Získané informace slouží k řízení činnosti verifikačního modelu jádra tak, aby jak vlastní implementace, tak model systému pracovaly nad stejnou množinou klasifikačních pravidel.

Při verifikaci bylo v rámci každého ze 16 testovacích případů zpracováváno vždy 100 000 vstupních požadavků a transakcí. Postupným opravováním odhalených chyb jsem dospěl do stavu, kdy je provádění všech 16 testů verifikace úspěšně dokončeno a jsou korektně zpracovány všechny vygenerované vstupní transakce. Úspěšným provedením verifikace jádra systému proces ladění firmwaru zdaleka nekončí. I tak se ale podařilo odhalit a opravit řadu chyb. Ve fyzické implementaci obvodu se mohou vyskytnou další chyby, které vznikly v době syntézy nebo procesu place & route a souvisejí například s nesprávným aplikováním omezujících podmínek definujících časování obvodu. Jejich ladění je problematické, protože již nemáme k dispozici pohodlné prostředí simulátoru a nelze jednoduše zobrazit hodnoty vnitřních signálů, a je nutné využít jiných prostředků umožňujících ladění firmwaru přímo na čipu FPGA.

Kapitola 6

Dosažené výsledky

Kapitola popisuje výsledky získané při hardwarovém testování a měření reálných výkonových parametrů provedené implementace firmwaru. Hlavním ze sledovaných parametrů byla dosahovaná propustnost realizovaného řešení pro různé případy hardwarového předzpracování síťových dat a různé délky příchozích ethernetových rámců. Dále jsem se zabýval analýzou dosahovaného zaplnění hašovací tabulky pro ukládání klasifikačních pravidel, která využívá metodu kukaččího hašování dle předchozího návrhu. Součástí je také sledování počtu provedených přístupů do externí paměti při operaci vkládání položek, v závislosti na aktuálním zaplnění hašovací tabulky, především v případě, kdy vznikají kolize vynucující provedení reorganizace položek uložených v paměti.

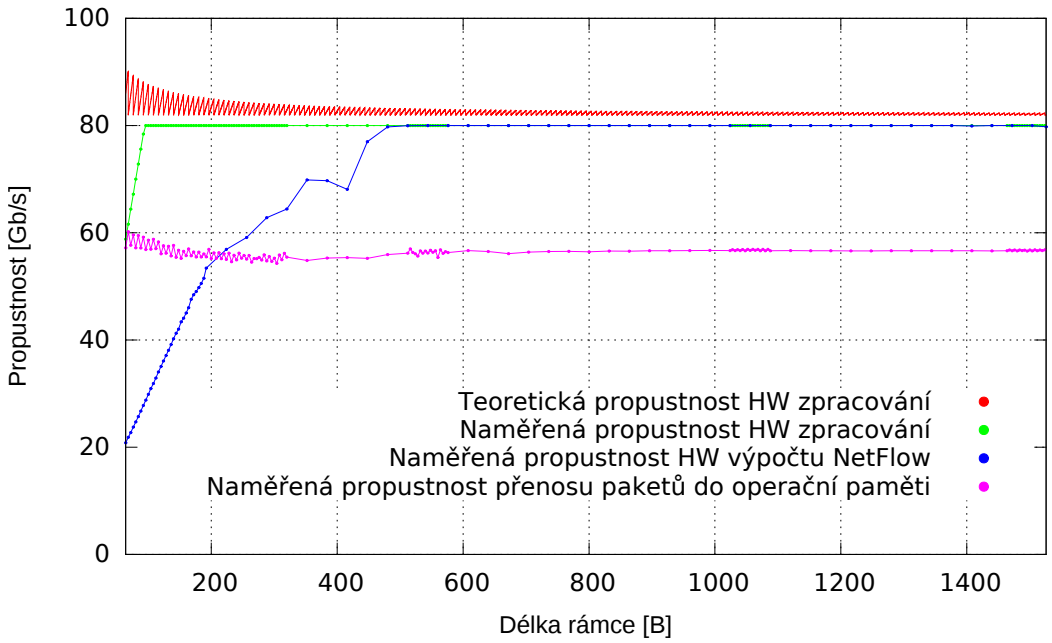
Veškerá měření probíhala na vývojové FPGA akcelerační kartě Fiberblaze FB8XG@-V7690, které byla představena již dříve. Použita byla verze firmwaru v konfiguraci síťových portů 8x10 Gb/s, pro niž se při překladu podařilo dosáhnout požadované pracovní frekvence pro zajištění plné propustnosti zpracování příchozích síťových dat při rychlosti 80 Gb/s. V rámci testovacího prostředí byla akcelerační karta zasazena do slotu PCI-Express 8x gen3 hostitelského počítače, který byl osazen dvěma 6-jádrovými procesory Intel Xeon E5-2620 pracujícími na frekvenci 2,00 GHz. Použitý server měl k dispozici 64 GB operační paměti a s aktivní podporou technologie Intel Hyper-Threading umožňoval běh až 24 výpočetních vláken.

6.1 Propustnost systému

Měření propustnosti zpracování příchozích síťových dat systémem probíhalo v laboratorním prostředí. Generování testovacího síťového provozu zajišťoval hardwarový tester Spirent TestCenter simulující provoz reálné sítě a umožňující konfiguraci délek odesílaných rámců a dalších parametrů generovaného provozu. Pro jednoduchost bylo nastaveno odesílání ethernetových rámců přenášejících pouze záhlaví protokolu IPv4 a náhodně vygenerovaná data, bez použití protokolu transportní vrstvy. Firmwarové jádro bylo nakonfigurováno tak, aby k identifikaci síťového toku používalo pouze zdrojové IP adresy. Zdrojové adresy rámců byly generovány rovnoměrně z rozsahu 192.168.0.0 – 192.168.7.255, čímž bylo vytvořeno 2048 různých síťových toků.

Proces měření propustnosti probíhal tak, že byly v pravidelných intervalech opakovaně vyčítány hodnoty čítačů rámců, které byly při plné saturaci všech připojených 10 Gb linek zpracovány v rámci vstupních síťových bloků na jednotlivých rozhraních. Dosažená propustnost zpracování pak byla odvozena jako podíl z celkového příchozího datového toku 80 Gb/s

na základě poměru úspěšně přijatých (zpracovaných) rámců a počtu všech příchozích rámců. Uvedený způsob měření byl opakován pro různé typy hardwarového předzpracování dat pro všechny síťové toky a různé délky ethernetových rámců z rozsahu 64 – 1526 bytů. Dále je rozlišena propustnost pouze hardwarového zpracování (realizováno zahazováním dat na vstupu DMA řadiče) a propustnost v případě, kdy jsou data přenášena do operační paměti počítače a projevuje se tak omezená propustnost systémové sběrnice PCI-Express. Za účelem dosažení vyšší přesnosti byl každý typ měření proveden pětkrát a výsledné hodnoty byly získány výpočtem aritmetického průměru hodnot jednotlivých měření. Na základě získaných údajů jsem vytvořil graf na obrázku 6.1, který vyjadřuje propustnost zpracování síťových dat systémem v Gb/s v závislosti na délce příchozích ethernetových rámců.



Obrázek 6.1: Propustnost systému v závislosti na délce rámce (v Gb/s)

Červená křivka představuje maximální teoretickou propustnost hardwarového zpracování při pracovní frekvenci jádra 160 MHz. Odhad byl proveden na základě znalosti šířky datových cest a prováděného způsobu zarovnání datových slov. Oscilace křivky je dána právě částečným zarovnáváním začátků přenášených bloků dat na násobky 8 bytů.

Zeleně je znázorněna propustnost hardwarového zpracování síťových dat, které zahrnuje provedení klasifikace paketu a odeslání buď jeho nezkrácené verze, nebo z něj extrahovaných dat na výstup do kanálu DMA (data nejsou v tomto případě pro potřeby měření přenášena až do operační paměti, ale zahazována na vstupu DMA řadiče). Pokles výkonnosti při zpracování (klasifikaci) krátkých rámců (64 – 96 B) je způsoben omezenou propustností externí paměti. Externí paměť QDR použitá na vývojové akcelerační kartě, na níž probíhalo testování, pracuje pouze na frekvenci 300 MHz a není schopná zpracovat takové množství požadavků, aby byla zajištěna plná propustnost při provádění klasifikace.

Modrá křivka vyjadřuje propustnost systému při výpočtu Netflow statistik ke každému síťovému toku generovaného provozu. Proces zpracování rámce pak zahrnuje nejen klasifikaci, ale také aktualizaci příslušného záznamu síťového toku v externí paměti. Právě vyšší počet přístupů do externí paměti při zpracování kratších rámců (64 – 480 B) je důvodem ještě výraznějšího poklesu výkonnosti. V rozsahu délek 192 – 480 B je horší propustnost

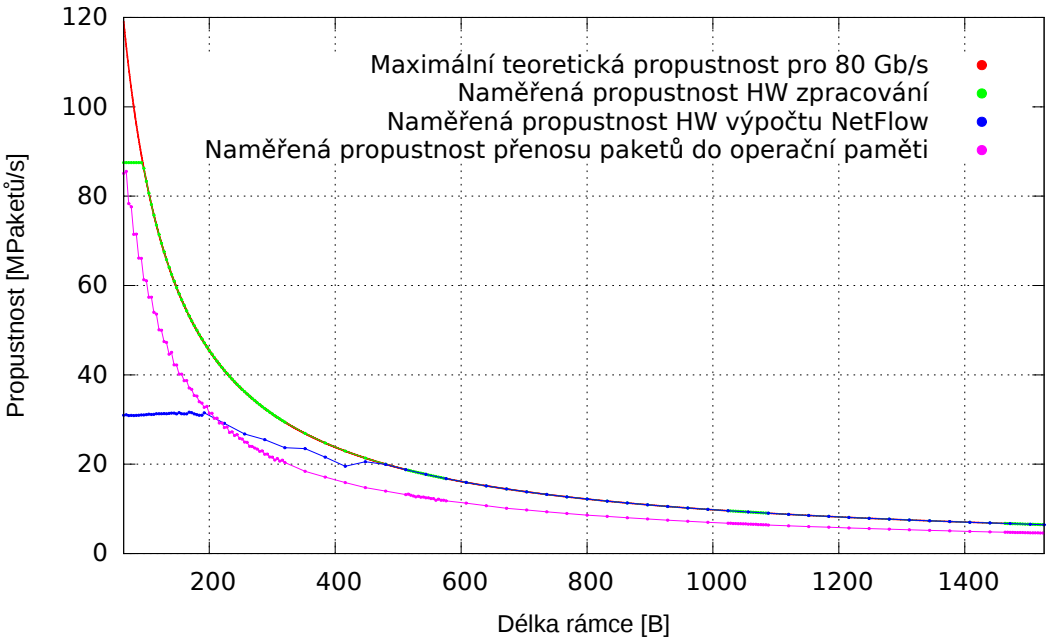
způsobena také dalším omezením – nedostatečnou kapacitou vyrovnávací FIFO paměti jednotky Packet Buffer. Tato chyba byla již úspěšně opravena a při opakovaném měření by měla zůstat zachována stejná lineární závislost propustnosti jako pro délky rámců do 192 B.

Pro delší rámce je maximální dosažená hodnota vždy 80 Gb/s. Skutečná maximální propustnost implementace však může být podle teoretického odhadu daleko vyšší. Z principu použité metody měření propustnosti však nelze tento předpoklad potvrdit.

Při přenosech celých, nezkrácených rámců až do operační paměti prostřednictvím 8 nezávislých DMA kanálů dosáhneme propustnosti v grafu vyznačené fialově, kde již narážíme na limit propustnosti systémové sběrnice PCI-Express.

Při přenosech pouze vyextrahovaných dat dochází především pro dlouhé rámce k velmi výrazné redukci objemu přenášených dat po systémové sběrnici a pozitivnímu vlivu na dosaženou propustnost systému. V grafu to již není speciálně zobrazeno, neboť výsledná křivka se nijak neliší od propustnosti čistě hardwarového zpracování (zobrazené zeleně).

Jiný pohled na naměřené hodnoty propustnosti zobrazuje graf na obrázku 6.2, kde je propustnost systému vyjádřena v počtu zpracovaných paketů za sekundu. Červená křivka



Obrázek 6.2: Propustnost systému v závislosti na délce rámce (v milionech paketů/s)

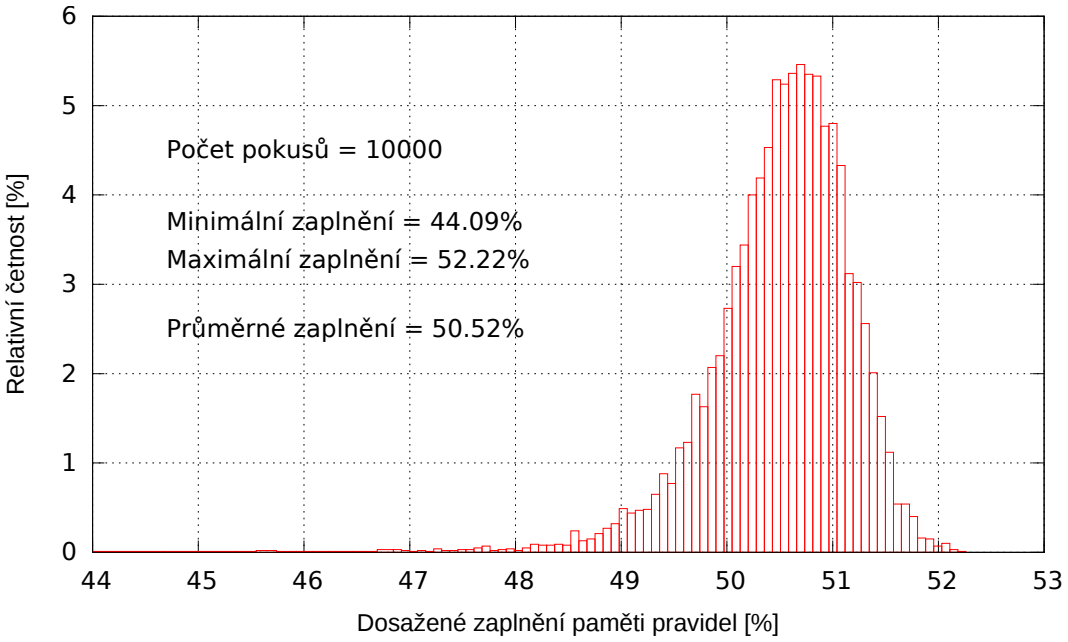
představuje propustnost 80 Gb/s přepočtenou na počet zpracovaných rámců dané délky. Význam zbylých křivek zůstává až na jiný způsob vyjádření propustnosti zachován. V tomto grafu lze lépe pozorovat omezení způsobené propustností paměti QDR – je znázorněno vodorovnými úseky. Vliv omezené propustnosti paměti QDR bude rovněž výrazně nižší v případě nasazení systému na reálné síti, neboť skladba délek rámců bude různá a podle provedené analýzy uvedené v [10] na reálné síti převládají (se zastoupení 57 %) velmi dlouhé rámce v rozsahu 1300 – 1526 B. Cílová akcelerační karta, podporující mimo jiné také 100G Ethernet, bude navíc osazena třemi moduly paměti QDRII+ pracujícími na vyšší frekvenci 500 MHz. Jeden z nich poskytne dostatečnou propustnost pro provádění klasifikace. Zbylé dva pak vhodně poslouží pro plně nezávislé uchovávání a správu záznamů o síťových tocích při zajištění plné propustnosti pro všechny přípustné délky ethernetových rámců. Dosažení maximální propustnosti 100 Gb/s na cílové akcelerační kartě je závislé také na tom, zda se

pro ni podaří vysyntetizovat firmware na dostatečné pracovní frekvenci jádra. V případě problémů by pak bylo potřeba přistoupit k dalším optimalizacím provedené implementace se zaměřením na části způsobující kritickou cestu.

6.2 Kapacita a latence paměti pravidel

Analýza dosahovaného zaplnění paměti pravidel probíhala zcela nezávisle na předchozím měření a bez nutnosti generování vstupního síťového provozu. Pro správu paměti pravidel je použita technika kukaččího hašování, která při vyhledávání pravidla přistupuje přímo na pozice určené dvěma hašovacími funkcemi a minimalizuje tak počet potřebných přístupů do externí paměti. Kolize vznikající při vkládání klasifikačních pravidel musí být řešeny vhodnou částečnou reorganizací paměti. Reorganizace však nemusí být úspěšná ve všech případech a mohou vnikat situace, kdy i přes volné pozice v externí paměti není vložení konkrétní položky možné.

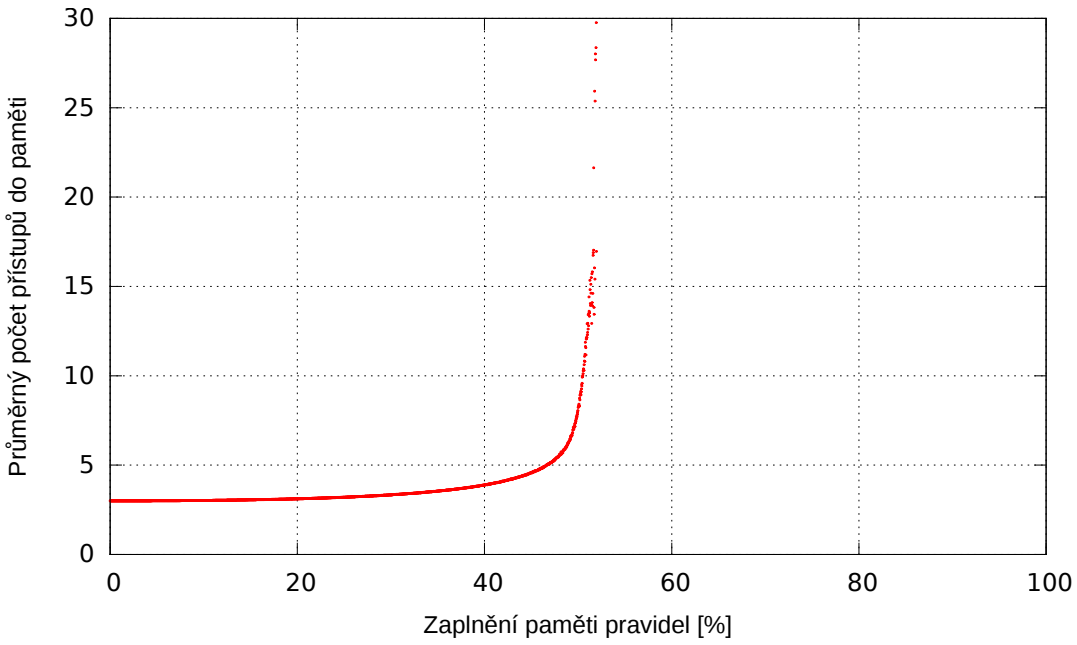
Teoretická kapacita paměti pravidel je na použité vývojové akcelerační kartě $2^{19} = 524\,288$ položek. Průběh měření vypadal tak, že bylo vždy vkládáno 524 288 klasifikačních pravidel s náhodně generovaným klíčem identifikujícím síťový tok. V případě, že jednotka SW Access nesignalizovala úspěch při umísťování aktuálně vkládaného pravidla do externí paměti v časovém limitu 1 sekunda, bylo vložení považováno na neúspěšné a pokus byl ukončen. Byla zaznamenána hodnota aktuálního stavu počtu vložených pravidel a celý experiment se opakoval. Firmwarové jádro bylo nakonfigurováno tak, aby používalo co nejjemnější rozlišení síťových toků a k jejich identifikaci se využívala jak dvojice IP adres, portů, tak identifikace transportního protokolu a vstupního síťového rozhraní. Pro dosažení maximálního rozsahu generovaných hodnot klíčů se záměrně používaly IPv6 adresy. Získané výsledky zobrazuje histogram na obrázku 6.3. Při provedeném počtu 10 000 pokusů bylo dosaženo průměrné zaplnění 50,52 % z celkové kapacity paměti, což odpovídá 264 870 úspěšně vloženým pravidlům. Dosahované zaplnění paměti se přitom pohybovalo v rozsahu



Obrázek 6.3: Zaplnění paměti pravidel při použití mechanismu kukaččího hašování

44,09 – 52,22 %. Naměřené hodnoty zcela odpovídají očekávaným výsledkům a provedené analýze v [10] pro metodu kukaččího hašování využívající dvě hašovací funkce. Dosahované zaplnění paměti pravidel však není nijak vysoké. Lepší výsledky by mohla umožnit například úprava návrhu doplňující mechanismus vyhledávání pravidla paralelní pamětí nižší kapacity, která by využívala jiný algoritmus a sloužila by pro odkládání kolizních pravidel.

Detailněji se procesem vkládání pravidel zabývá další z měření, které sleduje průměrný počet provedených přístupů do paměti na jedno vložení pravidla v závislosti na jejím aktuálním zaplnění. Tuto závislost zobrazuje graf na obrázku 6.4. Minimální počet přístupů



Obrázek 6.4: Průměrný počet přístupů do externí paměti v závislosti na míře jejího zaplnění při vkládání mechanismem kukaččího hašování

připadajících na jedno pravidlo jsou 3. Tento údaj vychází z mechanismu vkládání pravidla, kdy musí být vždy nejprve vyčteny hodnoty na dvou pozicích určených hašovacími funkcemi. Dále je ověřena jejich obsazenost a je proveden zápis na některou z nich. S každým krokem případně prováděné reorganizace paměti přibývají dva přístupy nutné pro vyčtení a zápis hodnoty alternativní pozice. Se vzrůstajícím zaplněním paměti roste náročnost a obecně klesá také úspěšnost prováděné reorganizace položek.

Měření bylo prováděno obdobně jako předchozí, opět jako pokus o vložení teoretického maximálního množství pravidel, pro náhodně generované identifikátory toků. Pro každé jednotlivé vložení byl ale navíc zaznamenán počet provedených přístupů do externí paměti. Uvedený postup byl opakován 10 000 krát a následně byl vypočten průměrný počet přístupů pro každé n-té vložení pravidla. Přesnost měření však se zaplněním paměti klesá, neboť vyšších zaplnění paměti bylo dosaženo v nižším počtu případů. Pro rozsah 0 až minimálně dosahované zaplnění paměti 44 % jsem následně změřil průměrnou dobu 3,25 μs potřebnou pro vložení jednoho pravidla. To odpovídá vložení přibližně 300 000 pravidel za sekundu. Hodnota byla určena na základě sledování celkové doby vložení cca 230 000 pravidel a jejím přepočtem pro jedno pravidlo. Údaj zahrnuje také režii spojenou s komunikací po systémové sběrnici a řídicím a konfiguračním rozhraní firmwaru.

Kapitola 7

Závěr

Cílem bakalářské práce byl návrh a implementace aplikačně specifického procesoru – firmwarové části systému umožňujícího hardwarovou akceleraci bezpečnostního monitorování vysokorychlostních sítí s využitím technologie FPGA. Návrh procesoru je založen na konceptu softwarově definovaného monitorování. V průběhu návrhu a při provádění implementace firmwaru byl kladen důraz především na dosažení požadované propustnosti zpracování síťových dat 100 Gb/s. Díky vhodnému rozložení činností mezi hardwarovou a softwarovou vrstvu poskytuje vytvořený systém nejen dostatečný výkon, ale i flexibilitu pro provádění vysokorychlostního monitorování síťových toků.

V rámci řešení bakalářská práce jsem se nejprve řádně seznámil s architekturou a základy dnešních počítačových sítí v podobě vrstvových modelů a principů použitých při síťové komunikaci. Nezbytným základem bylo také studium a detailní pochopení běžně používaných komunikačních protokolů a systému adresování a identifikace koncových uzlů a komunikujících aplikací. Další v pořadí byla problematika správy a bezpečnostního monitorování vysokorychlostních sítí na bázi síťových toků s ohledem na dosažení dostatečné výkonnosti. Trendem v oblasti hardwarové akcelerace zpracování síťových dat se stává použití programovatelných hradlových polí FPGA. Dále jsem se proto zaměřil na technologii FPGA Xilinx Virtex-7, specializovaný návrhový jazyk VHDL a možnost využití vývojové platformy NetCOPE. Teoretický rozbor jsem pak uzavřel částí věnovanou konceptu SDM, softwarově definovanému monitorování.

Na základě získaných poznatků jsem provedl podrobný implementační návrh aplikačně specifického procesoru pro potřeby bezpečnostního monitorování vysokorychlostních sítí. Návrh počítá s využitím externí paměti QDR pro ukládání klasifikačních pravidel a vytvářených záznamů o sledovaných síťových tocích a je proveden s ohledem na dodržení klíčových požadavků na výkonnost systému. Na základě podrobného návrhu jsem firmwarové jádro systému implementoval v jazyce VHDL, s využitím vývojové platformy NetCOPE. Provedenou implementaci navrženého jádra systému jsem pak otestoval v rámci základních simulací. Nad rámec zadání bakalářské práce jsem navíc provedl jeho podrobnou funkční verifikaci, včetně celého návrhu a implementace verifikačního prostředí a verifikačního modelu systému.

Začleněním jádra do infrastruktury platformy NetCOPE a následným překladem s pomocí syntézního nástroje Xilinx Vivado jsem sestavil firmware pro FPGA pro vývojovou akcelerační kartu Fiberblaze FB8XG@V7690. Na základě dále provedených optimalizací VHDL implementace se mně podařilo pro konfiguraci portů 8x10 Gb/s dosáhnout dostatečné pracovní frekvence firmwaru pro splnění plné propustnosti 80 Gb/s připojených vstupních síťových rozhraní.

Funkčnost procesoru jsem ověřil ve fyzické implementaci s využitím hardwarového testeru Spirent TestCenter umožňujícího generování a analýzu síťového provozu na plné propustnosti připojených linek. V laboratorním prostředí jsem pak provedl měření dosahované propustnosti systému. Zároveň jsem se zabýval analýzou dalších výkonových parametrů – dosahovanou efektivní kapacitou paměti klasifikačních pravidel při navrženém využití mechanismu kukaččího hašování, počtem prováděných přístupů do externí paměti a dobou potřebnou pro vložení pravidla do vyhledávací struktury. Nad rámec zadání práce jsem rovněž provedl návrh architektury jednotky vstupního síťového bloku, který rozšiřuje funkčnost platformy NetCOPE o dosud chybějící podporu příjmu ethernetových rámců na rychlosti 100 Gb/s. Zároveň jsem podle návrhu vytvořil jeho prvotní VHDL implementaci a realizovaný vstupní síťový blok úspěšně zverifikoval.

Výsledky práce byly prezentovány a publikovány v rámci konference Student EEICT [13]. Jejím plánovaným budoucím pokračováním je přenesení realizace firmwaru na akcelerační síťovou kartu podporující technologii Ethernet s přenosovou rychlostí 100 Gb/s. Současně rovněž probíhají ve spolupráci se sdružením CESNET přípravy na možné nasazení realizované varianty firmwaru 8x10 Gb/s na páteřní lince akademické sítě CESNET2 pro ověření a využití systému v praxi na reálné produkční síti. Jednou z možností pokračování je také další zvyšování efektivní kapacity paměti klasifikačních pravidel použitím například paralelní paměti nižší kapacity pro odkládání kolizních pravidel. Rozšiřovat lze také funkcionalitu systému například o podporu výpočtu jiných druhů statistik o síťových tocích a vůbec nových typů monitorovacích metod.

Seznam použitých zdrojů

[1] *CESNET, zájmové sdružení právnických osob* [online]. 1996–2014. [cit. 2014-02-15]. Dostupné na URL: http://www.cesnet.cz/

[2] CLAISE, B. *Cisco Systems NetFlow Services Export Version 9* [RFC 3954 (Informational)]. October 2004. Dostupné na URL: http://www.ietf.org/rfc/rfc3954.txt

[3] CLAISE, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Trafic Flow Information* [RFC 7011 (Internet Standard)]. September 2013. Dostupné na URL: http://www.ietf.org/rfc/rfc7011.txt

[4] DEERING, S. E., HINDEN, R. M. *Internet Protocol, Version 6 (IPv6) Specification* [RFC 2460 (Draft Standard)]. December 1998. Dostupné na URL: http://www.ietf.org/rfc/rfc2460.txt

[5] FIBERBLAZE. *Products: Unconfigured FPGA Cards* [online]. [cit. 2014-03-01]. Dostupné na URL: http://www.fiberblaze.com/products/unconfigured-fpga-cards.html

[6] IEEE Computer Society. *IEEE Standard VHDL Language Reference Manual* [IEEE Std 1076-2008]. January 2009, ISBN 978-0-7381-5800-6.

[7] IEEE Computer Society. *IEEE Standard for Ethernet* [IEEE Std 802.3-2012]. December 2012, ISBN 973-07381-7312-2.

[8] IEEE Computer Society. *IEEE Standard for SystemVerilog – Unified Hardware Design, Specification, and Verification Language* [IEEE Std 1800-2012]. February 2013, ISBN 978-0-7381-8110-3.

[9] JIROVSKÝ, V. *Vademecum správce sítě*. První vydání. Praha: Grada, 2001. 428 s. ISBN 80-7169-745-1.

[10] KEKELY, L. *Hardwarová akcelerace aplikací pro monitorování a bezpečnost vysokorychlostních sítí*. Diplomová práce. Brno: FIT VUT v Brně, 2013.

[11] KEKELY, L., PUŠ, V., KOŘENEK, J. Software Defined Monitoring of Application Protocols. *Proceedings of the 33rd IEEE International Conference on Computer Communications*, INFOCOM 2014. Toronto, Canada, 2014.

[12] KORČEK, P., KOŠAŘ, V., ŽÁDNÍK, M., aj. Hacking NetCOPE to run on NetFPGA-10G. *Proceedings of the 7th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS 2011. Brooklyn, New York, USA, 2011, s. 217–218. ISBN 978-0-7695-4521-9.

[13] KUČERA, J. Application-specific Processor for Stateful Network Traffic Processing. *Proceedings of the 20th Student Conference on Electrical Engineering, Information and Communication Technologies*, Student EEICT 2014. Brno, 2014, s. 198–200. ISBN 978-80-214-4922-0.

[14] LIBEROUTER / CESNET TMC group. *Technologies* [online]. [cit. 2014-02-15]. Dostupné na URL: https://www.liberouter.org/technologies/

[15] MARTÍNEK, T., KOŠEK, 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, s. 1–6. ISBN 978-1-4244-2277-7.

[16] MATOUŠEK, P. *Síťové aplikace a správa sítí*. Brno: VUTIUM, 2014. 396 s. ISBN 978-80-214-3766-1.

[17] PAGH, R., RODLER, F. F. Cuckoo Hashing. *Proceedings of the 9th Annual European Symposium on Algorithms*, ESA 2001. Aarhus, Denmark: Springer, 2001, s. 121–133. ISBN 978-3-540-42493-2.

[18] POSTEL, J. *User Datagram Protocol* [RFC 768 (Internet Standard)]. August 1980. Dostupné na URL: http://www.ietf.org/rfc/rfc768.txt

[19] POSTEL, J. *Internet Protocol* [RFC 791 (Internet Standard)]. September 1981. Dostupné na URL: http://www.ietf.org/rfc/rfc791.txt

[20] POSTEL, J. *Transmission Control Protocol* [RFC 793 (Internet Standard)]. September 1981. Dostupné na URL: http://www.ietf.org/rfc/rfc793.txt

[21] PUŠ, V., KEKELY, L., KOŘENEK, J. Low-latency Modular Packet Header Parser for FPGA. *Proceedings of the 8th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS 2012. New York, USA: ACM, 2012, s. 77–78. ISBN 978-1-4503-1685-9.

[22] RAMAKRISHNAN, K., FLOYD, S., BLACK, D. *The Addition of Explicit Congestion Notification (ECN) to IP* [RFC 3168 (Proposed Standard)]. September 2001. Dostupné na URL: http://www.ietf.org/rfc/rfc3168.txt

[23] SEKANINA, L. *Evoluční hardware: od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. První vydání. Praha: Academia, 2009. 321 s. ISBN 978-80-200-1729-1.

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

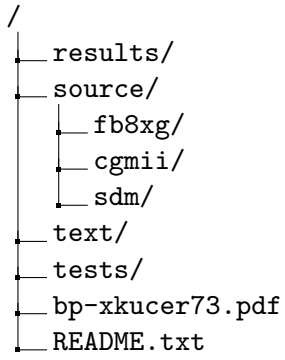
[25] SPRING, N., WETHERALL, D., ELY, D. *Robust Explicit Congestion Notification (ECN) Signaling with Nonces* [RFC 3540 (Experimental)]. June 2003. Dostupné na URL: http://www.ietf.org/rfc/rfc3540.txt

[26] ŠIMKOVÁ, M. *Hardwarově akcelerovaná funkční verifikace*. Diplomová práce. Brno: FIT VUT v Brně, 2011.

[27] XILINX. *All Programmable FPGAs* [online]. 2014. [cit. 2014-02-18]. Dostupné na URL: http://www.xilinx.com/products/silicon-devices/fpga/index.htm

Příloha A

Obsah DVD



Adresář `results/` obsahuje vytvořený bitstream pro FPGA pro vývojovou akcelerační kartu Fiberblaze FB8XG@V7690. Součástí jsou také soubory s naměřenými hodnotami při hardwarovém testování a shellové skripty pro zpracování hodnot a vykreslení grafů s pomocí nástroje `gnuplot`.

Adresář `source/` obsahuje zdrojové kódy implementovaných VHDL komponent, včetně vytvořených simulačních a verifikačních prostředí a potřebných součástí pro opětovné provedení syntézy. Jednotlivé podadresáře `fb8xg/`, `cgmii/` a `sdm/` obsahují popořadě zdrojové soubory firmwaru pro vývojovou akcelerační kartu FB8XG@V7690, implementaci vstupního síťového bloku a implementaci aplikačně specifického procesoru (implementaci aplikačního jádra firmwaru FPGA).

Adresář `text/` obsahuje zdrojové soubory textu bakalářské práce pro její možnou úpravu a opětovné vysázení systémem LaTeX, včetně zdrojových souborů použitých obrázků.

Adresář `tests/` obsahuje konfigurační a zdrojové soubory nástrojů a skriptů vytvořených nebo použitých pro účely testování a automatizaci prováděných měření.

Soubor `bp-xkucer73.pdf` obsahuje vysázený text bakalářské práce ve formátu PDF.

Soubor `README.txt` obsahuje informace o adresářové struktuře a obsahu přiloženého DVD. Dále poskytuje podrobné pokyny pro umístění implementovaných komponent do adresářové struktury platformy NetCOPE a instrukce pro využití překladového systému platformy pro spuštění procesu syntézy, simulace nebo verifikace firmwaru.