VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

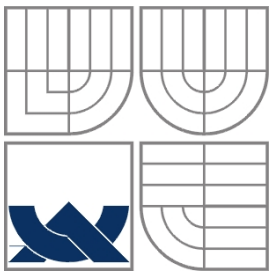
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PLATFORMA PRO RYCHLÝ VÝVOJ SÍŤOVÝCH ZAŘÍZENÍ

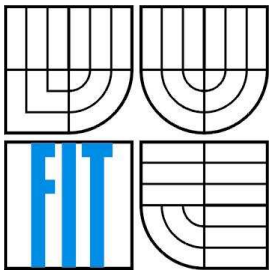
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JIŘÍ TOBOLA
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PLATFORMA PRO RYCHLÝ VÝVOJ SÍŤOVÝCH ZAŘÍZENÍ

PLATFORM FOR RAPID DEVELOPMENT OF NETWORK DEVICES

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE — Bc. JIŘÍ TOBOLA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KOŘENEK
SUPERVISOR

BRNO 2007

Zadání diplomové práce

1. Seznamte se s hardwarovou architekturou karet rodiny COMBO a nastudujte problematiku návrhu síťových zařízení.
2. Navrhněte architekturu platformy, která využije potenciál COMBO karet a umožní rychlý vývoj síťových zařízení.
3. Vytvořte popis platformy v jazyce VHDL s ohledem na syntézu do FPGA.
4. S využitím navržené platformy vytvořte dvě různá síťová zařízení, které budou demonstrovat možnosti navrženého řešení. Jako síťové zařízení můžete zvolit různé monitorovací nebo měřicí systémy, případně prvky síťové infrastruktury.
5. Funkci obou zařízení ověřte na kartách rodiny COMBO.
6. Zhodnoťte dosažené výsledky a možnosti využití navržené platformy v různých aplikacích.

Licenční smlouva

Licenční smlouva poskytovaná k výkonu práva užít školní dílo je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato práce popisuje návrh a implementaci platformy pro rychlý vývoj síťových zařízení na rodině karet COMBO. Navržená platforma zahrnuje jednotný protokol pro přenos dat ve formě paketů, sadu nástrojů pro manipulaci s daty ve formátu tohoto protokolu, vstupní a výstupní síťové bloky pro gigabitový ethernet, vysokorychlostní propojení se softwarovou vrstvou přes systémové sběrnice PCI, PCI-X nebo PCI-Express a množinu bloků pro analýzu a zpracování síťového provozu. Využití navržené platformy je demonstrováno při návrhu a implementaci síťové karty, hardwarového firewallu a exportéru unifikovaných hlaviček paketů.

Klíčová slova

Platforma pro síťové aplikace, NetCOPE, FPGA, VHDL, síť

Abstract

This thesis deals with the design and implementation of an FPGA-based platform for rapid development of network applications for the COMBO cards family. The proposed platform includes a generic data transfer protocol – FrameLink, a set of tools for FrameLink manipulation, network interface blocks for 1 Gigabit Ethernet, high-speed connection to the software layer via PCI, PCI-X or PCI Express bus and a set of IP cores for network traffic analysis and processing. The benefits of the proposed platform are demonstrated on design and implementation of a network interface card, hardware firewall and exporter of unified packet headers.

Keywords

Platform for network applications, NetCOPE, FPGA, VHDL, network

Citace

Jiří Tobola: Platforma pro rychlý vývoj síťových zařízení, diplomová práce, FIT VUT v Brně, 2007.

Platforma pro rychlý vývoj síťových zařízení

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jana Kořenka. Další informace mi poskytli kolegové z projektu Liberouter. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Tobola
22. května 2007

Poděkování

Především bych rád poděkoval vedoucímu své diplomové práce panu Ing. Janu Kořenkovi za odborné vedení a čas věnovaný konzultacím této práce. Také bych chtěl poděkovat kolegům z projektu Liberouter za poskytnutí informací a zajištění technické podpory při návrhu a implementaci.

Obsah

Kapitola 1

Úvod

Rozvoj počítačových sítí a zvláště Internetu přináší stále rychlejší technologie pro přenos dat. Zároveň s rychlostmi se v poslední době zvyšují i nároky na bezpečnost a sledování sítí. Právě vysoké přenosové rychlosti ale přinášejí problém, jak data efektivně zpracovávat, monitorovat a analyzovat. Na rychlostech linek v řádech jednotek až desítek gigabitů za vteřinu musí být řešeny úlohy směrování, filtrování dat, monitorování síťových toků, či výpočetně velmi náročné prohledávání obsahu paketů při detekci nebezpečného síťového provozu. Zatímco pro nižší přenosové rychlosti mohou být tyto úlohy řešeny v programovém vybavení počítače – softwaru, při vyšších rychlostech výkonnostně nedostačují ani dedikované servery založené na platformě PC. Limitujícími prvky standardních řešení jsou zejména výkonnost procesoru a také rychlost přenosu po systémové sběrnici.

Typickým řešením problému s nedostatečnou výkonností procesoru je hardwarová akcelerace dané úlohy. Příkladem běžně používaných hardwarově akcelerovaných zařízení pro síťovou oblast jsou směrovače nebo hardwarové firewally. Obecně mohou být tato hardwarová řešení založena na technologiích ASIC (Application-Specific Integrated Circuit) či FPGA (Field Programmable Gate Array). Zatímco ASIC je integrovaný obvod speciálně navržený pro konkrétní aplikaci a jeho struktura je pevně vytvořena při výrobě, FPGA je programovatelné pole hradel a jeho konfiguraci je možné měnit. Obě technologie nabízejí potencionálně daleko vyšší výkonnost pro určitou aplikaci než obecné procesory. S technologií ASIC lze většinou dosáhnout vyšší výkonnosti a menší spotřeby elektrické energie, technologie FPGA je naopak díky možnosti rekonfigurace flexibilnější a nabízí jistý kompromis mezi výkonem ASIC obvodů a flexibilitou obecných procesorů.

Komerční výrobky jsou ve většině případů založené na technologii ASIC. Toto řešení umožňuje dosahovat vysokou výkonnost a při velkém počtu vyrobených kusů i nízkou cenu. Specializovaná vysokorychlostní síťová zařízení se ale většinou ve velkých sériích nevyrábějí, a drahý vývoj zařízení se odráží ve vysokých cenách finálních výrobků. Navíc každá změna v návrhu znamená další náklady při výrobě nových masek. Proto se nejprve v akademické a následně i komerční sféře začala zkoumat možnost využití technologie FPGA.

Praktické výsledky ukazují, že použití této technologie je v určitých případech velmi výhodné. Hlavními důvody jsou snížení doby vývoje zařízení, snížení jeho ceny a možnost využití aktualizace funkce změnou konfigurace. Tyto aktualizace se s výhodou uplatňují zejména při opravě chyb a při zahrnování nových síťových standardů. Úspěšným příkladem využití technologie FPGA je výzkumný akademický projekt *Programmable Hardware* [1] nebo evropské projekty *6NET* [2], *SCAMPI* [3] či *GEANT2* [4].

Cílem této práce je navrhnout a implementovat obecnou generickou platformu pro rychlý vývoj síťových zařízení založených na technologii FPGA. Platforma poskytne vývojářům abstrakci od použitých hardwarových prostředků a nabídne řešení základních úloh vyskytujících se při implementaci síťových aplikací. Vlastnosti platformy a možnosti jejího využití budou demonstrovány při návrhu a implementaci tří síťových zařízení. Konkrétními hardwarovými prostředky pro realizaci návrhu bude rodina karet COMBO [5].

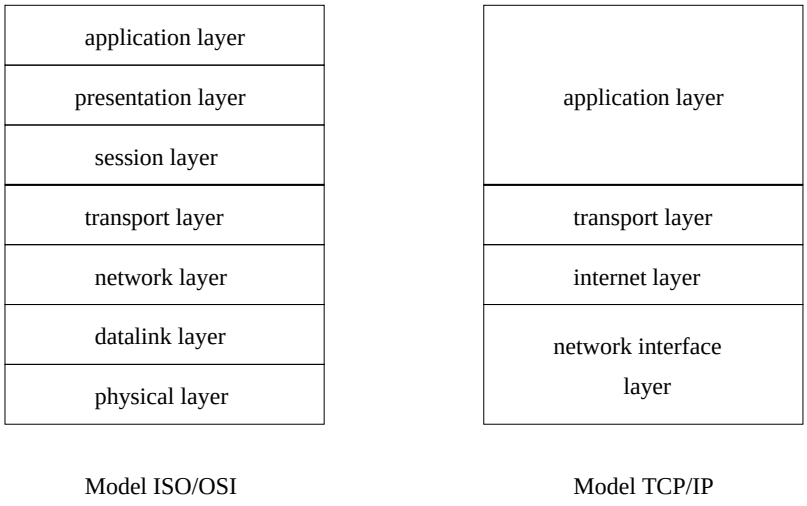
Práce je dále členěna do několika kapitol, které jsou organizovány následovně. Kapitola 2 shrnuje základní teoretické poznatky z oboru počítačových sítí důležité při návrhu platformy, představuje rodinu karet COMBO a uvádí do problematiky návrhu síťových zařízení. V kapitole 3 je prezentován návrh generické platformy pro vývoj síťových aplikací na rodině karet COMBO a v následující kapitole jsou představena tři síťová zařízení vyvinutá na navržené platformě. Závěrečnou kapitolu 5 tvoří shrnutí dosažených výsledků a zhodnocení architektury z hlediska dalšího vývoje.

Kapitola 2

Teoretický rozbor

2.1 Síťové modely ISO/OSI a TCP/IP

Pro řízení síťových komunikací se od samotného vzniku počítačových sítí používá princip vrstev. Síťové architektury jsou popisovány vrstvovými modely a v rámci příslušných vrstev jsou definovány služby a protokoly pro zajištění komunikace. Zpočátku vznikaly firemní uzavřené síťové architektury, které ovšem neumožňovaly propojování sítí různých výrobců. Logicky tedy vznikl tlak na otevřenost síťových architektur, který vyústil přijetím referenčního modelu ISO/OSI. Tento standard pro propojení otevřených systémů byl vytvořen mezinárodní organizací ISO (International Standarts Organization) a následně přijat i Mezinárodní telekomunikační unií (International Telecommunications Union, ITU).



Obrázek 2.1: Referenční model ISO/OSI a model TCP/IP.

Standard ISO/OSI nespecifikuje implementaci systémů, ale uvádí všeobecné principy sedmivrstvé síťové architektury. Jedná se zejména o účel vrstev, funkce přiřazené vrstvám, služby poskytované vyšší vrstvě a služby požadované od vrstvy nižší. Schéma tohoto modelu je uvedeno na obrázku 2.1. Funkce jednotlivých vrstev jsou popsány v následující bodech.

Fyzická vrstva se zabývá přenosem bitů komunikačním kanálem, ale nevěnuje už pozornost významu přenášených bitů. Definuje úrovně signálu pro nulové a jedničkové bity, rychlost přenosu jednoho bitu, přenosové médium a další parametry, které určují mechanické a elektrické rozhraní.

Linková vrstva vytváří nad fyzickou vrstvou datový spoj. Standardně je datový spoj vytvářen mezi bezprostředními sousedy, tedy na dvoubodovém spoji. Linková vrstva organizuje bitový proud fyzické vrstvy do rámců, což jsou obvykle bloky dat o velikosti desítek až tisíců bajtů. Rámec má svou přesnou strukturu a je v něm obvykle uvedena zdrojová a cílová adresa rámce. Vrstva umožňuje rozpoznávat chybně přenesené rámce a zajistit jejich opravný přenos.

Síťová vrstva řeší směrování toku dat v síti od odesilatele k adresátovi. Přenos může probíhat přes několik mezilehlých uzlů. Data jsou pro účely směrování dělena na pakety. Pro zajištění toku dat sítí je potřeba jednoznačná identifikace uzlu, která je nezávislá na fyzické adresaci. Této identifikaci se říká síťová adresa a je přiřazována všem uzlům sítě. Služby síťové vrstvy se dělí na spojované a na nespojované. Oba typy služeb mohou přitom být spolehlivé či nespolehlivé. Zatímco ve spolehlivých sítích je určitým mechanismem zaručeno spolehlivé doručení dat, u nespolehlivých se o tuto funkci stará až vyšší vrstva modelu ISO/OSI.

Transportní vrstva zaručuje adresování koncových komunikujících prvků, které působí v jednotlivých uzlech sítě. Mezi tyto prvky patří například procesy nebo uživatelské relace. Data jsou přenášena po segmentech, ve kterých je uložena adresa koncového prvku uzlu. Na úrovni síťové vrstvy jsou segmenty rozděleny na pakety, do kterých se přidají adresy komunikujících uzlů. Po uskutečnění přenosu jsou segmenty z paketů opět sestaveny. Na úrovni transportní vrstvy mohou existovat spojované i nespojované služby. U spojovaných služeb je v této vrstvě zajištěno ustavení komunikace a spolehlivé doručení paketů. Naopak u nespojovaných služeb se o spolehlivé doručení dat musí postarat přímo aplikace.

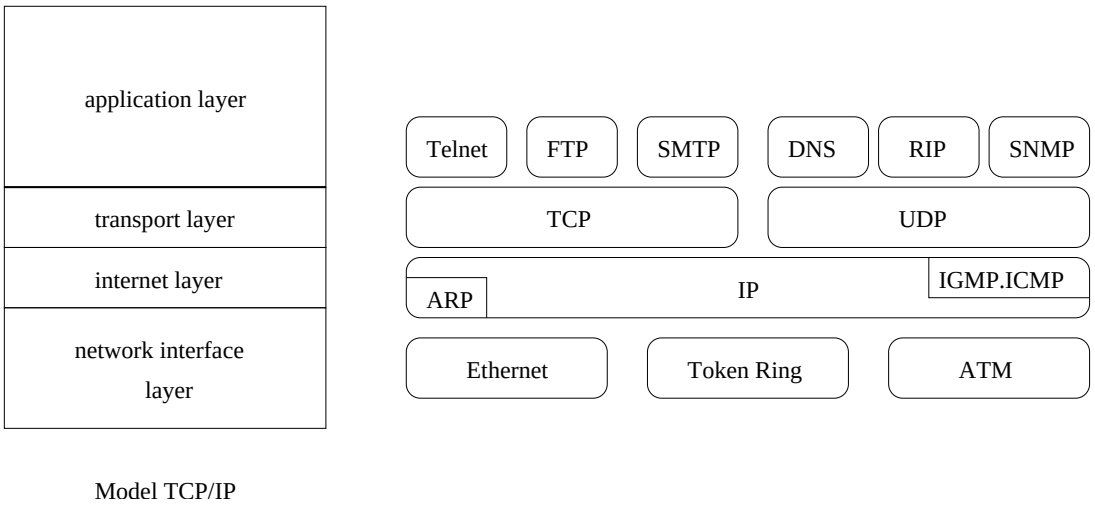
Relační vrstva umožňuje, aby si procesy nebo uživatelé v různých uzlech ustanovili mezi sebou relace a následně mohli koordinovat výměnu dat mezi těmito uzly.

Prezentační vrstva obsahuje funkce, které jsou prováděny tak často, že je pro ně vhodné mít obecné řešení. Uživatelé pak nemusí tyto funkce řešit ve vlastní režii. Mezi funkce prezentační vrstvy patří typicky šifrování dat, převod mezi různými kódováními nebo komprimace.

Aplikační vrstva realizuje aplikačně orientované služby. Poskytuje různá aplikační rozhraní jako například služby pro implementaci elektronické pošty, přenosu souboru nebo elektronického obchodu. Součásti této vrstvy bývají přímo procesy, které tyto aplikační funkce plní.

Přestože je referenční model ISO/OSI mezinárodně standardizován a uznáván, hlavní roli v oblasti síťových architektur hraje v současnosti model TCP/IP. Tento model je od modelu ISO/OSI odvozen, ale nabízí jednodušší a efektivnější architekturu rozdělenou pouze do čtyř vrstev. Srovnání obou zmiňovaných modelů znázorňující vzájemně si odpovídající vrstvy je uvedeno na obrázku 2.1.

Rodina protokolů TCP/IP vznikla v době počátků Internetu. K prvotní síti ARPANET se připojovaly další sítě často fungující na jiných technologických platformách a bylo nutné sjednotit adresování a propojování sítí. Za tímto účelem vznikl model TCP/IP. Hlavní koncepce tohoto modelu jsou abstrakce od fyzické a linkové vrstvy, definice protokolu síťové vrstvy IP (Internet Protocol) pro jednotné adresování, definice převodních mechanismů mezi linkovou a síťovou vrstvou (protokol ARP) a definice transportních protokolů TCP a UDP. Schéma modelu TCP/IP je zachyceno na obrázku 2.2. Tento model byl primárně navržen pro použití v Internetu a z toho vycházejí jeho vlastnosti. Model neobsahuje žádné centrální části, je velice robustní a je odolný vůči výpadkům částí sítě.



Obrázek 2.2: Síťový model TCP/IP.

Vrstva síťového rozhraní není modelem TCP/IP definována a předpokládá se použití různých přenosových technologií. Příklady těchto technologií jsou Token Ring, ATM, Sonet nebo dále popisovaný Ethernet. TCP/IP se pouze zabývá tím, jak tyto technologie co nejlépe využít.

Síťová vrstva zakrývá rozdíly mezi jednotlivými přenosovými technologiemi – různými způsoby adresování, velikostmi rámců a charakterem poskytovaných služeb. Cílem je vytvoření jednotného komunikačního prostředí nezávislého na konkrétních technologiích přenosu dat. Pro tuto vrstvu je definován protokol IP (Internet Protocol), který zajišťuje výše uvedené požadavky. V současné době se používají dvě verze IP protokolu: rozšířená verze 4 (IPv4) a moderní verze 6 (IPv6), kterou masovější rozšíření teprve čeká. Podrobněji jsou obě verze popsány v následujících kapitolách.

Transportní vrstva přizpůsobuje možnosti nižších vrstev požadavkům vyšších vrstev, může zajišťovat spojovanou komunikaci a spolehlivost. Nejčastěji používané protokoly této vrstvy jsou TCP a UDP.

Aplikační vrstva je nejvyšší vrstvou síťové architektury Internetu. Patří do ní uživatelské aplikace (HTTP, SMTP, FTP, telnet a další) a administrativní aplikace (DNS, DHCP, SNMP a další).

2.2 Linková vrstva a Ethernet

Z hlediska implementace obecné platformy pro vývoj síťových zařízení hraje důležitou roli linková vrstva modelu ISO/OSI. Platforma musí poskytnout automatický příjem a vysílání dat na této vrstvě. Data vyšších vrstev budou z hlediska platformy pouze propagována do jádra aplikace. Mezi protokoly pracující na linkové vrstvě patří Ethernet, Token Ring, ATM, Sonet či SDH. Nejrozšířenějším z těchto protokolů pro lokální a metropolitní sítě je technologie Ethernet [6], která je v této kapitole stručně představena a bude využita v navrhované platformě.

Technologie Ethernet v referenčním modelu ISO/OSI pokrývá fyzickou a linkovou vrstvu, v rámci TCP/IP modelu spadá do vrstvy síťového rozhraní. Norma IEEE 802.3, kterou je Ethernet specifikován, definuje komunikaci mezi dvěma sousedními uzly sítě, řeší problémy přístupu na přenosové médium, vymezuje typy přenosových médií, typy kódování dat, napěťové úrovně, vlnové délky u laserových spojů, konektory, maximální dosahy, přenosovou rychlost atd. Jako média pro přenos jsou definovány koaxiální kabely, kroucené dvojlinky, optická vlákna či bezdrátové spoje.

První Ethernet standard definoval přenosová rychlost 10 Mb/s a metodou přístupu ke společnému médiu CSMA/CD, tzn. hromadný přístup k médiu s detekcí kolizí. V roce 1995 byla přijat standard Fast Ethernet zvyšující přenosovou rychlost na 100 Mb/s a zachovávající stejnou metodu pro přístup k médiu, v roce 1998 byl standardizován gigabitový ethernet a v roce 2002 desetigigabitová verze tohoto protokolu. V gigabitové verzi je pomalu opouštěna metoda CSMA/CD a prakticky se využívá pouze plně duplexní přenos. V desetigigabitové verzi už jiný než plně duplexní způsob přenosu není definován. Na příští verzi se začalo pracovat v roce 2006 a výsledkem by měla být specifikace stogigabitového Ethernetu. Pro navrhovanou platformu jsou relevantní gigabitová a desetigigabitová verze této technologie.

Data jsou v ethernetu přenášena ve formě rámců, přičemž v současné době se používají dva typy. Standardní ethernetový rámec dle IEEE 802.3 a ethernetový rámec s VLAN tagem dle IEEE 802.1q [7]. Struktura obou rámců je znázorněna na obrázku 2.3 a stručný popis jednotlivých polí je uveden v následujícím seznamu.

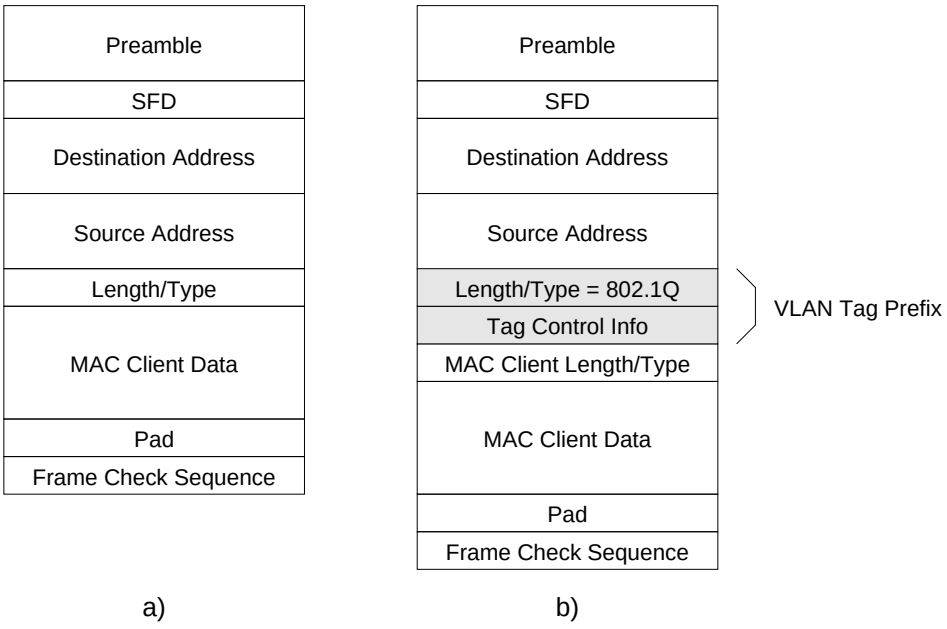
Preamble (7 bajtů) – vzorek dat, který zajišťuje časovou synchronizaci. Je tvořen střídajícími se hodnotami 0 a 1.

Starting Frame Delimiter (1 bajt) – identifikace začátku rámce. Je podobně jako Preamble tvořen posloupností střídajících se nul a jedniček, jen poslední nula je nahrazena hodnotou jedna. Vždy má tedy hodnotu *10101011*.

Destination Address (6 bajtů) – adresa cílového uzlu. Může se jednat o individuální adresu konkrétního uzlu, skupinovou (multicastovou) adresu více uzlů nebo všeobecnou (broadcastovou) adresu.

Source Address (6 bajtů) – adresa zdrojového uzlu, vždy individuální.

Length/Type (2 bajty) – význam položky se mění s hodnotou, která je v ní uložena. Pokud je hodnota vyšší než 600 hexadecimálně, obsahuje typ dat vyšší vrstvy. V opačném případě se jedná o velikost dat.



Obrázek 2.3: a) Standardní ethernetový rámec, b) ethernetový rámec s VLAN tagem.

MAC Client Data a Pad (46 až 1500 bajtů) – data protokolu vyšších vrstev. Pokud je velikost dat menší než je minimální požadovaná délka 46B, rámec je rozšířen o položku Pad obsahující libovolná data. Tato položka zajišťuje minimální délku nutnou pro správnou funkci metody CSMA/CD.

Frame Check Sequence (4 bajty) – dvaatřicetibitový cyklický kontrolní kód, který se počítá přes všechna pole kromě Preamble, SFD a FCS.

Ethernetový rámec s VLAN tagem vznikl s vytvořením konceptu *virtuálních lokálních sítí* [7]. Konstrukce virtuálních sítí umožňuje vytvoření několika logických lokálních sítí v rámci jedné fyzické. Hlavní výhodou tohoto přístupu je snadná rekonfigurace a údržba sítí. Pro rozlišení jednotlivých logických sítí a paketů, které k nim náleží, se používá rozšířený formát ethernetového rámce (viz obrázek 2.3b). Položka Length/Type obsahuje hodnotu 8100 (hexadecimálně) značící typ rámce 802.1q. Původní položka Lenght/Type je posunuta za položku Tag Control Info. Tato nová položka o velikosti 2 bajty obsahuje řídicí informace pro VLAN jako jsou priorita či VLAN identifikátor sloužící pro rozlišení mezi jednotlivými logickými sítěmi.

2.3 Síťová vrstva a IP protokol

Hlavním účelem síťové vrstvy je poskytnout síťové spojení otevřeným systémům, které spolu chtějí komunikovat a přitom spolu přímo nesousedí. Na základě síťové adresace je tato vrstva zodpovědná za vlastní komunikaci v komplexní síti, směrování a přenos datových jednotek označovaných jako pakety od zdroje k cíli. Síťová služba může být se spojením (spolehlivá) nebo bez spojení (nespolehlivá, datagramová). V modelu TCP/IP je pro síťovou vrstvu

definován jednotný datagramový protokol – Internet Protocol (IP). V současné době je přitom dominantní původní verze tohoto protokolu označovaná jako IPv4 [8], v budoucnosti se předpokládá rozšíření novější verze IPv6 [9].

Protože IP protokol tvoří spolu s protokoly transportních vrstev základ Internetu, budou v této a následující kapitole zmíněné protokoly stručně představeny. Tyto protokoly a jejich formáty paketů hrají hlavní roli při konstrukci síťových zařízení – od směrovačů, přes firewally a IDS/IPS systémy po monitorovací adaptéry, které často sledují právě toky (flow) na základě IP adres a portů transportní vrstvy.

2.3.1 IPv4

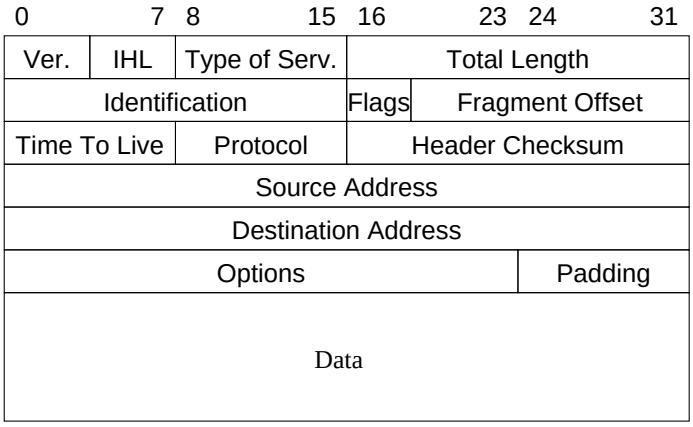
Pro zajištění základních funkcí síťové vrstvy je nutné definovat schéma adresace a formát datagramu pro přenos datových jednotek. Dalšími funkcemi síťové vrstvy jsou řízení provozu a podpora skupinového vysílání. Za účelem splnění těchto požadavků byly definovány následující protokoly:

- Internet Protocol, IP, RFC 791
- protokol mapování adres – Address Resolution Protocol, ARP, RFC 826
- protokol obráceného mapování adres – Reverse ARP, RARP, RFC 903
- protokol řídicích hlášení – Internet Control Message Protocol, ICMP, RFC 792
- protokol správy skupin – Internet Group Management Protocol, IGMP, RFC 3376
- směrovací protokoly OSPF a IGRP

Dále se budeme věnovat pouze vlastnímu IP protokolu. Popis zbývajících protokolů této vrstvy je možné nalézt v příslušných RFC dokumentech nebo například v [10]. IP protokol realizuje vysílání datagramů na základě síťových adres obsažených v jejich záhlavích. Protokol poskytuje síťovou službu bez spojení. Nenavazuje tedy spojení pro přenos datagramů a neudržuje informace o předávaných datagramech. Každý datagram je samostatná datová jednotka, která musí obsahovat všechny informace o adresátovi i odesílateli. Při použití fragmentace obsahuje také číslo pořadí ve zprávě, neboť datagramy jsou odesílány nezávisle na sobě a pořadí jejich doručení nemusí odpovídat jejich původnímu pořadí.

Fragmentace a následné znovusestavení datagramů patří mezi další funkce síťové vrstvy. K fragmentaci dochází pokud je nejvyšší povolená velikost datové jednotky nižší vrstvy menší než celková velikost IP datagramu. Každý fragment má své IP záhlaví, v němž se opakují hodnoty původního datagramu v polích zdrojové a cílové adresy IP adresy, identifikace a čísla protokolu. Pro rozlišení pořadí fragmentů se používá ve formátu datagramu IPv4 pole fragment offset. Fragmentace se provádí na směrovačích a to i vícenásobně, opětovné sestavování fragmentů se realizuje na cílové stanici.

Internet protokol je nespolehlivá služba, protože nezaručuje doručení datagramu. Služba pracuje v režimu *best effort* s maximální vůlí datagram doručit. IP nemá zabudovaný žádný mechanismus pro detekci a korekci chyb v přenášených datech. Datagramy se při přenosu mohou ztratit, duplikovat, přijít v jiném pořadí nebo se mohou porušit přenášená data, aniž by to IP služba detekovala. V modelu TCP/IP je případné zajištění spolehlivosti přenosu ponecháno na vyšší transportní vrstvě. Formát datagramu protokolu IPv4 je znázorněn na obrázku 2.4.



Obrázek 2.4: Formát IPv4 datagramu.

Version (4 bity) – definice verze protokolu. Pro IPv4 je tato hodnota rovna číslu 4.

Internet Header Length (4 bity) – počet 32-bitových slov v IP hlavičce.

Type of Service (8 bitů) – pole nesoucí informaci o požadované kvalitě při přenosu.

Total length (16 bitů) – délka celého datagramu (včetně hlavičky) v bajtech.

Identification (16 bitů) – identifikace datagramu pro podporu fragmentace.

Flags (3 bity) – pole využíváno pro mechanismus fragmentace. Obsahuje bity pro zákaz fragmentace, označení posledního fragmentu datagramu a označení že budou následovat další fragmenty.

Fragment Offset (13 bitů) – pořadí fragmentu v rámci datagramu, používá se vzdálenost začátku pole dat fragmentu od začátku původního datagramu v násobcích 64 bitů.

Time To Live (8 bitů) – počítadlo nastavené ve zdrojové stanici a snižované při průchodu směrovačem. Pokud jeho hodnota dosáhne nuly, je datagram zničen a je vygenerováno chybové hlášení ICMP protokolu. Tento mechanismus zajišťuje, že nedojde k nekonečnému bloudění datagramu na síti.

Protocol (8 bitů) – číslo protokolu vyšší vrstvy. Například protokol ICMP se označuje číslem 1, TCP číslem 6, UDP číslem 17.

Header Checksum (16 bitů) – kontrolní součet zabezpečující záhlaví proti chybám (zabezpečení dat v protokolu IP chybí).

Source Address (32 bitů) – adresa zdrojového uzlu (vždy individuální).

Destination Address (32 bitů) – adresa cílového uzlu (individuální, skupinová nebo všeobecná adresa).

Options – obsahuje volitelné položky IPv4 hlavičky, např. pro zabezpečení, záznam cesty sítí nebo dodržení předepsané cesty. Vzhledem ke složitosti zpracování se rozšiřující hlavičky příliš nepoužívají a běžně je využívána minimální délka IP hlavičky – 20 oktetů. Bližší informace o významu volitelných položek jsou uvedeny v [8].

Padding – zarovnání hlavičky na násobek 32 bitů. Položka složena z nulových bitů.

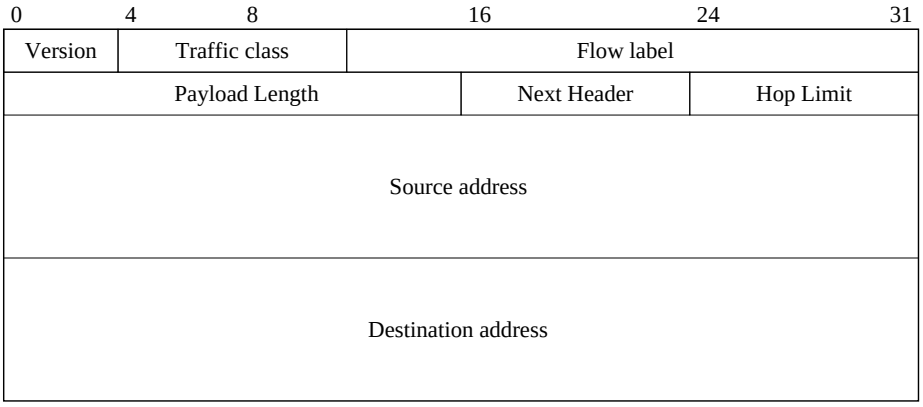
Data – informace vyšší vrstvy, délka dat je maximálně (65535 − délka záhlaví) oktetů.

Mimo formát datagramu protokol IP definuje způsob adresace, rozdělení adres do tříd, skupinové vysílání (multicast), protokoly pro mapování adres (ARP a RARP) či protokol řídicích hlášení ICMP. Z pohledu síťových zařízení je však nejdůležitější formát datagramu. Ostatní uvedené body nejsou dále z důvodu rozsahu publikace rozebrány a lze je najít v příslušných RFC nebo v [10].

2.3.2 IPv6

Nová verze protokolu IP (IP next generation) s číslem 6 (IPv6, [9]) byla navržena pro odstranění hlavních nedostatků předchozí verze 4. Jedná se zejména o nedostačující adresový prostor a jeho nesystematické využívání. Mimo možnosti využití až 10^{38} jedinečných adres přináší nová verze protokolu i další novinky jako autokonfiguraci, podporu mobility a integrované zabezpečení. Díky těmto inovacím je IPv6 lépe využitelný pro nové aplikace typu VoIP, videokonference, on-line hry a další.

Protokol IPv6 inovuje způsob adresace, zavádí nový protokol pro řídicí hlášení (ICMPv6) a další. Stejně jako v předcházející kapitole se ale zaměříme na z hlediska síťových zařízení nejdůležitější část, a to formát datagramu. IPv6 efektivně přesouvá informace, jejichž výskyt není vždy nutný, do volitelných rozšíření záhlaví. Povinné záhlaví (viz obrázek 2.5) je tím redukováno na pouhých 8 polí o fixní délce 40 oktetů. Za povinným záhlavím mohou následovat další volitelná záhlaví proměnné délky, určená buď pro zpracování v koncových uzlech nebo směrovačích. Směrovače by se ale většinou měly zabývat pouze povinným záhlavím pevné délky a méně komplikované struktury, čímž by se měla urychlit jejich práce.



Obrázek 2.5: Formát IPv6 hlavičky.

Version (4 bity) – definice verze protokolu. Pro IPv6 je tato hodnota rovna číslu 6.

Traffic Class (8 bitů) – položka umožňuje identifikovat prioritu datagramu. Ovlivňuje zpracování paketu ve směrovačích i koncovém uzlu.

Flow Label (20 bitů) – označuje datagramy vyžadující speciální péči při zpracování ve směrovačích. Označení toku a předchozí pole Traffic Class umožňují přímo řešit priority provozu, QoS a řízení využití šířky pásma.

Payload Length (16 bitů) – délka IPv6 datagramu mimo délky povinné hlavičky (tzn. délka všech doplňkových záhlaví plus délka dat).

Next Header (8 bitů) – identifikace typu záhlaví, které následuje za povinným záhlavím IPv6 datagramu. Pokud není další záhlaví použito, identifikuje typ dat vyšší vrstvy.

Hop Limit (8 bitů) – povolený počet zbývajících směrovačů v cestě. Číslo je při průchodu směrovačem dekrementováno o jedničku a pokud hodnota dosáhne nuly, datagram je odstraněn.

Source Address (128 bitů) – adresa zdrojového uzlu.

Destination Address (128 bitů) – adresa cílového uzlu, pokud není použita rozšiřující směrovací hlavička.

Za povinným záhlavím IPv6 mohou následovat některá z rozšiřujících záhlaví (extension headers). Každé z těchto záhlaví v poli Next Header identifikuje další následující záhlaví. Pokud žádné takové nenásleduje, specifikuje se transportní protokol prostřednictvím čísla protokolu, které je většinou stejné jako pro IPv4 protokol. IPv6 definuje následující rozšiřující hlavičky:

Hop-by-Hop options header – záhlaví definující speciální možnosti, které vyžadují zpracování každým směrovačem na cestě.

Destination options header – volitelné informace pro cílovou stanici nebo všechny cíle podle směrovacího záhlaví.

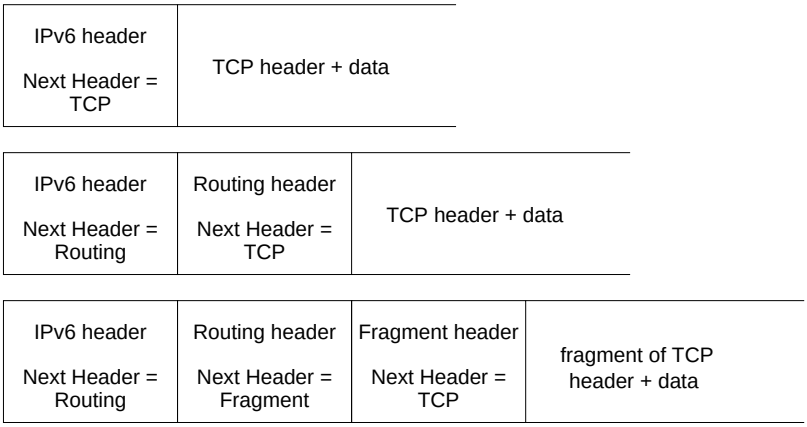
Routing header – poskytuje směrovací informace typu adres směrovačů, které musí být na cestě navštíveny.

Fragment header – obsahuje informace určené pro fragmentaci a znovusestavení datagramů. Fragmentace může být provedena pouze zdrojovou stanicí, nikoliv směrovači po cestě.

Authentification header – zabezpečuje integritu dat a věrohodnost datagramu

Encapsulating security payload header – ochrana přenášených dat v datagramu.

Možný způsob řazení hlaviček je ilustrován na obrázku 2.6. Pořadí rozšiřujících záhlaví je normou pouze doporučeno, nikoliv určeno. Dokonce se v datagramu podle normy může vyskytovat více hlaviček stejného typu.



Obrázek 2.6: Struktura IPv6 datagramu s rozšiřujícími hlavičkami.

2.4 Transportní vrstva

Hlavním úkolem transportní vrstvy je zajistit přenos dat mezi dvěma koncovými body sítě. V modelu TCP/IP jsou pro tuto úlohu definovány dva protokoly – Transmission Control Protocol (TCP, RFC 793, [11]), který poskytuje transportní službu se spojením včetně řízení koncového spojení (tzv. spolehlivý protokol), a User Datagram Protocol (UDP, RFC 768, [12]) poskytující transportní službu bez spojení (tzv. nespolehlivý protokol). Pro identifikaci aplikačního protokolu využívajícího služby transportních protokolů se v obou případech používají čísla portů.

2.4.1 TCP

Transmission Control Protocol poskytuje spolehlivý přenos dat mezi koncovými aplikacemi, který není v modelu TCP/IP zajištěn protokolem síťové vrstvy. TCP přijímá informace od vyšší vrstvy jako souvislý blok dat, z nich vytvoří transportní segmenty a ty předá nižší síťové vrstvě. Dále protokol zabezpečuje řízení koncového spojení a datového toku. Pro řízení dat se využívá princip klouzavého okna, pořadová čísla oktetů dat a pozitivní potvrzování doručených segmentů. Navázání spojení se realizuje pomocí mechanismu *three way handshaking* (trojité potřásání rukou) [11]. Z hlediska síťových zařízení je opět nejdůležitější formát TCP paketu, který je znázorněn na obrázku 2.7.

Source Port (16 bitů) – číslo zdrojového portu, identifikuje zdrojový aplikační proces.

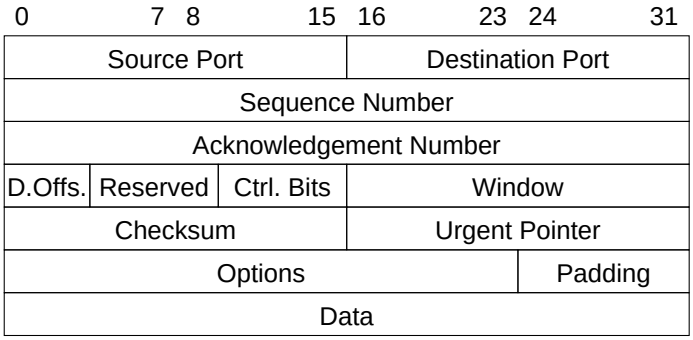
Destination Port (16 bitů) – číslo cílového portu, identifikuje cílový aplikační proces.

Sequence number (32 bitů) – pořadové číslo prvního oktetu dat v paketu.

Acknowledgement Number (32 bitů) – pořadové číslo dat, které očekává příjemce jako následující.

Data Offset (4 bity) – délka záhlaví v násobku počtu 32 bitů.

Reserved (6 bitů) – rezervováno pro budoucí použití, nastaveno na nulovou hodnotu.



Obrázek 2.7: Formát TCP paketu.

Control Bits (6 bitů) – pole příznaků pro řízení datového spojení. Význam jednotlivých bitů je následující: URG – urgentní data pro přednostní doručení, ACK – označuje platnost pole s číslem potvrzení, PSH – segment nese data vyšší vrstvy, RST – reset spojení, SYN – žádost o navázání spojení, FIN – žádost o ukončení spojení.

Window (16 bitů) – udává velikost okna (počet oktetů, které lze vyslat bez průběžného potvrzení).

Checksum (16 bitů) – kontrolní součet zabezpečující celý segment plus některá pole z IP hlavičky (tzv. pseudohlavička).

Urgent Pointer (16 bitů) – specifikace posledního oktetu urgentních dat.

Options – volitelné položky, které mohou následovat za povinnou TCP hlavičkou. Každá taková položka se skládá z typu a délky volitelné položky a hodnoty. Bližší popis volitelných položek lze nalézt v [11].

Padding – zarovnání TCP hlavičky na délku rovnající se násobku 32 bitů. Skládá se z nulových bitů.

2.4.2 UDP

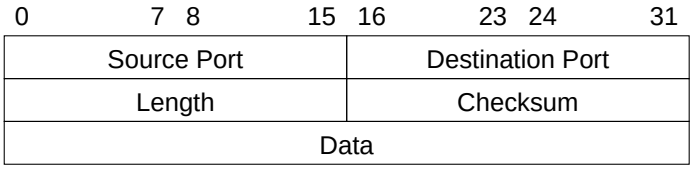
User Datagram Protocol poskytuje nespolehlivou transportní službu pro aplikace, které nevyžadují zabezpečení přenosu v takovém rozsahu jako u TCP protokolu nebo vyžadují rychlý přenos dat bez ohledu na chyby (VoIP, audio-video-streaming). UDP nezaručuje, zda se přenášený paket neztratí, zda se nezmění pořadí doručení paketů nebo zda nebude některý paket doručen vícekrát. Protokol je bezstavový a nenabízí funkci řízení toku dat nebo funkci opravy chyb. Formát paketu tohoto protokolu je uveden na obrázku 2.8.

Source Port (16 bitů) – číslo zdrojového portu, identifikuje zdrojový aplikační proces.

Destination Port (16 bitů) – číslo cílového portu, identifikuje cílový aplikační proces.

Length (16 bitů) – délka celého paketu v násobcích počtu 32 bitů.

Checksum (16 bitů) – kontrolní součet zabezpečující celý paket plus některá pole z IP hlavičky (tzv. pseudohlavička).



Obrázek 2.8: Formát UDP paketu.

2.5 Návrh síťových zařízení

Mezi prvky síťové infrastruktury pro něž se využívá hardwarové akcelerace patří směrovače, firewally, systémy pro detekci či prevenci nežádoucího provozu, monitorovací adaptéry, testovací sondy, generátory paketů a jiné. Úkolem hardwarové akcelerace je zajistit dostatečnou výkonnost těchto zařízení pro zpracování dat na rychlostech linky, protože softwarové zpracování může být v závislosti na typu aplikace limitováno už rychlostmi v řádu stovek megabitů za vteřinu.

Jedním ze základních prvků síťové infrastruktury je směrovač. Jeho úkolem je zpracovat pakety ze vstupního rozhraní, extrahovat směrovací informace ze síťové hlavičky (L3), provést vyhledání ve směrovacích tabulkách a na základě výsledku tohoto vyhledání upravit paket, zabalit jej do hlavičky linkové vrstvy a odeslat na výstupní rozhraní. Pro desetigigabitový ethernet přitom může přijít téměř 20 miliónů paketů za vteřinu a počet portů směrovače bývá zpravidla minimálně čtyři. Pro zpracování takového množství dat je hardwarová akcelerace prakticky nutností.

Řádově jednodušší úlohou je paketový filtr a tedy základní zabezpečovací mechanismus sítí – firewall. Jeho úkolem je extrahovat pole z hlaviček linkové, síťové a transportní vrstvy a pomocí vyhledání ve filtračních pravidlech rozhodnout zda paket zahodit nebo poslat dále. Mimo přímé filtrování paketů umožňují některé firewally i stavové filtrování, kdy je pomocí flagů v TCP hlavičce paketu monitorován stav TCP toku (posloupnost příkazů SYN, ACK atd). Při takovém filtrování je provoz považován za dvousměrnou výměnu paketů v rámci relace, přičemž pro každou relaci je udržována dynamická tabulka jejího stavu. Pakety v daném toku jsou porovnávány s očekávanými pakety relace a pokud jejich typ neodpovídá jsou filtrovány. Přestože dokážou firewally efektivně filtrovat síťový provoz a omezit jej pouze na požadované služby, zdaleka nedokáží zajistit úplnou bezpečnost sítě. Existují útoky, které firewally neodhalí, a ani ze své podstaty odhalit nemohou. K těmto útokům jsou používány standardní služby a porty, které není možné obecně zakázat a pro detekci takových útoků je nutné použít další mechanismy.

Mezi takovéto mechanismy patří systémy pro detekci či prevenci narušení (Intrusion detection/prevention system, IDS/IPS). Tyto systémy provádějí komplexní analýzu síťových toků včetně monitorování dat paketů. Systémy pro detekci narušení fungují jako pasivní síťové prvky a reportují zjištěné narušení administrátorovi, systémy pro prevenci narušení fungují podobně jako firewally a zjištěný nežádoucí provoz mohou přímo filtrovat. Vyhledávání vzorů v datech paketu je však velmi výpočetně náročná úloha, a pokud mají být v datech paketu hledány stovky nebo tisíce vzorů (databáze pravidel referenčního programu *Snort* [13]), je čistě softwarové řešení limitováno propustnostmi v řádu stovek megabitů za vteřinu [14]. Pro vyšší rychlosti je opět nutné použít jiná řešení.

Pro zvýšení bezpečnosti na sítích se používají různé monitorovací adaptéry. Může se jednat o dynamické nebo statické monitorovací sondy sloužící ke sledování dat procházejících síti. Příkladem jsou dynamické sondy exportující data ve formátu *NetFlow* [15] či *IPFIX* [16], které zpracovávají všechny pakety na lince, agregují statistiky na základě IP toků a exportují je na kolektor. Na kolektoru jsou data uložena do databáze a je nad nimi možné provádět detekci útoků, analýzy vytíženosti sítě, sekundární agregace pro dlouhodobé uložení dat a jiné. Základním požadavkem na exportující sondy je deterministické zpracování všech paketů na vstupu i při útoku, což v softwarovém řešení není možné zajistit. Zatímco hardwarová sonda může pakety deterministicky či adaptivně vzorkovat, v softwaru při útocích z důvodu nedostatečné propustnosti sběrnicového systému či přílišné náročnosti na výpočetní zdroje procesoru dochází k nedeterministickému zahazování paketů a exportovaná data nemusí odpovídat reálnému provozu na síti [17].

Mezi další síťová zařízení vyžadující akceleraci patří například sondy pracující s přesným časem nebo výkonné paketové generátory. Pro přesná časová rozlišení v řádu nanosekund je nutné pracovat s časovými značkami už v okamžiku příjmu paketu ze síťového rozhraní, protože přenos paketu do softwaru po systémové sběrnici může trvat nedeterministickou dobu. V případě paketových generátorů pro testování sítí je často nutné generovat objemy provozu přesahující možnosti systémové sběrnice. V obou případech nabízí vhodné řešení hardwarová realizace dané úlohy.

Při návrhu a implementaci všech výše zmíněných síťových zařízení je možné se setkat s mnoha přístupy. Patří mezi ně čistě softwarové řešení na obecném procesoru, distribuce zátěže na víceprocesorové řešení, využití aplikačně specifických síťových procesorů, specializovaná úzce zaměřená hardwarová řešení založená na technologii ASIC, či flexibilnější architektury využívající technologie FPGA. Častým přístupem při návrhu akcelerátoru je kombinace hardwarového a softwarového přístupu, kdy je většina výpočetně náročného zpracování realizována v hardwaru a zbývající menší část je zpracována v softwaru.

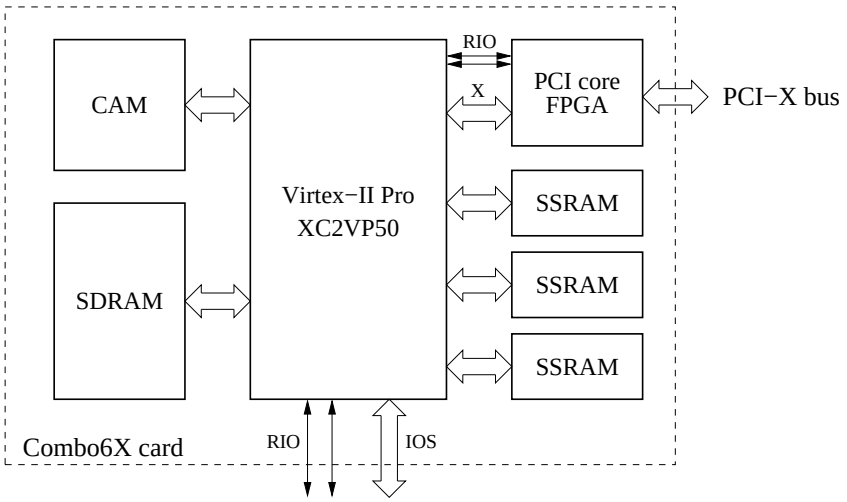
Přestože každý přístup pracuje na zcela jiném principu, lze mezi nimi vysledovat určitou podobnost. V každém ze zmíněných řešení je nutné řešit úkoly na různých úrovních. Od těch nejnižších tvořících jistou míru abstrakce nad systémem jako je například příjem paketů do systému nebo definice komunikačních rozhraní, přes obecné úlohy typu distribuce zátěže na více pracovních jednotek, po nejvyšší úroveň vlastní aplikace – algoritmus zpracování. Při návrhu platformy pro rychlý vývoj síťových zařízení je proto vhodné respektovat zmíněné úrovně a vytvořit takové řešení, které poskytne abstrakci od konkrétní hardwarové platformy, bude integrovat nejnižší vrstvy síťových operací jako jsou příjem a vysílání paketů nebo přenos dat do softwaru, poskytne jednotné komunikační rozhraní a množinu základních knihovních funkcí pro typické opakující se úlohy v síťových aplikacích. Návrháři tím bude umožněno pracovat pouze na vlastním algoritmu jádra aplikace a navržená platforma mu bude sloužit jako základní vývojové prostředí.

2.6 Rodina karet COMBO

Cílové hardwarové prostředky pro realizaci síťové platformy tvoří rodina karet COMBO. Tato rodina karet je koncipována jako hardwarový akcelerátor do slotu systémové sběrnice klasického PC. Akcelerátor je rozdělen do dvojice karet – základní karty a karty rozhraní. Rozdělením na základní kartu a kartu rozhraní je získána flexibilita pro různé síťové tech-

nologie, kdy při stejné základní kartě je možné použít různé karty rozhraní – např. kartu se čtyřmi metalickými gigabitovými rozhraními, čtyřmi optickými gigabitovými rozhraními nebo kartu s rozhraním k desetigigabitovému ethernetu.

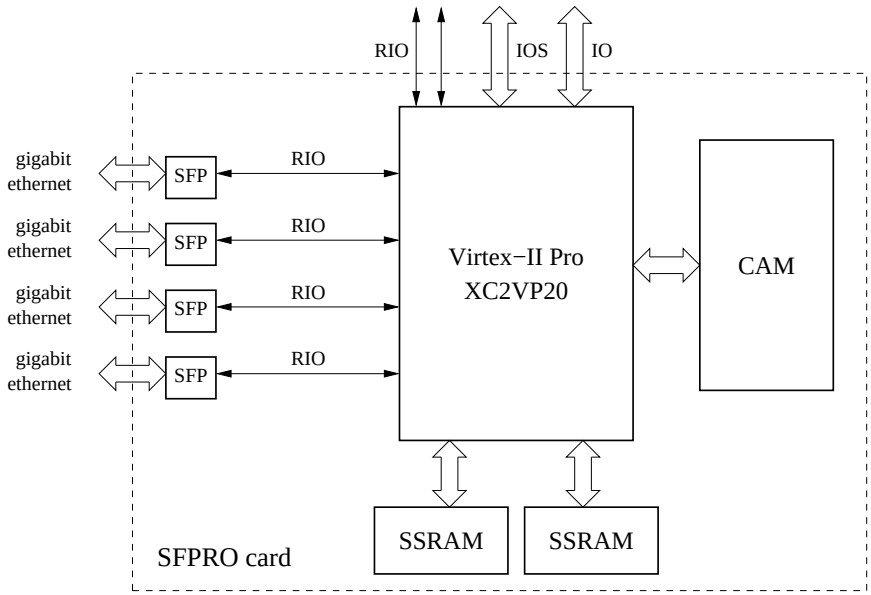
Základní karta obsahuje hlavní výpočetní prostředky – výkonné FPGA Virtex-II [18] nebo Virtex-II Pro [19] a podpůrné paměťové moduly (statické, dynamické či asociativní). Dále je karta vybavena konektorem systémové sběrnice, který je v závislosti na konkrétním typu karty určen pro PCI, PCI-X nebo PCI-Express rozhraní, pomocným hradlovým polem CPLD a menším PCI core programovatelným hradlovým polem pro komunikaci se systémovou sběrnicí. Jako propojení malého a velkého FPGA slouží 40 paralelních vodičů a dva multigigabitové transceivery RocketIO [19]. Podobně pro propojení základní karty a karty rozhraní je použita skupina 104 paralelních vodičů a dva kanály RocketIO. Schéma karty COMBO6X je zachyceno na obrázku 2.9.



Obrázek 2.9: Schéma karty COMBO6X.

Rozšiřující karta obsahuje FPGA Virtex-II či Virtex-II Pro, podpůrné paměťové moduly a síťová rozhraní pro příjem a odesílaní paketů. Přídavných karet bylo vyvinuto více typů lišícími se zejména druhem síťových konektorů. Karta COMBO4-MTX je osazena čtyřmi metalickými porty gigabitového ethernetu (GE), karta COMBO4-SFP a COMBO4-SFPRO čtyřmi optickými rozhraními GE a karta XFP dvěma optickými rozhraními pro desetigigabitový ethernet. Pro propojení se základní kartou jsou použity paralelní vodiče IOS a u modernějších typů karet i multigigabitové transceivery RocketIO. Schéma karty rozhraní SFPRO je znázorněno na obrázku 2.10.

Pro použití v navrhované síťové platformě je velmi zajímavá i další z rodiny karet COMBO – karta PTM. Ta umožňuje generovat přesná 64-bitová časová razítka a přenášet je na kartu rozhraní, kde můžou být využity k zaznamenání přesného času příchodu paketů. Karta PTM je navržena jako samostatná rozšiřující karta do systémové sběrnice PCI, je osazena mikroprocesorem Texas Instruments, hradlovým polem pro komunikaci se softwarem a pro řízení TimeStamp jednotky, přesným hodinovým krystalem a rozhraním k GPS pro přesnou časovou synchronizaci. S kartou rozhraní, případně základní kartou, je možné ji propojit pomocí paralelního IO konektoru.



Obrázek 2.10: Schéma karty SFPRO.

Architektura síťové platformy je primárně určena pro použití v modernějších základních kartách COMBO6X a COMBO6E, které umožní rychlé propojení s pamětí hostujícího počítače přes sběrnice PCI-X a PCI-Express. Cílové síťové rozhraní je v první fázi gigabitový a v druhé fázi desetigigabitový Ethernet. Pro první fázi bude využita karta SFPRO, karta pro druhou fázi je v současné době ve stádiu výroby prototypu.

Kapitola 3

Architektura síťové platformy

Platforma pro rychlý vývoj síťových zařízení byla pojmenována *NetCOPE*. Při jejím návrhu hrály nejdůležitější roli snaha o co nejobecnější řešení využitelné pro širokou škálu aplikací, snaha o maximální využití potenciálu karet COMBO a snaha o návrh komplexního prostředí pro vývojáře síťových aplikací. Platforma musí odstínit vývojáře od nízkoúrovňových problémů, poskytnout řešení pro základní operace s pakety a umožnit soustředit se na návrh a implementaci vlastního aplikačního jádra.

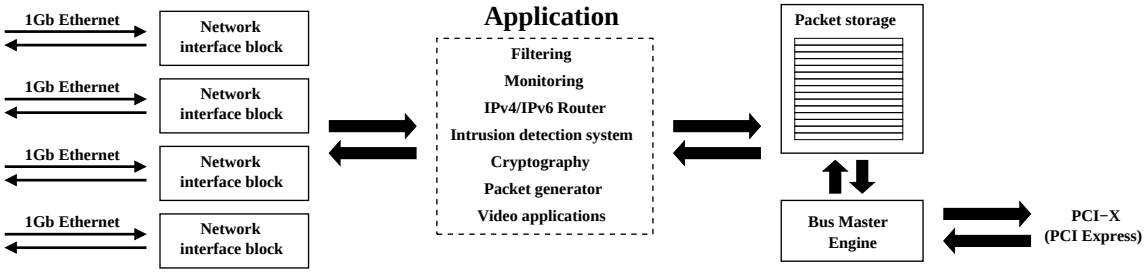
Pro splnění těchto požadavků a vytvoření vhodné abstraktní vrstvy nad hardwarovými prostředky bylo identifikováno několik dílčích úloh. První z nich je definice komunikačního rozhraní pro přenos paketů v rámci jednoho programovatelného pole i mezi několika programovatelnými hradlovými poli navzájem. Za tímto účelem je v této práci specifikován generický protokol pro přenos paketů – *FrameLink*. Druhým úkolem je vytvoření sady nástrojů pro manipulaci s daty přenášenými tímto protokolem. Tato sada nástrojů bude plnit základní úkony typu rozdělování a spojování datových toků a v uživatelských aplikacích bude využívána podobně jako knihovní funkce ve vyšších programovacích jazycích.

Třetí úlohou je abstrakce od nízkoúrovňových operací jako jsou příjem paketů ze síťového rozhraní, odesílání paketů na síťové rozhraní, přenos paketů mezi adaptérem a pamětí RAM hostujícího počítače a řízení podpůrných externích prvků na kartě. Pro realizaci těchto úloh jsou využity IP cores, které byly vyvinuty v rámci projektu *Liberouter* [1]. Zatímco některé byly vyvinuty přímo pro platformu NetCOPE, některé musely být přepracovány, aby splňovaly nároky na požadované rozhraní a obecnou funkcionalitu.

Pro příjem a odesílání paketů jsou využity vstupní a výstupní síťové bloky pro GMII rozhraní. Vstupní bloky zpracovávají data z tohoto rozhraní a transformují je na FrameLink protokol, kterým jsou data předávány do aplikace. Výstupní bloky naopak zpracovávají data z aplikace ve formátu FrameLink protokolu a vysílají je na výstupní rozhraní. Pro přenos dat mezi softwarem hostujícího počítače a adaptérem byl v rámci projektu Liberouter navržen a implementován propojovací systém sběrnic [20]. Jde o škálovatelný systém, který nabízí rychlou sběrnici pro přenos paketů, pomalejší a úspornější sběrnici pro přenos konfiguračních a jiných dat a programovatelný DMA kontrolér s kontrolní sběrnicí pro řízení bus master přenosů. Pro podporu externích komponent na kartách rodiny COMBO zahrnuje platforma NetCOPE řadiče všech typů dostupných pamětí (statické, asociativní a dynamické) a řadič phyterů. Specifickým požadavkem na navrženou platformu vyplývajícím z architektury COMBO karet je zajištění komunikace mezi základní kartou a kartou rozhraní přes RocketIO kanály a skupinu paralelních vodičů.

Na vyšší úrovni je návrh a implementace vlastní aplikace. Pro tuto úlohu je možné využít řadu bloků pro zpracování a analýzu síťového provozu, které byly vyvinuty v rámci projektu *Liberouter* a upraveny pro použití na platformě NetCOPE. Jedná se například o procesní jednotky pro analýzu či editaci paketů [21, 22], klasifikační procesor [23], jednotku pro vyhledávání vzorů [24] nebo systém pro stavové zpracování TCP/IP toků [25].

Architektura navržené platformy je znázorněna na obrázku 3.1. Platforma odstíní vývojáře aplikace od nízkoúrovňových problémů, poskytne mu jednotné rozhraní pro komunikaci se síťovým rozhraním a rozhraním do softwaru a nabídne sadu předpřipravených řešení ve formě IP cores bloků. Úkolem vývojáře bude vytvořit vlastní jádro aplikace, přičemž i při jeho vývoji může využít množinu dostupných komponent.



Obrázek 3.1: Architektura platformy NetCOPE.

Následující seznam shrnuje klíčové prvky navržené infrastruktury:

- Jednotný a generický paketově orientovaný komunikační protokol *FrameLink* pro propojení interních komponent na čipu.
- Sada nástrojů pro manipulaci s daty ve formátu FrameLink protokolu umožňující základní operace nad daty jako distribuování toku dat do více paralelních cest, transformace šířky dat či spojování toků dat.
- Propojovací systém sběrnic s vysokou propustností pro přenos dat mezi komponentami v hradlovém poli a systémovou pamětí RAM hostujícího počítače.
- Programovatelný DMA řadič realizovaný ve vestavěného procesoru PowerPC pro řízení DMA operací mezi pamětí RAM a adaptérem.
- Vstupní a výstupní bloky pro příjem a odesílání paketů přes síťové rozhraní s podporou gigabitového Ethernetu.
- Sada základních stavebních bloků ve formě IP cores pro analýzu, zpracování a monitorování síťového provozu.
- Komponenty pro zajištění spolehlivého přenosu dat mezi kartami po multigigabitových kanálech RocketIO.
- Sada komponent pro řízení externích prvků na kartě.

Architektury všech zmíněných prvků jsou popsány v následujících kapitolách. Nejprve je uvedena specifikace protokolu *FrameLink*. Ve druhé kapitole je popsána sada nástrojů pro manipulaci s daty ve formátu tohoto protokolu a ve třetí kapitole je představen propojovací systém sběrnic. Čtvrtou kapitolu tvoří popis předpřipravených bloků ve formátu IP cores, do kterých jsou zahrnuty i některé klíčové bloky platformy *NetCOPE* jako jsou vstupní a výstupní síťové bloky, softwarové buffery pro uložení paketů nebo komponenty pro řízení přenosu dat po RocketIO kanálech. V poslední kapitole je představena kompletní blokové schéma navržené síťové platformy *NetCOPE* pro základní kartu COMBO6X a kartu rozhraní SFPRO.

3.1 FrameLink protokol

Jedním z klíčových prvků navrhované platformy je použití jednotného generického paketově orientovaného protokolu pro přenos dat mezi jednotlivými funkčními bloky. Za tímto účelem byl navržen protokol *FrameLink*. Použití tohoto striktně definovaného protokolu umožní rychlou vzájemnou integraci komponent a jednoduché skládání aplikací. Hlavní vlastnosti navrženého protokolu jsou shrnuty v následujících bodech:

- uživatelem definovatelná generická šířka dat,
- přenos dat ve formě rámců libovolné délky,
- synchronní point-to-point komunikace,
- oboustranná kontrola toku dat,
- účinné využití přenosové kapacity linky.

Protokol FrameLink částečně vychází z protokolu *Xilinx LocalLink* [26], ale omezuje množství definovaných variant tohoto protokolu a upravuje některé jeho vlastnosti. Hlavním důvodem proč není použit přímo protokol firmy Xilinx je nezarovnaná struktura paketu v tomto protokolu. Paket se dle definice LocalLinku může skládat až ze tří částí – kontrolních dat na začátku (header), vlastních dat paketu (payload) a kontrolních dat na konci (footer). Při přenosu vždy následují data těchto částí hned za sebou bez mezer a v jednom taktu mohou být přenesena v závislosti na datové šířce protokolu i všechny části paketu najednou. Výhodou tohoto řešení je využití maximální možné propustnosti datových vodičů.

Nevýhodou je nutnost složitých přerovnávacích obvodů pro každou komponentu pracující s obsahem dat přenášených protokolem. Typicky bývá takových komponent v aplikaci více a použití nezarovnaného přenosu dat by znamenalo výrazné zvýšení nároků na zdroje FPGA. Proto byl raději navržen nový protokol, který snižuje náročnost na zdroje programovatelného hradlového pole za cenu nižší efektivity využití přenosového pásma.

Přestože schéma komunikace je u protokolu FrameLink jiné, rozhraní obou protokolů jsou do značné míry podobná a umožňují snadnou vzájemnou integraci funkčních bloků. Hlavní rozdíly protokolů FrameLink a LocalLink jsou shrnuty v následujících bodech. Začátky dat všech částí rámce jsou ve FrameLinku zarovnány na šířku slova, v jednom cyklu není možné přenášet více částí rámce a není omezen počet částí rámce. Protokol FrameLink explicitně neoznačuje o kterou část rámce se jedná, význam je dán kontextem. Signály

Signál	Směr	Celé jméno
CLK	vstup	Clock
DATA	zdroj → cíl	Data Bus
REM	zdroj → cíl	Remainder Bus
SRC_RDY_N	zdroj → cíl	Source Ready
DST_RDY_N	cíl → zdroj	Destination Ready
SOF_N	zdroj → cíl	Start-of-Frame
EOF_N	zdroj → cíl	End-of-Frame
SOP_N	zdroj → cíl	Start-of-Part
EOP_N	zdroj → cíl	End-of-Part

Tabulka 3.1: Rozhraní protokolu FrameLink.

SOP_N a *EOP_N* označují každou část rámce, jejich význam je změněn z označení začátku resp. konce vlastních dat (Start-of-Payload, End-of-Payload) na označení začátku resp. konce každé části (Start-of-Part, End-of-Part). Uspořádání bajtů ve slově je u FrameLinku definováno jako *Little Endian*, LocalLink v základní verzi definuje použití opačného uspořádání *Big Endian*.

Rozhraní protokolu FrameLink je tvořeno signály uvedenými v tabulce 3.1. Zdrojová a cílová aplikace komunikují v simplexním (jednosměrném) režimu. Aplikací může být libovolná uživatelská jednotka či IP core blok. Při nutnosti oboustranné komunikace je možné použít dvě FrameLink rozhraní. Datový tok může být regulovaný zdrojovou i cílovou aplikací pomocí jednoduchého handshake protokolu. Cíl i zdroj mohou monitorovat rozhraní protokolu neustále, ale jen pokud oba signalizují připravenost vysílat resp. přijímat data, dojde v příslušném cyklu k datové transakci. Zdroj signalizuje svou připravenost vysílat data aktivním signálem *SRC_RDY_N*, cíl aktivním signálem *DST_RDY_N*. Při vystavování těchto signálů na sebe zdroj a cíl vzájemně nečekají a indikují svou schopnost vysílat či přijímat data hned, když je to možné.

Struktura přenášeného paketového rámce je signalizována ze zdroje k cíli čtyřmi signály – *SOF_N* (Start-Of-Frame), *EOF_N* (End-Of-Frame), *SOP_N* (Start-Of-Part), *EOP_N* (End-Of-Part). První dva z těchto signálů označují začátek a konec přenášeného rámce, druhé dva určují rozdělení rámce na části. Těchto částí může být libovolné množství (minimálně jedna), přičemž každá z těchto částí musí být ohraničena danými signály a musí být zarovnána na šířku přenášeného slova. Pokud je přenášená část kratší než šířka slova na datové sběrnici, jsou zároveň aktivní signály *SOP_N* i *EOP_N*. Počet částí je závislý na konkrétní aplikaci, typicky jsou to ale dvě nebo tři – kontrolní data před vlastním paketem (header), tělo paketu (payload) a kontrolní data za paketem (footer), přičemž kontrolní data na začátku nebo na konci mohou chybět. Všechny čtyři kontrolní signály jsou platné pouze když vysílací i přijímací strana signalizují připravenost vysílat/přijímat prostřednictvím příslušných signálů *SRC_RDY_N* a *DST_RDY_N*.

Pořadí bajtů v přenášených slovech je definováno uspořádáním Little Endian, tedy nejméně významný bajt je na nejnižší adrese datové sběrnice. Jelikož šířka datové sběrnice může být libovolná, je nutné řešit indikaci posledního platného bajtu. To zajišťuje signál REM, který je přenášen paralelně s daty a nese binárně zakódovanou informaci o

posledním platném bajtu v přenášeném slově. Bajty jsou přitom číslovány od nuly. Šířka signálu REM je dvojkový logaritmus šířky datové sběrnice v bajtech. Například pro sběrnici širokou 4 bajty je šířka signálu REM 2 bajty a hodnoty tohoto signálu 0, 1, 2, 3 postupně značí platnost jednoho, dvou, tří a čtyř (tj. všech) bajtů. Signál REM je platný pouze pokud je přenášen konec libovolné části a jeho platnost je potvrzována signálem EOP_N.

Stručný popis jednotlivých signálu rozhraní protokolu je uveden v následujícím seznamu. Směry a zkratky názvů signálů jsou uvedeny ve výše zmíněné tabulce 3.1.

Clock: vstupní hodiny, vůči kterým jsou synchronní všechny signály rozhraní protokolu.

Data Bus: datová sběrnice po níž jsou přenášeny rámce. Šířka tohoto signálů je generická a může se jednat o libovolný násobek 8 bitů.

End-of-Frame Remainder Bus: signál určující pozici posledního platného bajtu v rámci nebo v části rámce. Šířka tohoto signálu je určena vztahem $\lceil \log_2 N \rceil$, kde N je šířka datové sběrnice v bajtech.

Source Ready: signál značící připravenost zdroje vysílat data. Pokud tento signál není aktivní, neproběhne na sběrnici žádný přenos dat. Zdroj může při jakékoliv probíhající části přenosu rámce deaktivovat tento signál z libovolného důvodu (např. zdroj nemá dočasně žádná data k odesílání), čímž pozastaví přenos dat. Signál je aktivní v nule.

Destination Ready: signál indikující připravenost cíle přijímat data. Pokud tento signál není aktivní, neproběhne na sběrnici žádná transakce. Cíl může tento signál kdykoliv deaktivovat pokud nemůže dočasně přijímat data, čímž pozastaví přenos dat. Signál je aktivní v nule.

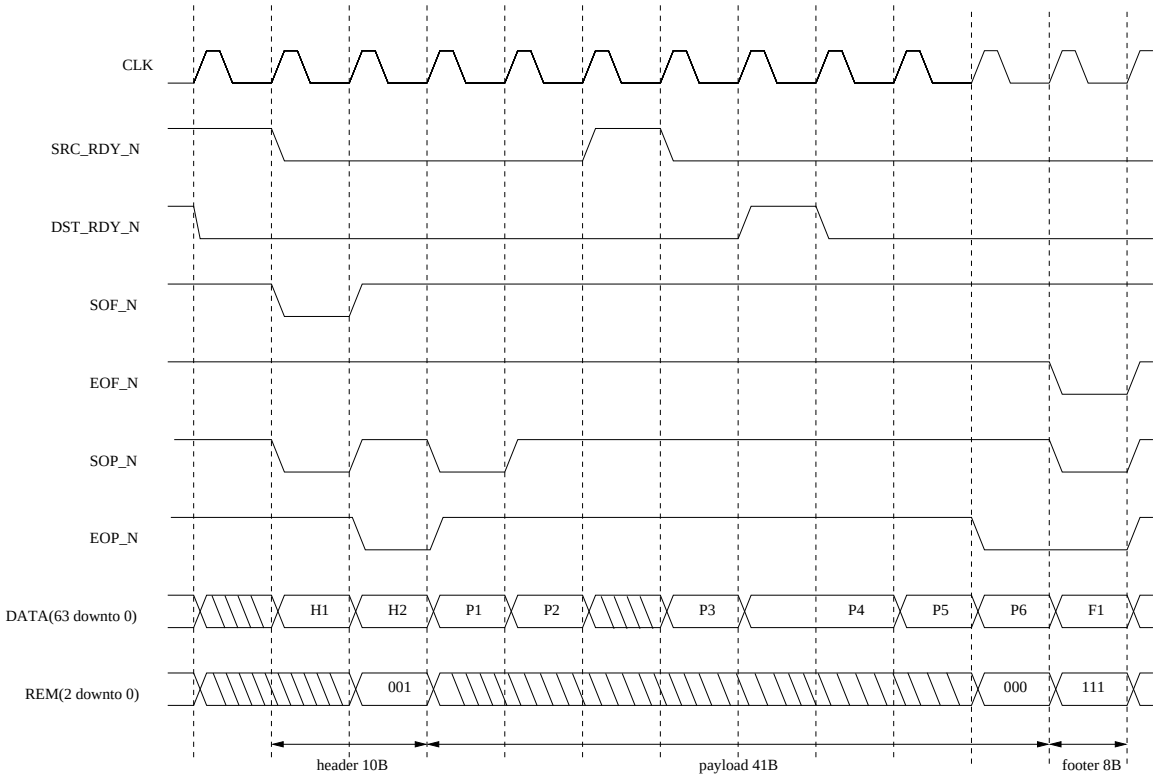
Start-of-Frame: signál indikující, že přenášené slovo je prvním slovem rámce. Signál je aktivní v nule.

End-of-Frame: signál označující poslední slovo rámce. Poslední platný bajt rámce ve slově je označen signálem REM. Signál je aktivní v nule.

Start-of-Part: signál označující první slovo v části rámce. Začátek části rámce je vždy zarovnaný na šířku datového slova. Signál je aktivní v nule.

End-of-Part: signál označující poslední slovo v části rámce. Poslední bajt příslušné části je označen signálem REM. Signál je aktivní v nule.

Možný průběh komunikace při přenosu rámce skládajícího se ze tří částí po sběrnici o šířce 8 bajtů je uveden na obrázku 3.2. Cíl signalizuje připravenost přijímat data a přenos je odstartován v následujícím taktu, kdy zdroj signalizuje připravenost vysílat data. Zdroj zároveň s nastavením signálu *SRC_RDY_N* nastaví kontrolní signály *SOF_N* a *SOP_N* a na datovou sběrnici vystaví první slovo přenášených dat. První a druhý takt se přenášejí kontrolní data na začátku (header), následujících osm taktů vlastní tělo paketu (payload) a poslední takt kontrolní data na konci (footer). Pokud zdroj nebo cíl nesignalizují připravenost k přenosu, transfer se neprovede. Přenos rámce končí vystavením signálu *EOF_N*. Hodnoty signálu *REM* 1, 0 a 7 (pro příslušné části) značí, že dva nejméně významné bajty (konkrétně DATA[7:0] a DATA[15:8]), jeden nejméně významný bajt a všech osm bajtů nesou poslední platné oktety příslušné části.



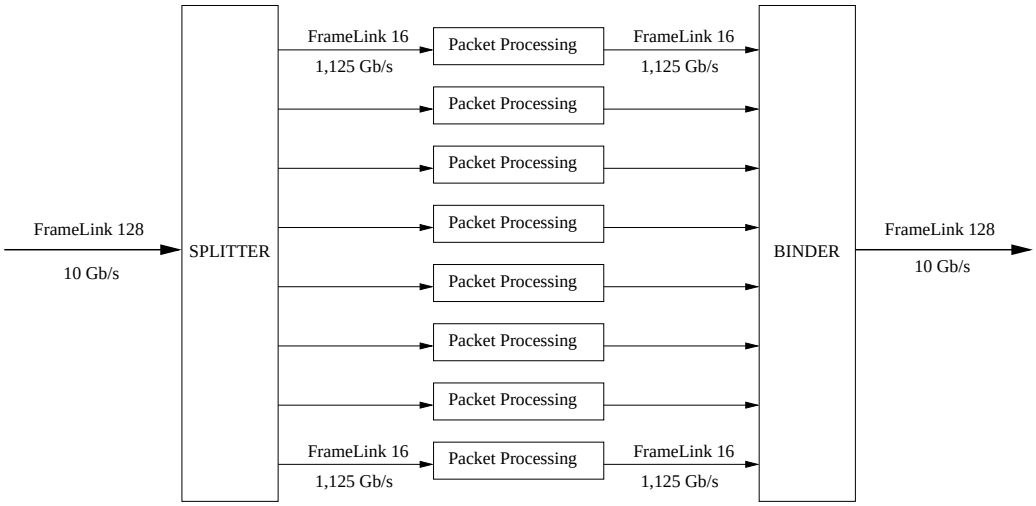
Obrázek 3.2: Časový diagram FrameLink protokolu.

3.2 Nástroje pro práci s FrameLink protokolem

Pro manipulaci s daty ve formátu FrameLink protokolu byla navržena a implementována sada nástrojů nazvaná *FrameLink tools*. Tato sada nástrojů obsahuje množinu generických komponent umožňujících různé operace nad FrameLink protokolem jako je rozdělování a spojování toků dat v tomto protokolu či transformace datové šířky protokolu. Generičností je u těchto komponent myšlena nejen generická šířka datových sběrnic, ale i generický počet vstupních a výstupních rozhraní.

Mezi nejdůležitější komponenty pro práci s FrameLink protokolem patří generické jednotky pro spojování a rozdělování toků pojmenované *FrameLink Binder* a *FrameLink Splitter*. Pomocí těchto jednotek lze libovolně distribuovat tok dat ve formě paketů do několika datových toků nebo naopak více datových toků spojovat do jednoho. Toho lze s výhodou využít při návrhu aplikací, kdy je často nutné řešit distribuci zátěže mezi více procesních jednotek nebo spojování a rozdělování toků z důvodu transportu po rychlé sériové sběrnici.

Příkladem může být rozdělení vstupního desetigigabitového toku paketů do osmi toků, které jsou následně zpracovány procesory pro analýzu a editaci dat (viz obrázek 3.3). Díky rovnoměrné distribuci stačí pro zajištění plné propustnosti použít osm procesorů s propustností 1,125 Gb/s. Po zpracování a případné editaci může být tok pomocí Binderu opět spojen a odeslán na výstupní 10 Gb rozhraní. Podrobnější popis všech FrameLink nástrojů je obsahem následujících kapitol.



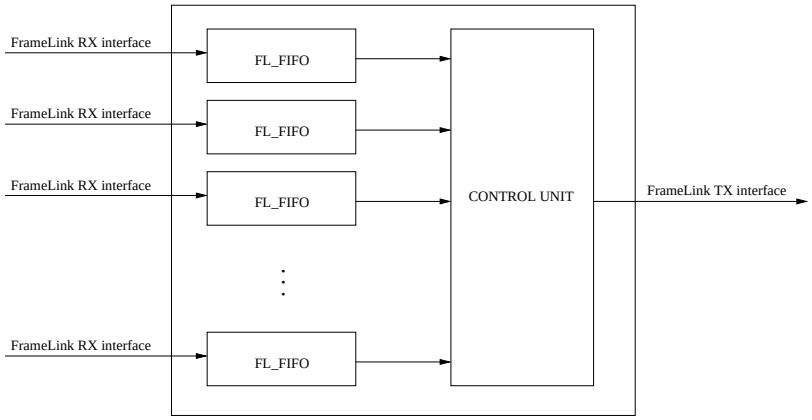
Obrázek 3.3: Paralelní zpracování pomocí komponent FrameLink Splitter a Binder.

3.2.1 FrameLink Binder

Komponenta je určena pro spojování vstupních toků dat do jednoho výstupního toku. Počet vstupních toků a šířka vstupních i výstupních toků jsou generické parametry komponenty. Příkladem využití jednotky může být spojení dat z více vstupních síťových rozhraní do jednoho kvůli přenosu po rychlé sériové sběrnici, spojení toku dat před odesláním do softwaru, nebo spojení paralelně zpracovávaných toků, které vznikly předchozím rozdělením pomocí jednotky Splitter (viz obrázek 3.3).

Architektura jednotky je založena na principu znázorněném na obrázku 3.4. Data z N vstupních rozhraní se ukládají do N příslušných front. Z těchto se následně na základě určité politiky vybere jedna fronta, ze které jsou data vyčítána na výstupní rozhraní. Výběr probíhá okamžitě při uvolnění výstupního rozhraní nezávisle na tom, zda je paket ve vstupní frontě uložený celý či nikoliv. Využitím tohoto principu se eliminuje latence dat při průchodu komponentou. Politiku výběru fronty je možné genericky zvolit ze tří možností: výběr z nejobsazenější fronty, spravedlivý cyklický výběr (*round robin*) a prioritní výběr z určité fronty. Každá z těchto politik se hodí pro jinou situaci.

Výběr z nejobsazenější fronty probíhá na základě zaplnění vstupních front a řídicí jednotka pro tento algoritmus je úsporná na zdroje programovatelného hradlového pole. Politika se uplatní při spojování souměrných toků dat. Příkladem mohou být toky, které vznikly předchozím rozdělením pomocí komponenty Splitter. Při spojování nezávislých datových toků je výhodnější spravedlivá politika s cyklickým výběrem, která zabraňuje zvýhodňování toku s nejvyšším datovým provozem. U této politiky je uplatňována rotující priorita, přičemž se vybírá ze všech front, ve kterých je uložen alespoň jeden celý paket. Pokud není celý paket uložen v žádné frontě, vybere se opět nejvíce obsazená fronta. Pro případy, kdy je nutné zvýhodnit výběr z určité fronty, je určena prioritní politika. Při použití této politiky je jedna z front označena jako prioritní, a pokud je v ní uložen celý paket, tak je vybrán na výstup. Pokud v prioritní frontě celý paket uložen není, vyčítá se podle politiky výběru z nejobsazenější fronty.



Obrázek 3.4: FrameLink Binder.

Při typickém použití jednotky Binder je šířka výstupního rozhraní dána součtem šířek vstupních rozhraní. Pakety jsou do front ukládány na rychlosti vstupního rozhraní a vyčítány vyšší rychlostí výstupního rozhraní. Aby nedocházelo k limitaci výstupní propustnosti rychlostí vstupního rozhraní, je vhodné, aby byl paket před vyčítáním ve frontě uložen celý. Pro zajištění dostatečné kapacity pro uložení paketů jsou proto vstupní fronty realizovány v blokových pamětech FPGA. Pokud nebude paket při vyčítání uložen ve frontě celý, dojde k dočasnému snížení propustnosti výstupního rozhraní. Tato situace ale nastane pouze při malém vytížení jednotky nebo při zpracování tzv. *jumbo* paketů, jejichž velikost je vyšší než kapacita blokových pamětí (2kB pro Virtek II Pro).

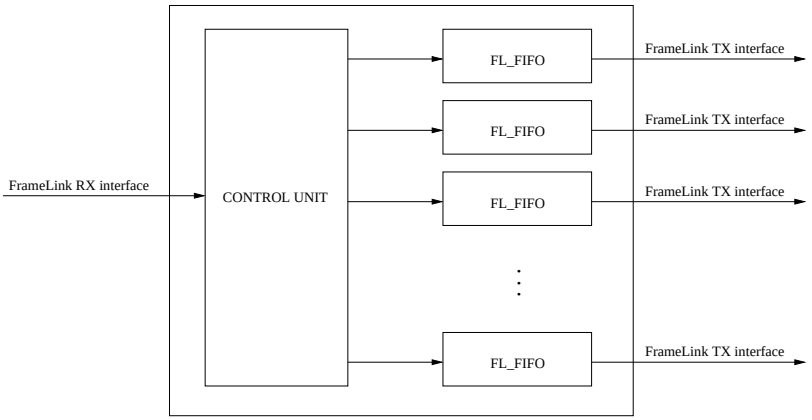
Při vyšších výstupních datových šířkách než 32 bitů (maximální šířka dat blokových pamětí) se uplatní speciální varianta front nazvaná *multififo*. Tento blok realizuje generický počet front v každé blokové paměti, čímž se odstraní nutnost mít pro každou frontu paměť o šířce dat výstupního rozhraní. Pomocí multififa je umožněno vyčítat v každém taktu slovo o šířce výstupního rozhraní z jedné vstupní fronty realizované ve více pamětech a není nutné mít pro každou vstupní frontu použito více blokových pamětí paralelně pro zajištění potřebné datové šířky. Pokud je to možné, použije se tedy jen tolik blokových pamětí, kolik je vstupních rozhraní.

3.2.2 FrameLink Splitter

Komponenta pro rozdělení jednoho vstupního toku dat do více výstupních toků. Počet výstupních toků a šířka vstupních a výstupních toků jsou generické parametry komponenty. Jednotka najde využití na všech místech, kde je nutné distribuovat vstupní datový tok do více toků s nižší propustností, a umožní kombinovat v aplikaci komponenty s různými propustnostmi. Příklad využití paralelního zpracování jednotkami s nižší propustností je uveden na obrázku 3.3.

Architektura jednotky se skládá z kontrolní jednotky a N výstupních front realizovaných v blokových pamětech FPGA. Schéma je zachyceno na obrázku 3.5. Kontrolní jednotka na základě informace o volném místu v každé frontě rozhoduje, do které z těchto front se přepošle aktuální paket ze vstupu. Jako rozhodovací algoritmus je použit princip výběru nejprázdnější fronty, který zajišťuje rovnoměrné rozložení paketů ze vstupního toku do toků

výstupních. Pomocí simulací byly zkoumány i sofistifikovanější metody výběru výstupní fronty, ale žádná nedosahovala lepších vlastností než uvedená základní metoda. Pokud se paket ze vstupu nevejde do fronty, která je mu přiřazena, dojde k dočasnému blokování dat na vstupu (než se příslušná fronta dostatečně nevyprázdní) a do ostatních front nemohou být ukládány data. K tomu ovšem nebude docházet pokud je celková propustnost jednotek na výstupních rozhraních vyšší nebo rovna propustnosti na rozhraní vstupním.



Obrázek 3.5: FrameLink Splitter.

3.2.3 Další komponenty

FrameLink FIFO – fronta pro uložení paketů ve formě FrameLink protokolu. Řeší zakódování kontrolních signálů FrameLink protokolu do paritních bitů blokových pamětí. Generické parametry umožňují vybrat typ paměti (bloková nebo distribuovaná paměť), šířku datového slova a počet položek fronty. Pro podporu řízení toku dat jsou implementovány signály označující stav zaplnění fronty a signál indikující přítomnost celého rámce ve frontě.

Packet FIFO – rozšířená verze komponenty FrameLink FIFO, která umožňuje zahazování paketů pomocí řídicích signálů na svém výstupním rozhraní.

Packet Releasing FIFO – speciální verze fronty, která při nedostatku místa pro uložení dalších dat zahodí právě ukládaný rámec a všechny následující data až do konce rámce. Toto chování je nutné pokud při zpracování vstupního toku dat, kdy nemůžeme pozastavit příjem paketů, nevadí ztráta paketů při zahlcení a přitom je nutné zachování konzistence dat FrameLink protokolu.

Transformer – komponenta pro transformaci datové šířky protokolu.

FL2CMD a CMD2FL – vzájemná transformace *command protokolu* a FrameLink protokolu. Command protokol se používal při vývoji aplikací na kartách COMBO, ale vykazoval nízkou efektivitou přenosu, a proto byl nahrazen novým FrameLink protokolem. Komponenty umožňující vzájemný převod těchto protokolů urychlují nasazení nového protokolu.

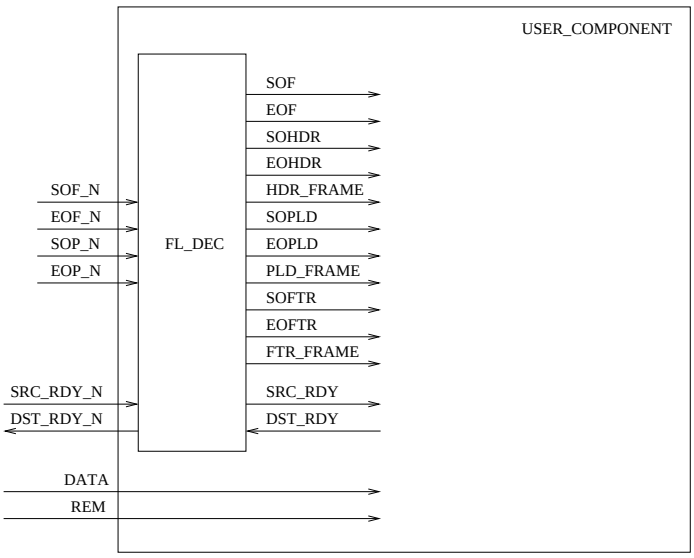
Switch – komponenta pro přepínaní a replikaci toků dat. Jednotka disponuje jedním vstupním a generickým počtem výstupních rozhraní. Pakety ze vstupního rozhraní jsou na základě kontrolních informací uvedených v jejich hlavičkách rozeslány na jedno popřípadě na více výstupních rozhraní zároveň.

Marker – komponenta modifikující obsah rámce dle genericky zadaných parametrů a jednoduchého kontrolního rozhraní. Příkladem využití je označování paketů hodnotou čítače pro určení pořadí příchodu nebo označování příznakem čísla vstupního rozhraní.

Relay – jednotka pro eliminaci kritických cest vzniklých na FrameLink rozhraní. Relay je implementován jako FIFO skládající se z posuvných registrů a nabízí zřetězení logiky pro řízení toku dat.

Watch – diagnostická pasivní komponenta sledující rozhraní FrameLink protokolu. Jednotka hlídá porušení struktur FrameLink rámců a zaznamenává počty paketů na každém rozhraní. Počet hlídaných rozhraní je generický. Uplatnění jednotky je zejména ve fázi testování.

FrameLink Decoder – komponenta určena pro použití v každé jednotce pracující s obsahem dat ve formátu FrameLink protokolu. Dekodér na základě generických parametrů určuje počet částí rámce a explicitně příslušné části označuje. Komponenta dále otáčí negativní logiku FrameLink protokolu, která je vhodná pro mezičipový přenos, do uživatelsky přívětivější pozitivní logiky. Zapojení dekodéru v uživatelské komponentě je zachyceno na obrázku 3.6. Dekodér má vyvedeny signály SOP a EOP pro jednotlivé části rámce a signály FRAME aktivní po celou dobu přenosu příslušných částí.



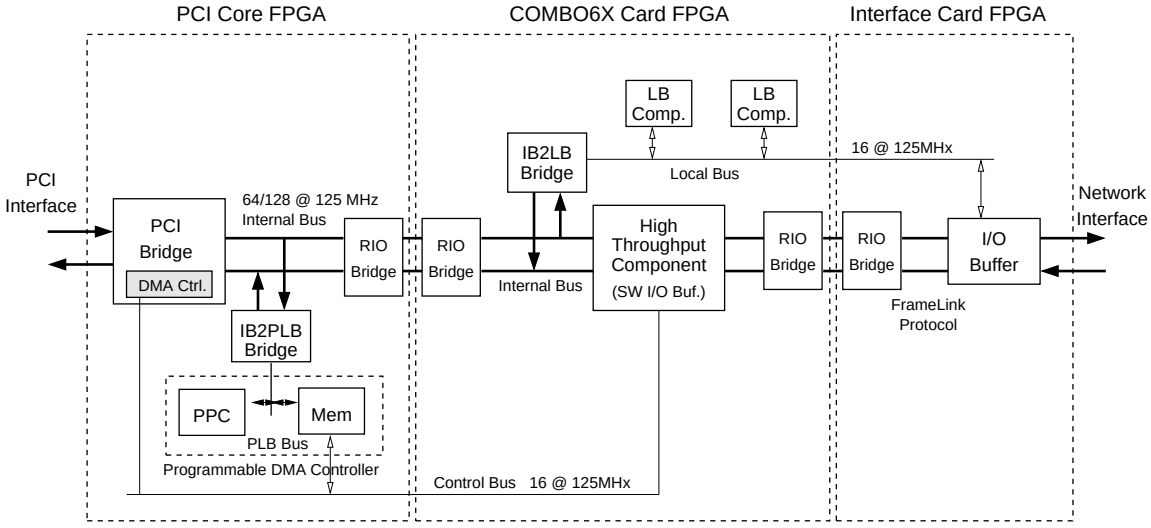
Obrázek 3.6: FrameLink Decoder.

3.3 Propojovací systém sběrnic

Propojovací systém platformy NetCOPE zahrnuje tři typy sběrnic, programovatelný DMA kontrolér a PCI bridge pro připojení na systémové rozhraní PCI, PCI-X či PCI-Express. Detailně je architektura tohoto systému popsána v [20]. Následující kapitola shrnuje nejdůležitější vlastnosti tohoto systému z hlediska jeho využití pro síťové aplikace nad hardwarovou platformou COMBO.

Základní sběrnicí propojovacího systému je *interní sběrnice (Internal Bus)*, která je určena pro připojení komponent vyžadujících vysokou propustnost dat mezi sebou, nebo mezi hardwarovým akcelerátorem a systémovou pamětí RAM. Tato sběrnice je přes PCI bridge připojena na systémovou sběrnici. Protože je interní sběrnice relativně náročná na zdroje programovatelného hradlového pole a ne všechny komponenty vyžadují rychlý přístup ze softwaru, je pro připojení pomalejších komponent určena *lokální sběrnice (Local Bus)*. Tato sběrnice nabízí nižší propustnost a menší nároky na zdroje FPGA. Zpravidla je použita pro přenos konfigurační dat nebo přenos řídicích a stavových informací. Na interní sběrnici je připojena prostřednictvím komponenty LB Bridge.

Posledním typem sběrnic je *řídicí sběrnice (Control Bus)*. Ta je určena pro přenos dat mezi programovatelným DMA kontrolérem a komponentami, které potřebují odesílat data do systémové paměti, nebo je naopak z této paměti přijímat. Úkolem zmíněného kontroléru je řídit veškeré DMA operace mezi programovatelným hradlovým polem a softwarovým ovladačem. Jelikož programovatelnost a rychlost těchto operací je jedním z nejdůležitějších požadavků na platformu NetCOPE, je jako hlavní prvek DMA řadiče využit procesor PowerPC.



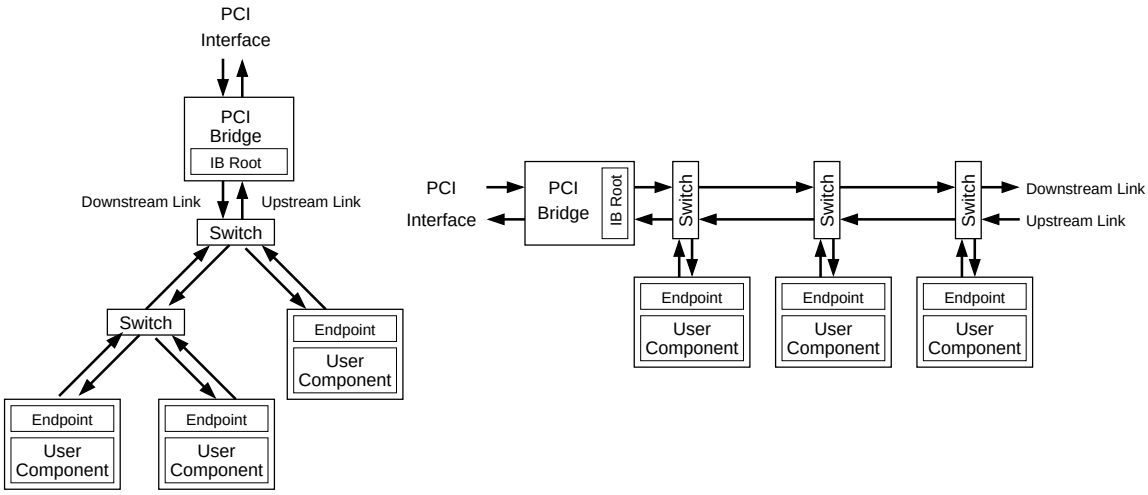
Obrázek 3.7: Architektura interního systému sběrnic.

Architektura navrhovaného propojovacího systému sběrnic pro kartu COMBO6X je zachycena na obrázku 3.7. PCI core FPGA obsahuje bridge systémové sběrnice a programovatelný DMA kontrolér. Interní sběrnice je k hlavnímu hradlovému poli přivedena přes rozhraní RocketIO, kontrolní sběrnice je přivedena přes sadu paralelních vodičů. Na hlavním

FPGA je rozvedena interní a lokální sběrnice. Na přídavnou kartu rozhraní je přivedena pouze lokální sběrnice a propojení přes RocketIO je využito k přenosu dat ve formátu FrameLink protokolu.

3.3.1 Interní sběrnice

Interní sběrnice je klíčovým prvkem navrženého propojovacího sytému sběrnic a její hlavní vlastností je vysoká propustnost. Sběrnice je určená pro přenosy dat mezi FPGA čipem a systémovou pamětí RAM a pro přenos dat mezi jednotlivými komponentami v rámci FPGA. Sběrnice je založená na stromové architektuře a její hlavní části tvoří komponenty Root, Switch a Endpoint (viz obrázek 3.8). Komponenta Root je součástí PCI Bridge, který zajišťuje komunikaci se systémovou sběrnicí PCI. Komponenty typu Switch směrují veškerou komunikaci v rámci interní sběrnice a jednotlivé Endpoint bloky připojují komponenty s požadavkem na vysokou propustnost dat. Typickými příklady takovýchto komponent jsou softwarové přijímací a odesílací buffery, interní DRAM kontrolér, bridge mezi interní a lokální sběrnicí a potenciálně i některý z vestavěných procesorů PowerPC.



Obrázek 3.8: Architektura interní sběrnice.

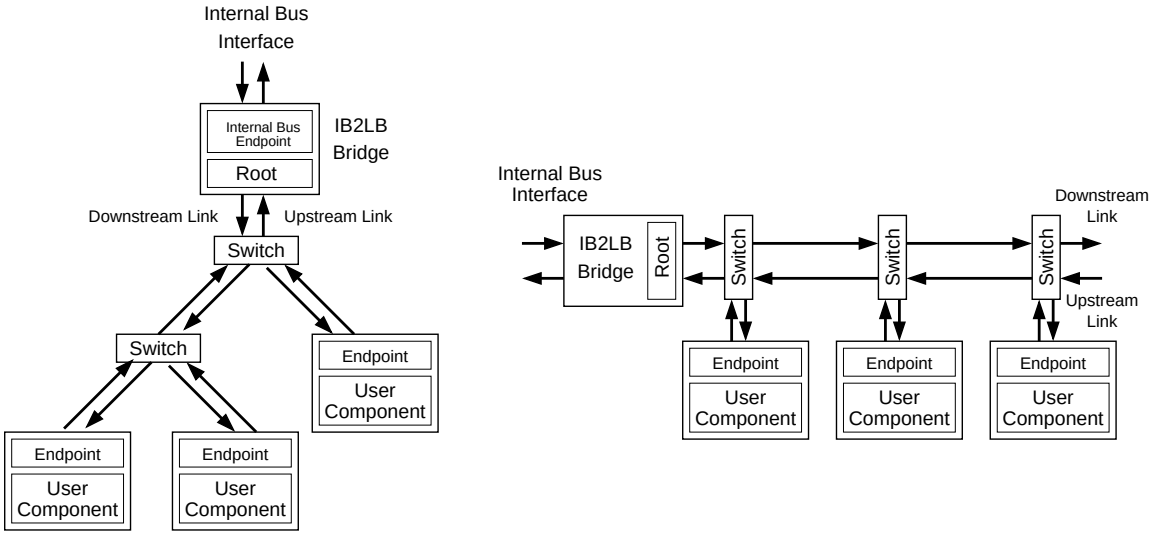
Linka interní sběrnice je široká 64 nebo 128 bitů (v závislosti na konfiguraci) a její pracovní frekvence je 125 MHz. Každá linka je plně duplexní, takže data mohou být přenášena oběma směry současně. Tato vlastnost umožňuje jednoduché zapojení na novou generaci sběrnic PCI-Express a je velice užitečná zejména pro oblast síťových aplikací. Stromová architektura umožňuje Endpointům komunikovat každý s každým a pokud oba uzly leží ve stejné větvi, nedochází k vytěžování ostatních linek interní sběrnice. Jak je naznačeno na obrázku 3.8, stromová architektura může být jednoduše převedena do formy sběrnice, ve které jednotlivé Switch komponenty tvoří stupně ve smyslu zřetězení. Tento druh zřetězení je velmi důležitý zejména pro technologii FPGA, kde vlivem omezených zdrojů pro propojování komponent jsou takto široké sběrnice velmi citlivé na vzdálenost.

Komunikační protokol na interní sběrnici je založen na posílání paketů. Podobně jako u sběrnice PCI-Express jsou definovány transakce typu *Read*, *Write* a *Completion*. Jednou z hlavních výhod použitého komunikačního protokolu je i jeho kompatibilita s protokoly

FrameLink a Xilinx LocalLink. Díky této kompatibilitě lze do infrastruktury interní sběrnice snadno připojovat specializované IP cores komponenty, jako je například *Aurora* od firmy Xilinx zajišťující komunikaci přes multigigabitové spoje *RocketIO*.

3.3.2 Lokální sběrnice

Lokální sběrnice je pomalejší a úspornější sběrnice sloužící k připojení komponent, které nevyžadují vysokou propustnost do softwaru. Jak je zřejmé z obrázku 3.9, tato sběrnice má také stromovou architekturu a hlavní komponenty jsou opět Root, Switch a Endpoint. Ve srovnání s interní sběrnicí je lokální sběrnice mnohem jednodušší. Veškeré čtecí nebo zápisové transakce mohou být iniciovány pouze komponentou Root. Komponenty Endpoint pracují v podřízeném módu a prvky Switch pouze přeposílají transakce z upstream portů na downstream porty. Podobně jako u interní sběrnice, komponenty Switch pomáhají zřetězit komunikaci na sběrnici a redukovat její citlivost na vzdálenost.



Obrázek 3.9: Architektura lokální sběrnice.

Linka lokální sběrnice je 16 bitů široká a pracuje na frekvenci 125 MHz. Přestože je každá linka obousměrná, komunikace probíhá vždy pouze jedním směrem. Oboustrannost je nutná pro odstranění třístavových sběrnic, které jsou nevhodné k realizaci v FPGA. Komunikační protokol rozlišuje pouze základní typ čtecí a zápisové transakce a jeho předností je možnost jednoduše jej použít i pro komunikaci mezi více FPGA čipy bez potřeby speciálních komponent typu Bridge.

3.3.3 Řídicí sběrnice

Řídicí sběrnice je specializována na přenos řídicích informací mezi programovatelným DMA kontrolérem a komponentami, které potřebují přenášet bloky dat mezi hradlový polem a systémovou pamětí RAM. Typicky ji využívají komponenty jako jsou DRAM kontrolér, přijímací a vysílací softwarový buffer, či procesor PowerPC. Podobně jako interní a lokální má i řídicí sběrnice stromovou strukturu. Komponenta Root je součástí programovatelného

DMA kontroléru, komponenty Switch pouze přeposílají pakety mezi upstream a downstream porty a Endpointy připojují komponenty vyžadující DMA přenosy. Komponenta Root může posílat pakety libovolnému Endpointu a libovolný Endpoint může poslat paket komponentě Root. Komunikace mezi Endpointy není povolena. Řídicí sběrnice je plně-duplexní, má šířku linky 16 bitů a pracuje na 125 MHz synchronně s ostatními částmi sběrnicového systému NetCOPE. Komunikační protokol je kompatibilní s protokolem FrameLink.

3.3.4 Programovatelný DMA řadič

Hlavní úlohou programovatelného DMA řadiče (PDMA) je řídit veškeré DMA operace mezi FPGA adaptérem a systémovou pamětí RAM. Důležitou vlastností komponenty je přitom právě programovatelnost, neboť každá aplikace běžící na platformě NetCOPE může mít jiné požadavky na DMA operace. Architektura PDMA je složena z procesoru PowerPC, komponenty Root řídicí sběrnice a lokálních pamětí pro program a data procesoru. Typicky PDMA pracuje v následujících krocích:

1. načte scatter-gather seznamy (SG) z RAM do lokální paměti,
2. zpracuje všechny položky SG seznamu, kde každá položka reprezentuje DMA operaci,
3. modifikuje příslušné parametry SG seznamu,
4. na závěr přenese upravený SG seznam zpět do paměti RAM (a eventuelně vygeneruje přerušení).

Jak již bylo uvedeno v části o řídicí sběrnici, veškerá komunikace mezi PDMA a komponentou Endpoint řídicí sběrnice je založena na posílaní paketů (zpráv). Například při příchodu paketu do systému je PDMA odeslána zpráva o této události, PDMA přenese paket do paměti RAM a jakmile je přenos dokončen, zašle PDMA směrem k Endpointu potvrzovací zprávu. Softwarový buffer připojený k příslušnému Endpointu následně paket uvolní ze své paměti. Tento princip eliminuje latence při zpracování příchozích i odchozích paketů a je vhodný pro použití v prostředí vysokorychlostních síťových aplikací.

3.4 IP cores

Do množiny bloků ve formátu IP cores jsou zařazeny tři skupiny komponent. První skupina je primárně určena pro použití v platformě NetCOPE a tvoří důležitou část její infrastruktury. Jedná se o vstupní a výstupní bloky síťového rozhraní, softwarové buffery pro uložení paketů a komponenty pro přenos dat po multigigabitových transceiverech RocketIO. Druhou skupinu tvoří sada bloků pro řízení externích prvků na kartě a třetí skupinu sada jednotek pro analýzu a zpracování síťového provozu. Komponenty druhé a třetí skupiny byly vyvinuty v rámci projektu Liberouter a v rámci platformy NetCOPE bylo pouze upraveno jejich rozhraní pro jejich snadné použití při vývoji aplikací nad touto platformou. Následující kapitoly vyvinuté IP cores bloky podrobněji představují.

3.4.1 Vstupní a výstupní bloky síťového rozhraní

Vstupní buffer (Input Buffer, IBUF) pro gigabitový ethernet a jeho podvrstvu pro nezávislý přístup k médiu GMII (Gigabit Media Independent Interface) implementuje přijímací část MAC vrstvy specifikované v IEEE standardu 802.3 [6]. Komponenta přijímá ethernetový rámec z GMII rozhraní, provádí kontroly rámce, ukládá data do vnitřního bufferu a poskytuje je na svém výstupním rozhraní ve formátu FrameLink protokolu. Komponenta dále umožňuje přidávat k datům paketu kontrolní data ve formě hlavičky či patičky FrameLink protokolu a také umožňuje vzorkování paketů na vstupním rozhraní.

Prováděné kontroly zahrnují kontrolu cyklického součtu CRC, kontrolu MAC adresy a kontrolu maximální a minimální délky rámce. Hodnoty těchto délek jsou stejně jako množina přijímaných MAC adres nastavitelné ze softwaru. Při nesplnění některé z kontrol je na základě softwarově nastavitelné masky rozhodnuto, zda je paket přijat či naopak zahozen. Softwarové řízení dále zahrnuje operace pro zapnutí a vypnutí příjmu paketů, možnost nastavení nižší rychlosti rozhraní pro kompatibilitu s desetimegabitovým a stomegabitovým ethernetem a možnost vyčítání statistik vstupního rozhraní. Tyto statistiky zahrnují celkový počet přijatých paketů, počet správně přijatých paketů a počet zahozených paketů.

Mimo vstupní GMII rozhraní a výstupní FrameLink rozhraní komponenta dále zahrnuje dvě externí řídicí rozhraní – jedno pro podporu vzorkování a jedno pro přidávání kontrolních dat k paketům. Rozhraní pro generování kontrolních dat bylo pojmenováno PACODAG (PAcket COntrol DAta Generator) a umožňuje generovat hlavičku a patičku FrameLink protokolu ke každému paketu. Rozhraní je obousměrné, uživatelské komponentě generující kontrolní data jsou posílány statistiky získané v jednotce IBUF a oznámení o příchodu nového paketu, uživatelská komponenta na každé takové oznámení musí reagovat vygenerováním kontrolních dat. Rozhraní pro podporu vzorkování je pojmenováno SAU (rozhraní k *Sampling Unit*). Po oznámení přijetí nového paketu přes PACODAG rozhraní může externí komponenta prostřednictvím dvou signálů rozhodnout zda paket přijmout či nikoliv. V případě zahození je uvolněno příslušné místo ve frontě a paket nebude dán k dispozici na výstupní FrameLink rozhraní, v opačném případě bude paket předán na výstupní rozhraní směrem k aplikaci.

Výstupní buffer (Output Buffer, OBUF) pro gigabitový ethernet a jeho podvrstvu pro nezávislý přístup k médiu GMII implementuje vysílací část MAC vrstvy. Komponenta přijímá data ve formě paketů FrameLink protokolu, ukládá je v dočasné paměti a vysílá je na výstupní GMII rozhraní. Stejně jako u vstupního bufferu má komponenta softwarové rozhraní umožňující zapínání a vypínání vysílání paketů, nastavení rychlosti linky (10/100/1000 Ethernet), nastavení MAC adresy a čtení statistik s počtem odeslaných paketů. Na základě řídicích dat uložených v patičce FrameLink rámce je možné zvolit zda se zdrojová MAC adresa přepíše MAC adresou uloženou v jednotce OBUF či nikoliv.

Pro větší variabilitu při příjmu a vysílání paketů na čtyřportové kartě byl navržen a implementován GMII přepínač (crossbar). Tato komponenta je určena pro zapojení na vstupní a výstupní rozhraní před komponenty IBUF a OBUF a umožňuje realizovat libovolné propojení vstupních a výstupních portů v závislosti na softwarovém nastavení. Výsledkem je možnost realizace opakovačů nad libovolnými porty karty rozhraní.

3.4.2 Softwarové buffery

Důležitými komponentami z hlediska rychlého přenosu dat mezi adaptérem a pamětí RAM hostujícího počítače jsou buffery pro uložení paketů. Úkolem přijímacího bufferu je dočasné uložení paketů přicházejících ze vstupních síťových rozhraní před transportem přes systém sběrnic do paměti RAM. Rozhraním na vstupu komponenty je FrameLink generické šířky. Pro přenos paketů do softwaru je využito rychlé *interní sběrnice* a pro předávání zpráv programovatelnému DMA řadiči je implementováno rozhraní *kontrolní sběrnice*.

Architektura jednotky se skládá z kruhového bufferu pro uložení dat paketů, řídicí jednotky a fronty pro uložení informací o příchozích paketech jakou jsou počáteční adresa, délka paketu a případně příznaky. Princip přenosu paketů do softwaru je následující. Po přijetí je paket uložen v paměti a po kontrolní sběrnici je programovatelnému řadiči zaslána zpráva obsahující adresu a délku paketu. PDMA následně vyvolá přenos paketu po interní sběrnici do softwaru a po jeho dokončení informuje softwarový buffer zasláním zprávy po kontrolní sběrnici, že paket byl přenesen. Následně je paket v kruhovém bufferu vymazán.

Inverzní komponentou zajišťující dočasné uložení paketů po přenosu paketu z paměti RAM do adaptéru a před vysláním na výstupní síťové rozhraní je odesílací softwarový buffer. Architektura i princip fungování jsou u něj přesně opačné. Paket je do paměti komponenty uložen přes interní sběrnici a PDMA poté zašle komponentě zprávu, že na příslušné adrese je uložen paket jisté délky. Buffer zpracuje zprávu a odešle paket z dané adresy na výstupní rozhraní, které je tvořeno FrameLink protokolem generické šířky. Po odeslání paketu je z komponenty do PDMA zaslána zpráva oznamující úspěšné provedení operace.

3.4.3 Jednotky pro přenos dat přes RocketIO

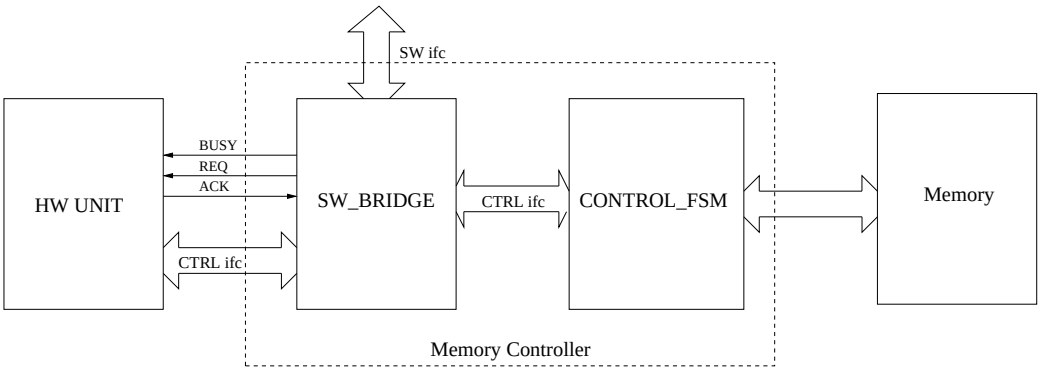
Pro přenos dat mezi kartami jsou využity multigigabitové transceivery (přijímače/vysílače) RocketIO. Tyto vysokorychlostní sériové rozhraní mohou fungovat na několika typech linkových vrstev. Příkladem může být technologie Ethernet. Pro uživatelské využití při přenosu mezi čipy nebo kartami Xilinx doporučuje používat jim definovaný protokol *Aurora* a dodávané IP cores. Ty poskytují LocalLink rozhraní pro přenos dat a uživateli nabízí možnost spojování více fyzických kanálů do jednoho virtuálního a rozhraní pro řízení toku dat. Na základě tohoto rozhraní jsou navrženy dvě komponenty pro přenos dat přes RocketIO.

První komponentou je *Aurora with Flow Control (AURFC)*, která rozšiřuje možnosti volně dostupného IP coru o řízení toku dat a převod do FrameLink protokolu. Komponenta je rozdělena do dvou částí. Odesílací část ukládá pakety ze vstupního rozhraní do vyrovnávací paměti a odesílá je po RocketIO do přijímací části. Přijímací část pakety ukládá do vyrovnávací paměti, převádí do FrameLink protokolu a odesílá na výstupní rozhraní. Pokud je vyrovnávací paměť na přijímací straně zaplněna, pošle se rozhraním pro řízení toku dat zpráva pro přerušení vysílání dat. Poté co se v bufferu místo opět uvolní, je zaslána zpráva pro pokračování přenosu dat.

Druhou komponentou je *Aurora with Virtual Channels (AURVC)* implementující v rámci jednoho fyzického přenosového kanálu generický počet virtuálních kanálů. Zatímco v případě *AURFC* je nutné při přenosu dat přes RocketIO spojit všechny toky dat do jednoho, tato komponenta umožňuje zachovat jejich oddělení a zabránit možnému blokování na výstupu. Důvodem pro implementaci této komponenty je zejména počet RocketIO kanálů na propojení karet rodiny COMBO, který nedovoluje pro každé síťové rozhraní použít jeden fyzický kanál.

3.4.4 Řadiče externích prvků

Mezi externí prvky na kartách COMBO patří zejména statické, asociativní a dynamické paměti. Řadiče těchto pamětí byly vyvinuty v rámci projektu Liberouter. V rámci projektu NetCOPE byla sjednocena koncepce těchto řadičů a bylo ujednoceno rozhraní k softwaru, které tyto řadiče využívají. Výsledná architektura řadičů je zachycena na obrázku 3.10. Řadič je rozdělen do dvou částí, kdy komunikaci s externí pamětí a vlastní řízení této paměti realizuje řídicí část (control fsm). Ta poskytuje obecné rozhraní (CTRL ifc) pro řízení příslušného typu paměti. Na toto rozhraní může být přímo připojena jednotka využívající služeb řadiče (HW unit). Pokud je zároveň nutný přístup k paměti ze softwaru, využije se druhá část řadiče – softwarový bridge. Ten zpřístupňuje veškeré operace s pamětí programovému vybavení počítače (SW ifc) a realizuje multiplexování komunikace s řídicí části mezi hardwarovou jednotku a softwarem.



Obrázek 3.10: Architektura řadičů pamětí.

Do kategorie řadičů externích prvků na kartách COMBO dále patří jednotka pro řízení phyterů přes I2C rozhraní. Tato jednotka umožňuje softwaru číst stavové a nastavit řídicí registry těchto prvků, které zajišťují fyzický příjem a vysílání paketů na síťovém rozhraní. Příkladem informace, kterou phytery poskytují, je ustanovená rychlost linky. V závislosti na hodnotě této rychlosti musí být nastaveny vstupní a výstupní síťové bloky.

3.4.5 Bloky pro analýzu a zpracování síťového provozu

Následující komponenty byly vyvinuty v rámci projektu Liberouter a pro použití v projektu NetCOPE bylo pouze upraveno jejich rozhraní. Jedná se o bloky umožňující základní operace nad síťovými daty jako je analýza, klasifikace a editace paketů. Kompletní seznam lze nalézt v [1].

GENA – generický nanoprocesor určený pro zpracování paketů. Šířka datového slova je 16 bitů, operační frekvence 100 MHz. Podrobný popis lze nalézt v [21].

Header Field Extractor – procesor založený na jádře GENA určený pro analýzu hlaviček paketů a extrakci informací z těchto hlaviček [21].

Output Packet Editor – proudový procesor založený na jádře GENA určený pro analýzu a editaci dat paketů [22].

Look Up Processor – klasifikační procesor založený na kombinaci vyhledávání v asociativní paměti a vykonání sekvenčního programu procesní jednotky [23].

Pattern Matching Unit – jednotka pro vyhledávání vzorů v datech paketu. Dostupné jsou dvě implementace, z nichž jedna využívá externí asociativní paměť [24], druhá je realizována prostřednictvím nedeterministických konečných automatů v FPGA [14].

Flow Context – systém pro zpracování stavů TCP/IP toků [25].

Time Stamp Unit – jednotka poskytující přesné časové značky [1]. Implementace na přídavné kartě COMBO-PTM, přesnost zajištěna synchronizací přes GPS.

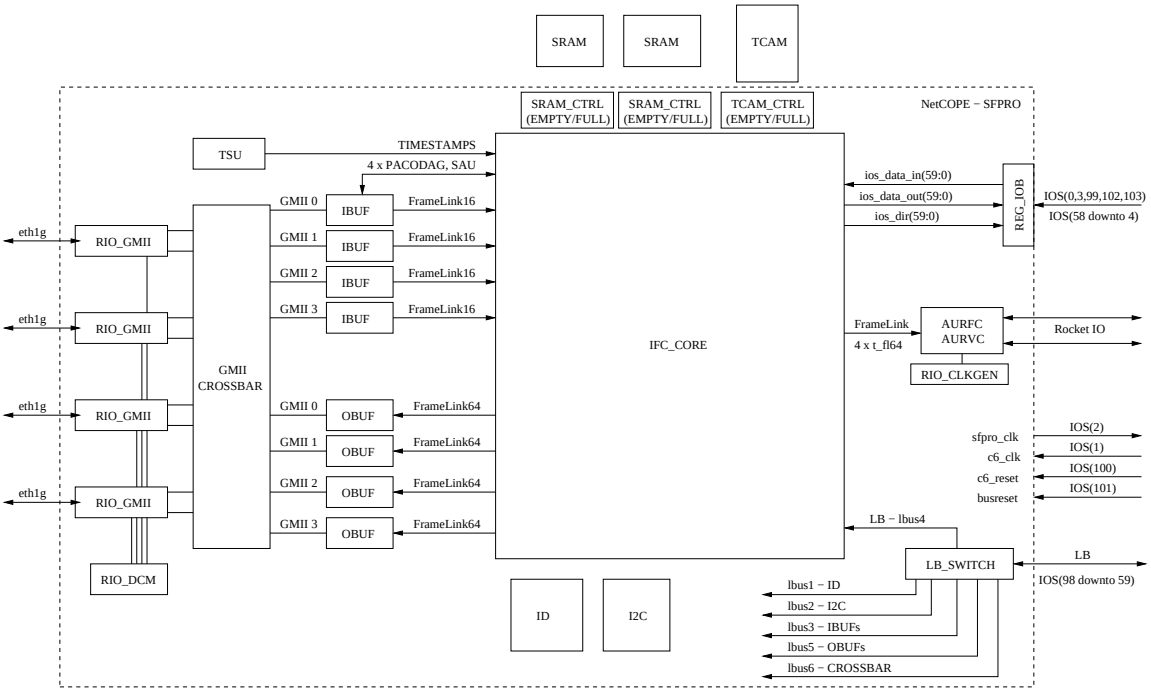
3.5 Architektura NetCOPE pro rodinu karet COMBO

Architektura navržené platformy je založena na blocích představených v předchozích kapitolách s důrazem na co možná nejlepší využití rodiny karet COMBO. Důležitým prvkem při návrhu architektury byla snaha o maximální možnou obecnost a generičnost platformy, s cílem využitelnosti pro co nejširší škálu zařízení. Architektura systému reflektuje způsob použití karet COMBO a je proto rozdělena do dvou částí – pro kartu rozhraní a pro základní kartu. Uživatelská aplikace může být s ohledem na zdroje FPGA libovolně rozdělena mezi obě karty.

3.5.1 Karta rozhraní

Referenční kartou rozhraní, pro kterou je platforma navržena a implementována, je karta SFPRO. Základní úlohou této karty je komunikace se síťovým rozhraním – příjem a vysílání paketů a konfigurace phyterů. Dalším důležitým úkolem je zajištění komunikace se základní kartou přes multigigabitové sériové transceivery RocketIO a skupinu paralelních vodičů IOS. Blokové schéma navržené architektury je zachyceno na obrázku 3.11. Architektura zahrnuje převodník ze síťových rozhraní na GMII protokol (rio gmii), GMII přepínač (crossbar), vstupní a výstupní síťové bloky (ibuf, obuf), řadiče statických a asociativní pamětí (sram ctrl, tcam ctrl), komponenty pro zajištění přenosu dat po paralelních vodičích IOS a po kanálech RocketIO (aurfc, aurvc), lokální sběrnici, ID komponentu, I2C komponentu pro komunikaci s phytery, jednotku pro generování přesných časových značek (tsu) a prostor pro uživatelskou aplikaci (ifc core).

Pro příjem paketů ze vstupních rozhraní je využit Xilinx IP core pro RocketIO, který převádí data do protokolu MAC vrstvy GMII. Tato data jsou následně zpracována vstupním síťovým blokem a převedena do formátu protokolu FrameLink. Pro přidávání kontrolních dat k paketu je využito kontrolní rozhraní PACODAG, pro vzorkování paketů ve vstupních síťových blocích je využito rozhraní SAU. Rozhraní PACODAG umožňuje z aplikace ke každému paketu generovat libovolný obsah hlavičky a patičky ve formátu FrameLink protokolu. Rozhraní SAU směrem k aplikaci oznamuje příjem nového paketu a směrem z aplikace je posíláno rozhodnutí, zda paket zpracovat či zahodit. Odesílání paketů je realizováno výstupním síťovým blokem. Vlastní příjem a vysílání paketů na fyzické vrstvě síťového rozhraní zajišťují externí phytery. Stavové a řídicí informace těchto phyterů je možné číst a zapisovat pomocí I2C komponenty.



Obrázek 3.11: Blokové schéma architektury NetCOPE – karta rozhraní.

Přenos dat mezi kartami je realizován přes kanály RocketIO a paralelní sběrnici IOS. Sběrnice IOS zahrnuje 104 paralelních vodičů, na kterých je možné realizovat spolehlivé přenosy do frekvence 100 MHz, což znamená propustnost 10 Gb/s. Čtyři z těchto signálu jsou použity pro kontrolní signály. Ze základní karty jsou přivedeny signály reset, busreset (reset pro sběrnice) a hodiny, naopak na základní kartu jsou posílány hodiny z karty rozhraní. Tyto hodiny umožňují na základní kartě ve správný okamžik vzorkovat data přicházející souběžně s těmito hodinami. Dalších 40 signálů je využito pro přivedení lokální sběrnice na kartu rozhraní. Zbývajících 60 signálů s propustností 6 Gb/s je dáno k dispozici uživateli pro využití při transportu dat v rámci aplikace.

Mezi základní kartou a kartou rozhraní jsou k dispozici dva plně duplexní RocketIO kanály s celkovou propustností 4 Gb/s v každém směru. Pro řízení těchto kanálů je možné z aplikace zvolit komponenty AURFC či AURVC. Komponenta AURFC (Aurora with flow control) nabízí jeden plně duplexní kanál pro 64-bitový FrameLink. Komponenta AURVC (Aurora with virtual channels) nabízí generický počet vzájemně oddělených kanálů pro 64-bitový FrameLink. Obě komponenty zajišťují kontrolu toku dat. Celková propustnost mezi kartami je tedy 14 Gb/s, v jednom směru maximálně 10 Gb/s.

Posledními bloky architektury NetCOPE pro kartu rozhraní jsou ID komponenta a komponenta pro generování přesných časových značek (Time Stamp Unit, TSU). ID komponenta je připojena na lokální sběrnici a slouží pro uložení informací typu jména a verze aplikace. TSU komponenta generuje přesné časové značky využitelné pro zaznamenání přesného času příchodu paketů. Tyto značky mohou být pomocí rozhraní PACODAG přidány do kontrolních dat rámců a po exportování do softwaru využity pro analýzu časových vlastností sítě. Funkčnost TSU jednotky je podmíněna připojením karty PTM ke kartě rozhraní.

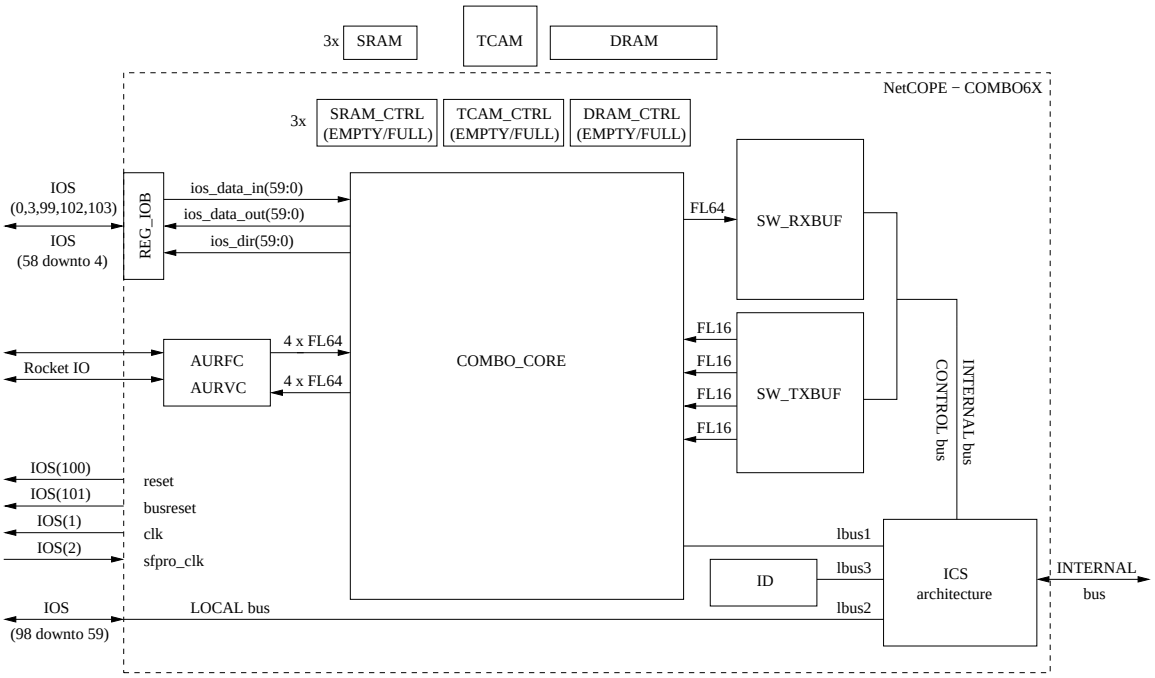
Ve VHDL je výše popsaná architektura implementována genericky a je možné si z aplikace volit, zda se mají či nemají instancovat řadiče pamětí, GMII přepínač, bloky výstupního síťového rozhraní, prvky pro zajištění přenosu po vodičích IOS a po kanálech RocketIO či jednotka pro generování přesných časových značek. Z aplikace je dále možné si volit důležité generické parametry jednotlivých komponent jako jsou například velikosti bufferů vstupních a výstupních síťových bloků.

Kompletní infrastruktura NetCOPE pro kartu SFPRO byla otestována v hardwaru na frekvenci 100 MHz a zabírá v závislosti na nastavení velikostí bufferů 4000 až 7000 sliců FPGA Virtex II Pro. Při nevyužitím některých z dostupných jednotek jsou nároky na zdroje programovatelného hradlového pole ještě nižší. Rozhraní k uživatelské aplikaci je naznačeno na obrázku 3.11 a je popsáno v rámci předchozích odstavců. Tvoří jej:

- řídicí signály – hodiny (100 MHz) a reset.
- 4 x IBUF rozhraní – FrameLink, generická datová šířka.
- 4 x PACODAG rozhraní – generování kontrolních dat k paketům.
- 4 x SAU rozhraní – vzorkování vstupních paketů.
- TimeStamp rozhraní – přesné časové značky.
- 4 x OBUF rozhraní – FrameLink, generická datová šířka.
- 4 x Aurora rozhraní – 64-bitový FrameLink, 4 vstupní a 4 výstupní rozhraní. Při použití AURVC se použijí všechny rozhraní, při použití AURFC se využívá pouze první rozhraní v obou směrech.
- IOS – rozhraní k 60 třístavovým vodičům, signály pro vstup dat a výstup dat a signály pro řízení směru přenosu.
- 2 x SRAM rozhraní – obecné rozhraní k paměťovému řadiči.
- 1 x TCAM rozhraní – obecné rozhraní k řadiči asociativní paměti.
- lokální sběrnice – přivedení sběrnice ze základní karty.

3.5.2 Základní karta

Architektura NetCOPE pro základní kartu je navržena pro karty COMBO6X a COMBO6E. Úkolem architektury je řídit komunikaci se systémovou sběrnicí, implementovat bloky pro realizaci rychlého přenosu dat mezi adaptérem a pamětí RAM hostujícího počítače, zajistit komunikaci s kartou rozhraní a vytvořit prostor pro realizaci jádra aplikace. Blokové schéma navržené architektury je znázorněno na obrázku 3.12. Platforma NetCOPE pro základní kartu zahrnuje propojovací systém sběrnic (interconnection system, ICS), softwarové buffery pro příjem a odesílání paketů (sw rxbuf, sw txbuf), řadiče statických pamětí (sram ctrl), řadič asociativní a dynamické paměti (tcam ctrl, dram ctrl), komponenty pro zajištění přenosu dat po vodičích IOS a přes RocketIO (aurfc, aurvc), ID komponentu a prostor pro uživatelskou aplikaci (combo core).

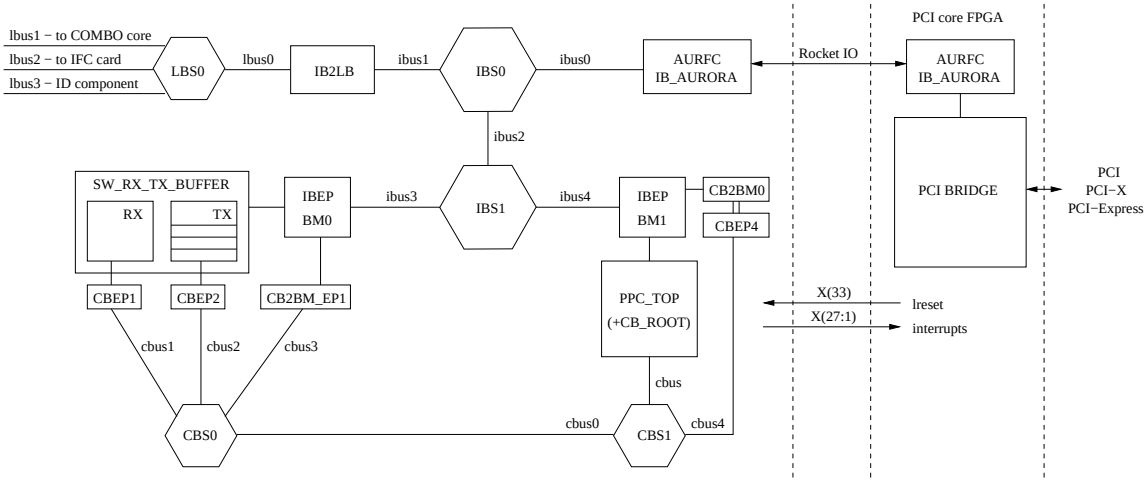


Obrázek 3.12: Blokové schéma architektury NetCOPE – základní karta.

Přenos dat mezi kartami je stejně jako na kartě rozhraní realizován přes sériové kanály RocketIO a paralelní sběrnici IOS. Stejnou funkci mají i paměťové řadiče a ID komponenta. Novými prvky jsou naopak softwarové buffery a propojovací systém sběrnic. Funkce softwarových bufferů je uložení paketů před jejich přenosem do softwaru, respektive uložení a následné odvysílání paketů po jejich přenosu ze softwaru. Instance přijímacího a vysílacího bufferu genericky podmíněné a je možné nainstancovat pouze RX buffer (pro monitorovací aplikace), nebo naopak pouze TX buffer (pro paketový generátor). Přijímací buffer je z důvodu využití distribuce toků a virtualizace rozhraní směrem k softwaru instancován jeden. U odesílacích bufferů je z aplikace možné genericky volit jednu nebo čtyři instance, neboť tato volba může být aplikačně závislá.

Architektura propojovacího systému je zachycena na obrázku 3.13 a zahrnuje všechny tři typy sběrnic a programovatelný DMA řadič. Vzájemný převod z interní na systémovou sběrnici a naopak je realizován v PCI core FPGA komponentou PCI bridge. Interní sběrnice je následně pomocí RocketIO a AURFC komponenty přivedena na hlavní FPGA základní karty. Zde je implementována stromová struktura této sběrnice se třemi endpointy. První tvoří *Local bus Root (IB2LB)*, který zajišťuje převod na lokální sběrnici, druhý endpoint je určen pro softwarové buffery a třetí pro programovatelný DMA řadič (ppc top).

Lokální sběrnice na základní kartě zahrnuje mimo komponenty *Local bus Root* jeden tříportový switch, který tvoří rozhraní této sběrnice pro rozšiřující kartu, uživatelskou aplikaci a ID komponentu. Kontrolní sběrnice propojuje programovatelný DMA řadič, softwarové buffery a endpointy interní sběrnice s podporou bus master přenosů. Komponenta *Control bus Root* je integrována v *PPC TOP*, ostatní prvky jsou k této sběrnici připojeny pomocí switchů a endpointů.



Obrázek 3.13: Architektura systému sběrnic.

Podobně jako v případě architektury pro kartu rozhraní je výše popsaná architektura genericky naimplementována ve VHDL. Možnosti volby zahrnují použití komponent AURFC či AURVC, volitelné instancování softwarových bufferů a paměťových řadičů a možnost definování všech vnitřních parametrů architektury. Kompletní infrastruktura byla otestována v hardwaru na frekvenci 100 MHz a zabírá v závislosti na nastavení velikostí bufferů 6000 až 8000 sliců FPGA Virtex II Pro. Rozhraní k uživatelské aplikaci je naznačeno na obrázku 3.12 a tvoří jej:

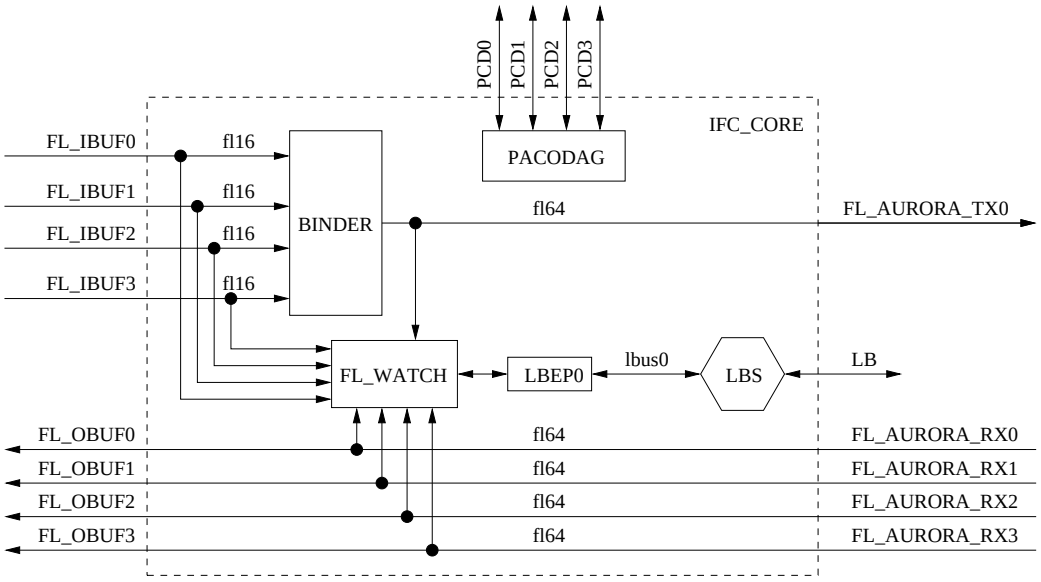
- řídicí signály – hodiny (100 MHz) a reset
- 1 x rozhraní k přijímacímu softwarovému bufferu – FrameLink, generická datová šířka.
- 4 x rozhraní z odesílacího softwarového bufferu – FrameLink, generická datová šířka. Při použití čtyř odesílacích bufferů se použijí všechny rozhraní, při použití jednoho se využívá pouze první rozhraní.
- 4 x Aurora rozhraní – 64-bitový FrameLink, 4 vstupní a 4 výstupní rozhraní. Při použití AURVC se použijí všechny rozhraní, při použití AURFC se využívá pouze první rozhraní v obou směrech.
- IOS – rozhraní k 60 třístavovým vodičům, tvořeno signály pro vstup dat, výstup dat a signálem pro řízení směru přenosu.
- 3 x SRAM rozhraní – obecné rozhraní k paměťovému řadiči.
- 1 x TCAM rozhraní – obecné rozhraní k řadiči asociativní paměti.
- 1 x DRAM rozhraní – obecné rozhraní k řadiči dynamické paměti.
- lokální sběrnice – softwarové rozhraní k uživatelské aplikaci.

Kapitola 4

Síťové aplikace

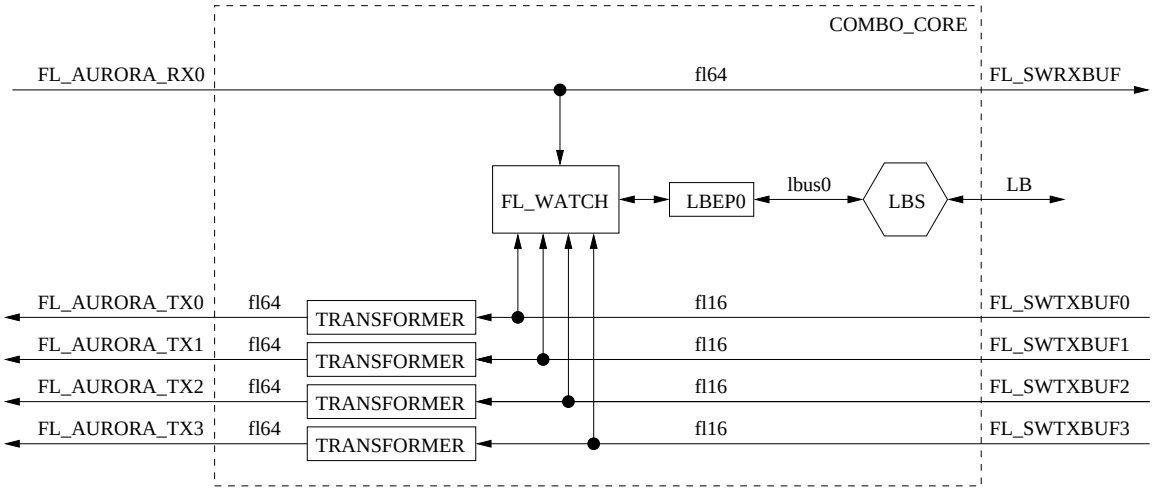
4.1 Síťová karta

První aplikací pro platformu NetCOPE je síťová karta (Network Interface Card, NIC). Tato aplikace byla vytvořena zejména k ověření výkonu propojovacího systému, odladění systémových ovladačů a otestovaní základních modulů IP cores. Architektura síťové karty z větší části pouze využívá prvky infrastruktury NetCOPE (viz obrázky 3.11 a 3.12) a v aplikačních jádrech je soustředěno pouze minimum funkčnosti. Schéma architektury síťové karty pro kartu rozhraní je zachyceno na obrázku 4.1, schéma architektury pro základní kartu na obrázku 4.2.



Obrázek 4.1: Architektura síťové karty pro kartu rozhraní.

Data ze vstupních síťových rozhraní jsou pomocí IP coru firmy Xilinx (rio gmii) převedena do protokolu MAC vrstvy a následně zpracována vstupním síťovým blokem. Ten realizuje příjem paketů a převedení do vnitřního protokolu FrameLink. Pomocí rozhraní



Obrázek 4.2: Architektura síťové karty pro základní kartu.

PACODAG se ke každému paketu do hlavičky FrameLink protokolu přiřadí číslo rozhraní, ze kterého paket přišel, a volitelně se může přiřadit časové razítko z TimeStamp komponenty. Vzorkovací rozhraní SAU u vstupních síťových bufferů není využito. Data ze čtyř vstupních rozhraní jsou poté komponentou Binder spojena do jednoho toku a pomocí jednotky AURFC přenesena na základní kartu. Tam jsou data stejnou komponentou přijata a přímo předána přijímacímu softwarovému bufferu. V jádru aplikace není mimo předání dat žádná funkcionalita. Ze softwarového bufferu jsou data paketů přenesena architekturou propojovacího systému do paměti RAM hostujícího počítače a pomocí ovladače jsou předána operačnímu systému.

V opačném směru je tok dat podobný. Nejprve jsou data transportována DMA přenosem z paměti hostujícího počítače do odesílacího softwarového bufferu. Následně jsou po odeslání z tohoto bufferu v jádře aplikace transformována z šířky 16 bitů na 64 bitů a předána komponentě AURVC pro přenos nezávislých kanálů na kartu rozhraní. Na této kartě jsou stejnou komponentou přijata a přímo předána výstupním blokům pro odeslání na síťové rozhraní. Zatímco u přijímacího bufferu se využívá spojení čtyř toků ze síťových rozhraní do jednoho a společný prostor pro uložení paketů. Vysílací buffer je naopak instancován čtyřikrát, jednou pro každé výstupní rozhraní. Důvodem je zamezení blokování na výstupu. Pokud by například první rozhraní bylo nakonfigurováno na rychlost deset megabitů za vteřinu, a společný kruhový buffer by byl naplněn daty pro toto rozhraní, případná data pro jiná rozhraní umístěná na konci bufferu by nemohla být vysílána, protože by byla blokována pomalu odesílanými daty pro první rozhraní.

Pracovní frekvence aplikace je 100 MHz. Pro transport jednoho gigabitového toku dat je vždy vyhrazen 16-ti bitový FrameLink kanál s propustností 1,6 Gb/s, čímž je zajištěna dostatečná propustnost i po přidání kontrolních informací k datům paketu. Při spojení čtyř toků dat je využíván 64 bitový FrameLink kanál s teoretickou propustností 6,4 Gb/s. Při použití osmi oktetů kontrolních dat klesá propustnost při nejkratších 64 bajtových paketech na 89% maximální hodnoty (5,7 Gb/s), což ale výrazně převyšuje maximální propustnost 4 Gb/s všech síťových rozhraní dohromady. Úzkým hrdlem aplikace zůstává pouze propust-

nost mezi malým a velkým FPGA na základní kartě a propustnost mezi základní kartou a kartou rozhraní. V obou případech se jedná o výrobní omezení karet způsobené použitím pouze dvou RocketIO kanálů pro propojení zmíněných prvků. Maximální teoretická propustnost těchto kanálů je 4 Gb/s, reálná propustnost závisí na délce transportovaných paketů a pohybuje se od 90 po 99 procent maximální hodnoty.

Nároky na zdroje FPGA jsou pro kartu rozhraní SFPRO 85% sliců a 72% blokových pamětí (tj. 7900 sliců a 64 blokových pamětí). Tyto relativně vysoká čísla jsou důsledkem využití všech prvků infrastruktury NetCOPE pro kartu rozhraní a naddimenzovanou velikostí bufferů pro vstupní a výstupní síťové bloky. V případě potřeby by se daly požadované zdroje razantně snížit. Na základní kartě COMBO6X využívá aplikace 39% sliců a 27% blokových pamětí (tj. 9200 sliců a 64 blokových pamětí). Většina těchto zdrojů je využita pro infrastrukturu NetCOPE.

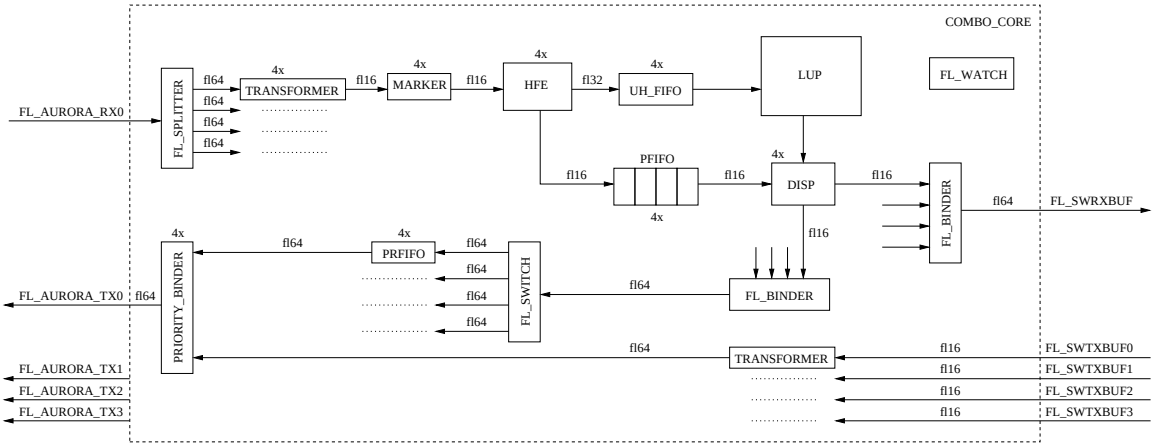
4.2 Hardwarový firewall

Komplexním síťovým zařízením plně využívajícím možností karet COMBO je hardwarový firewall s přeposíláním paketů. Klíčovými prvky tohoto zařízení jsou analýza a klasifikace vstupních paketů. Ostatní důležité prvky jsou pokryty v rámci platformy NetCOPE. Mimo samotnou filtraci paketů na rychlostech linky umožňuje navržené zařízení na základě výsledků klasifikace provádět hardwarové přeposílání paketů. Nasazení tohoto zařízení tedy není omezeno pouze na funkci firewallu, ale umožňuje realizovat i další úlohy jako například rozdělování vstupního toku do několika výstupních na základě L2, L3 i L4 pravidel, přeposílání podezřelých toků do honeypotu či replikaci síťových toků.

Architektura této aplikace pro kartu rozhraní je shodná s architekturou síťové karty (viz obr. 4.1). Jediným jejím úkolem je příjem a odesílání paketů. Architektura jádra aplikace pro základní kartu je znázorněna na obrázku 4.3. Při návrhu architektury se maximálně využily IP cores bloky vyvinuté v rámci platformy NetCOPE i IP cores bloky vyvinuté v rámci projektu Liberouter. Jediným novým prvkem architektury je menší komponenta *Dispatcher (DISP)* .

Pakety z karty rozhraní jsou po přijetí komponentou AURFC rozděleny do čtyř toků pomocí jednotky Splitter, čímž je zajištěna jejich rovnoměrná distribuce do čtyř procesorů pro analýzu vstupních paketů (HFE). Tyto procesory vytvářejí z dat potřebných pro klasifikaci *unifikovanou hlavičku* [21] a ukládají ji do příslušné fronty těchto hlaviček (UH FIFO). Data paketů jsou v nezměněné podobě ukládána do paketové fronty (PFIFO). Nad unifikovanou hlavičkou provádí klasifikaci vyhledávací procesor (LUP). Výsledkem klasifikace je záznam, v němž je uloženo, jak má být paket dále zpracován. Možnosti jsou zahození paketu, přeposlání paketu do softwaru nebo přímé přeposlání v hardwaru na některé výstupní rozhraní. Provedení tohoto zpracování je úkolem jednotky Dispatcher.

Tato jednotka je v aplikaci instancována čtyřikrát, jednou pro každý tok paketů z příslušného HFE rozhraní. Dispatcher přebírá výsledek klasifikace od vyhledávacího procesoru určený pro dané rozhraní, seřadí výsledky do správného pořadí (vyhledávací procesor nezaručuje pořadí na výstupu odpovídající pořadí na vstupu) a zajistí požadované zpracování. Při filtraci paketů se jedná o využití zahazovacího rozhraní paketové fronty, při přeposlání do softwaru o vyčtení dat z paketové fronty a zaslání na softwarový výstup a při hardwarovém přeposlání dat o vyčtení a přeposlání na hardwarový výstup. Při hardwarovém



Obrázek 4.3: Architektura hardwarového firewallu s přeposíláním paketů.

přeposlání je navíc do kontrolních dat přeposílaného rámce přidána informace o výstupním rozhraní, na které má být paket odeslán. Může se přitom jednat o libovolnou kombinaci výstupních rozhraní, například i přeposlání na všechny čtyři rozhraní zároveň. Seřazení výsledků z vyhledávacího procesoru je zajištěno na základě označení paketů jednotkou Marker, která přiděluje paketům v toku identifikační číslo. Vyhledávací procesor následně propaguje na svůj výstup spolu z výsledkem zpracování vstupní rozhraní, ze kterého byl paket přijat, a identifikační číslo paketu v tomto rozhraní. Na základě těchto údajů si dispatcher ukládá pouze výsledky určené pro jim spravovaný tok dat a provádí požadované seřazení.

Pakety všech čtyřech toků určené pro zpracování v softwaru jsou spojeny jednotkou Binder a přeposlány do softwarového bufferu. Podobně jsou spojeny i pakety všech čtyř toků určené pro přímé hardwarové přeposlání. Ty jsou po spojení zpracovány jednotkou Switch, která podle přidané kontrolní informace zajistí přeposlání na příslušné výstupní rozhraní (kterých může být i více najednou). Výsledné čtyři toky dat určené pro čtyři výstupní síťové rozhraní jsou poté uloženy do front s automatickým zahazováním paketů (PRFIFO) a spojeny s odpovídajícími toky dat paketů zaslaných na výstupní síťová rozhraní ze softwaru. Při spojování těchto toků je využit prioritní binder a zmíněná fronta, která při zaplnění zahazuje pakety ze vstupu. Důvodem jsou gigabitové propustnosti obou cest, přičemž odesílací síťové rozhraní má také pouze kapacitu 1 Gb/s. Pokud by tedy došlo k výstupním blokování, budou hardwarově přeposílané pakety zahozeny.

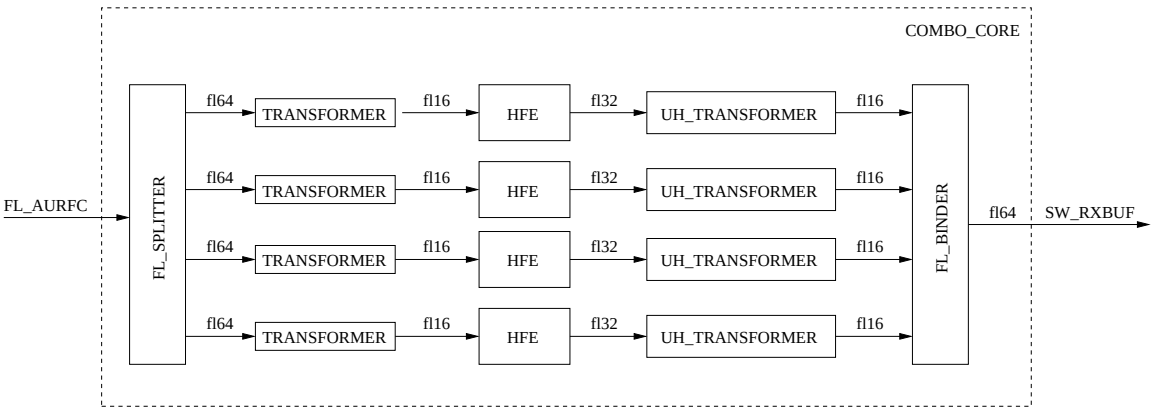
Pracovní frekvence aplikace je opět 100 MHz. Nároky na zdroje FPGA jsou 18000 sliců a 143 blokových pamětí (tj. 78% sliců a 61% blokových pamětí na kartě COMBO6X). Výkonnost jádra aplikace je dána propustností HFE procesorů a LUP procesoru. Zatímco vyhledávací procesor nabízí propustnost až 10 Gb/s v závislosti na složitosti filtračních pravidel [23], procesory pro analýzu síťového provozu disponují gigabitovou propustností [1]. Použitím čtyř těchto procesorů je zajištěna možnost zpracování paketů ze všech čtyřech vstupních síťových rozhraní na plné rychlosti. Jediné omezení propustnosti aplikace je stejně jako v případě síťové karty dáno architekturou rodiny karet COMBO. Samotná aplikace je dobře škálovatelná a při dostupnosti desetigigabitové platformy snadno upravitelná na tuto vyšší propustnost. Vyhledávací procesor nabízí dostatečnou výkonnost a použil by se ve

stejné verzi, procesory pro analýzu paketů by se instancovaly v takovém počtu, aby jejich celková propustnost také tvořila 10 Gb/s. Vzhledem ke generičnosti komponent Binder, Splitter i dalších by se jednalo o velmi rychlou úpravu, která by nebyla časově náročná.

4.3 Exportér paketových hlaviček

Posledním zařízením navrženým v rámci diplomové práce je exportér paketových hlaviček. Cílem tohoto pasivního zařízení je předzpracovat vstupní tok paketů, extrahovat z nich důležité informace do pevné šablony a ty exportovat do softwaru pro další zpracování. V softwaru nad exportérem paketových hlaviček může běžet například monitorovací sonda typu NetFlow[15] nebo IPFIX[16]. Hardwarové předzpracování zajistí zejména redukci datového toku přenášeného po systémové sběrnici a dále přesun analýzy paketů a vytváření šablon ze softwaru do hardwaru, čímž se získá více procesorového času pro vlastní monitorovací aplikaci realizovanou v programovém vybavení počítače.

Architektura exportéru pro kartu rozhraní je podobná s architekturou síťové karty (viz obr. 4.1). Jediným rozdílem je použití pouze vstupních síťových rozhraní, výstupní rozhraní nejsou v aplikaci použita. Architektura aplikace pro základní kartu je znázorněna na obrázku 4.4. Při návrhu aplikace se opět maximálně využilo IP cores bloků vyvinutých v rámci platformy NetCOPE i IP cores bloků vyvinutých v rámci projektu Liberouter a jedinou novou komponentou aplikace je komponenta pro transformaci a odvysílání unifikovaných hlaviček do softwaru (uh transformer).



Obrázek 4.4: Architektura exportéru paketových hlaviček.

Tok dat je podobný jako v předchozí aplikaci. Pakety z karty rozhraní jsou přijaty komponentou AURFC a rovnoměrně distribuovány do čtyř procesorů pro analýzu vstupních paketů. Tyto procesory extrahují položky hlaviček paketů, vytvářejí *unifikované hlavičku* a odesílají ji do fronty těchto hlaviček realizované v UH transformeru. Konkrétní položky, které mají být exportovány, a strukturu šablonu je možné v závislosti na potřebách aplikace měnit. Při použití generického nanoprocesoru HFE nahráním nového programu pro tento procesor, při použití analyzátoru realizovaného v jazyce *Handel-C* (HFE C) novým překladem firmwaru. Typickými exportovanými položkami jsou zdrojová a cílová adresa, zdrojový a cílový port, protokol, TCP flagy, časové razítko atp.

UH transformer pracuje nad dvěmi blokovými pamětmi, přičemž střídavě do jedné ukládá hlavičku právě zpracovávanou HFE procesorem a z druhé odesílá přeuspořádanou výslednou hlavičku na výstupní rozhraní. Na vstupním rozhraní jednotka přijímá 32-bitové slovo obsahující adresu a hodnotu položky, na výstupním rozhraní už je pouze seřazená posloupnost 16-bitových položek v podobě paketu FrameLink protokolu. Data ze všech čtyř takto vytvořených rozhraní jsou následně spojená komponentou Binder a odeslána do softwarového bufferu, odkud jsou přenesena DMA přenosem do paměti RAM hostujícího počítače a předána aplikaci.

Nároky na zdroje FPGA jsou pro kartu rozhraní SFPRO redukovány zhruba na poloviční hodnoty proti síťové kartě. Důvodem je použití pouze vstupních síťových bloků a nevyužití výstupní cesty. Aplikace zabírá 45% sliců a 42% blokových pamětí (tj. 4200 sliců a 38 blokových pamětí). Na základní kartě COMBO6X využívá aplikace 36% sliců a 21% blokových pamětí (tj. 8600 sliců a 48 blokových pamětí). Ušetření proti síťové kartě je opět dáno nepoužitím výstupní cesty.

4.4 Další zařízení

Navržená platforma je vhodná k realizaci libovolných síťových zařízení, u kterých je možné uplatnit hardwarovou akceleraci v programovatelném hradlovém poli. Mimo představené aplikace se může jednat o systémy pro detekci či prevenci průniku (IDS/IPS), různé monitorovací sondy, či výkonné paketové generátory. Například v rámci projektu Liberouter se předpokládá nasazení platformy pro implementaci monitorovací sondy *FlowMon* a implementaci systému pro detekci průniků *Traffic Scanner*.

Cílové použití platformy není omezeno jen na síťové aplikace a je možné jej využít i pro jiné aplikace. Těmto aplikacím může platforma NetCOPE nabídnout zejména rychlý přenos dat mezi adaptérem a pamětí RAM hostujícího počítače, abstrakci od konkrétně použitých hardwarových prostředků a sadu generických komponent pro práci s FrameLink protokolem. Příklady oborů, pro něž by mohla být hardwarová akcelerace v FPGA velmi zajímavá, jsou kryptografie, zpracování obrazu nebo bioinformatika.

Kapitola 5

Závěr

Na základě znalostí rodiny karet COMBO a analýzy síťových zařízení byla navržena obecná generická platforma pro rychlý vývoj síťových aplikací nad technologií FPGA. Platforma nabízí jednotný komunikační protokol, sadu nástrojů pro snadnou manipulaci s daty ve formátu tohoto protokolu, rychlé rozhraní do softwaru a sadu funkčních jednotek ve formě IP cores umožňujících rychlý vývoj aplikací skládáním dostupných komponent.

Návrh byl implementován v jazyce VHDL a otestován při implementaci základního stavebního bloku sítí – síťové karty. Po úspěšné evaluaci byla platforma použita pro rychlý vývoj komplexnějších zařízení – hardwarového firewallu a exportéru paketových hlaviček. Funkcionalita těchto aplikací byla ověřena v simulacích a otestována v hardwaru na kartách COMBO6X a SFPRO. Nasazení navržené platformy při vývoji zmíněných aplikací zkrátilo čas potřebný k jejich návrhu, implementaci i otestování.

Vzhledem k úspěšnému nasazení platformy NetCOPE v aplikacích popsaných v kapitole 4 se předpokládá její brzké využití ve všech aktivně vyvíjených aplikacích projektu Liberouter. Použitím navržené platformy pro stávající zařízení se výrazně zvýší jejich propustnost do paměti RAM hostujícího počítače. Pro nové aplikace bude hlavním přínosem dostupnost vývojového prostředí a redukce implementace na vývoj vlastního aplikačního jádra, které bude nezávislé na konkrétních hardwarových prostředcích. Přestože byla platforma primárně cílena pro vývoj síťových zařízení, její použití se nabízí i v jiných oblastech, které pracující s technologiemi FPGA a využijí abstrakci od hardwarové vrstvy či rychlý přenos dat mezi hardwarem a softwarem. Příklady těchto oblastí mohou být kryptografie, zpracování obrazu nebo bioinformatika.

Z hlediska dalšího vývoje architektury NetCOPE bude hrát klíčovou roli vývoj programovatelných hradlových polí. Infrastruktura bude upravována pro nové hardwarové prostředky a nové typy FPGA. Bude zvyšována pracovní frekvence IP core bloků, ale rozhraní k aplikaci zůstane díky své generičnosti stejné. Také většina komponent by měla vzhledem ke své generické implementaci a volitelné datové šířce sloužit nejen při současně uvažovaných rychlostech 1-10 Gb/s ale i v budoucnu při rychlostech vyšších.

Literatura

[1] WWW stránka projektu *Liberouter*, http://www.liberouter.org (květen 2007).

[2] WWW stránka projektu *6NET*, http://www.6net.org/ (květen 2007).

[3] WWW stránka projektu *SCAMPI*, http://www.ist-scampi.org (květen 2007).

[4] WWW stránka projektu *GÉANT2*, http://www.geant2.net (květen 2007).

[5] Novotný, J., Fučík, O., Kokotek, R.: *Schematics and PCB of COMBO6*. Technical report 14/2002, CESNET, Praha, 2002.

[6] IEEE, 3 Park Avenue, New York, NY 10016-5997, USA. *Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*. IEEE, Std 802.3 edition, 2005, ISBN 0-7381-4741-9.

[7] IEEE, 3 Park Avenue, New York, NY 10016-5997, USA. *Virtual Bridged Local Area Networks*. IEEE Std 802.1Q, 2005, ISBN 0-7381-4877-6.

[8] Postel, J.: *Internet Protocol* , RFC 791, 1981. Dokument dostupný na http://www.ietf.org/rfc/rfc0791.txt (květen 2007).

[9] Deering, S., Hinden, R.: *Internet Protocol Version 6 (IPv6) Specification*, RFC 2460, 1998. Dokument dostupný na http://www.ietf.org/rfc/rfc2460.txt (květen 2007).

[10] Pužmanová, R.: *Moderní komunikační sítě od A do Z*, 2. aktualizované vydání, Praha, Computer Press, 2006, ISBN 80-251-1278-0.

[11] Postel, J.: *Transmission Control Protocol*, RFC 793, 1981. Dokument dostupný na http://www.ietf.org/rfc/rfc0793.txt (květen 2007).

[12] Postel, J.: *User datagram protocol*, RFC 768, 1980. Dokument dostupný na http://www.ietf.org/rfc/rfc0768.txt (květen 2007).

[13] Sourcefire: *Snort*: The Open Source Network Intrusion Detection System, http://www.snort.org (květen 2007).

[14] Kobierský, P., Kořenek, J., Hank, A.: *Traffic Scanner*. Technical report 33/2006, CESNET, Praha, 2006.

[15] Claise, B.: *Cisco Systems NetFlow Services Export Version 9*, RFC 3954, IETF, 2004.

[16] Quittek, J.: *Requirements for IP Flow Information Export*, RFC 3917, IETF, 2004.

[17] Čeleda, P., Kováčik, M., Koníř, T., Krmíček, V., Špringl, P., Žádník, M.: *FlowMon Probe*. Technical report 31/2006, CESNET, Praha, 2006.

[18] Xilinx, Inc. 2100 Logic Drive, San Jose, CA 95124-3400, USA. *Virtex-II Platform FPGAs: Complete Data Sheet*, březen 2005. Dokument dostupný na URL http://direct.xilinx.com/bvdocs/publications/ds031.pdf (květen 2007).

[19] Xilinx, Inc. 2100 Logic Drive, San Jose, CA 95124-3400, USA. *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet*, březen 2007. Dokument dostupný na URL http://direct.xilinx.com/bvdocs/publications/ds083.pdf (květen 2007).

[20] Martínek, T., Tobola, J.: *Interconnection System for the NetCOPE Platform*. Technical Report 34/2006, CESNET, Praha, 2006.

[21] Mikušek, P.: *Návrh a implementace procesní jednotky pro analýzu vstupních paketů*, bakalářská práce, Brno, FIT VUT v Brně, 2005.

[22] Pazdera, J.: *Procesní jednotka pro analýzu a editaci síťového provozu*, diplomová práce, Brno, FIT VUT v Brně, 2007.

[23] Málek, T.: *Návrh a implementace procesoru pro klasifikaci IPv4/IPv6 paketu*, bakalářská práce, Brno, FIT VUT v Brně, 2006.

[24] Tobola, J.: *Vyhledávání řetězců v payloadu paketu s využitím TCAM*, bakalářská práce, Brno, FIT VUT v Brně, 2005.

[25] Košek, M.: *Stavové zpracování TCP/IP toků*, bakalářská práce, Brno, FIT VUT v Brně, 2007.

[26] Xilinx, Inc. 2100 Logic Drive, San Jose, CA 95124-3400, USA. *Local-Link Interface Specification*, říjen 2006. Dokument dostupný na URL http://direct.xilinx.com/bvdocs/publications/sp006.pdf (květen 2007).

[27] Xilinx, Inc. 2100 Logic Drive, San Jose, CA 95124-3400, USA. *Virtex-5 Family Overview LX and LXT Platforms*, říjen 2006. Dokument dostupný na URL http://direct.xilinx.com/bvdocs/publications/ds100.pdf (květen 2007).