VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Jakub Cabal

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

ZPRACOVÁNÍ SÍŤOVÉHO PROVOZU NA VELMI VYSOKÝCH RYCHLOSTECH
NETWORK TRAFFIC PROCESSING AT VERY HIGH SPEED

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Jakub Cabal

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Lukáš Fujcik, Ph.D.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**
Ústav mikroelektroniky

Student: Bc. Jakub Cabal ***ID:*** 154688
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Zpracování síťového provozu na velmi vysokých rychlostech

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s technologií FPGA a standardy Ethernet 100 a 400 Gb/s. Identifikujte problémy při zpracování a generování hlaviček paketů na rychlostech 100 Gb/s a vyšších. Navrhněte obvodovou realizaci parseru a generátoru paketů vhodnou pro technologii FPGA. V rámci navazující diplomové práci implementujte obvody ve zvoleném jazyce a vyhnoťte výsledky. Zvažte možnost automatického generování těchto obvodů z vysokoúrovňového popisu v jazyce P4.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 25.5.2017

Vedoucí práce: doc. Ing. Lukáš Fujcik, Ph.D.
Konzultant:

doc. Ing. Lukáš Fujcik, Ph.D.
předseda oborové rady

ABSTRAKT

Různá síťová zařízení vyžadují zpracování síťového provozu. Pro zpracování síťového provozu je většinou nutné extrahovat hlavičky jednotlivých protokolů obsažených v přijatých ethernetových rámcích. Zpracované hlavičky pak lze upravit a znovu poskládat do ethernetových rámců a odeslat zpět do sítě. Tato práce se zabývá návrhem a implementací obvodu pro analýzu a extrakci hlaviček protokolů a obvodu pro skládání ethernetových rámců z hlaviček protokolů. Obvody budou navrženy pro přenosovou rychlost až 400 Gb/s a budou implementovány prostřednictvím technologie FPGA.

KLÍČOVÁ SLOVA

FPGA, VHDL, Ethernet, Analýza, Zpracování, Internetový provoz

ABSTRACT

Different network devices require processing of the network traffic. To process the network traffic, it is necessary to parse headers of particular protocols packed in incoming ethernet frames. The processed headers can be repackaged to ethernet frames and sent back to the network. The goal of this thesis is to design and implement a circuit for analysis and parsing of ethernet frames, together with circuit for deparsing ethernet frames. The circuits are designed for throughputs of up to 400 Gb/s. The circuits are implemented for the FPGA technology.

KEYWORDS

FPGA, VHDL, Ethernet, Analysis, Processing, Internet traffic

CABAL, Jakub. *Zpracování síťového provozu na velmi vysokých rychlostech*. Brno, Rok, 53 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav mikroelektroniky. Vedoucí práce: Doc. Ing. Lukaš Fujcik, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Zpracování síťového provozu na velmi vysokých rychlostech“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Doc. Ing. Lukáši Fujcikovi, Ph.D. za vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych chtěl poděkovat odbornému konzultantovi Ing. Lukáši Kekelymu za odborné vedení, konzultace a užitečné rady k práci. Poděkování si také zaslouží celý vývojový tým Liberouter a zejména pak Ing. Pavel Benáček, Ph.D. Speciální poděkování patří i mé rodině a snoubence Kláře za jejich trpělivost a podporu.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

ÚVOD

Internetový síťový provoz neustále zrychluje, tomu odpovídá i vydávání nových a rychlejších standardů Ethernet. V současnosti je poslední standard Ethernet určen pro přenosouvou rychlost 100 Gb/s, ale již se aktivně připravuje nový standard pro přenosovou rychlost 400 Gb/s.[1][2] Různá síťová zařízení potřebují pro svou činnost znát políčka z hlaviček jednotlivých protokolů, které jsou zapouzdřeny v ethernetových rámcích. Získávání těchto políček lze realizovat pomocí běžných procesorů, avšak pokud je nutné dodržet vysoké přenosové rychlosti, je vhodné použití speciálně navržených obvodů. Tyto specializované obvody dokáží extrahovat hlavičky protokolů mnohem efektivněji a především rychleji.[3][4]

Extrahované hlavičky protokolů lze dále libovolně zpracovat a analyzovat. V některých případech můžeme chtít hlavičky jednotlivých protokolů upravit nebo přidat hlavičku protokolu, který v ethernetovém rámci původně nebyl. V takovém případě potřebujeme poskládat z upravených a přidaných hlaviček nový ethernetový rámec a ten například odeslat zpět do sítě. Takovou úlohu je opět možné realizovat pomocí běžných procesorů, ale opět je vhodné z důvodu vyšší rychlosti využít specializovaných obvodů.[5]

Tato práce se zabývá návrhem specializovaného obvodu pro extrakci hlaviček z ethernetových rámců a návrhem obvodu pro skládání ethernetového rámce z hlaviček jednotlivých protokolů. Oba navržené specializované obvody by měly být schopny dosáhnout přenosové rychlosti až 400 Gb/s. Dále se tato diplomová práce zabývá implementací těchto obvodů. Pro implementaci obvodů byla vybrána technologie FPGA, která umožní rychlejší a levnější implementaci než například technologie ASIC.[6]

Kapitola 1 je teoretický rozbor popisující model ISO/OSI, Ethernet, vybrané internetové protokoly, technologii FPGA a proprietární sběrnici MFB. Kapitola 2 rozebírá problematiku, návrh a implementaci obvodu pro analýzu a extrakci hlaviček internetových protokolů zabalených v ethernetovém rámci. Kapitola 3 popisuje problematiku, návrh a implementaci obvodu pro skládání ethernetových rámců z hlaviček jednotlivých internetových protokolů. Kapitola 4 popisuje vyhodnocení dosažených výsledků implementovaných obvodů. V kapitole 5 je rozebírána možnost využití vysokoúrovňového jazyka P4 k automatizovanému generování implementovaných obvodů. Závěr shrnuje přínos a dosažené výsledky práce.

1 TEORETICKÝ ROZBOR

Tato práce má za cíl navrhnout způsob zpracování internetového provozu na vysokých rychlostech. Ke splnění zadaného cíle je nutné znát a použít vhodné technologie. K přenosu internetového provozu se dnes nejčastěji používá technologie Ethernet v různých verzích, které se liší především rychlostí přenosu. V této kapitole je stručně popsán referenční model ISO/OSI, standard Ethernet, a zmíněn je také Ethernet pro rychlosti 100 Gbit/s a 400 Gbit/s. Dále jsou popsány vybrané a často používané internetové protokoly, které budou při zpracovávání internetového provozu analyzovány.

Samotné zpracování internetového provozu při takto vysokých rychlostech již nelze provádět pomocí běžných procesorů, ale je nutné navrhnout speciální digitální obvod, který zadaný úkol zvládne. Pro samotnou implementaci navrhovaného obvodu bude použita technologie FPGA (Field Programmable Gate Array), jejíž popis je zde také uveden. V poslední části teoretického rozboru je popsána proprietární sběrnice MFB, použitelná pro přenos ethernetových rámců.

1.1 Referenční model ISO/OSI

Referenční model ISO/OSI vytvořila organizace ISO při snaze o standardizaci počítačových sítí nazvaných OSI. V roce 1984 byl model přijat jako mezinárodní norma ISO 7498. Referenční model ISO/OSI popisuje sedmivrstvou architekturu počítačové sítě. Každá z vrstev má jasně definované funkce pro komunikaci. V jednotlivých vrstvách se komunikace řídí protokoly.[7] V následujícím textu je postupně a stručně popsáno všech sedm vrstev modelu ISO/OSI od nejnižší po nevyšší.

Fyzická vrstva je nejnižší vrstvou ISO/OSI modelu. Fyzická vrstva určuje elektrické a fyzikální vlastnosti zapojených zařízení. Také určuje způsob přenosu dat po přenosovém mediu.

Spojová vrstva (občas také nazývána jako linková vrstva) má za úkol umožnit spojení mezi dvěma systémy. Data z fyzické vrstvy transformuje do logických celků označovaných jako rámce. Spojová vrstva nastavuje rámcům fyzické adresy, zajišťuje synchronizaci pro fyzickou vrstvu a signalizuje případné chyby.

Síťová vrstva zajišťuje směrování dat v síti a síťové adresování. Pomocí obsažených funkcí dokáže překlenout rozdílné přenosové technologie.

Transportní vrstva má za úkol zajistit přenos dat mezi koncovými uzly. Účelem této vrstvy je také zajistit kvalitu přenosu, kterou vyžaduje vyšší vrstva. Protokoly pro transportní vrstvu lze rozdělit na spojované a nespojované.

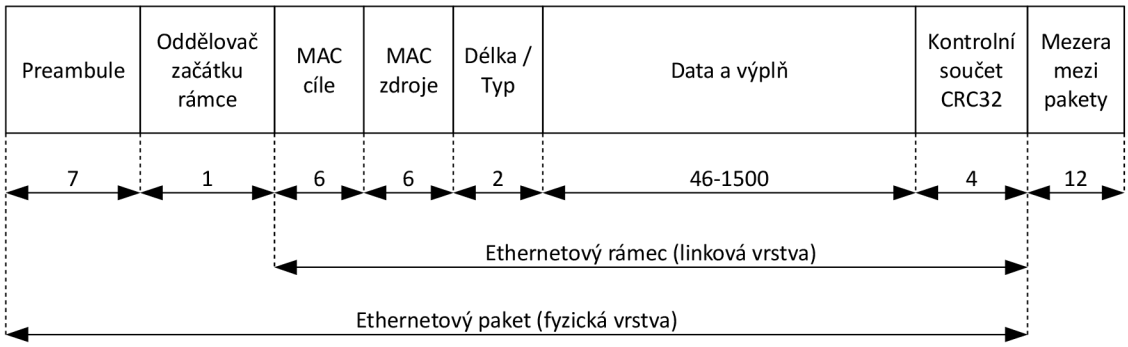
Relační vrstva má za úkol řídit dialog a výměnu dat mezi spolupracujícími relačními vrstvami komunikujících systémů. Umožňuje vytvoření, ukončení, synchronizaci a nebo obnovu spojení relačních vrstev.

Prezentační vrstva má za úkol transformovat data do podoby, která bude srozumitelná pro aplikace. Umožňuje řešit převody kódů, šifrování a jiné transformace dat. Prezentační vrstva neřeší význam dat, ale jen jejich strukturu.

Aplikační vrstva má za úkol poskytnout různým aplikacím přístup ke komunikaci, a tím umožnit jejich spolupráci. Z protokolů a služeb, které patří do této vrstvy, lze například uvést: FTP, DHCP, Telnet nebo SSH.

1.2 Ethernet

Technologie Ethernet dnes tvoří nejpoužívanější základ pro přenos dat v počítačových sítích, které používají především kabely s kroucenou dvoulinkou nebo optické kabely. Z velké části je Ethernet standardizován jako IEEE 802.3. V referenčním modelu ISO/OSI zajišťuje Ethernet fyzickou a spojovou vrstvu. Ethernet tak umožňuje přenášet data protokolů vyšších vrstev, například síťové protokoly IPv4 a IPv6.[8]



Obr. 1.1: Ethernetový rámec uvnitř ethernetového paketu.

Na fyzické vrstvě je základní datová jednotka ethernetový paket, který obsahuje preambuli, oddělovač začátku rámce (často označován zkratkou SFD) a ethernetový rámec. Jeho struktura je znázorněna na obrázku 1.1 s délkami jednotlivých políček v bytech. Preambule slouží k synchronizaci přijímače a vysílače a je tvořena posloupností 7 bytů. Oddělovač začátku rámce odděluje preambuli od ethernetového rámce a má hodnotu 10101011 (od LSB po MSB). Na spojové vrstvě je základní datovou jednotkou samotný ethernetový rámec.[8] V následujícím textu jsou blíže popsána jednotlivá políčka ethernetového rámce.

Cílova MAC adresa (6 bytů) obsahuje adresu cílového zařízení.

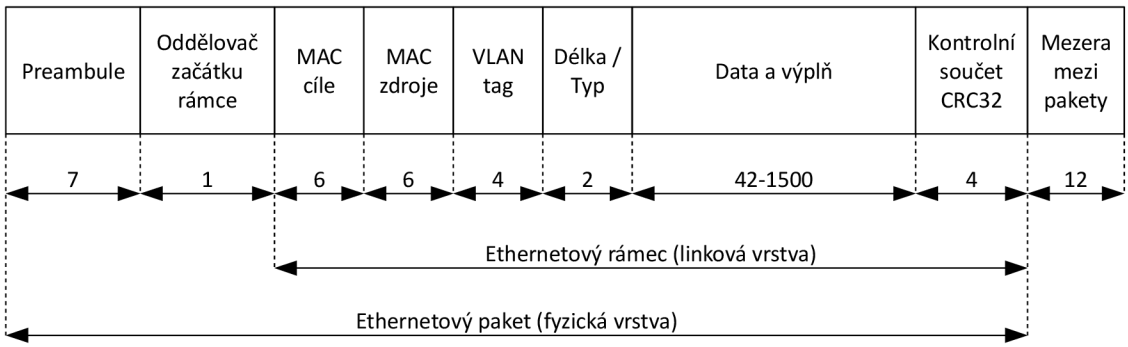
Zdrojová MAC adresa (6 bytů) obsahuje adresu zdrojového zařízení.

Délka/typ rámce (2 byty) může nabývat dvou významů, pokud je jeho hodnota menší nebo rovna 1500, udává délku následujícího políčka, ale pokud je jeho hodnota větší nebo rovna 1536, udává typ zapouzdřeného protokolu.

Data a výplň (46 až 1500 bytů) obsahuje přenášená data a výplň pro zarovnání, také může obsahovat zabalený protokol vyšší vrstvy.

Kontrolní součet CRC32 (4 byty) obsahuje kontrolní součet vypočtený ze všech výše jmenovaných políček.

Ethernetový rámec lze rozšířit vložením jednoho nebo více VLAN tagů. Tyto tagy, označované také jako IEEE 802.1Q, jsou vkládány mezi políčko *zdrojová MAC adresa* a políčko *délka/typ rámce*. Rozšířený ethernetový rámec s jedním VLAN tagem je znázorněn na obrázku 1.2. Jeden VLAN tag má délku 4 byty a je složen ze dvou částí TPID a TCI. TPID část má délku 2 byty a obsahuje identifikaci VLAN tagu. TCI část má délku také 2 byty a obsahuje kontrolní informace VLAN tagu. Jelikož je potřebné zajistit dodržení minimální délky ethernetového rámce, je nutné zmenšit minimální délku políčka *data a výplň* o délku vložených VLAN tagů.[9]



Obr. 1.2: Rozšířený ethernetový rámec s jedním VLAN tagem.

Standard Ethernet definuje strukturu rámce a základní vlastnosti popsané výše. Ethernet umožňuje komunikaci při různých standardizovaných rychlostech. Z nich jsou pro tuto práci nejvíce zajímavé přenosové rychlost 100 Gb/s a 400 Gb/s. Ethernet pro přenosovou rychlost 100 Gb/s byl poprvé schválen v roce 2010 v rámci standardu IEEE 802.3ba, který rovněž definoval Ethernet pro přenosovou rychlost 40 Gb/s.[1] Jelikož požadavky na přenosovou rychlost neustále rostou, tak již v roce 2013 vytvořila organizace IEEE pracovní skupinu připravující standard Ethernet pro přenosovou rychlost 400 Gb/s. Dle současného plánu organizace IEEE by měl být Ethernet pro přenosovou rychlost 400 Gb/s schválen na konci roku 2017 v rámci standardu IEEE 802.3bs.[2]

1.3 Vybrané internetové protokoly

V této části jsou stručne popsány vybrané internetové protokoly. Popis se zaměřuje především na obsah hlavičky vybraných protokolů, které je nutné znát zejména pro hlubší analyzování ethernetových rámců.

1.3.1 Multiprotocol Label Switching

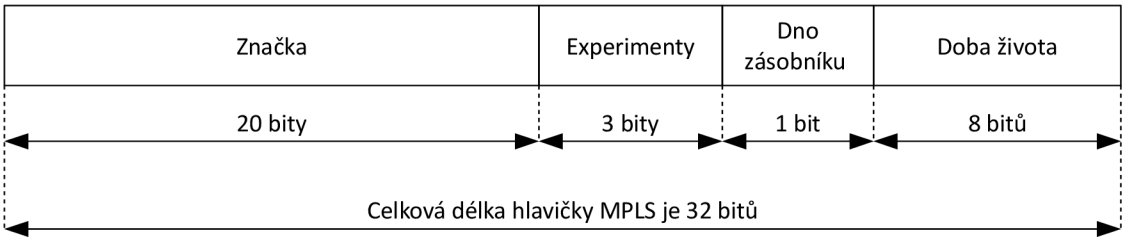
Protokol MPLS (Multiprotocol Label Switching) slouží ke směrování paketů ve vysokorychlostních sítích. Pro směrování nepoužívá dlouhé směrovací adresy, ale krátké značky. Tyto značky pak určijí virtuální spoje, na které se má paket odeslat. V jednom paketu může být vložena jedna nebo více hlaviček protokolu MPLS, tím vzniká zásobník protokolů MPLS. Obrázek 1.3 zobrazuje strukturu hlavičky protokolu MPLS včetně délky jednotlivých políček. Hlavička protokolu MPLS je jednoduchá a obsahuje pouze čtyři políčka. Význam jednotlivých políček je popsán v následujícím textu.[10]

Značka (20 bitů) slouží k určení cílového virtuálního spoje.

Experimenty (3 bity) experimentální pole, používá se například pro řízení kvality přenosu nebo pro určování priority.

Dno zásobníku (1 bit) bit identifikující nejnižší protokol MPLS v zásobníku.

Doba života (8 bitů) každý směrovač tuto hodnotu dekrementuje. Pokud se hodnota dostane na 0, paket se zahodí.

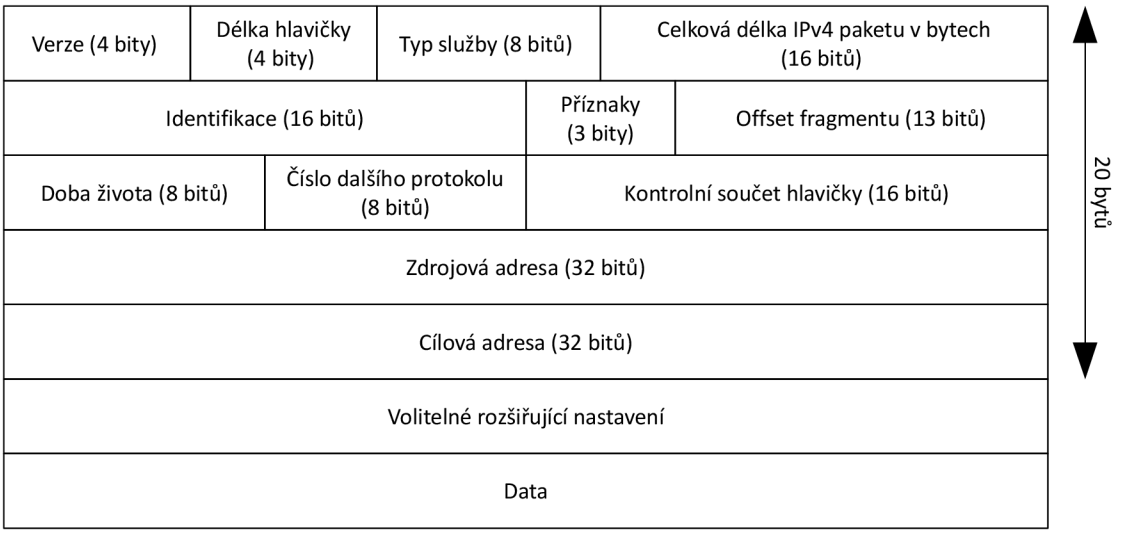


Obr. 1.3: Popis struktury hlavičky protokolu MPLS.

1.3.2 Internet Protocol version 4

Protokol IPv4 (Internet Protocol version 4) byl popsán v roce 1981 v dokumentu RFC 791 a stále tvoří základ pro komunikaci v síti internet. IPv4 je protokol síťové vrstvy. Tento protokol negarantuje doručení ani zachování pořadí přenášených dat. IPv4 protokol také neřeší integritu přenášených dat. Protokol IPv4 poskytuje až 2^{32} adres používaných pro adresaci síťových zařízení. V současnosti jsou tyto adresy již

prakticky rozebrány. Hlavička protokolu IPv4 nemá konstantní délku, ale informace o její délce je obsažena v hlavičce.[11]



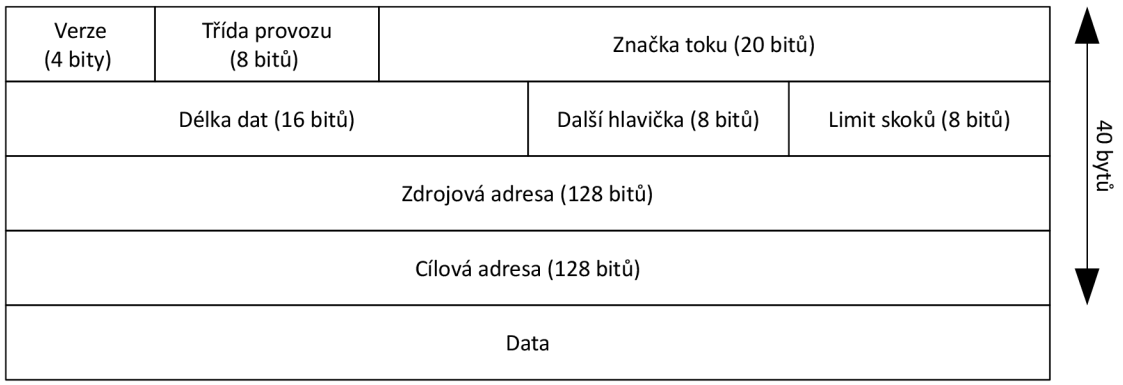
Obr. 1.4: Popis struktury paketu protokolu IPv4.

Obrázek 1.4 popisuje strukturu paketu protokolu IPv4 a zejména jeho hlavičky včetně délky jednotlivých políček. Z obrázku je patrné, že hlavičku tvoří povinná část s konstantní celkovou délkou 20 bytů a volitelná rozšiřující nastavení s variabilní délkou. Následující text popisuje význam vybraných políček, která jsou důležitá z pohledu analýzy internetového provozu.[11]

Verze (4 bity) obsahuje číslo verze protokolu. Hodnota políčka musí být 4.
Délka hlavičky (4 bity) udává delku hlavičky po 32 bitových slovech.
Typ služby (8 bitů) hodnota udává prioritu paketu.
Celková délka paketu (16 bitů) délka paketu včetně hlavičky v bytech.
Délka života (8 bitů) každý směrovač tuto hodnotu dekrementuje. Pokud se hodnota dostane na 0, paket se zahodí.
Číslo dalšího protokolu (8 bitů) definuje protokol zabalený v datech.
Zdrojová adresa (32 bitů) obsahuje zdrojovou IP adresu.
Cílová adresa (32 bitů) obsahuje cílovou IP adresu.

1.3.3 Internet Protocol version 6

Protokol IPv6 (Internet Protocol version 6) je nástupce protokolu IPv4. Poprvé byl definován v roce 1998 v dokumentu RFC 2460. Protokol IPv6 přináší především výrazně větší adresní prostor (celkem 2^{128} adres), který umožnuje lepší hierarchické uspořádání sítě a tím i snadnější směrování.[12]



Obr. 1.5: Popis struktury paketu protokolu IPv6.

Obrázek 1.5 zobrazuje popis struktury paketu protokolu IPv6 včetně podrobně popsané struktury jeho hlavičky. Oproti staršímu protokolu IPv4 má tento novější protokol hlavičku mírně zjednodušenou. Význam jednotlivých políček je popsán v následujícím textu.[12]

Verze (4 bity) obsahuje číslo verze protokolu. Hodnota políčka musí být 6.
Třída provozu (8 bitů) udává prioritu paketu.
Značka toku (20 bitů) specifikuje speciální zacházení s paketem ve směrovačích.
Délka dat (16 bitů) udává délku celého paketu bez hlavičky v bytech.
Další hlavička (8 bitů) definuje hlavičku protokolu zabaleného v datech.
Limit skoků (8 bitů) každý směrovač tuto hodnotu dekrementuje. Pokud se hodnota dostane na 0, paket se zahodí.
Zdrojová adresa (128 bitů) obsahuje zdrojovou IP adresu.
Cílová adresa (128 bitů) obsahuje cílovou IP adresu.

1.3.4 Rozšiřující hlavičky IPv6

Rozšiřující hlavičky IPv6 (IPv6 EXT) jsou součástí protokolu IPv6. Tyto hlavičky jsou volitelné a umožnují přenášet doplňující informace. Každá rozšiřující hlavička musí následovat hlavičku IPv6 nebo předchozí rozšiřující hlavičku IPv6 a musí mít délku o násobku 8 bytů. Existuje více typů rozšiřujících hlaviček a nové mohou být přidány v budoucnu. První byte u všech rozšiřujících hlaviček IPv6 obsahuje políčko, které definuje typ následujícího protokolu. Některé rozšiřující hlavičky mohou mít konstantní délku, ale většina má delku proměnnou. Rozšiřující hlavičky s proměnnou délkou obsahují na druhém bytu políčko udávající délku rozšiřující hlavičky po osmi bytových blocích bez prvního bloku.[12]

1.3.5 Transmission Control Protocol

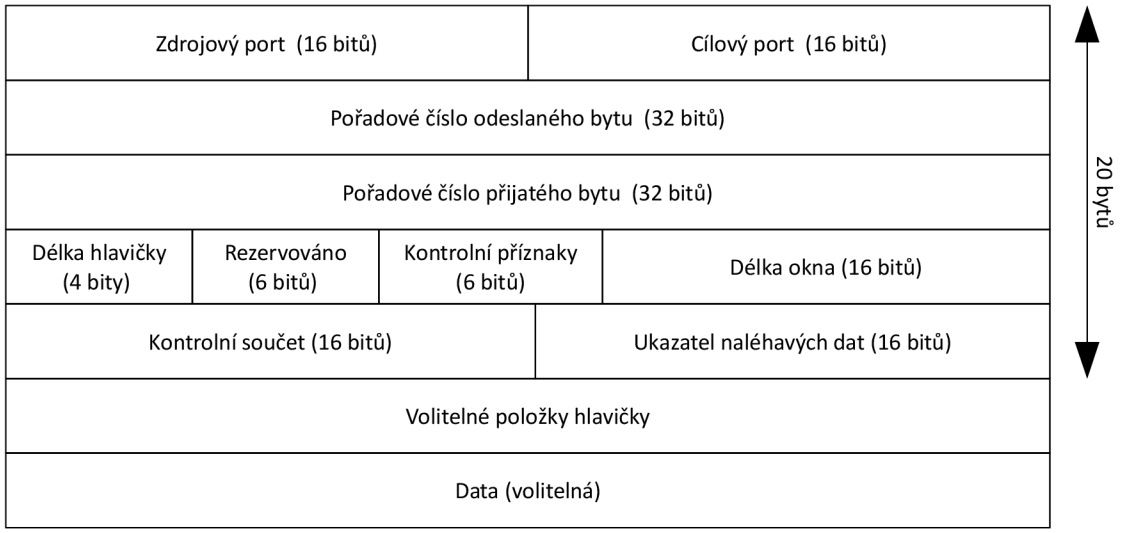
Protokol TCP (Transmission Control Protocol) patří do transportní vrstvy a je velmi používaný v síti internet. Protokol TCP je podrobně popsán v dokumentu RFC 793. Protokol garantuje doručení dat a také správnost jejich pořadí. Protokol TCP často využívají aplikační protokoly, jako například SSH nebo email. Na obrázku 1.6 je znázorněn podrobný popis paketu protokolu TCP. Obrázek popisuje jednotlivá políčka hlavičky protokolu a udává jejich délky. Následující text popisuje význam vybraných políček, která jsou důležitá z pohledu analýzy internetového provozu.[13]

Zdrojový port (16 bitů) obsahuje číslo portu zdrojové aplikace. Políčko není povinné. Pokud není využito, musí obsahovat pouze nuly.

Cílový port (16 bitů) obsahuje číslo portu cílové aplikace.

Délka hlavičky (4 bity) udává délku hlavičky po 32 bitových slovech. Minimální délka hlavičky je 20 bytů.

Kontrolní příznaky (6 bitů) tyto příznaky určují speciální vlastnosti paketu.



Obr. 1.6: Popis struktury paketu protokolu TCP.

1.3.6 User Datagram Protocol

Protokol UDP (User Datagram Protocol) patří také do transportní vrstvy, ale narozdíl od protokolu TCP negarantuje doručení a správné pořadí dat. Protokol UDP je popsán v dokumentu RFC 768. Jde o jednoduchý protokol s malou režií, který využívají aplikační vrstvy jako například DHCP nebo DNS. Obrázek 1.7 zobrazuje

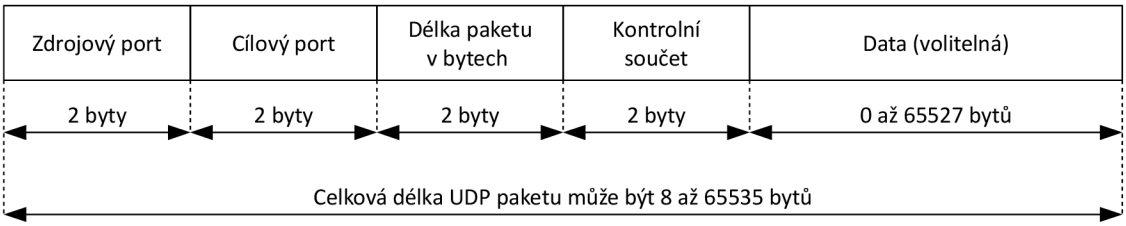
strukturu paketu protokolu UDP a jeho hlavičky včetně délky jednotlivých políček. Hlavička protokolu UDP je velmi jednoduchá a obsahuje pouze čtyři políčka. Význam jednotlivých políček je popsán v následujícím textu.[14]

Zdrojový port (16 bitů) obsahuje číslo portu zdrojové aplikace. Políčko není povinné. Pokud není využito, musí obsahovat pouze nuly.

Cílový port (16 bitů) obsahuje číslo portu cílové aplikace.

Délka paketu (16 bitů) udává delku paketu v bytech. Celková délka paketu může být 8 až 65535 bytů.

Kontrolní součet (16 bitů) slouží k detekci chyb. Políčko není povinné. Pokud není využito, musí obsahovat pouze nuly.



Obr. 1.7: Popis struktury paketu protokolu UDP.

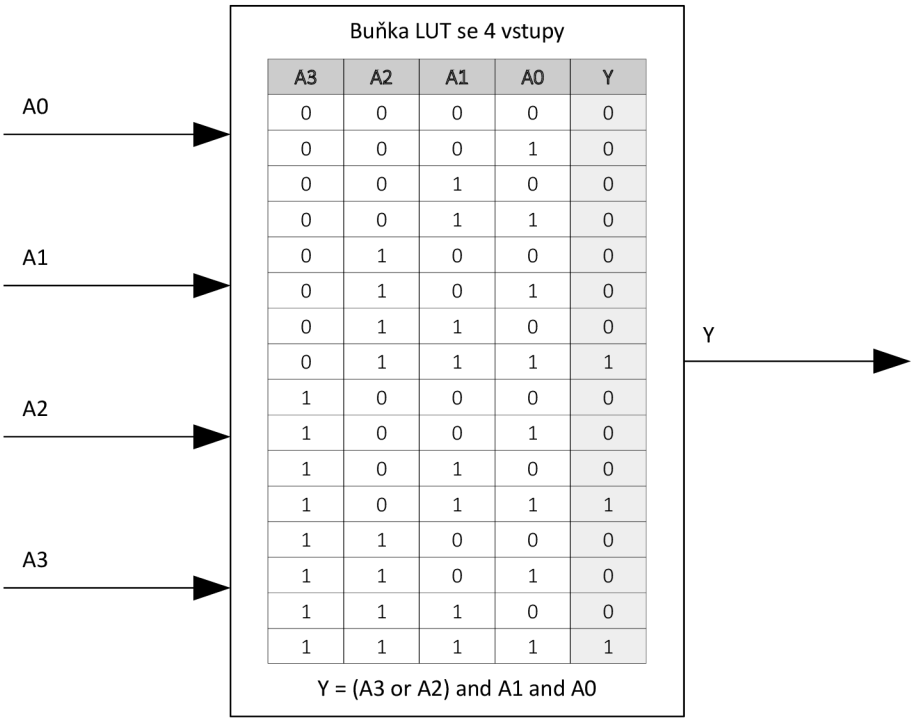
1.4 Obvody FPGA

Obvody FPGA (Field Programmable Gate Array) se řadí mezi programovatelná hradlová pole. Historie obvodů FPGA sahá až do roku 1984, kdy byl představen první obvod FPGA od firmy Xilinx. Obvody FPGA lze rozdělit do dvou druhů: FPGA s volatilní konfigurací a FPGA s nevolatilní konfigurací. Každý druh FPGA má své výhody i nevýhody. Volatilní FPGA používá paměťové buňky typu SRAM a drží se technologicky napřed, jeho nevýhoda je nutnost načíst konfuguraci po zapnutí obvodu. Nevolatilní FPGA používají paměti typu FLASH nebo EEPROM, typicky mají lepší odolnost proti radiaci, ale jejich konfiguraci je obtížnější změnit. V současnosti se výrobou obvodů FPGA zabývají zejména firmy Altera (Intel), Lattice a Xilinx.[6]

Obvody FPGA umožnují snadno implementovat vlastní digitální obvod navržený v HDL jazyce. Nejčastěji se používají HDL jazyky Verilog a VHDL. K implementaci digitálních obvodů je uvnitř FPGA připravena konfigurovatelná struktura základních a specializovaných bloků, které lze konfigurovatelně propojovat pomocí propojovací struktury. Mezi tyto bloky patří logické bloky, blokové paměti, DSP bloky, PLL bloky a transceivery. Logický blok obsahuje čtyřvstupou nebo šestivstupou buňku

LUT (LookUp Table), která slouží k implementaci logických funkcí, a konfigurovatelný klopný obvod.[6]

Každá buňka LUT umí implementovat libovolnou logickou funkci o tolika proměnných, kolik vstupů má buňka LUT. Buňka LUT je vytvořena pomocí jednoduché paměťové buňky, do které je nahrána konfigurace dle zvolené logické funkce. Jako vstupy buňky LUT slouží adresové vodiče, které reprezentují vstupní proměnné logické funkce. Pro každou adresu, čili kombinaci vstupních proměnných, je v paměťové buňce uložena odpovídající výsledná hodnota. Takže pokud přivedeme nějakou kombinaci bitů na vstupy LUT, na výstupu LUT dostaneme odpovídající výslednou hodnotu pro implementovanou logickou funkci. Na obrázku 1.8 je znázorněna čtyřvstupá buňka LUT s nastavenou pravdivostní tabulkou, která odpovídá na obrázku uvedené logické funkci. Pokud je nutné implementovat složitější logickou funkci, rozdělí se taková funkce mezi několik buněk LUT a jejich vhodným propojením dojde k implementaci požadované složitější logické funkce.[6]

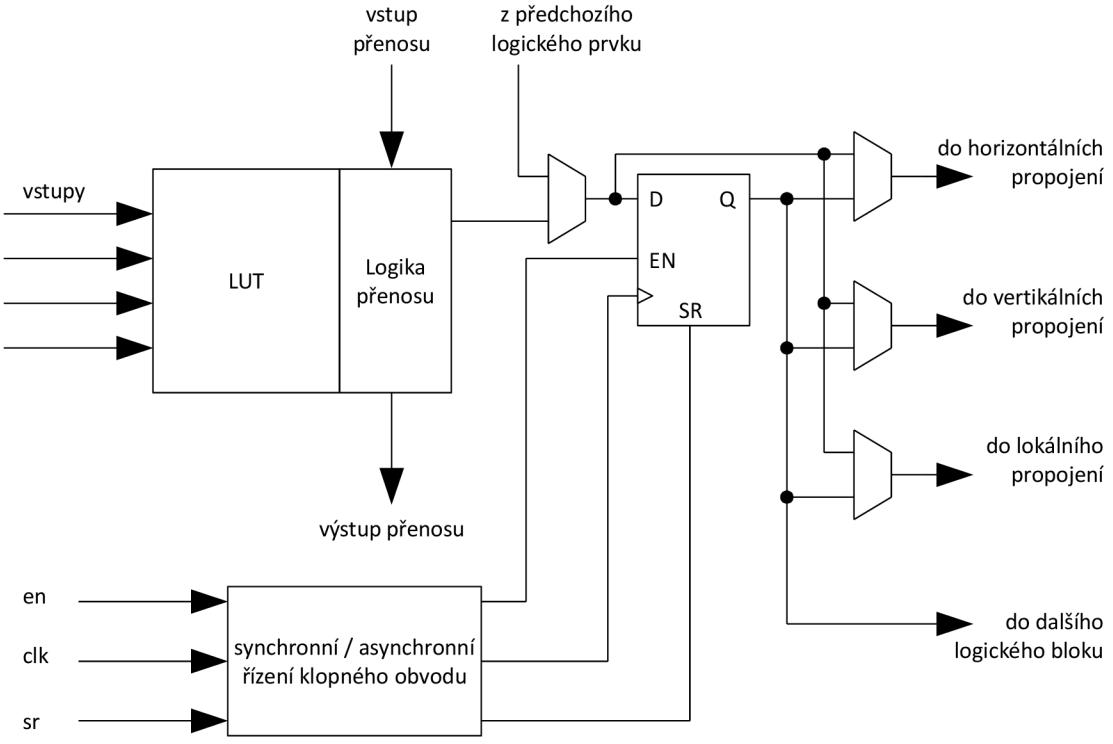


A3	A2	A1	A0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Obr. 1.8: Příklad čtyřvstupé buňky LUT s nastavenou logickou funkcí.

Konfigurovatelný klopný obvod mimo obvyklé vstupy a výstupy obsahuje také několik řídících vstupů, mezi které patří zejména povolovací vstup a synchronní nebo asynchronní reset. Na vstup klopného obvodu je zpravidla přiveden výstup z buňky LUT, ale většina dnešních FPGA umožňuje na vstup klopného obvodu přivést také jeden ze vstupů buňky LUT nebo výstup klopného obvodu z vedlejšího logického

bloku.[6] Některé moderní FPGA, například Xilinx Virtex 7, mohou mít v logickém bloku umístěny hned dva klopné obvody zapojené v sérii.[15] Zjednodušené schéma zapojení logického bloku je uvedeno na obrázku 1.9. Na tomto obrázku je možné vidět propojení LUT a klopného obvodu, zapojení řídicích signálů klopného obvodu a možné způsoby propojení s okolními logickými bloky.



Obr. 1.9: Zjednodušené schéma logického bloku.

Další důležitou částí FPGA jsou blokové paměti, které umožnují snadno implementovat různé typy a velikosti pamětí. Blokové paměti se nejčastěji používají pro implementace paměti typu ROM, RAM a FIFO. Jedna bloková paměť disponuje menší kapacitou, ale jednotlivé blokové paměti lze různě propojovat, a tak sestavit paměť s požadovanou datovou šířkou a hloubkou.[6]

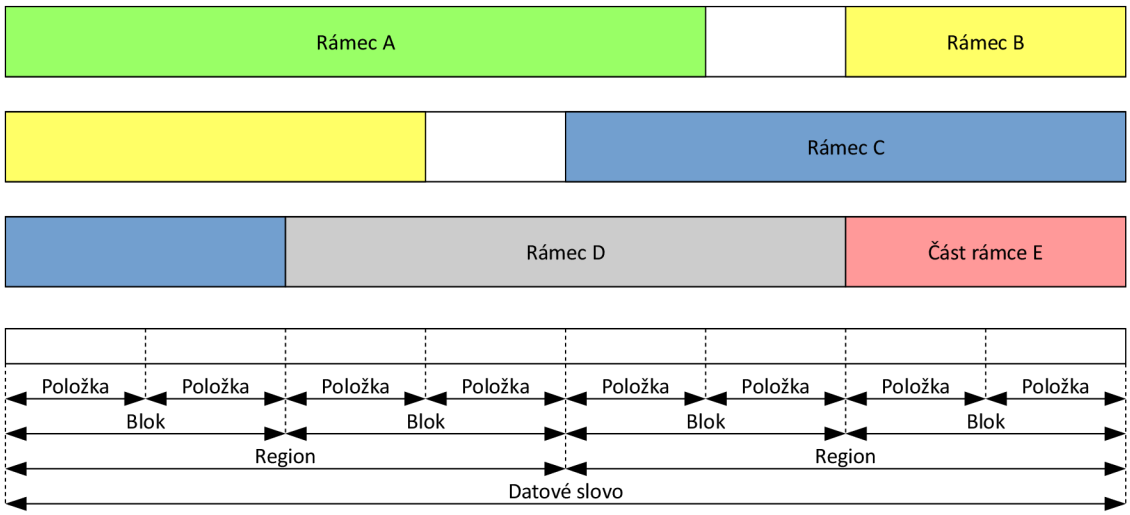
Neméně důležitou částí FPGA jsou DSP (Digital Signal Processing) bloky. Původně DSP bloky obsahovali pouze násobičku, v moderních FPGA už zvládají mimo násobení i další aritmetické funkce. Moderní DSP bloky jsou také silně konfigurovatelné a umožnují tak například povolit nebo vypnout vstupní a výstupní registry.[6]

Moderní FPGA obsahují až statisíce logických elementů a stovky megabitů blokových pamětí. Takto vybavené FPGA obvody již umožňují implementovat opravdu velké digitální obvody. Důležitým parametrem digitálních obvodů je i jejich rychlost, obvody implementované pomocí dnešních FPGA lze provozovat na frekvenci v řádu

stovek MHz. Výhodou obvodů FPGA je jejich univerzálnost a možnost opakované konfigurace. Naopak nevýhodou oproti obvodům ASIC je vyšší spotřeba a menší výkon. Obvody FPGA jsou především vhodné k implementaci digitálních obvodů v menších sériích, kdy jsou výrazně levnější než obvody ASIC. S výhodou lze využít opakované konfigurace obvodů FPGA a použít je také pro testování digitálních obvodů nebo pro sdílení hardwaru mezi více aplikací.[6]

1.5 Sběrnice Multi Frame Bus

Vývojový tým Liberouter, který je součástí sdružení CESNET, umí pracovat se standardem Ethernet pro přenosovou rychlost 100 Gb/s a v současné době se připravuje na příchod standardu Ethernet pro rychlost 400 Gb/s. Součástí firmwaru současné i budoucí síťové karty s obvodem FPGA bude i obvod, který zajišťuje fyzický příjem ethernetových rámců. Datový výstup takového obvodu je realizován pomocí sběrnice MFB (Multi Frame Bus).[16]



Obr. 1.10: Struktura datového slova sběrnice MFB s ukázkou rámců.

Už z názvu sběrnice je patrné, že byla navržena tak, aby zvládla přenést jeden nebo i více rámců v jednom datovém slově. Celé datové slovo sběrnice MFB se dělí na regiony, bloky a položky. Nejmenší díl datového slova se nazývá položka (Item). Šířka jedné položky odpovídá nejmenší granularitě rámců, pro Ethernet je šířka položky rovna 1 bytu, respektive 8 bitů. Konce rámců jsou zarovnávány na jednotlivé položky. Začátky rámců jsou zarovnávány na začátky bloků (Block). Jeden blok je složen z určitého počtu položek tak, že jeho šířka odpovídá hodnotě, na kterou jsou rámce zarovnávány. Nejhrubší díl datového slova se nazývá region. Region je složen

z bloků a jeho šířka odpovídá minimální možné délce, jakou může mít jeden rámec, pro ethernetový rámec to je 64 bytů. Celé datové slovo se pak skládá z počtu regionů, který lze zvolit libovolně dle potřeby.

Stanovením velikosti regionu, bloku, položky a počtu regionů můžeme sběrnici přizpůsobovat pro různé typy rámců. Díky tomu je sběrnice MFB velmi obecná a lze ji použít v mnoha digitálních obvodech. Velikost regionu, bloku, položky a počet regionů lze konfigurovat pomocí čtyř parametrů, které jsou v následujícím textu podrobně popsány.

Šířka položky - (Item Width, iw) definuje šířku položky v bitech, která odpovídá nejmenší granularitě rámců

Velikost bloku - (Block Size, bs) definuje velikost bloku v počtu položek

Velikost regionu - (Region Size, rs) definuje velikost regionu v počtu bloků

Počet regionů - (Regions, r) definuje počet regionů v jednom datovém slově

Na obrázku 1.10 je zobrazeno rozdělení datového slova na regiony, bloky a položky. Dále tento obrázek zobrazuje ukázku umístění rámců v datovém slově, zarovnání začátků rámců na bloky a zarovnání konců rámců na položky. Každý řádek na obrázku odpovídá jednomu datovému slovu. Jednotlivé řádky jsou po sobě jdoucí datová slova, kde první řádek je první slovo.

Signál	**Šířka (b)**	**Směr**
DATA	$r \cdot rs \cdot bs \cdot iw$	zdroj → cíl
SOF	r	zdroj → cíl
EOF	r	zdroj → cíl
SOF_POS	$r \cdot \log_2(rs)$	zdroj → cíl
EOF_POS	$r \cdot \log_2(rs \cdot bs)$	zdroj → cíl
SRC_RDY	1	zdroj → cíl
DST_RDY	1	cíl → zdroj

Tab. 1.1: Signály sběrnice MFB (Multi Frame Bus).

Tabulka 1.1 uvádí přehled jednotlivých signálů sběrnice MFB. Z této tabulky je patrné, že sběrnice obsahuje celkem sedm signálů, z toho šest je řídících a jeden datový. Popis těchto signálů sběrnice MFB je uveden v následujícím textu.

DATA je vícebitový signál, který přenáší data od zdroje do cíle. Data jsou platná, když je SRC_RDY v log.1.

Start Of Frame (SOF) je vícebitový signál, který určuje platnost začátku rámce v jednotlivých regionech. Signál je platný, když je SRC_RDY v log.1.

End Of Frame (EOF) je vícebitový signál, který určuje platnost konce rámce v jednotlivých regionech. Signál je platný, když je SRC_RDY v log.1.

Start Of Frame Position (SOF_POS) je vícebitový signál, který určuje pozice začátku rámce pro jednotlivé regiony.

End Of Frame Position (EOF_POS) je vícebitový signál, který určuje pozici konce rámce pro jednotlivé regiony.

Source Ready (SRC_RDY) je jednobitový řídící signál, který značí připravenost odesílatele odeslat data. Signál také určuje platnost signálů směřujících od zdroje k cíli.

Destination Ready (DST_RDY) je jednobitový řídící signál, který značí připravenost příjemce přijmout data. Přenos dat může probíhat pouze, když je současně připraven odesílatel i příjemce.

Konfigurace MFB sběrnice	**Šířka datové sběrnice (b)**	**Frekvence (MHz)**	**Přenosová rychlost (Gb/s)**
MFB(1,8,8,8)	512	200	100
MFB(2,8,8,8)	1024	100	100
MFB(4,8,8,8)	2048	50	100
MFB(1,8,8,8)	512	800	400
MFB(2,8,8,8)	1024	400	400
MFB(4,8,8,8)	2048	200	400

Tab. 1.2: Porovnání různých kombinací frekvencí a šířek datové sběrnice.

Konfigurace sběrnice MFB s konkrétní konfigurací parametrů bude dále značena ve tvaru MFB(r,rs,bs,iw). S vhodně zvolenou frekvencí a konfigurací sběrnice lze dosáhnout požadované přenosové rychlosti. Tabulka 1.2 uvádí přehled vybraných kombinací frekvence a datové šířky slova respektive konfigurace MFB sběrnice pro přenosové rychlosti 100 Gb/s a 400 Gb/s. Z tabulky lze vyčíst, že požadované přenosové rychlosti dosáhneme buď zvolením vysoké frekvence anebo široké datové sběrnice. Obvody FPGA zatím neumožnují dosahovat vysokých frekvencí jako obvody ASIC, proto vývojový tým Liberouter v současnosti provozuje většinu designů na frekvenci 200 MHz.[16] Pro přenosouvou rychlost 100 Gb/s tak stačí datová šířka 512 b respektive MFB(1,8,8,8). Pro přenosovou rychlost 400 Gb/s je nutné rozšířit datovou šířku na 2048 b respektive MFB(4,8,8,8); důsledek takové datové šířky je větší náročnost na zdroje FPGA.

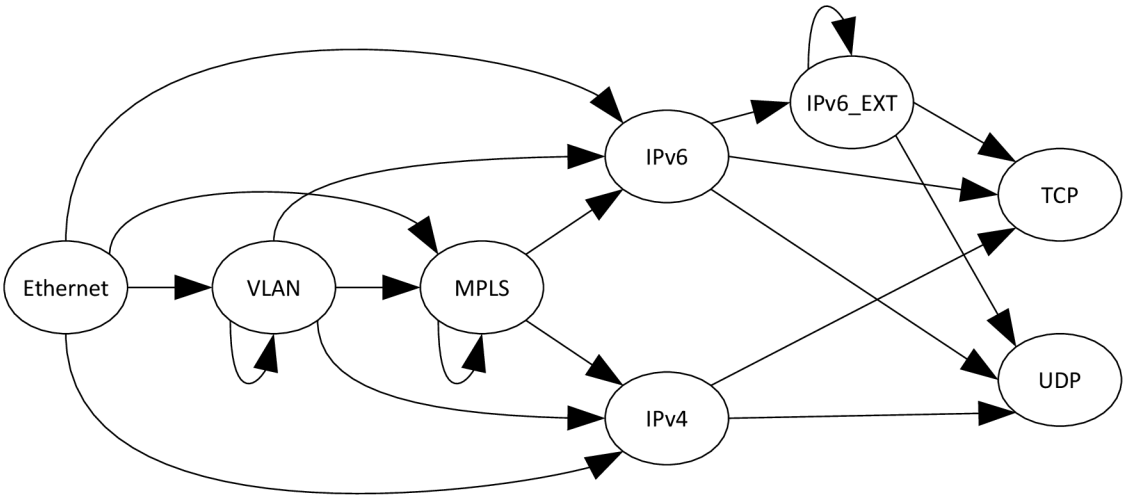
2 ANALÝZA ETHERNETOVÝCH RÁMCŮ

Tato kapitola se zabývá analýzou internetového provozu při rychlostech 100 Gb/s a 400 Gb/s. Jak již bylo popsáno v teoretickém rozboru, internetový provoz je tvořený ethernetovými rámci. Při analýze jsou z nich extrahovány hlavičky jednotlivých internetových protokolů. Informace obsažené v těchto hlavičkách lze následně zpracovávat dle potřeby.

Jedním z hlavních cílů této práce je navrhnout takový obvod analyzátoru, který bude schopný dosahovat požadované přenosové rychlosti při všech délkách ethernetových rámců. Jeho architektura bude natolik obecná, aby bylo možné nastavit požadovanou přenosovou rychlost pouhou změnou parametrů tohoto analyzátoru.

2.1 Problematika

Při analýze ethernetového rámce je vhodné postupovat od jeho začátku a postupně analyzovat hlavičky jednotlivých zabalených protokolů. Na reálné síti mohou nastat situace, kdy není dodržován princip, kdy protokoly jedné vrstvy předávají data protokolům z nižší vrstvy modelu ISO/OSI. Takové situace nastávají například při tunelování protokolu IPv6 pomocí protokolu IPv4, ale podobných možností, jak takové situace docílit, je více. Každý ethernetový rámec může obsahovat jinou sadu protokolů a stejný protokol nemusí být v ethernetovém rámci vždy na stejné pozici.



Obr. 2.1: Schéma možného zapouzdření uvedených protokolů.

Obrázek 2.1 zobrazuje schéma možného zapouzdření protokolů v ethernetovém rámci. Šipka směřující z protokolu A do protokolu B znamená, že v protokolu A může

být zapouzdřen protokol B. Šipka směřující z protokolu A a zpět do něj znamená, že v protokolu A může být zapouzdřen opět protokol A. Toto schéma je platné pro sadu protokolů, která obsahuje: Ethernet, VLAN, MPLS, IPv4, IPv6, rozšíření IPv6 (IPv6 EXT), TCP a UDP. Čím komplexnější podporovaná protokolová sada bude, tím větší část internetového provozu dokážeme analyzovat.

Začátek protokolu, který je zapouzdřený v ethernetovém rámci, dokážeme určit pomocí známé délky ethernetové hlavičky. Typ zapouzdřeného protokolu je obsažen v ethernetové hlavičce (políčko Délka/Typ). Pokud analyzátor podporuje zjištěný typ zapouzdřeného protokolu, může být tento protokol analyzován a lze z něj určit informace k rozeznání dalšího zapouzdřeného protokolu. Tento postup hledání a identifikace následujících zapouzdřených protokolů se opakuje až do nalezení posledního zapouzdřeného protokolu nebo do nalezení nepodporovaného protokolu.

Podporovanou sadu protokolů může být v budoucnu nutné rozšířit, proto by měl být analyzovací obvod navržen s dostatečně modulární architekturou, která umožní snadné přidávání dalších analyzovatelných protokolů. Díky tomu bude obvod natolik obecný, že jej bude možné využít pro mnoho síťových aplikací.

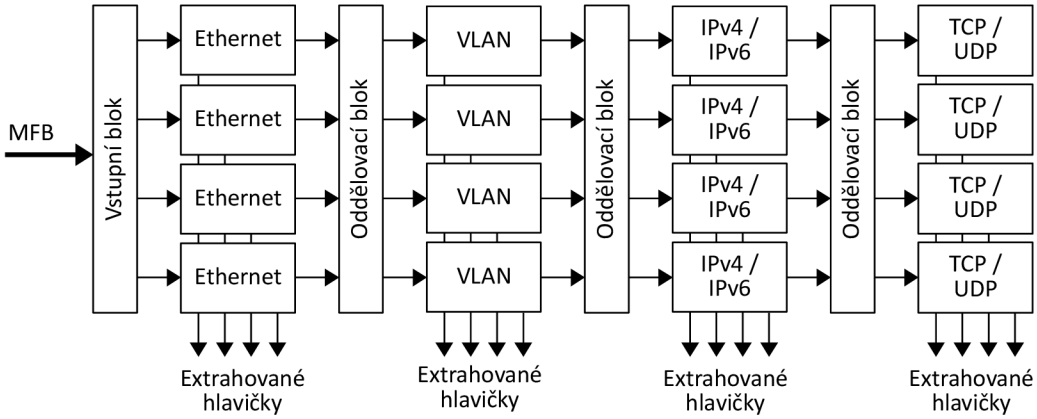
2.2 Návrh obvodu

Navrhovaný analyzovací obvod, dále označovaný pod interním názvem MFB HFE (Multi Frame Bus Header Field Extractor), využívá konfigurovatelnou datovou sběrnici MFB. Požadovaných přenosových rychlostí 100 Gb/s a 400 Gb/s dosáhne obvod kombinací vhodné MFB konfigurace a frekvence hodinového signálu. Obvod MFB HFE využívá zřetězeného zpracování dat, za účelem dosažení vysoké propustnosti.

Architekturu obvodu, která je znázorněna v blokovém schématu na obrázku 2.2, tvoří vhodně poskládané dílčí bloky. Funkce obvodu MFB HFE je taková, že na vstupní sběrnici MFB přichází do obvodu ethernetové rámce, které jsou v každém stupni zřetězeného zpracování analyzovány na konkrétní protokol. Pokud je analyzovaný protokol v ethernetovém rámci obsažen, je z rámce extrahována jeho hlavička.

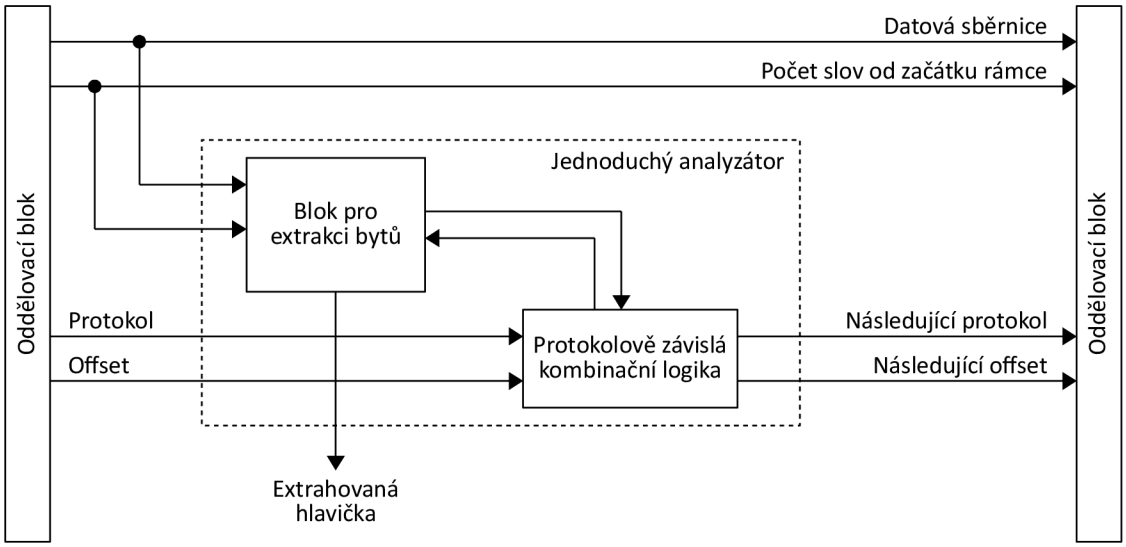
Jednoduchý analyzátor musí zvládnout analyzovat jeden ethernetový rámec za hodinový takt. Při přenosové rychlosti 100 Gb/s a sběrnici MFB(1,8,8,8) může přijít po sběrnici maximálně jeden celý ethernetový rámec, ale při přenosové rychlosti 400 Gb/s a sběrnici MFB(4,8,8,8) mohou přijít až čtyři celé ethernetové rámce současně. Navrhovaný obvod MFB HFE tak musí mít, ve variantě pro přenosovou rychlost 400 Gb/s, čtyři paralelně zapojené jednoduché analyzátory, díky čemuž celý obvod dokáže analyzovat jeden protokol až u čtyř ethernetových rámců současně v jednom hodinovém taktu.

Na začátku celého obvodu MFB HFE je vstupní blok, který má za úkol počítat počet slov od začátku každého ethernetového rámce a určit pozici prvního protokolu



Obr. 2.2: Blokové schéma návrhu obvodu pro přenosovou rychlost 400 Gb/s.

(Ethernet) v datovém slově na sběrnici MFB. Tento blok následně tyto udaje předá prvnímu stupni analyzátorů, které analyzují protokol Ethernet. Jednotlivé stupně analyzátorů jsou odděleny oddělovacími bloky, tyto bloky slouží k přenosu řídicích dat mezi jednotlivými stupni. Řídicí data jsou v těchto blocích distribuována do regionů tak, aby je dostali všechny odpovídající jednoduché analyzátory. Tyto oddělovací bloky mohou obsahovat registry pro dosažení vyšších frekvencí, pro které může být obvod implementován.

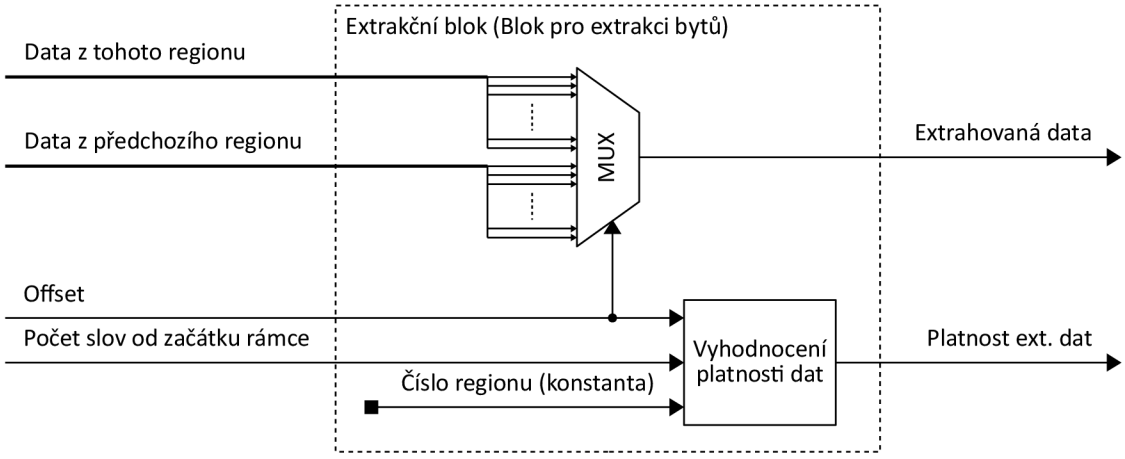


Obr. 2.3: Blokové schéma jednoduchého analyzačního bloku.

Jeden analyzátor je tvořený blokem pro extrakci určitého počtu bytů a řídicí kombinační logikou, která se liší pro každý analyzovaný protokol. Každý analyzátor má mít na vstupu informace o typu příchozího protokolu, o začátku protokolu v datovém slově na sběrnici MFB (tzv. offset) a o počtu slov od začátku současného ethernetového rámce. Tyto údaje získá analyzátor z předchozího oddělovacího bloku nebo od startovního bloku. Z nich analyzátor vyhodnotí, zda příchozí protokol umí analyzovat a zda je začátek protokolu na pozici, kterou dotyčný analyzátor vidí.

Každý analyzátor přísluší k jednomu regionu sběrnice MFB. Vnitřní struktura a zapojení analyzátoru je znázorněna v blokovém schématu na obrázku 2.3. Analyzátor musí umět analyzovat protokol, když indentifikuje konec jeho hlavičky ve svém regionu, ale protože jednotlivé protokoly nejsou zarovnány na regiony, musí analyzátor vidět na data ve svém a předchozím regionu. Z toho plyne, že maximální délka analyzovaného protokolu je rovna délce jednoho regionu. Pokud dojde k situaci, kdy v jednom regionu končí dvě hlavičky protokolů, první hlavičku analyzuje příslušný analyzátor, a druhou hlavičku analyzuje v rámci výpomoci následující analyzátor, který druhou hlavičku také ještě vidí. Pokud je analyzátor zaneprázdněn výpomocí, musí hlavičky protokolů v jeho regionu analyzovat jeho následující analyzátor.

Na výstupu z analyzátoru musí být jednotlivá extrahovaná políčka z hlavičky analyzovaného protokolu, ale především musí mít analyzátor na výstupu řídicí informace o pozici a typu následujícího protokolu. Tyto řídicí informace zpracuje následující oddělovací blok a předá je dalšímu stupni analyzátorů.



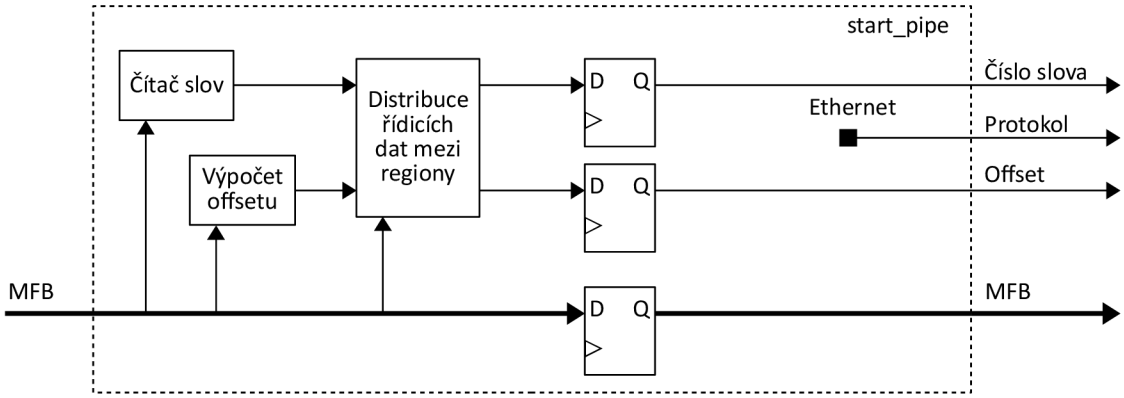
Obr. 2.4: Blokové schéma struktury extrakčního bloku.

Extrakční blok, který je součástí každého analyzátoru, tvoří především parametricky nastavitelný multiplexor a kombinační logika. Tato kombinační logika určuje platnost extrahovaných dat. Struktura extrakčního bloku je znázorněna v blokovém

schématu na obrázku 2.4. Pro každý analyzátor určitého protokolu, lze extrakční blok parametricky nastavit tak, aby extrahoval stanovený počet bytů. Pozice začátku analyzované hlavičky (offset) slouží z části přímo jako adresa pro multiplexor. Platnost extrahovaných dat je pak určena porovnáním offsetu s počtem slov od začátku ethernetového rámce a s číslem regionu, ke kterému je extrakční blok přidělen.

2.3 Implementace obvodu

Tato kapitola popisuje jednotlivé části obvodu MFB HFE tak, jak byl obvod implementován podle uvedeného návrhu. Obvod MFB HFE využívá zřetězeného zpracování dat. Vstupní blok z návrhu je implementován jako komponenta s názvem start_pipe. Blokové schéma této komponenty je zobrazeno na obrázku 2.5.



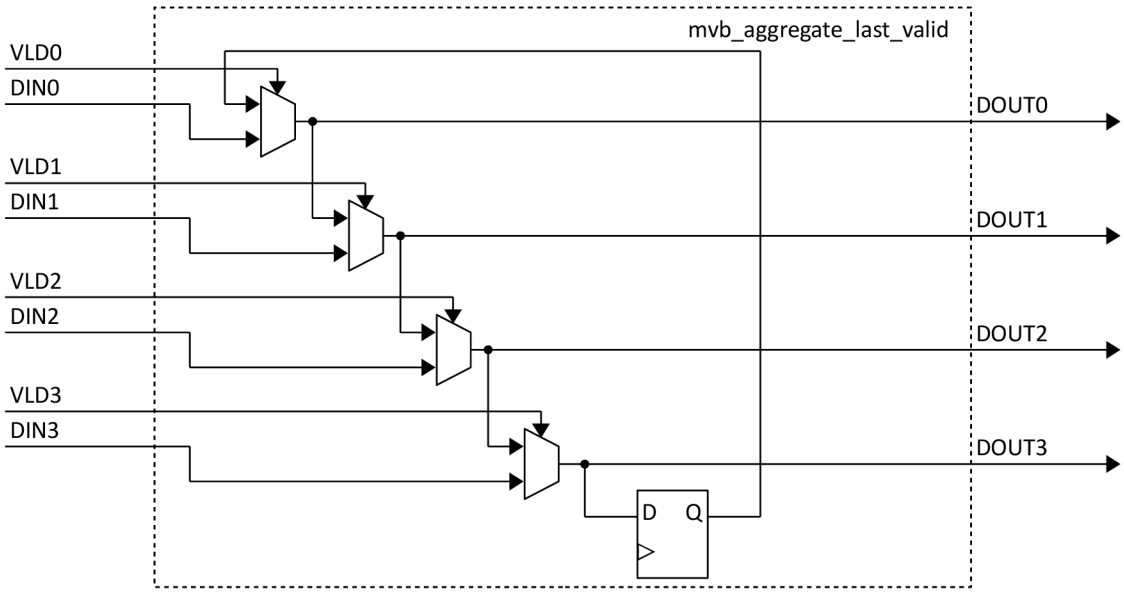
Obr. 2.5: Zjednodušené blokové schéma komponenty start_pipe.

Vstupní blok v sobě obsahuje čítač platných slov ze vstupní sběrnice MFB. Čítač počítá výslednou hodnotu pro každý region zvlášť. Hodnota čítače je resetována, pokud v regionu začíná nový ethernetový rámec. Další částí vstupního bloku je logika pro výpočet offsetů začátků ethernetových rámců. Offset se počítá z řídicích signálů sběrnice MFB. Vypočtené hodnoty z čítače a hodnoty offsetů jsou správně rozdistribuovány blokem mvb_aggregate_last_valid, který slouží pro distribuci dat mezi regiony. Na konci komponenty start_pipe jsou volitelné výstupní registry, které zajišťují lepší splnění časových podmínek obvodu.

Vnitřní struktura komponenty mvb_aggregate_last_valid je zjednodušeně znázorněna na obrázku 2.6. Součástí této komponenty je registr a řada multiplexorů 2 na 1, jejichž počet odpovídá počtu regionů. Tato komponenta předá na výstupy pro jednotlivé regiony poslední platná data. Jednotlivé multiplexory jsou řízeny bitem značícím platnost dat. Pokud jsou vstupní data platná, jsou přenesena na výstup,

pokud nesjou platná, na výstup jsou přenesena výstupní data z předchozího regionu. Registr v tomto obvodu slouží, k uložení výstupních dat z posledního regionu do následujícího platného slova.

Za tímto vstupním blokem se nachází analyzátor protokolu Ethernet. Pro každý region sběrnice MFB je zde zapojen jeden takový analyzátor. Tento blok se nazývá ethernet_analyzer a jeho úkolem je extrahovat jednotlivá políčka z hlavičky protokolu Ethernet, konkrétně cílovou MAC adresu, zdrojovou MAC adresu a následující protokol. Z extrahovaných dat obvod vypočítá nové řídicí data pro analyzování dalšího internetového protokolu.



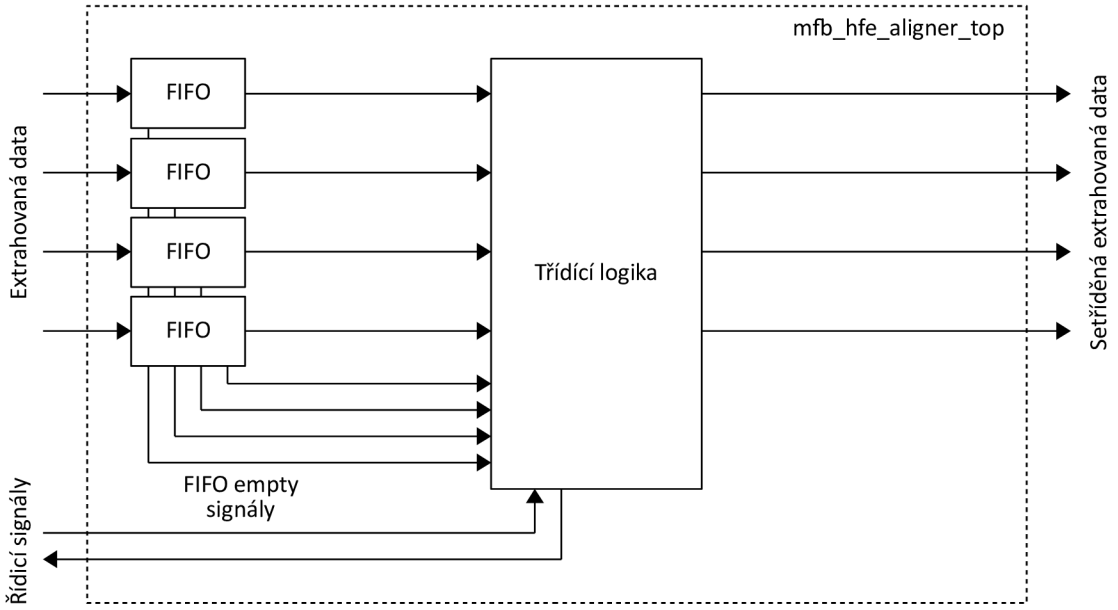
Obr. 2.6: Zjednodušené schéma struktury distribučního bloku.

Vnitřní struktura komponenty ethernet_analyzer, platná i pro analyzátory jiných protokolů, je následující: Analyzátor pro konkrétní protokol je implementovaný pouze jako obálka obecného bloku analyze_block doplněná o specifickou logikou pro konkrétní internetový protokol. Tento obecný blok zahrnuje doplňující logiku a především komponentu get_bytes, která slouží pro extrakci parametricky konfigurovatelného počtu bytů z datového slova sběrnice MFB. Komponentu get_bytes tvoří parametrický multiplexor a logika pro řízení extrakce bytů.

V následujícím stupni zřetězeného zpracování, po analyzátoru protokolu Ethernet, je připojena komponenta s názvem mid_pipe, která v návrhu odpovídá oddělovacímu bloku. Tato komponenta slouží k distribuci platných řídicích dat z předchozích analyzátorů do analyzátorů, které se nachází za tímto oddělovacím stupněm. Komponenta mid_pipe obsahuje již popsaný blok mvb_aggregate_last_valid pro

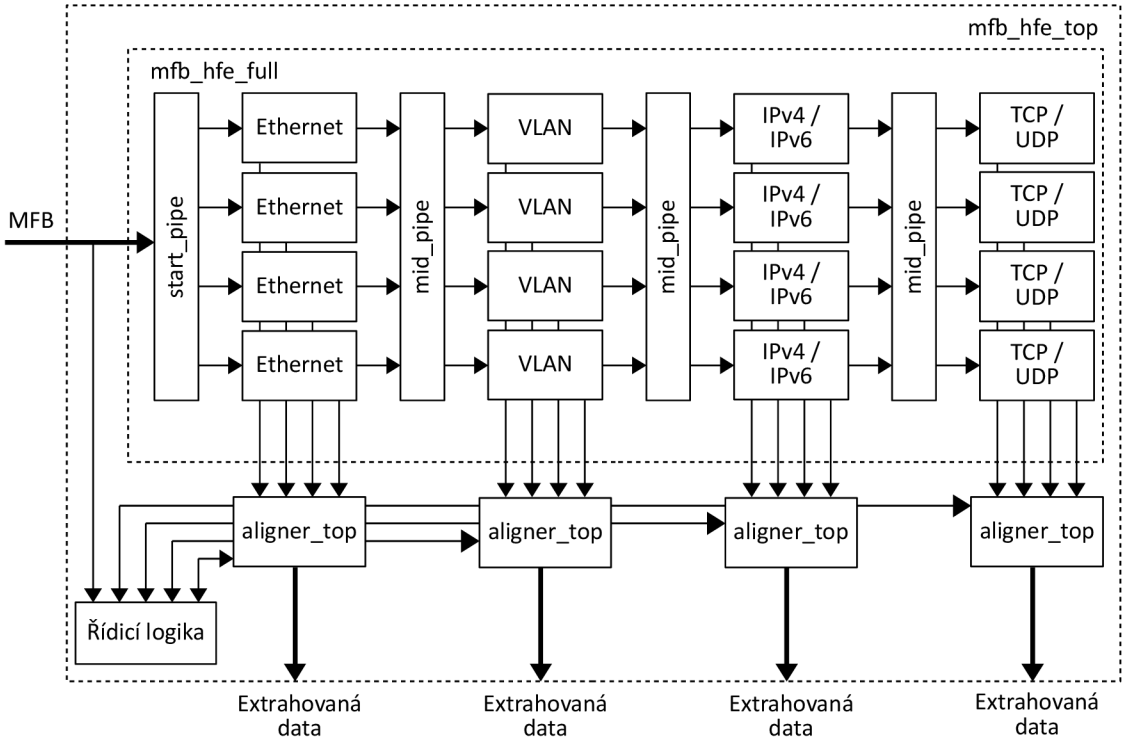
distribuci řídicích dat a registry pro zlepšení časování. Registry jsou parametricky volitelné a jsou umístěny v řídicí i v datové cestě.

Další stupeň zřetězeného zpracování slouží k analýze dalšího protokolu a obsahuje příslušné bloky. Dále následuje oddělovací stupeň s komponentou mid_pipe. Tato struktura se dále opakuje až po stupeň analyzující poslední podporovaný protokol. Pořadí analýzy protokolů odpovídá nejdelší cestě ve schématu popisující zapouzdření internetových protokolů. Pro zjednodušenou sadu protokolů bylo toto schéma znázorněno a popsáno v kapitole 2.1. Implementovaný obvod MFB HFE ve výchozím stavu podporuje následující internetové protokoly: Ethernet, VLAN, MPLS, IPV4, IPV6, vybrané rozšíření IPV6 (IPV6 EXT), UDP, TCP, ICMP a SCTP. Ethernetové rámce mohou obsahovat některé protokoly vícekrát, proto i implementovaný obvod obsahuje více analyzovacích stupňů pro takové protokoly (4x VLAN, 4x MPLS, 2x IPV6 EXT). Zapojení celého zřetězeného zpracování vstupních ethernetových rámců tvoří komponenta s názvem mfb_hfe_full.



Obr. 2.7: Zjednodušené blokové schéma komponenty aligner_top.

Extrahovaná data z jednotlivých analyzačních stupňů se dostávají na výstup tak, jak postupně prochází ethernetový rámec celým zřetězeným zpracováním. Aby bylo možné mít všechna extrahovaná data k jednomu ethernetovému rámci k dispozici v jednu chvíli (v jeden hodinový takt), jsou extrahovaná data ukládána do vyrovnávacích pamětí FIFO. Jakmile jsou všechna extrahovaná data pro jeden ethernetový rámec uložena v paměti FIFO, pak mohou být současně vyčtena na výstup celého implementovaného obvodu.



Obr. 2.8: Zjednodušené blokové schéma celého obvodu MFB HFE.

Počet výstupů pro jednotlivá extrahovaná data odpovídá počtu regionů sběrnice MFB. Pokud je počet regionů sběrnice MFB větší než jedna, pak extrahovaná data některých analyzovaných protokolů nemusí přijít na výstupu, který odpovídá číslu regionu, kde analyzovaný ethernetový rámec začínal. Proto je nutné výstupní paměti FIFO doplnit o logiku, která zajistí roztřídění extrahovaných dat. Díky tomu jsou všechna extrahovaná data analyzovaného ethernetového rámce na výstupu, který odpovídá číslu regionu, kde daný ethernetový rámec začínal. Tato logika včetně paměti FIFO je součástí komponenty aligner_top, která je znázorněná zjednodušeným blokovým schématem na obrázku 2.7. Celý implementovaný obvod MFB HFE je pak zabalen do obálky s názvem mfb_hfe_top, která je znázorněná zjednodušeným blokovým schématem na obrázku 2.8.

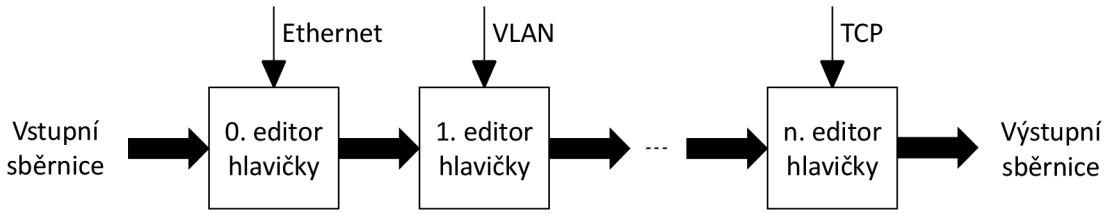
3 SKLÁDÁNÍ ETHERNETOVÝCH RÁMCŮ

Tato kapitola se zabývá návrhem skládání ethernetových rámců z připravených hlaviček jednotlivých internetových protokolů. Tuto problematiku lze řešit běžným procesorem, ale při vysokých přenosových rychlostech je vhodnější použít specializovaný obvod. Skládání ethernetových rámců může být vyžadováno v případě, kdy chceme generovat úplně nové ethernetové rámce nebo v případě, kdy chceme například do existujícího ethernetového rámce vložit hlavičky dalších protokolů.

Pokud máme připravena data, respektive hlavičky jednotlivých protokolů, které bude ethernetový rámec obsahovat, pak stačí tyto data poskládat do správného pořadí za sebe. Takto lze získat kompletní požadovaný ethernetový rámec. Z pohledu návrhu obvodu, který bude generovat požadované ethernetové rámce při vysoké rychlosti až 400 Gb/s, je problematika složitější zejména z důvodu možnosti vytvořit více než jeden rámec za takt a s tím související náročnosti na spotřebu zdrojů.

3.1 Problematika

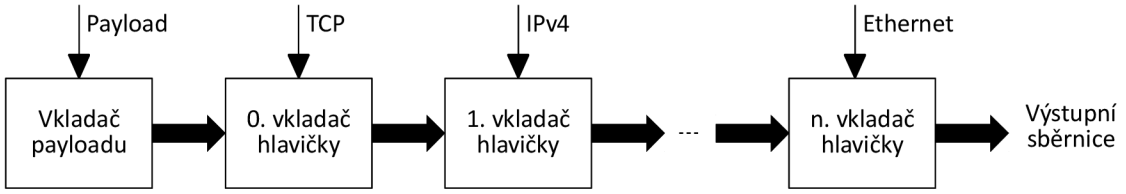
Ethernetové rámce můžeme upravovat, anebo je celé znovu poskládat z hlaviček jednotlivých protokolů. Pokud nechceme měnit strukturu protokolů v ethernetovém rámci, pak stačí přepsat upravené hlavičky protokolů. Obvod pro takovou modifikaci ethernetových rámců (editor) pak potřebuje znát pouze pozice jednotlivých protokolů, aby dokázal změnit požadované byty. V případě, kdy je požadované i odebíraní vybraných protokolů z ethernetového rámce, musí editor umět posunout část rámce za odebraným protokolem. Implementace editoru s možností posunu je však více náročná na zdroje než obyčejný editor ethernetových rámců.



Obr. 3.1: Blokové schéma editoru rámců.

V mnoha případech může být nutné požadovat, kromě editace a odebírání jednotlivých protokolů, také přidávání nových protokolů, které nejsou v původním ethernetovém rámci zabaleny anebo je nutné protokoly v ethernetovém rámci kompletně

přeskládat. V takovém případě je původní rámec v podstatě přepisován novým rámcem. Editor tak musí být schopný posouvat častmi rámců tak, aby uměl vytvořit místo pro nový protokol nebo odebrat místo po odstraněném protokolu. Blokové schéma editační architektury je zobrazeno na obrázku 3.1, kde každý dílčí editor pracuje s jedním protokolem. Jednotlivé dílčí editory jsou zapojeny zasebou a postupně tak dochází k editaci celého ethernetového rámce.

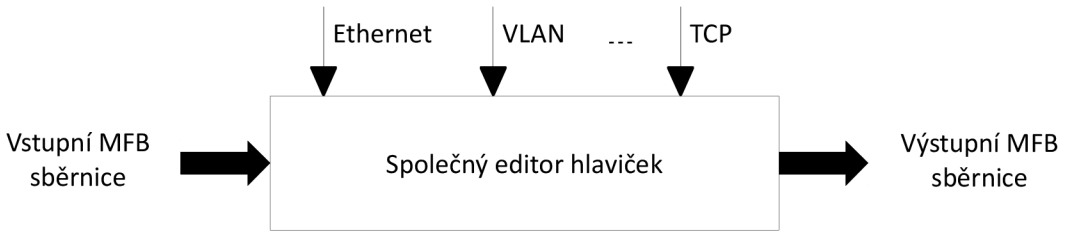


Obr. 3.2: Blokové schéma generátoru rámců.

Alternativní možností je generátor ethernetových rámců. Blokové schéma architektury generátoru je zobrazeno na obrázku 3.2. Generátor ethernetových rámců generuje na prázdnou datovou sběrnici nové ethernetové rámce z hlaviček jednotlivých protokolů a k nim přidává datovou část (payload). Takový generátor může využívat zřetězené zpracování dat a ethernetové rámce skládat postupně po jednom protokolu v každém stupni.

3.2 Návrh obvodu

Při srovnání dvou výše popsaných architektur lze dojít k závěru, že generátor i editor musí umět vkládat jednotlivé hlavičky na různá místa na datové sběrnici. Generátor navíc musí vkládat také payload, ale editor zase musí posouvat části rámců. Ve výsledku v obou případech tyto funkcionality zajistí velký počet multiplexorů. Za předpokladu, že bude použita datová sběrnice o šířce až 2048 bitů, bude množství těchto multiplexorů mít značný vliv na celkovou spotřebu zdrojů.



Obr. 3.3: Blokové schéma obvodu s jedním společným editorem.

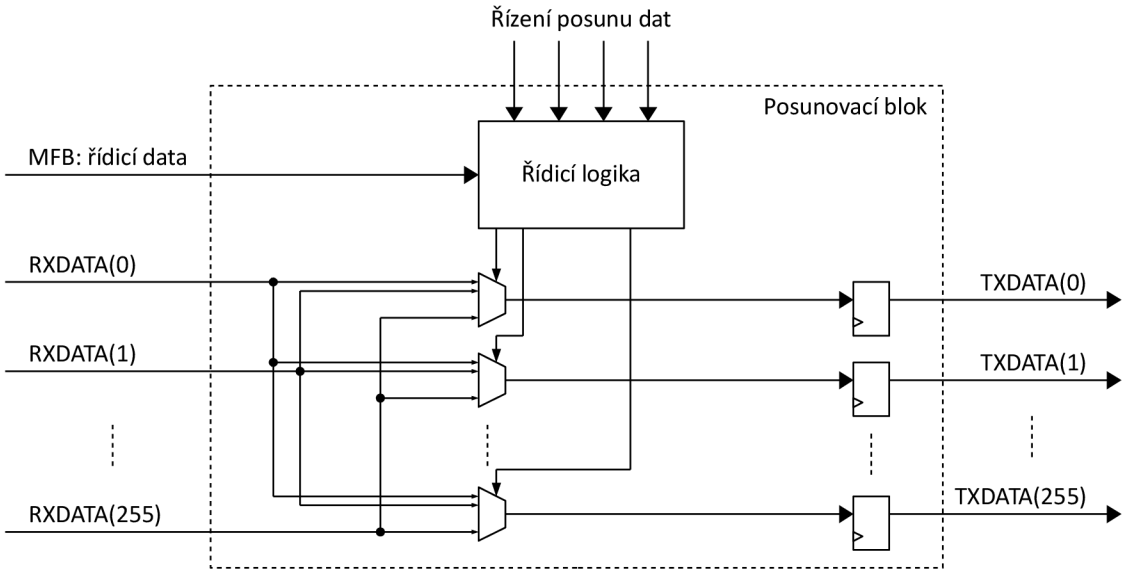
Zřetězené zapojení dílčích obvodů můžeme nahradit jedním velkým blokem, který bude všechny editace a posuny dat provádět sám. Takovou architekturu znázornuje blokové schéma na obrázku 3.3. Taková změna sníží celkový počet multiplexorů a celkový počet multiplexorových vstupů, ale naroste počet multiplexorových vstupů na jeden multiplexor. Díky této změně ušetříme určité množství multiplexorových vstupů. Každý multiplexor totiž obsahuje vstup na přivedení předchozí hodnoty dat a pokud se sníží počet multiplexorů, sníží se počet i těchto vstupů.

Počet vstupů	**Šířka vstupu (b)**	**Spotřeba LUT**
1024	8	15945
512	8	2591
256	8	584
128	8	272
64	8	152
32	8	72

Tab. 3.1: Porovnání spotřeby zdrojů po syntéze u různě velkých multiplexorů.

Mohlo by se zdát, že takové řešení přinese úsporu zdrojů, ale opak je pravdou. Problém je v tom, že generické multiplexory s velkým počtem vstupů se hůře mapují na zdroje v FPGA čipech. Efektivnější mapování multiplexorů s menším počtem vstupů do FPGA dokazuje i tabulka 3.1. Tato tabulka uvádí přehled spotřeby zdrojů pro multiplexory s různými počty vstupů. U multiplexorů s menším počtem vstupů (do 256) se spotřeba zdrojů přibližně zdvojnásobí, pokud se zdvojnásobí počet vstupů. Ale když zdvojnásobíme počet vstupů na 512, spotřeba zdrojů vzroste více než čtyřikrát. Při zdvojnásobení počtu vstupů na 1024 se spotřeba LUT vyšplhá až na hodnotu 15945. Takto velký multiplexor spotřebuje 4,4 % LUT z celého FPGA xc7vh580thcg1931-2 (řada Virtex 7). Pokud bychom chtěli použít větší počet takových multiplexoru tak, brzy vyčerpáme zdroje tohoto FPGA. Hodnoty spotřeby zdrojů uvedené v tabulce byly získány při syntéze multiplexorů ve vývojovém prostředí Xilinx Vivado 2016.3 pro FPGA z řady Virtex 7 (xc7vh580thcg1931-2).

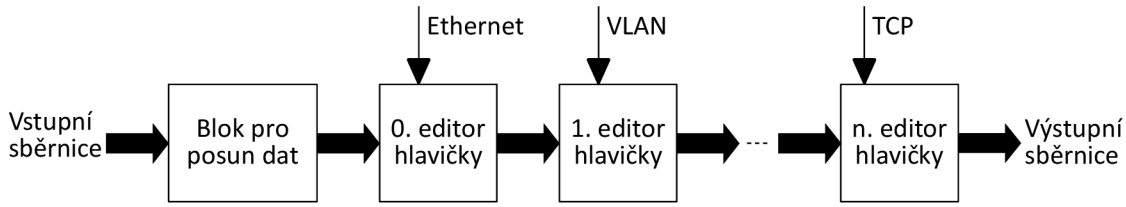
Z předchozí úvahy plyne, že je vhodnější použít obvod tvořený zřetězeným zapojení dílčích bloků, který v každém stupni bude obsahovat editor pro určitý protokol. Tato architektura je také lépe strukturovatelná a lze do ní snadněji přidávat podporu nových protokolů. Editační architekturu lze využít nejen k modifikaci vstupních ethernetových rámcu, ale i k efektivnímu generování zcela nových ethernetových rámců. Stačí na vstup přivést prázdné rámce, tedy jen značky začátků a konců, platná data pak lze postupně zapsat pomocí dílčích editorů, přičemž není nutné data posouvat.



Obr. 3.4: Zjednodušená struktura bloku pro posun částí rámce.

Aby bylo možné odebírat a přidávat protokoly do modifikovaného ethernetového rámce, musí tyto editory umět posouvat ostatní části rámců, jak již bylo psáno výše. Ale schopnost posouvání vyžaduje větší počet vstupů u multiplexorů v editoru, což vede k velkému navýšení spotřeby zdrojů. Pokud by všechny dílčí editory podporovaly tuto funkcionalitu, byl by celý obvod velmi náročný na zdroje FPGA.

Optimálnější řešení je na začátek obvodu přidat speciální blok, který bude posouvat části příchozích ethernetových rámců po sběrnici MFB tak, aby vznikla mezera pro přidání protokolu nebo byla odstraněna mezera po odebraném protokolu. Tuto architekturu znázorňuje blokové schéma na obrázku 3.5.

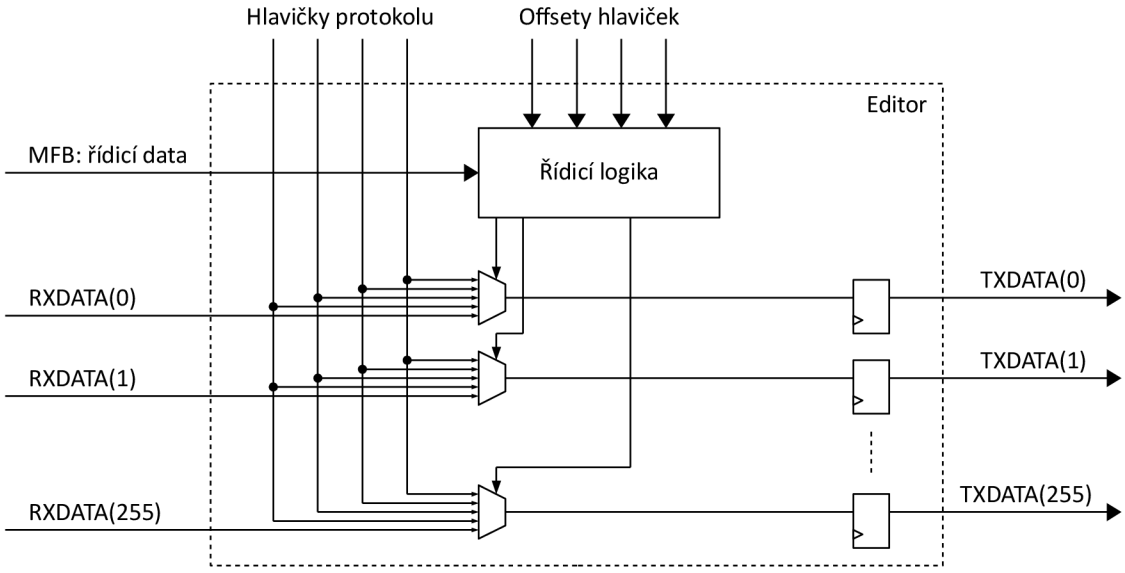


Obr. 3.5: Blokové schéma navrženého obvodu pro skládání rámců.

Blok pro posun dat bude obsahovat multiplexory s největším počtem vstupů. Při datové sběrnici o šířce 2048 bitů a granularitě 8 bitů musí mít tyto multiplexory 256 vstupů. Tyto multiplexory budou tvořit největší část spotřebovaných zdrojů pro implementaci celého obvodu. Pro snažší splnění časových podmínek může být

na konci tohoto bloku vložena řada registrů pro každý bit z datového slova sběrnice MFB. Následující bloky (editory) pak již jen budou editovat, respektive vkládat hlavičky protokolů na odpovídající místa v datovém slově.

Při datové šířce 2048 bitů (přenosová rychlost 400 Gb/s) může nastat situace, kdy bude potřeba vložit až čtyři hlavičky do jednoho datového slova. Pro splnění toho požadavku má editační blok přístup ke čtyřem hlavičkám daného protokolu. Řídicí logika určuje místo vložení hlavičky z její stanovené pozice (offset) a aktuálního stavu MFB sběrnice. Pro lepší plnění časových podmínek může být na konci editovacího bloku vložena řada registrů pro každý bit z datového slova sběrnice MFB. Zjednodušenou strukturu popsaného editovacího bloku pro přenosovou rychlost 400 Gb/s znázornujě obrázek 3.6.

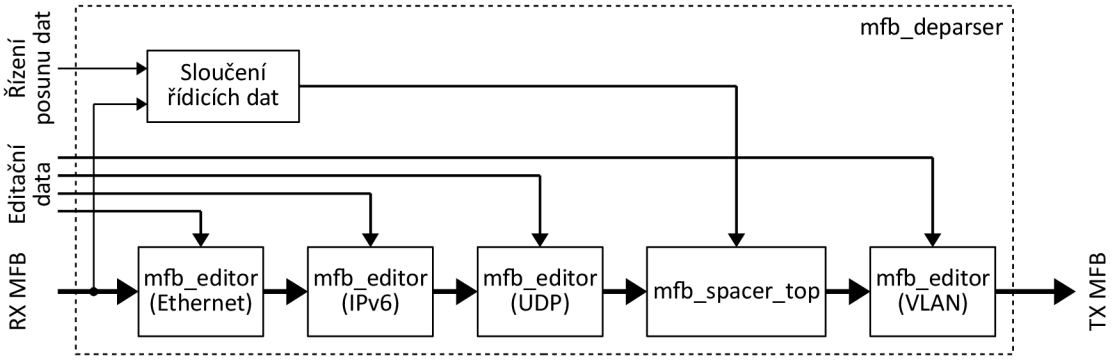


Obr. 3.6: Zjednodušené schéma navrženého editoru hlaviček.

Celý obvod bude využívat datovou sběrnici MFB, která je parametricky konfigurovatelná. Tyto parametry budou také využity pro snadné škálování celého obvodu pro přenosové rychlosti 100 Gb/s a 400 Gb/s. Stejně jako obvod pro analýzu ethernetových rámců (MFB HFE) tak bude i tento obvod pro skládání ethernetových rámců možné snadno použít při různých přenosových rychlostech bez nutných úprav navrženého obvodu.

3.3 Implementace obvodu

Navržený obvod je implementovaný jako komponenta mfb_deparser, interně označovaný jako MFB Deparser, tvoří zřetězené zapojení dílčích komponent. Obrázek 3.7 zobrazuje zjednodušené blokové schéma komponenty mfb_deparser. Ke vstupní datové sběrnici je připojena komponenta mfb_editor, která slouží k editaci hlavičky protokol Ethernet. Za ní je připojen editor protokolu IPv6 následovaný editorem protokolu UDP. Dále následuje komponenta mfb_spacer_top, která posouvá data v ethernetovém rámci tak, aby bylo možné přidat nebo odebrat VLAN hlavičku. Posledním dílčím blokem je editor hlavičky protokolu VLAN.

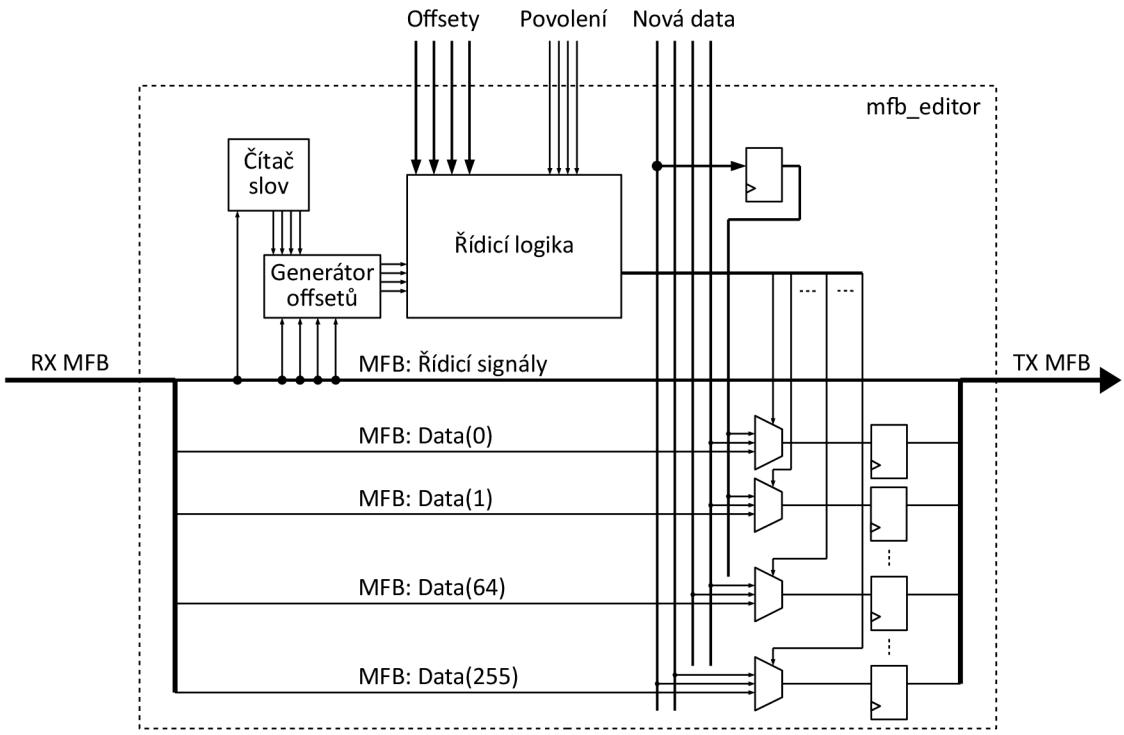


Obr. 3.7: Zjednodušené blokové schéma obvodu MFB Deparser.

Celý implementovaný obvod MFB Deparser v současné verzi tedy podporuje internetové protokoly Ethernet, VLAN, IPv6 a UDP. Informace obsažené v hlavičkách podporovaných protokolů lze upravovat a protokol VLAN lze zcela odebrat či přidat do ethernetového rámce. V případě potřeby je možné podporu protokolů rozšiřovat.

Jednou z dílčích částí obvodu MFB Deparser je editor hlaviček ethernetových protokolů. Tato komponenta je implementována jako obecný editační obvod označený názvem mfb_editor. Zjednodušené schéma obvodu mfb_editor je znázorněno na obrázku 3.8. Tato komponenta umožňuje přepsat/editovat určenou část ethernetového rámce. Délka přepisované části je parametricky volitelná. Nová data přichází po vedlejší sběrnicí spolu s offsetem a povolovacím signálem, který určuje, zda se má přepis dat opravdu provést.

Komponenta mfb_editor obsahuje čítač slov, který spolu s řídicími signály vstupní sběrnice MFB slouží k určení aktuálního offsetu ethernetových rámců na vstupní sběrnici MFB. Obvod také obsahuje multiplexory, které umožnují přiřadit na výstupní sběrnici MFB původní hodnoty ze vstupní sběrnice nebo hodnoty nové. Offset nových dat je porovnáván s aktuálním offsetem sběrnice MFB. V případě



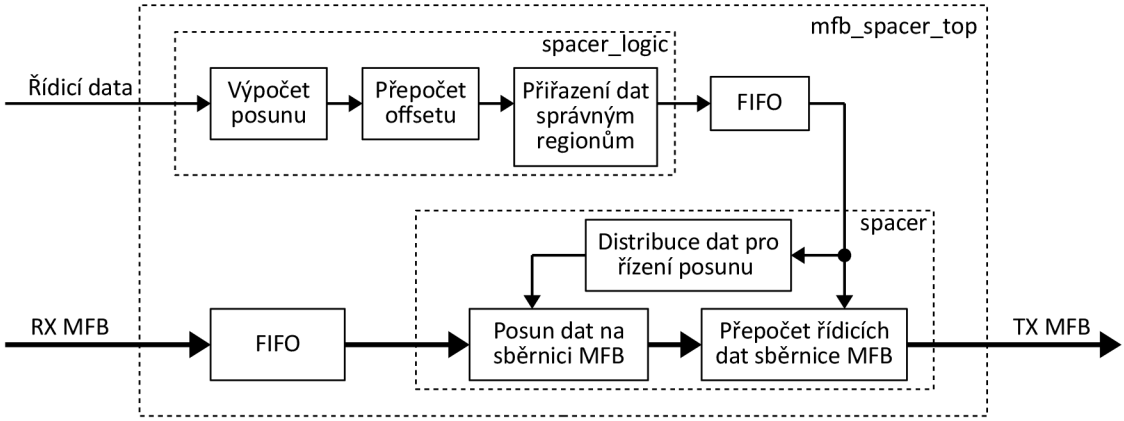
Obr. 3.8: Zjednodušené schéma implementovaného editoru.

shody porovnávaných offsetů a aktivního povolovacího signálu zajistí řídicí logika správné nastavení multiplexorů.

Komponenta je implementována s využitím datové sběrnice MFB a využívá možnosti její parametrické konfigurace. Pro zjednodušení řídicí logiky může v každém regionu sběrnice MFB začínat pouze jeden přepis/editace dat. Dále platí, že nová (přepisující) data musí přijít po vedlejší sběrnici ve stejném regionu, ve kterém bude jejich začátek i na výstupní sběrnici MFB. Maximální povolená délka přepisované části je rovna délce jednoho regionu sběrnice MFB.

Dalším důležitým obvodem je blok pro posun částí rámce, který je implementován jako komponenta mfb_spacer_top. Obrázek 3.9 zobrazuje blokové schéma komponenty mfb_spacer_top, která umožnuje posouvat částmi ethernetováho rámce tak, aby bylo možné odebrat nebo přidat určený internetový protokol. Komponenta mfb_spacer_top obsahuje paměti FIFO na datové i řídicí cestě a tvoří ji především dva hlavní bloky: řídicí (komponenta spacer_logic) a datový (komponenta spacer).

Řídicí komponenta spacer_logic obsahuje logiku, která postupně vypočítá posuny jednotlivých ethernetových rámců a přepočítá značky začátků a konců jednotlivých ethernetových rámců. Tyto řídicí údaje jsou přiřazeny ke správným regionům a odeslány do komponenty spacer. Implementovaná řídicí logika je tvořena soustavou sčítaček a multiplexorů, které jsou vhodně proloženy registry.



Obr. 3.9: Zjednodušené blokové schéma komponenty mfb_spacer_top.

Komponenta spacer realizuje samotné posuny dat (vytváření a odebírání mezer) na sběrnici MFB. Komponentu tvoří řada širokých multiplexorů, které vhodně posouvají data. Řídicí data z komponenty spacer_logic jsou vhodně rozdistribuovány mezi multiplexory a slouží jako jejich adresa pro výběr vstupu. Distribuce řídicích dat mezi multiplexory se odvíjí podle offsetu začátku posunu dat v datovém slově sběrnice MFB.

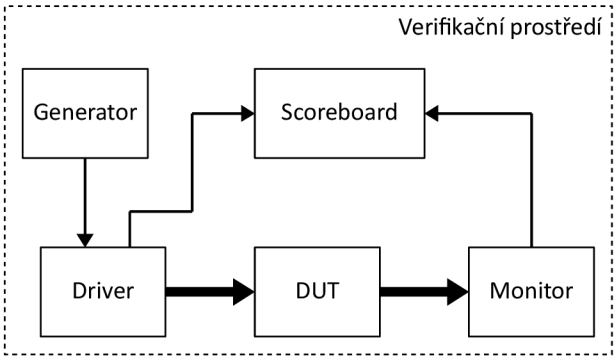
4 VYHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ

Tato kapitola popisuje testování fukčnosti a dosažené výsledky obou implementovaných obvodů. Výsledky implementovaného obvodů jsou zkoumány především z hlediska náročnosti obvodů na zdroje FPGA a odhad maximální frekvence po syntéze implementovaných obvodů.

4.1 Verifikace obvodů

Abychom ověřili, zda jsou implementované obvody opravdu funkční, je nutné provést nějaký kontrolní test. Často se pro ověření funkčnosti obvodů poživají simulace, ale vzhledem k velkému počtu kombinací vstupních dat je vhodnější použít funkční verifikaci, která je schopna ověřit funkčnost verifikovaného obvodu mnohem rychleji.

Implementované obvody byly nad rámec zadání testovány pomocí funkční verifikace napsané v jazyce System Verilog. Pro funkční verifikaci jsou často využívány metodiky OVM a UVM.[17][18] Tyto metodiky jsou poměrně robustní, a proto vývojový tým Liberouter používá vlastní verifikační prostředí znázorněné na obrázku 4.1. Použité verifikační prostředí je jednodušší, ale vychází ze struktury definované v OVM/UVM.



Obr. 4.1: Struktura verifikačního prostředí.

Použité verifikační prostředí se skládá z několika částí. Generátor generuje požadované, zpravidla náhodné, transakce, které mohou reprezentovat například ethernetové rámce. Driver tyto transakce odesílá pomocí zvolené sběrnice (například MFB) do verifikovaného obvodu, který je často označován zkratkou DUT (design under test). Driver transakci odešle také do Scoreboardu, kde se zpracuje podle verifikačního modelu testovaného obvodu a uloží se do tabulky. Monitor přijímá transakce od DUT na zvolené sběrnici a po přijetí je odešle do scoreboardu, kde jsou transakce

porovnány s transakcemi, které byly zpracovány verifikačním modelem. V případě, že se transakce neshodují, se verifikace zastaví.

V případě obvodu MFB HFE generuje verifikační prostředí platné ethernetové rámce s náhodně zvolenou sadou internetových protokolů. Generátor umožňuje nastavit maximální a minimální délku generovaných ethernetových rámců a také jejich celkový počet. Vygenerované ethernetové rámce jsou odesílány do verifikovaného obvodu, který je zpracuje a výsledky odešle zpět do verifikačního prostředí. Přijaté výsledky z obvodu MFB HFE jsou následně porovnány s výsledky verifikačního modelu, které jsou považovány za referenční. Pokud se výsledky neshodují, verifikace se zastaví a upozorní na nalezenou chybu. Díky tomu je jednodušší nalezenou chybu identifikovat a opravit. Finální verze implementovaného obvodu při testování funkční verifikací správně analyzovala přes deset miliónů Ethernetových rámců.

Obdobný postup je aplikován také při verifikaci implementovaného obvodu MFB Deparser. Opět jsou generována platná vstupní data, která v tomto případě reprezentují hlavičky vybraných internetových protokolů doplněné o řídící data. Tato řídicí data určují, zda a kde mají být tyto hlavičky použity. Výstupní data verifikovaného obvodu tvoří platné ethernetové rámce na datové sběrnici MFB. Tyto výstupní ethernetové rámce jsou porovnávány s referenčními daty, které byly získány z verifikačního modelu.

4.2 Vyhodnocení obvodu MFB HFE

Tabulka 4.1 ukazuje náročnost implementovaného obvodu MFB HFE na zdroje FPGA a odhad maximální dosažitelné frekvence po syntéze pro vybrané konfigurace obvodu, respektive vstupní sběrnice MFB. Tyto konfigurace odpovídají přenosovým rychlostem 100, 200, 400 a 800 Gb/s při frekvenci 200 MHz. Jako cílové FPGA bylo zvoleno FPGA z řady Virtex 7 (xc7vh580thcg1931-2) od firmy Xilinx.

Konfigurace MFB sběrnice	**Spotřeba LUT**	**Spotřeba LUTRAM**	**Spotřeba FF**	**Maximální frekvence**
MFB(1,8,8,8)	8147 (2,3 %)	1022 (0,7 %)	15018 (2,1 %)	267,3 MHz
MFB(2,8,8,8)	18138 (5,0 %)	2174 (1,5 %)	24858 (3,4 %)	239,7 MHz
MFB(4,8,8,8)	38967 (10,7 %)	4600 (3,3 %)	43628 (6,0 %)	203,5 MHz
MFB(8,8,8,8)	87935 (24,2 %)	9700 (6,9 %)	81534 (11,2 %)	170,7 MHz

Tab. 4.1: MFB HFE - Výsledky syntézy pro FPGA z řady Virtex 7.

Z tabulky je vidět, že spotřeba roste více než dvojnásobně při zdvojnásobení počtu regionů sběrnice MFB. Spotřeba LUT udává celkový počet využitých buněk

LUT a spotřeba LUTRAM udává, kolik z celkových LUT je využito v režimu distribuční paměti. Hodnoty v závorkách uvadí procentuální podíl z dostupných zdrojů v uvedeném FPGA. Obvod MFB HFE v konfiguracích MFB(1,8,8,8), MFB(2,8,8,8) a MFB(4,8,8,8) spotřeboval řádově jednotky procent dostupných zdrojů, díky tomu zbývá dostatek zdrojů pro další doplňující obvody. Maximalní dosažitelná frekvence klesá s počtem regionů. Obvod MFB HFE v konfiguraci MFB(8,8,8,8) spotřebuje téměř čtvrtinu LUT dostupných v tomto FPGA a nedosáhne očekávané frekvence 200 MHz, a tudíž nedosáhne ani očekávané propustnosti 800 Gb/s. Kritická cesta prochází řídicí logikou, která zajišťuje správné třídění extrahovaných dat. Pro splnění očekávaných vlastnosti obvodu v této konfiguraci je nutné použít lepší FPGA čip. Blokové paměti (BRAM) nejsou v tomto obvodu využívány, proto není jejich spotřeba uvedena v tabulkách. Syntéza obvodu byla prováděna nástrojem Xilinx Vivado 2016.3 s výchozím nastavením.

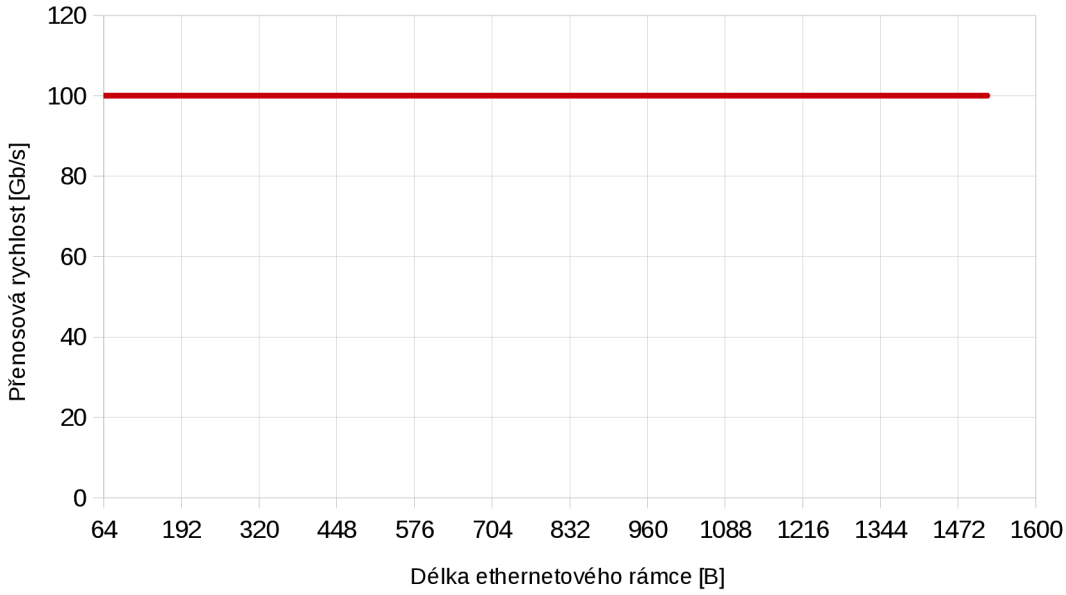
Konfigurace MFB sběrnice	**Spotřeba LUT**	**Spotřeba LUTRAM**	**Spotřeba FF**	**Maximální frekvence**
MFB(1,8,8,8)	7625 (1,0 %)	922 (0,2 %)	14991 (1,0 %)	484,4 MHz
MFB(2,8,8,8)	17565 (2,2 %)	2090 (0,5 %)	24915 (1,6 %)	402,0 MHz
MFB(4,8,8,8)	37359 (4,7 %)	4200 (1,1 %)	43751 (2,8 %)	336,4 MHz
MFB(8,8,8,8)	85875 (10,9 %)	9344 (2,4 %)	81694 (5,2 %)	251,1 MHz

Tab. 4.2: MFB HFE - Výsledky syntézy pro FPGA z řady UltraScale+.

Na trh se již dostává novější řada FPGA UltraScale+ od firmy Xilinx.[19] Tato řada FPGA bude v budoucnu častěji využívána, a proto byla syntéza obvodu ve stejných konfiguraci provedena nad rámec zadání i na FPGA z řady UltraScale+ (xcvu7p-flvb2104-2-i-EVAL). Náročnost obvodu na zdroje FPGA a odhad maximální dosažitelné frekvence syntéz pro vybrané konfigurace obvodu na FPGA UltraScale+ jsou uvedeny v tabulce 4.2. Při srovnání výsledků syntézy na Virtex 7 a UltraScale+ jsou jasně patrné výhody novější řady FPGA UltraScale+. Nové moderní FPGA umožňuje dosáhnout téměř dvojnásobných frekvencí a mírně lepší spotřeby LUT a LUTRAM. Toto moderní FPGA umožnilo splnit očekávané vlastnosti obvodu MFB HFE i v konfiguraci MFB(8,8,8,8), která při frekvenci 200 MHz odpovídá přenosové rychlosti 800 Gb/s.

V konfiguraci odpovídající přenosové rychlosti 100 Gb/s byl obvod také otestován přímo v FPGA Xilinx Virtex 7 jako součást firmwaru síťové karty COMBO.[16] Test byl prováděn s rámci o délce 64 bytů až 1522 bytů. Ethernetové rámce byly generovány profesionálním nástrojem Spirent, který spolehlivě zvládne generovat ethernetové rámce na plné rychlosti 100 Gb/s. Obrázek 4.2 zobrazuje graf závislosti

dosažené přenosové rychlosti na délce ethernetových rámců. Z grafu je patrné, že obvod dosahuje plné propustnosti při všech testovaných délkách rámců.



Obr. 4.2: MFB HFE - Propustnost v závislosti na délce ethernetových rámců.

Nad rámec zadání byl obvod MFB HFE také úspěšně otestován na živé síti CESNET. Tato síť propojuje největší univerzitní města v ČR a poskytuje vysoké přenosové rychlosti až 100 Gb/s. Obvod MFB HFE byl na této síti nasazen jako součást firmwaru pro monitorování vysokorychlostních síťových linek na kartě COMBO.[16] Test na živé síti probíhal bez přestání po dobu pěti dnů. Během celého testu fungoval obvod bezchybně. To dokazuje přínos funkční verifikace, která umožnila rychle identifikovat a odstranit chyby brzy po dokončení implementace.

4.3 Vyhodnocení obvodu MFB Deparser

V této části práce je popsáno vyhodnocení implementovaného obvodu MFB Deparser. Tento implementovaný obvod byl analyzován z hlediska náročnosti implementovaného obvodu na zdroje FPGA a odhadu maximální dosažitelné frekvence po syntéze pro vybrané konfigurace obvodu, respektive konfigurace sběrnice MFB. Dva cílové FPGA čipy byly opět zvoleny z řady Virtex 7 (xc7vh580thcg1931-2) a UltraScale+ (xcvu7p-flvb2104-2-i-EVAL) od firmy Xilinx. Syntézy obvodu byly opět prováděny nástrojem Xilinx Vivado 2016.3 s výchozím nastavením.

Konfigurace MFB sběrnice	Spotřeba LUT	Spotřeba FF	Spotřeba BRAM	Maximální frekvence
MFB(1,8,8,8)	10032 (2,8 %)	4550 (0,6 %)	8 (0,9 %)	234,1 MHz
MFB(2,8,8,8)	28188 (7,8 %)	9052 (1,2 %)	15 (1,6 %)	266,5 MHz
MFB(4,8,8,8)	93813 (25,9 %)	19433 (2,7 %)	30 (3,2 %)	232,6 MHz
MFB(8,8,8,8)	330641 (91,1 %)	46604 (6,4 %)	59 (6,2 %)	206,7 MHz

Tab. 4.3: MFB Deparser - Výsledky syntézy pro FPGA z řady Virtex 7.

Konfigurace MFB sběrnice	Spotřeba LUT	Spotřeba FF	Spotřeba BRAM	Maximální frekvence
MFB(1,8,8,8)	10024 (1,3 %)	4550 (0,3 %)	8 (0,6 %)	415,6 MHz
MFB(2,8,8,8)	28088 (3,6 %)	9052 (0,6 %)	15 (1,0 %)	431,2 MHz
MFB(4,8,8,8)	92853 (11,8 %)	19430 (1,2 %)	30 (2,1 %)	367,5 MHz
MFB(8,8,8,8)	329846 (41,9 %)	46553 (3,0 %)	59 (4,1 %)	337,9 MHz

Tab. 4.4: MFB Deparser - Výsledky syntézy pro FPGA z řady UltraScale+.

Tabulka 4.3 ukazuje výsledky syntézy implementovaného obvodu pro FPGA z řady Virtex 7. Tabulka 4.4 ukazuje výsledky syntézy implementovaného obvodu pro FPGA z řady UltraScale+. Z obou tabulek je vidět, že spotřeba zdrojů výrazně roste s počtem regionů sběrnice MFB. Oproti obvodu MFB HFE roste spotřeba LUT výrazně rychleji. Obvod v konfiguraci MFB(4,8,8,8) spotřebuje více než čtvrtinu LUT v uvedeném FPGA z řady Virtex 7. U konfigurace MFB(8,8,8,8), která odpovídá datové šířce 4096 bitů, vzroste spotřeba LUT ještě mnohem více. Spotřeba přesáhne 90 % všech LUT v FPGA z řady Virtex 7. Největší podíl na této velké spotřebě LUT má komponenta spacer, která musí umět posouvat data alespoň po 2 bytech, což odpovídá nejhoršímu případu zarovnání většiny internetových protokolů. Ve výsledcích pro FPGA z řady Virtex 7 se maximální frekvence drží nad hodnotou 200 MHz. Ve výsledcích pro FPGA z řady UltraScale+ si lze povšimnout opět výrazně lepších maximálních frekvencí a drobných rozdílů ve spotřebě zdrojů FPGA. Novější FPGA z řady UltraScale+ dosahuje vyšších frekvencí především díky modernější 16 nm FinFET výrobní technologii.[19]

5 MOŽNOSTI VYUŽITÍ JAZYKA P4

Vždy, když se změní podporovaná sada internetových protokolů, je nutné příslušně upravit implementovaný obvod pro analýzu nebo skládání ethernetových rámců. Například v případě obvodu MFB HFE to znamená naimplementovat dílčí analyzátory pro nově zvolené internetové protokoly a příslušně upravit zřetězené zapojení dílčích analyzátorů. Takové upravy jsou často časově náročné. Jazyk P4 může být jedna možnost, jak takové úpravy zautomatizovat, a tím i výrazně zkrátit potřebný čas.

Jazyk P4 (Programming Protocol-independent Packet Processors) slouží pro programování síťových procesorů nezávysle na síťových (internetových) protokolech. Jazyk P4 se řadí mezi vysokoúrovňové jazyky a jeho úkolem je volné definování zpracování paketů (například ethernetových rámců) v síťovém zařízení. Program v jazyce P4 lze interně zkompilovat do reprezentace pro konkrétní síťové zařízení. To umožňuje využít libovolnou vnitřní architekturu síťového zařízení (například CPU, FPGA, specializovaný síťový procesor a další). V současnosti existují dvě verze tohoto jazyka: P4 14 a novější P4 16 ve fázi návrhu.[20][21] V této práci je použita široce podporovaná verze P4 14.

Výpis 5.1: Ukázka popisu hlavičky protokolu Ethernet v jazyce P4.

```
1 header ethernet {
2     fields {
3         dst_mac  : 48; // šířka v bitech
4         src_mac  : 48;
5         eth_type : 16;
6     }
7 }
```

Jazyk P4 může posloužit k automatizovanému generování implementovaných obvodů s libovolnou podporou internetových protokolů dle konkrétního P4 programu. V jazyce P4 lze snadno popsat libovolný internetový protokol. Výpis 5.1 zobrazuje popis hlavičky protokolu Ethernet v jazyce P4. Výpis 5.2 zobrazuje popis hlavičky protokolu IPv6. Jak lze z uvedených výpisů vidět, syntaxe pro popis internetových protokolů je poměrně jednoduchá. Je nutné definovat jednotlivá políčka hlaviček protokolů ve správném pořadí za sebou a jejich délku v bitech.

Jazyk P4 přímo disponuje prostředky pro snadný popis extrakce hlaviček ze síťových paketů a také pro jejich skládání. Velice snadno tak lze v jazyce P4 popsat obvody implementované v této práci. Výpis 5.3 zobrazuje nejjednodušší P4 program pro popis extrakce ethernetové hlavičky. Příkaz *extract* určuje, co se bude extrahovat, v tomto případě hlavička protokolu Ethernet. Příkaz *switch* zde popisuje výběr

následujícího internetového protokolu (zabaleného uvnitř protokolu Ethernet) z uvedených možností na základě hodnoty políčka *eth_type*.

Výpis 5.2: Ukázka popisu hlavičky protokolu IPv6 v jazyce P4.

```
header ipv6 {
    fields {
        version       : 4;
        traffic_class : 8;
        flow_label    : 20;
        payload_lng   : 16;
        next_header   : 8;
        hop_limit     : 8;
        src_addr      : 128;
        dst_addr      : 128;
    }
}
```

Kompilátor P4 programu musí být dobře napsán tak, aby dokázal popis v P4 programu převést do konkrétního VHDL popisu. Aby nebylo nutné generovat celý VHDL popis implementovaného obvodu, lze využít vnitřní bloky implementované v této práci, které jsou protokolově nezávislé. Kompilátor analyzuje P4 program, a tak zjistí, jaké protokoly musí obvod z ethernetových rámců umět analyzovat a také pořadí analýzy těchto protokolů. Na základě tohoto zjištění kompilátor vygeneruje VHDL popis analyzátorů zvolených protokolů. Kompilátor připravené dílčí bloky správně zapojí a vygeneruje VHDL popis tohoto zapojení.[22]

Výpis 5.3: Ukázka popisu extrakce ethernetové hlavičky v jazyce P4.

```
header ethernet eth;
parser ethernet {
    extract(eth);
    switch(eth.eth_type) {
        case 0x8100: vlan;
        case 0x9100: vlan;
        case 0x9200: vlan;
        case 0x9300: vlan;
        case 0x0800: ipv4;
        case 0x86DD: ipv6;
    }
}
```

Pokud bude potřeba popsaný obvod pro analýzu ethernetových rámců rozšířit o nový síťový protokol, stačí příslušně upravit P4 program a ten znovu zkompilovat. Kompilátor znovu vygeneruje potřebný VHDL popis dílčích analyzátorů a vygeneruje zapojení vnitřních komponent.

Nad rámec zadání byl připraven kompilátor P4 programů pro automatizované generování implementovaného obvodu MFB HFE. Tento kompilátor byl napsán v jazyce Python. Po několika experimentech se podařilo vygenerovat VHDL popis obvodu MFB HFE s podporu protokolů odpovídající připravenému P4 programu.

Automatizované generování obvodů přináší výrazné zrychlení nutných úprav při rozšiřování sady podporovaných protokolů, ale generované části obvodů nemusí být tak dobře optimalizované jako v případě ruční implementace.[22]

6 ZÁVĚR

Diplomová práce je zaměřena na problematiku analýzy a skládání ethernetových rámců. Výsledkem této práce je návrh a implementace obvodů pro analýzu a skládání ethernetových rámců. Tyto obvody jsou navrženy tak, aby je bylo možné použít na sítích Ethernet o přenosové rychlosti 100 Gb/s a 400 Gb/s.

V první části diplomové práce jsem nastudoval problematiku sítě Ethernet a vybraných internetových protokolů, protože znalost internetových protokolů je nutná pro správnou a efektivní analýzů ethernetových rámců. Dále jsem nastudoval základní teorii k obvodům FPGA, které budou použity pro implementaci navržených obvodů. Také jsem popsal datovou sběrnicí MFB, která je využívána pro přenos ethernetových rámců.

Jedním z úkolů diplomové práce bylo návrhnout a implementovat obvody pro analýzů a skládání ethernetových rámců. Obvody jsem navrhl s parametricky volitelnou přenosouvou rychlostí 100 Gb/s a více. Limitem může být pouze nedostatek zdrojů v FPGA nebo nesplnění časových podmínek. Obvody byly navrženy modulárně, aby bylo snadné rozšiřovat podporu internetových protokolů. V navazující části detailněji popisuji samotnou implementaci navržených obvodů se zaměřením na zajímavé implementační části.

V následující části práce rozebírám vysledky implementovaných obvodů především z pohledu náročnosti na zdroje FPGA a odhadu maximální dosažitelné frekvence. Spotřeba zdrojů roste s počtem regionů datové sběrnice MFB. Přesto u konfigurací implementovaných obvodů pro přenosové rychlosti 100 Gb/s až 400 Gb/s není zjištěná spotřeba zdrojů pro moderních FPGA limitující. Obvody se podařilo implementovat tak, že splňují časové podmínky pro dosažení frekvence 200 MHz po syntéze na vhodný FPGA čip, tudíž splňují i očekávané přenosové rychlosti. Implentovaný obvod MFB HFE byl nad rámec zadání otestován na živé síti CESNET jako součást firmwaru síťové karty COMBO.

V poslední části práce diskutuji o možnosti generování implementovaných obvodů pomocí vysokoúrovňového jazyka P4. Automatizované generování obvodů přináší výrazné zrychlení nutných úprav při rozšiřování sady podporovaných protokolů, ale generované části obvodů nemusí být tak dobře optimalizované jako v případě ruční implementace. Nad rámec zadání jsem experimentálně vyzkoušel generování implementovaného obvodu MFB HFE s využitím jazyka P4.

Výsledky této práce umožní zpracovávat internetový provoz při velmi vysokých rychlostech i s využitím FPGA čipů. Implementované obvody jsou již nyní použity v reálných síťových kartách s FPGA čipy. Další práce v této oblasti může navázat optimalizací implementovaných obvodů a pokračování využití automatizovaného generování pomocí jazyka P4.

LITERATURA

[1] IEEE P802.3ba 40 Gb/s and 100 Gb/s Ethernet Task Force. *IEEE 802.3 ETHERNET WORKING GROUP* [online]. [cit. 2016-11-01]. Dostupné z: `http://www.ieee802.org/3/ba/`

[2] IEEE P802.3bs 200 Gb/s and 400 Gb/s Ethernet Task Force. *IEEE 802.3 ETHERNET WORKING GROUP* [online]. [cit. 2016-11-01]. Dostupné z: `http://www.ieee802.org/3/bs/`

[3] POLČÁK, Libor. *Hardwarová akcelerace extrakce položek z hlaviček paketů* [online]. Brno, 2010 [cit. 2016-11-27]. Dostupné z: `https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=117458`. Diplomová práce. FIT VUT v Brně.

[4] PUŠ, Viktor, Lukáš KEKELY a Jan KOŘENEK. *Design Methodology of Configurable High Performance Packet Parser for FPGA*. Brno, 2014.

[5] BENÁČEK, Pavel. *Generation of High-Speed Network Device from High-Level Description*. Praha, 2016 [cit. 2017-04-20]. Dostupné z: `http://www.fit.cvut.cz/sites/default/files/PhDThesis-Benacek.pdf`. Disertační práce. České vysoké učení technické v Praze.

[6] PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL*. 1. vyd. Praha, ČR: BEN, 2006, 349 s. ISBN 80-730-0198-5.

[7] ISO 7498. *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. Second edition. Switzerland, 1994.

[8] IEEE 802.3 ETHERNET WORKING GROUP. *IEEE 802.3 ETHERNET WORKING GROUP* [online]. [cit. 2016-11-01]. Dostupné z: `http://www.ieee802.org/3/`

[9] 802.1Q - Virtual LANs. *Institute of Electrical and Electronics Engineers* [online]. [cit. 2016-11-01]. Dostupné z: `http://www.ieee802.org/1/pages/802.1Q.html`

[10] ROSEN, E. et al. *RFC 3032: MPLS Label Stack Encoding* [online]. 2001 [cit. 2016-11-30]. Dostupné z: `https://tools.ietf.org/pdf/rfc3032.pdf`

[11] *RFC 791: Internet Protocol, Specification* [online]. California, USA, 1981 [cit. 2016-11-01]. Dostupné z: `https://tools.ietf.org/pdf/rfc791.pdf`

[12] DEERING, S. et al. *RFC 2460: Internet Protocol, Version 6 (IPv6) Specification* [online]. California, USA, 1998 [cit. 2016-11-01]. Dostupné z: https://tools.ietf.org/pdf/rfc2460.pdf

[13] *RFC 793: Transmission Control Protocol, Specification* [online]. California, USA, 1981 [cit. 2016-11-01]. Dostupné z: https://tools.ietf.org/pdf/rfc793.pdf

[14] POSTEL, J. *RFC 768: User Datagram Protocol* [online]. California, USA, 1980 [cit. 2016-11-01]. Dostupné z: https://tools.ietf.org/pdf/rfc768.pdf

[15] XILINX. *UG474: 7 Series FPGAs Configurable Logic Block* [online]. San Jose, USA, 2016 [cit. 2016-11-01]. Dostupné z: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[16] CESNET. *Liberouter* [online]. Brno, CZ, 2016 [cit. 2016-11-27]. Dostupné z: https://www.liberouter.org

[17] *OVM User Guide: Version 2.1.2* [online]. 2011 [cit. 2017-05-14]. Dostupné také z: http://www.specman-verification.com/source_bank/ovm-2.1.2/ovm-2.1.2/OVM_UserGuide.pdf

[18] ACCELLERA. *Universal Verification Methodology: 1.2 User's Guide* [online]. 2015 [cit. 2017-05-14]. Dostupné také z: http://www.accellera.org/images//downloads/standards/uvm/uvm_users_guide_1.2.pdf

[19] XILINX. *Virtex UltraScale+ FPGAs: Product brief* [online]. San Jose, USA, 2016 [cit. 2017-05-14]. Dostupné také z: https://www.xilinx.com/support/documentation/product-briefs/virtex-ultrascale-plus-product-brief.pdf

[20] THE P4 LANGUAGE CONSORTIUM. *The P4 Language Specification: Version 1.0.3* [online]. 2016 [cit. 2017-05-14]. Dostupné také z: http://p4.org/wp-content/uploads/2016/11/p4-spec-latest.pdf

[21] THE P4 LANGUAGE CONSORTIUM. *P4 16 Language Specification: Draft* [online]. 2016 [cit. 2017-05-14]. Dostupné také z: http://p4.org/wp-content/uploads/2016/12/P4_16-prerelease-Dec_16.pdf

[22] BENÁČEK, Pavel, Viktor PUŠ a Hana KUBÁTOVÁ. *P4-to-VHDL: Automatic Generation of 100 Gbps Packet Parsers* [online]. 2016 [cit. 2017-05-14]. Dostupné z: https://www.liberouter.org/wp-content/uploads/2016/05/p4.pdf

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ASIC	Application Specific Integrated Circuit
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DHCP	Dynamic Host Configuration Protocol
DSP	Digital Signal Processing
DUT	Design Under Test
EEPROM	Electrically Erasable Programmable Read-Only Memory
FF	Flip-Flop
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
HDL	Hardware Description Language
HFE	Header Field Extractor
ICMP	Internet Control Message Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LUT	LookUp Table
MAC	Media Access Control
MFB	Multi Frame Bus
MPLS	Multiprotocol Label Switching
OVM	Open Verification Methodology
P4	Programming Protocol-independent Packet Processors
PLL	Phase-Locked Loop
SCTP	Stream Control Transmission Protocol
SRAM	Static Random Access Memory
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UVM	Universal Verification Methodology

A OBSAH PŘILOŽENÉHO CD

Tato kapitola popisuje obsah přiloženého CD. Součástí obsahu CD jsou především zdrojové kódy implementovaných obvodů napsané v jazyce VHDL a elektronická verze diplomové práce. Dále CD obsahuje externí zdrojové kódy a skripty, které byly použity v této práci. Popis obsahu CD je též uveden v souboru readme.txt v kořenovém adresáři přiloženého CD.

```
/ ..... kořenový adresář přiloženého CD
├── latex ..... latex soubory diplomové práce
├── source ..... adresář s přiloženými zdrojovými soubory
│   ├── build ..... externí skripty překladového systému
│   ├── deparser ..... zdrojové soubory obvodu MFB Deparser
│   │   ├── comp ..... adresář obsahující dílčí komponenty
│   │   ├── synth ..... adresář obsahující skripty pro syntézu
│   │   ├── ver ..... adresář obsahující skripty pro verifikaci
│   │   ├── mfb_deparser.vhd
│   │   └── mfb_deparser_pkg.vhd
│   ├── external ..... využité externí zdrojové soubory
│   ├── hfe ..... zdrojové soubory obvodu MFB HFE
│   │   ├── comp ..... adresář obsahující dílčí komponenty
│   │   ├── synth ..... adresář obsahující skripty pro syntézu
│   │   ├── ver ..... adresář obsahující skripty pro verifikaci
│   │   ├── hfe_full.vhd
│   │   ├── hfe_top.vhd
│   │   └── hfe_top_pkg.vhd
│   └── p4 ..... zdrojové soubory kompilátoru P4
│       ├── generated_files ..... vygenerované VHDL soubory
│       ├── hfe_compiler ..... P4 kompilátor obvodu MFB HFE
│       ├── headers.p4
│       └── parser.p4
├── diplomova_prace_xcabal05.pdf ..... elektronická verze diplomové práce
└── readme.txt ..... soubor s popisem obsahu CD
```