FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Hardwarové zrcadlo paketů
Student:	Bc. Karel Hynek
Vedoucí:	Dr. Ing. Sven Ubik
Studijní program:	Informatika
Studijní obor:	Návrh a programování vestavných systémů
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Analyzujte existující softwarová řešení zrcadla paketů, které odesílá přijaté síťové pakety zpět na IP adresu odesílatele nebo zadanou adresu. Navrhněte architekturu modulů pro hardwarové řešení zrcadla paketů v obvodu FPGA. Zrcadlo musí umět přijímat a znovu vysílat zpět síťové pakety určené pomocí zadaných hodnot v UDP hlavičce přes 1G Ethernet až do plné rychlosti rozhraní. Implementujte zrcadlo v jazyce VHDL. Ověřte jeho činnost v simulaci a na FPGA desce poskytnuté zadavatelem. Doplňte zrcadlo softwarovou částí umožňující ruční a automatickou konfiguraci pomocí DHCP a monitorování paketových statistik.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Hardwarové zrcadlo paketů

Bc. Karel Hynek

Katedra číslicového návrhu
Vedoucí práce: Dr. Ing. Sven Ubik

6. května 2019

Poděkování

V prvé řadě bych na tomto místě rád poděkoval Dr. Ing. Svenu Ubikovi za množství podnětů a dobrých rad, které značným způsobem přispěly ke zkvalitnění této práce. Dále bych rád poděkoval své přítelkyni Kateřině za jazykovou korekturu a pomoc s úpravou obrázků a Bc. Tomáši Benešovi za morální podporu. A v neposlední řadě děkuji své rodině za podporu během celé doby mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 6. května 2019 .....

Odkaz na tuto práci

Hynek, Karel. *Hardwarové zrcadlo paketů*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Diplomová práce se zabývá návrhem a realizací zrcadla paketů určeného k testování 1Gbps Ethernetových sítí. Zařízení dokáže přeposílat Ethernetové IP pakety až do plné rychlosti rozhraní. Nad datovým tokem je taktéž prováděna analýza variace síťového zpoždění paketů.

Cíle bylo dosaženo pomocí specializovaných číslicových jednotek implementovaných v obvodu FPGA. Práce se v prvé řadě zabývá číslicovým návrhem hardwaru, dále řeší návrh a realizaci softwarového vybavení určeného k ovládání vytvořených jednotek a integraci již existujících knihoven.

Zrcadlo paketů bylo implementováno a jeho funkčnost byla ověřena na dvou FPGA obvodech od společnosti Xilinx. Zařízení slouží ve sdružení CESNET k testování Ethernetových sítích určených pro přenos videa s nízkou latencí.

Klíčová slova Počítačová síť, Zrcadlo, FPGA, MicroBlaze, Linux, Jitter, Testování

Abstract

The thesis presents the design and construction of a packet reflector for 1Gbps Ethernet networks. The device can forward Ethernet IP packets at full speed of the interface. The device is also capable of packet delay variation analysis of incoming packet stream.

The thesis concerns the digital design implemented in an FPGA circuit and it also deals with the implementation of software used for controlling designed units. The goal of this thesis has been accomplished by these units.

The packet reflector was successfully tested and implemented in two Xilinx FPGA chips. The device is already used in CESNET z.s.p.o. for testing Ethernet networks used for low-latency video transmissions.

Keywords Network, Ethernet, Mirror, FPGA, MicroBlaze, Linux, IPDV, Testing

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V posledních letech se celosvětově zvyšuje poptávka po multimediálních datech z on-line služeb, jako je Netflix a YouTube, které generují více než 50 % veškerého síťového provozu v Severní Americe [1].

Přenos multimediálních dat je citlivý na kvalitu připojení. Z toho důvodu se používají různé druhy vyrovnávacích pamětí, které zakrývají nedostatky sítě. Tyto paměti ovšem zvyšují zpoždění propagace signálu k přijímači.Velikost vyrovnávacích pamětí roste s kvalitou streamovaného videa. I když využijeme výkonné kompresní algoritmy, jako je H.264/H.265, může být celkové zpoždění dat v řádu sekund. U jednosměrných přenosů, právě jako Youtube a Netflix, není ani takto vysoká latence příliš velký problém.

U obousměrného přenosu, jako jsou videohovory nebo vzdálené ovládání strojů, je ovšem nezbytné komunikovat v reálném čase. Zachování co nejnižší latence je v těchto případech velice důležité, a to i za předpokladu, že se jedná o přenosy napříč jednotlivými kontinenty.

Zájmové sdružení právnických osob CESNET se již několik let zabývá přenosy videa s nízkou latencí pomocí jimi vyvinuté platformy MVTP. Jedná se o hardware s FPGA obvody, jež dovoluje vysokou flexibilitu konfigurace při malých rozměrech.

Vyvinutý aparát má širokou škálu použití, jak ve videohovorech, tak v tvůrčí práci. Hlavním limitem, který brání v jeho použití je ale kvalita síťového připojení. Z toho důvodu je nutné ověřovat kvalitu spojení mezi oběma koncovými body realizovaného přenosu. Toto testování stojí poměrně velké finanční prostředky vynaložené na cestovní náklady a čas vysoce kvalifikovaných pracovníků.

Aby nebylo nutné vysílat dva pracovníky na nákladnou služební cestu, bylo rozhodnuto vyvinout malé zařízení, které je možné odeslat poštou, a které dokáže přeposílat pakety zpět odesílateli. Kvalitu spojení je tedy možné otestovat pouze na jednom konci linky, což šetří čas i prostředky.

Kapitola 1

Představení práce

1.1 Cíl diplomové práce

Cílem práce je vytvoření síťového zařízení určeného k testování 1Gbps Ethernetových sítí nazvané Hardwarové zrcadlo paketů. Jak již název napovídá, aparát dokáže zrcadlit pakety k odesílateli, nebo je přeposílat na jinou adresu v plné rychlosti 1Gbps Ethernetu. Jako datový tok se předpokládá video přenos z platformy MVTP, který vyhovuje standardu *SMPTE ST 2110-21:201*. Nad těmito daty je vytvořený aparát schopný analyzovat variaci síťového zpoždění paktů (jitter) s vysokou přesností. Softwarové vybavení podporuje ruční i automatickou konfiguraci Ethernetového rozhraní, a zároveň je možné provádět vzdálenou administraci.

Pro splnění cíle diplomové práce byla na počátku definována následující sada požadavků, které musí zařízení splňovat:

- Přeposílání UDP paketů generovaných platformou MVTP do uživatelem definovaného cíle plnou rychlostí 1Gbps rozhraní
- Automatická konfigurace síťového rozhraní
- Vzdálení administrace celého zařízení
- Zjednodušená GUI administrace paketového zrcadla a měření variace síťového zpoždění paketů
- Implementovat zařízení s jednotkou zrcadla paketů na dvou vývojových deskách:
  - Numato Mimas A7 - Artix 7 FPGA
  - Avnet AES-KU040-DB-G - Kintex UltraScale FPGA
- Měření variace síťového zpoždění na proudu dat generovaných platformou MVTP na desce Avnet AES-KU040-DB-G

1.2 Struktura práce

Kapitola 1 popisuje hlavní cíle práce

Kapitola 2 obsahuje nezbytné informace, které musel autor nastudovat, aby byl schopný kvalifikovaně a správně navrhnout celé zařízení.

Kapitola 3 se zabývá identifikací hlavních problémů návrhu a nastiňuje možná řešení.

Kapitola 4 se zabývá vlastním postupem návrhu a výslednou realizací.

Kapitola 5 obsahuje popis metod, které sloužily k testování a ověření funkčnosti zařízení jako celku.

Kapitola 6 popisuje měření výkonu implementovaných jednotek.

Kapitola 2

Rešeršní část

V této kapitole jsou popsány zásadní informace, bez kterých by nebylo možné navrhnout celé zařízení. Nejedná se v žádném případě o veškeré znalosti, které si musel autor osvojit, neboť ke splnění zadání bylo třeba obsáhnout velké množství materiálů a získané informace značně převyšují rozsah diplomové práce.

2.1 Komunikační rozhraní Ethernet

Počátek technologie Ethernet datujeme v 70. letech ve vývojových laboratořích Xerox Palo Alto Research Center. Konsorcium firem DEC, Intex a Xerox je následně v roce 1980 standardizovalo pod jménem DIX. V témže roce začala práce na dnes známém standardu IEEE 802.3, který byl publikován v roce 1985 [2]. Standard IEEE 802.3 celým názvem *Carrier Sense Multiple Acces with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification* je rozvíjen dodnes a zahrnuje mnohá rozšíření popisující různorodou signálovou komunikaci přes rozličná přenosová média jako optický kabel, koaxiální kabel a kroucená dvoulinka. Dnes je Ethernet standardizován i normou ISO 8802/3 [2].

Komunikační rozhraní Ethernet se v referenčním modelu ISO/OSI popsaného v [3] vyskytuje na fyzické a spojové vrstvě [4].

2.1.1 Ethernetový rámec

V Ethernetových sítích jsou data posílána po tzv. rámcích [4]. Struktura Ethernetového rámce je zobrazena v obrázku 2.1 a skládá se z:

Preambule 56bitové pole, kde každý oktet obsahuje hexadecimální hodnotu *0x55*. Tato sekvence slouží k synchronizaci hodin u příjemce.

Oddělovač rámce (SFD) 8bitové pole obsahující hexadecimální hodnotu *0x5D*. Tento oktet určuje začátek rámce.

Obrázek 2.1: Struktura Ethernetového rámce podle 802.3 [4]

7 bajtů	1 bajt	6 bajtů	6 bajtů	2 bajty		4 bajty
Preambule	SFD	Adresa cíle	Adresa zdroje	Délka/typ	Data	CRC

Adresa cíle Fyzická adresa příjemce dlouhá 48 bitů.

Adresa zdroje Fyzická adresa odesílatele dlouhá 48 bitů.

Typ/délka Toto 16bitové pole může být použito dvěma způsoby. Hodnota menší než 1 500 udává délku datového pole v oktetech, zatímco hodnota 1 536 a více znamená, že se jedná o číslo indikující typ protokolu zapouzdřeného v Ethernetovém rámci.

Data Pole o proměnné délce sloužící pro přenášená data.

Kontrolní součet FCS Jedná se o algoritmus CRC32 popsaný v [5], který slouží k detekci integrity rámce.

2.1.2 Ethernetové vrstvy

Standard IEEE 802.3 specifikuje jednotlivé vrstvy v komunikačním modelu. Ty jsou různé pro jednotlivá rozšíření standardu. Komunikační vrstvy pro rychlý Ethernet (100 Mbps až 10 Gbps) jsou zobrazeny v obrázku 2.2.

2.1.2.1 Vrstva řízení přístupu k médiu (MAC)

V MAC vrstvě je zahrnuta detekce kolizí a chyb v integritě dat zajištěná nezávisle na fyzickém médiu [4]. Zároveň jsou zde také rozdělena data do rámců (preambule, oddělovač a kontrolní sekvence). K datům je také přidána MAC adresa odesílatele a příjemce (viz sekce 2.1.1).

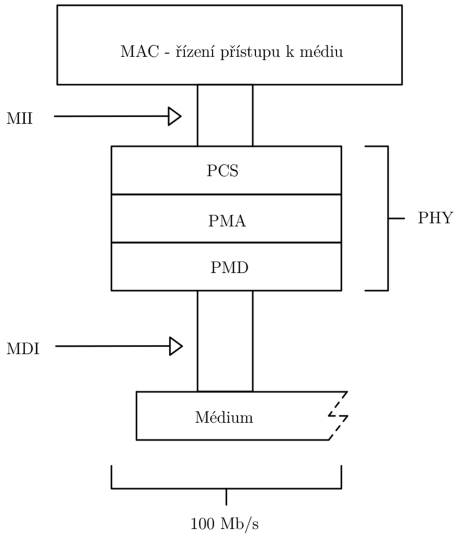
2.1.2.2 Rozhraní nezávislé na typu média (MII)

Toto rozhraní definuje jednoduché a plně duplexní komunikační propojení mezi MAC a fyzickými vrstvami (PHY).

Součástí MII je také rozhraní MDIO, které je používáno k vyčítání a zápisu registrů ve vrstvách PHY. Pomocí těchto registrů lze vrstvy PHY nejen nastavit, ale i zjišťovat informace o připojení kabelu atd.

Původní MII používalo pro komunikaci čtyřbitové obousměrné rozhraní taktované na frekvenci 25 MHz. Tím bylo dosaženo rychlosti 100 Mbps. Avšak další rozšíření dosahují rychlosti i 10 Gbps (XGMII) [4].

Obrázek 2.2: Rozdělení vrstev v komunikačním modelu podle IEEE 802.3 pro rychlý Ethernet. [4]



2.1.2.3 Podvrstva fyzického kódování (PCS)

Podvrstva PCS se stará o překlad dat mezi MII a PMA. Jednotlivá čtyřbitová slova přicházející po MII jsou zakódována do pětibitových kódových skupin, která jsou následně posílána po fyzickém médiu.

Zároveň jsou zde generovány signály protokolu přístupu ke sdílenému médiu (CSMA). Rovněž serializace a deserializace dat z/do formy pro vrstvu PMA je prováděna v této vrstvě [4].

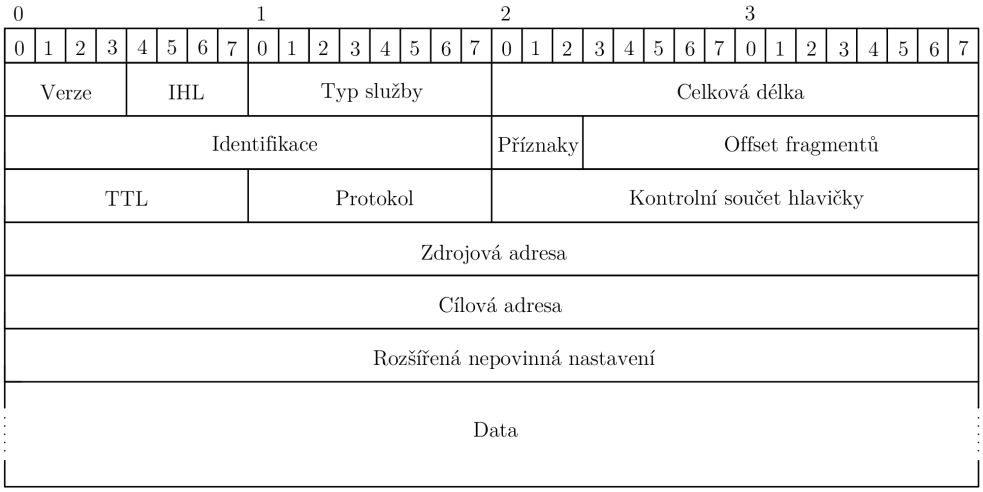
2.1.2.4 Připojení k fyzickému médiu (PMA)

Tato podvrstva mapuje přenosové kódy pro podvrstvu závislou na fyzickém médiu (PMD). Generuje signály značící dostupnost PMD a zajišťuje zpětné dekódování hodin z NRZI kódování. V tomto kódování je binární jednička kódována změnou stavu signálu a nula je přečtena, pokud k žádné změně nedošlo [6]. Dále se v této vrstvě nachází (pokud je implementována) automatická negociace.

2.1.2.5 Podvrstva závislá na fyzickém médiu (PMD)

Tato podvrstva se stará o konverzi signálů PMA na fyzické médium. To zahrnuje časování jednotlivých bitů, kódování signálů a buzení média [4].

Obrázek 2.3: Struktura paketu protokolu IPv4 [7].



2.2 Síťové protokoly

Na vyšších vrstvách referenčního modelu ISO/OSI popsaného v [3] jsou definovány další protokoly popisující komunikaci na větší úrovni abstrakce. V této sekci se budu věnovat hlavním síťovým protokolům, které se týkají mé práce.

2.2.1 Internetový protokol

Internetový protokol (IP) je základním protokolem celé rodiny TCP/IP, která je v dnešní době brána jako standard pro komunikaci v rozsáhlých počítačových sítích [2].

IP definuje nad linkovou vrstvou další logickou adresaci, díky které je možné komunikovat mezi více sítěmi. Adresa definovaná protokolem IP se nazývá IP adresa a jedná se o unikátní identifikátor. V současné době se aktivně využívají dva typy IP protokolů, IPv6 a IPv4 [2].

2.2.1.1 Struktura IP paketu

Struktura paketu IPv4 je zobrazena v obrázku 2.3 a skládá se z následujících polí [7]:

Verze protokolu 4bitové pole.

Délka hlavičky v 32bitových slovech (IHL) 4bitové pole Délka hlavičky na rozdíl od protokolu IPv6 může mít proměnnou délku.

Typ služby 8bitové pole.

Celková délka datagramu 16bitové pole.

Identifikace 16bitové pole sloužící k opětovnému složení fragmentovaných datagramů.

Příznaky 3bitové pole obsahující následující jednobitové příznaky:

Rezervováno (0) Jedná se o rezervovaný bit, který musí být nulový.

Nefragmentovat (DF) Je-li tento příznak nastaven, nesmí být zpráva po cestě fragmentována.

Více fragmentů (MF) Není-li tento příznak nastaven, pak se jedná o poslední fragment.

Offset fragmentu 13bitové pole udávající offset fragmentu v původním datagramu. Offset je udáván po 64 bitech.

Životnost (TTL) 8bitové pole. Při průchodu směrovačem se tato hodnota snižuje o jedna, přičemž jakmile dosáhne hodnoty nula, směrovač má povinnost zprávu zahodit. Tímto mechanismem dochází k zabránění kruhovému přeposílání dat mezi přepínači.

Protokol 8bitové pole označující vnořený protokol

Kontrolní součet hlavičky 16bitové pole sloužící k ověření integrity hlavičky.

Zdrojová IP adresa 32bitové pole s adresou odesílatele

Cílová IP adresa 32bitové pole s adresou příjemce

Rozšířená nepovinná nastavení Jedná se o pole s proměnnou délkou. Minimální délka je 32bitové slovo a maximální jsou čtyři 32bitová slova.

Data Datové pole, které může být velké až 64 KiB.

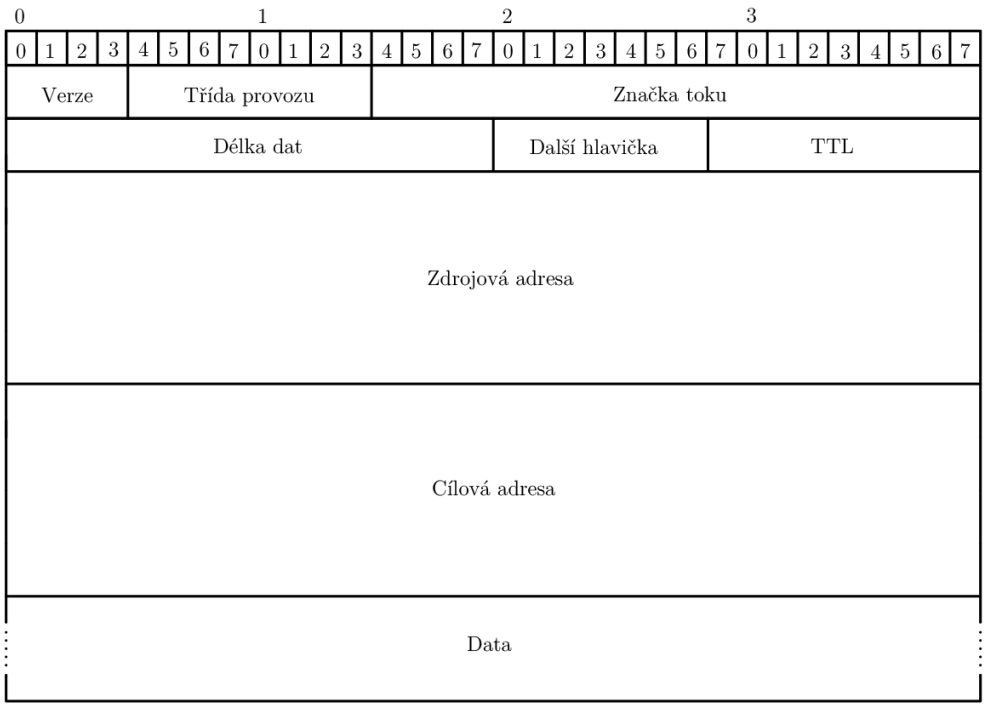
Oproti IPv4 nabízí novější protokol IPv6 podstatně větší počet adres, protože rozšířil její velikost na 128 bitů. Větší počet adres je jeden z hlavních důvodů nasazování novějšího protokolu, jelikož v roce 2011 byly alokovány poslední volné adresní bloky [8].

IPv6 nabízí oproti staršímu protokolu i další výhody. Jedná se například o efektivnější směrování, identifikátor toku (QoS) a absenci redundantního kontrolního součtu, který zbytečně zahlcuje směrovače. Struktura IPv6 paketu je zobrazena v obrázku 2.4 a skládá se z následujících polí [9]:

Verze 4bitové pole obsahující verzi IP protokolu.

Třída provozu 8bitové pole určující prioritu paketu.

Obrázek 2.4: Struktura paketu protokolu IPv6 [9].



Značka toku 20bitové pole sloužící pro správu QOS. Jedná se o nepoužívané pole, které mělo sloužit pro aplikace reálného času.

Délka dat 16bitové pole určující délku datové části paketu.

Další hlavička 8bitové pole, typ následující hlavičky. Může se jednat o jednu z rozšiřujících hlaviček definující např. fragmentaci, šifrování a směrování, nebo o vnořený protokol.

Životnost (TTL) 8bitové pole určující maximální počet směrovačů, kterými může paket projít. Jedná se o stejný mechanizmus jako u IPv4 hlavičky.

Zdrojová IP adresa 128bitové pole s adresou odesílatele.

Cílová IP adresa 128bitové pole s adresou příjemce.

Data Datové pole, které může být velké až 64 KiB ve standardním režimu. IPv6 zavádí i tzv. jumbogramy s velikostí až 4 GiB.

2.2.1.2 Výpočet kontrolního součtu

Hlavička internetového protokolu 4. verze obsahuje kontrolní součet zajišťující integritu hlavičky(viz sekce 2.2.1.1). Ten je třeba vypočítat při každé její

úpravě. Protokol IPv6 kontrolní součet již neobsahuje, protože jeho výpočet zbytečně zatěžuje síťové prvky např. přepínače, které při každém průchodu paketu snižují hodnotu TTL a spoléhá na zajištění integrity hlavičky ve vyšší vrstvě ISO/OSI modelu popsaného v [3].

Algoritmus je počítán jako prostý aritmetický součet 16bitových hodnot hlavičky paketu. Přičemž jakmile dojde k přetečení, je přičtena jednička navíc. Na tento výsledek je aplikován operátor logické negace, čímž vznikne finální hodnota kontrolního součtu. V pseudokódu 2.1 je zapsána funkce implementující algoritmus [7].

Pseudokód 2.1: Algoritmus implementující kontrolní součet IPv4 protokolu.

```
ui16 calculate_checksum(ui16[] header, int length)
{
    ui16 sum = 0
    int i = 0;
    while(i < length(header))
    {
        sum = sum + header[i];
        if(overflow(sum))
        {
            sum = sum + 1;
        }
    }
    return not(sum);
}
```

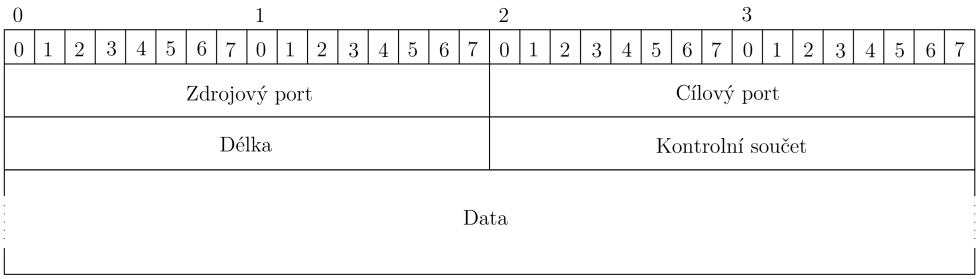
2.2.1.3 Fragmentace IP paketů

Velikost odeslaného paketu může být větší, než podporují síťové prvky po cestě. V takovém případě je třeba IP paket rozdělit na menší části. Tento proces se nazývá fragmentace. Jednotlivé verze IP protokolů přistupují k tomuto procesu odlišně.

V protokolu IPv4 může odesílatel odeslat jakkoliv velký paket a o jeho případné rozdělení se postarají jednotlivé prvky po cestě [7], zatímco protokol IPv6 vyžaduje, aby fragmentace byla prováděna pouze u odesílatele [9]. To v praxi znamená, že pokud k síťovému prvku dorazí větší paket a on jej není schopný odeslat, tak přijatý paket zahodí.

Dalším rozdílem je absence podpory fragmentace v základní hlavičce u IPv6, kde jsou údaje o fragmentovaném paketu uloženy v tzv. rozšiřující hlavičce [9].

Obrázek 2.5: Struktura paketu protokolu UDP [10].



2.2.2 User Datagram Protokol (UDP)

V hlavičce IP protokolu můžeme naleznout pole definující vnořený protokol (viz sekce 2.2.1.1). Jedním z těchto protokolů může být User Datagram Protocol (UDP).

Pomocí tohoto protokolu probíhá komunikace bez navázání přenosového kanálu mezi vysílačem a přijímačem. To znamená menší režijní nároky a malou hlavičku, ale na druhou stranu se jedná o nespolehlivé spojení. Protokol nezaručuje přijetí dat ve správném pořadí, dokonce negarantuje přijetí vůbec. Díky těmto vlastnostem se protokol využívá v aplikacích reálného času a pro přenos multimédií, kde je třeba minimalizovat síťové zpoždění.

UDP definuje čísla portů. Ty slouží k rozlišení typu síťové aplikace, podle kterých jsou pakety na zdrojových stanicích přiřazovány jednotlivým procesům, které si daný port zaregistrovaly.

Struktura UDP paketu je zobrazena v obrázku 2.5 a skládá se z následujících polí [10]:

Zdrojový port 16bitové pole označující číslo zdrojového portu.

Cílový port 16bitové pole označující číslo cílového portu.

Délka zprávy 16bitové pole.

Kontrolní součet 16bitové pole s kontrolním součtem. Jedná se o stejný algoritmus jako v případě IP protokolu (viz sekce 2.2.1.2) a je počítaný přes UDP pseudohlavičku a datové pole.

UDP pseudohlavičkou je myšlena UDP hlavička s přidanou zdrojovou a cílovou IP adresou, typem vnořeného protokolu a délkou dat IP paketu.

Data Datové pole, které může být velké až 64 Kib.

Kontrolní součet UDP paketu je nepovinný (může být nulový), pokud je jako síťový protokol použit IPv4, naopak při použití protokolu IPv6 je jeho jeho výpočet nutný [11] [12].

2.2.3 Adress Resolution Protocol (ARP)

Tento protokol slouží k získání fyzické adresy (viz sekce 2.1.1) z IP adresy. Využívá se ho zejména ve chvíli, kdy je potřeba komunikovat pomocí IP protokolu ve stejné síti.

Pokud taková situace nastane, je třeba odeslat žádost *ARP request* linkovým broadcastem. Jedná se o speciální fyzickou adresu, která zaručí, že bude doručena všem zařízení v lokální síti. Příjemce žádosti s příslušnou IP adresou odpovídá pomocí *ARP reply* s fyzickou adresou [13].

Odesílatel si následně uloží tuto informaci do překladové tabulky mezi IP a fyzickou adresou nazývanou *ARP table*. Z toho důvodu není nutné odesílat dotaz na fyzickou adresu při každém pokusu o komunikaci, avšak každý záznam v tabulce je potřeba po vypršení jeho platnosti obnovit [14].

2.2.4 Dynamic Host Configuration Protocol (DHCP)

DHCP protokol umožňuje konfiguraci síťových zařízení přímo přes Ethernetové rozhraní. Jedná se především o IP adresu, masku sítě a výchozí bránu.

Pomocí DHCP lze přiřadit IP adresu dvěma různými způsoby:

Dynamické Adresa je klientovi přiřazena s časovým omezením.

Statická Server má alokované IP adresy k příslušným MAC adresám.

Dynamické přidělení umožňuje znovupoužitelnost adres v síti. Po určitém čase musí klient znovu požádat o přidělení adresy. Pokud tak neučiní, může ji server přidělit někomu jinému.

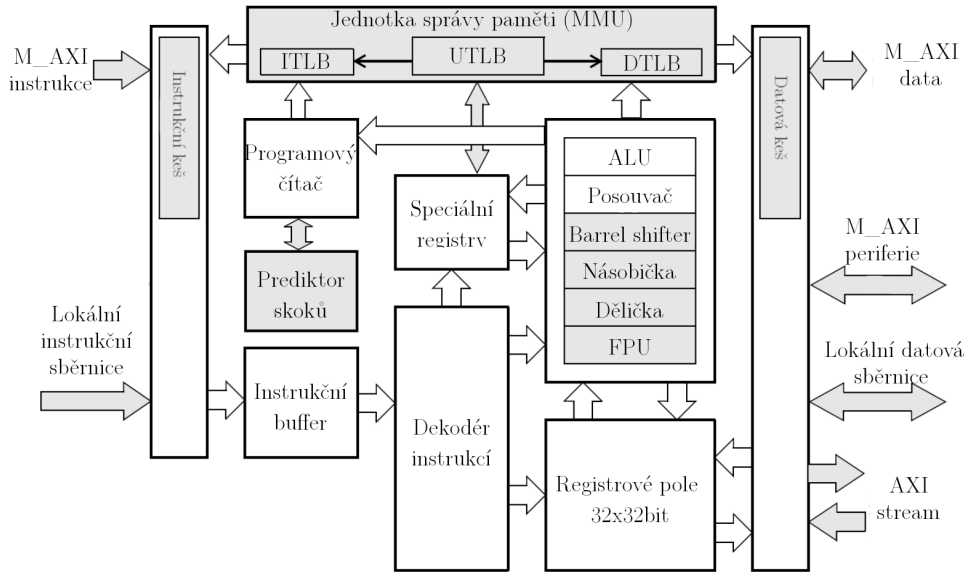
Klienti žádají o přidělení adresy pomocí protokolu UDP (viz sekce 2.2.2) na portu 68, přičemž server poslouchá na portu 67. Klient po připojení vysílá linkovým broadcastem *DHCPDISCOVER* paket. Na ten server odpoví paketem *DHCPOFFER* se seznamem nabízených IP adres. Klient si následně vybere jednu z nabízených adres, požádá server o přidělení vybrané adresy pomocí *DHCPREQUEST* a server mu ji potrvdí pomocí *DHCPACK* [15]. Po přijetí DHCPACK může začít klient používat vybranou adresu.

2.3 Hardwarová platforma společnosti Xilinx

2.3.1 Microblaze

MicroBlaze je 32bitový RISC soft mikroprocesor navržený pro implementaci na platformě Xilinx FPGA. Jedná se o široce konfigurovatelný mikroprocesor registrové architektury, který dovoluje návrhářům vybrat pouze podmnožinu funkcionalit potřebnou pro jejich cílovou aplikaci. MicroBlaze může fungovat jako malý mikrokontrolér s 8 Kbit operační paměti, nebo jako plnohodný jedno-jádrový procesor s jednotkou správy paměti (MMU), predikcí skoků a aritmetickým akcelerátorem čísel v plovoucí řádové čárce [16].

Obrázek 2.6: Architektura procesoru MicroBlaze [16].



Ačkoliv je MicroBlaze všestranně konfigurovatelný, následující vlastnosti nelze změnit:

- Celkem 32 registrů pro všeobecné použití o šířce registru 32 bitů.
- Instrukční slovo dlouhé 32 bitů.
- Pipeline vykonávající maximálně jednu instrukci za takt.

Blokové schéma procesoru je zobrazeno v obrázku 2.6.

2.3.1.1 Pipeline

Ve vykonávání instrukcí u procesoru MicroBlaze je využito pipeliny, přičemž většina instrukcí potřebuje pouze jeden hodinový cyklus k dokončení stupně pipeline. Velikost pipeline značným způsobem ovlivňuje rychlost zpracování instrukcí. Procesor lze nastavit do tří následujících konfigurací ovlivňujících velikost pipeline [16].

Minimum spotřebovaných prostředků FPGA Pipeline je v tomto nastavení minimalizována na tři stupně (*Načtení, Dekódování a Vykonání instrukce*)

Maximalní výkon Pipeline je rozdělena do pěti stupňů (*Načtení, Dekódování, Vykonání instrukce, Přístup do paměti a Zápis do registru*)

Maximalní frekvence Pipeline obsahuje osm stupňů, přičemž jejich rozdělení je stejné jako v nastavení *Maximální výkon*, pouze přístup do paměti je členěno do dalších čtyř různých částí.

2.3.1.2 Paměťová architektura

MicroBlaze je navržen s Harwardskou architekturou, což znamená, že paměť instrukcí a programu jsou dva rozdílné adresové prostory. Celkem může mít mikroprocesor až tři různá paměťová rozhraní [16].

Lokální paměťová sběrnice (LMB) Jedná se o paměťovou sběrnici, která se připojuje k nízkokapacitním pamětem umístěných přímo v FPGA. Velikost této paměti se pohybuje obvykle v řádech desítek kilobajtů.

AXI4 rozhraní s koherenčním rozšířením (AXI4–ACE) Toto rozhraní je obvykle používáno pro přístup k externím pamětem DDR4, jelikož obsahuje rozšíření pro koherentní přístup do cache.

AXI4 Toto rozhraní je používáno pro přístup k periferiím.

Adresové rozsahy jednotlivých rozhraní a periferií je možné nastavit s ohledem na to, že maximální velikost adres je 32, nebo 64 bitů. Také je při nastavování adres třeba brát ohled na fakt, že MicroBlaze začíná po resetu číst instrukce z adresy *0x0* [16].

2.3.2 Komunikační rozhraní AMBA AXI

Advanced eXtensible Interface *AXI* je otevřený standard sběrnicové architektury vyskytující se přímo na čipu. Tato architektura byla poprvé představena v roce 2003 společností ARM jako součást sběrnicových architektur Advanced Microcontroller Bus *AMBA*. Poslední generace AMBA AXI4 byla představena v roce 2010 a je v dnešní době široce používaná v mikrokontrolérech, FPGA a Systémech na čipu (SoC) [17].

Existují tři typy rozhraní AXI4. Jedná se o AXI4, AXI4–Lite a AXI4–Stream.

2.3.2.1 AMBA AXI4

Toto rozhraní, také občas nazývané AXI4–Full, je používáno v aplikacích s vysokými nároky na propustnost [17].

Standard AXI4 protokolu definuje dvoubodové rozhraní mezi mastrem a slavem pro vzájemnou komunikaci. Pro realizaci spojení více masterů k více slaveům je nutné použít křížový přepínač, který podle cílové adresy specifikované při inicializaci transakce provede spojení k cíli s daným adresním rozsahem.

Dvoubodové spojení AXI4 je podle standardu rozděleno do pěti nezávislých částí [18] :

- Rozhraní pro čtení adresy
- Rozhraní pro zápis adresy
- Rozhraní pro čtení dat
- Rozhraní pro zápis dat
- Rozhraní pro potvrzení zápisu

Šířka dat a adres obsažených v jednotlivých rozhraních není definována a lze si ji libovolně zvolit v závislosti na aplikaci.

Komunikace pro AMBA AXI4 probíhá mezi mastrem a slavem v transakcích, které vždy zahajuje master. Data se následně mohou přenášet v obou směrech po dávkách. Jedna dávka je sekvence dat, která může být velká až 256 sběrnicových slov a její velikost je vždy definována při zahájení transakce. Obrázek 2.7 zobrazuje průběh zápisové transakce na rozhraní AXI4. Přesné názvy signálů jednotlivých částí, jejich průběhy a veškeré funkcionality jsou popsány v [18].

2.3.2.2 AMBA AXI4–Lite

AXI4–Lite je velmi podobné rozhraní AMBA AXI4. Je vhodná při použití v aplikaci s nízkými nároky na propustnost, např pro vyčítání a zápis kontrolních registrů. Hlavním rozdílem je velikost dávky, která je u AXI4–Lite omezena na jedna. Dále postrádá další funkcionality jako podporu cache, vyrovnávacích pamětí a atomicitu transakcí [18].

2.3.2.3 AMBA AXI4–Stream

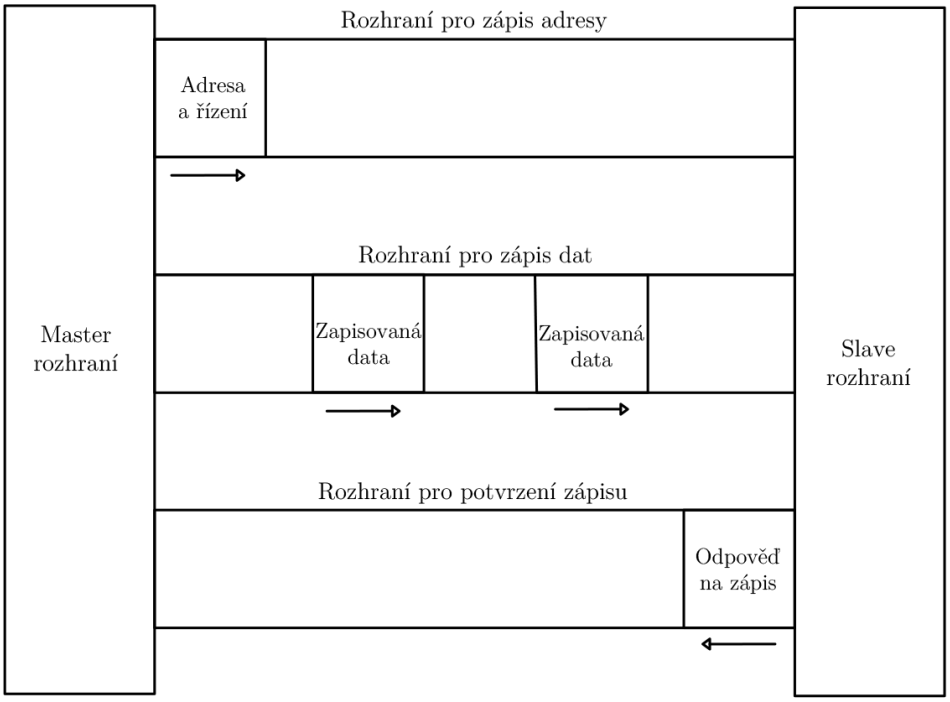
Rozhraní AXI4–Stream je rozhraní používané pro přenos vysokorychlostního proudu dat. Jedná se o jednosměrnou komunikaci, kdy master vysílá a slave nemůže na data žádným způsobem odpovědět. Slave ovládá pouze jeden signál, kterým může mastera upozornit na zahlcení, a data pozastavit.

Ve skutečnosti je podmnožina AXI4–Stream protokolu použita při komunikaci na jednotlivých nezávislých rozhraních protokolu AXI4 a AXI4–Lite 2.3.2.1. Přesné názvy signálů, průběhy a funkcionality protokolu jsou popsány v [19].

2.4 Měření variace síťového zpoždění paketů (síťového jitteru)

Podle *RFC3393* je variace síťového zpoždění paketů definován jako odchylka zpoždění paketů vůči průměrnému zpoždění. Tento jev se ale častěji označuje

Obrázek 2.7: Zapisová transakce AXI4 [18].



jako síťový jitter a z toho důvodu bude tento termín používán i v následujícím textu diplomové práce [20].

Kolísání doby propagace paketu sítí může nastat kvůli zahlcení, špatnému řazení do front aktivních prvků nebo i nesprávným nastavením [21]. Jitter je jedním z parametrů kvality připojení, který výrazným způsobem ovlivňuje latenci u streamovaných dat, jelikož jediný možný způsob, jak jev maskovat na straně přijímače je použití vyrovnávacích bufferů.

Síťový jitter je v dnešní době většinou měřen v softwaru. Tyto techniky obvykle dosahují maximálně milisekundové přesnosti, což je ve většině aplikací zcela dostačující, avšak pro testování sítí určených pro přenosy s latencí nižší než 1 ms je takováto přesnost nepoužitelná.

V následující sekci je popsáno několik metod pro měření síťového jitteru s vysokou přesností.

2.4.1 Vkládání časových značek

Asi nejjednodušší metodou, jak měřit síťový jitter, je pomocí vkládání časových značek do paketů. Při příjmu je následně spočítáno síťové zpoždění daného paketu. Z něj je možné počítat i jitter. V [22] byla tato technika použitá pro měření kruhového zpoždění optické sítě. Tedy přijímač byl

nakonfigurován jako nízkolatenční síťové zrcadlo a výpočet byl realizován na vysílači. Tímto způsobem bylo dosaženo přesnosti 10 µs.

Pokud bychom chtěli tyto metody použít pro měření zpoždění paketu v jednom směru, je nutné provádět vyhodnocení na přijímači. To však přináší nový problém synchronizace času mezi vysílačem a přijímačem.

Při použití moderního protokolu PTP jsme sice schopni na Ethernetové síti dosáhnout přesnosti až 100 ns [23], avšak jeho použití potřebuje hardwarovou podporu na všech síťových prvcích po cestě mezi přijímačem a odesílatelem [24].

Rozšířenější protokol NTP, který nepotřebuje žádnou podporu mezilehlých síťových prvků, dokáže podle [23] dosáhnout přesnosti až 1 ms. Použití metody vkládání časových značek používající protokoly pro synchronizaci je tedy hlavně limitováno přesností protokolů samotných a nikdy nemůže být lepší.

2.4.2 Metoda Interarrival Histograms

Další populární způsob měření síťového jitteru je metoda Inter-Arrival histogram [25]. Tato metoda předpokládá, že zdroj odesílá proud rovnoměrně rozestoupených a stejně velkých paketů. Přijímací strana pouze značkuje příchody paketů a počítá rozdíl intervalů mezi příchozími pakety.

2.4.2.1 Princip měření

Metoda Interarrival Histograms počítá jitter jako rozdíl mezi naměřeným a očekávaným intervalem mezi pakety. Očekávaná hodnota je rovna intervalu mezi pakety nastavenému na vysílači. Pokud přijímač tuto dobu nezná, je nutné ji odhadnout.

Nechť Ts_i reprezentuje i-tou dobu mezi pakety na vysílači, Tr_i reprezentuje i-tou dobu mezi pakety na přijímači a D_i reprezentuje i-tou změnu zpoždění sítě. Pak jistě platí:

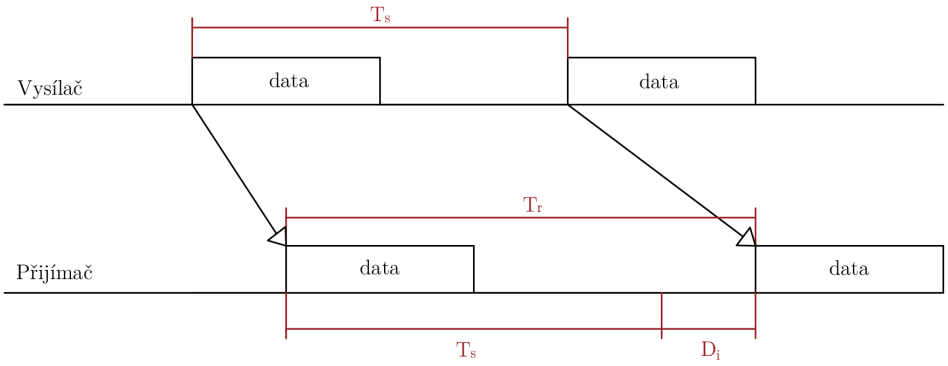
$$Tr_i = Ts_i + D_i \tag{2.1}$$

Podle metody Interarrival Histograms je Ts_i konstanta, kterou je nutné odhadnout na přijímači. Dále v textu budu tedy předpokládat, že:

$$Ts_i = Ts \tag{2.2}$$

Pokud předpokládáme, že Tr_i je náhodná veličina s normálním rozdělením, které má střední hodnotu ($\mathbb{E}$) rovnou Ts, pak s dostatečně velkým statistickým vzorkem můžeme střední hodnotu odhadnout na přijímací straně pomocí výběrového průměru změřených dob [25]. Výpočet odhadu Ts je vyjádřen ve vztahu 2.3.

Obrázek 2.8: Princip měření metody Interrarival Histrograms, kde Ts reprezentuje dobu mezi pakety na vysílači, Tr_i reprezentuje i-tou dobu mezi pakety na přijímači a D_i reprezentuje i-tou změnu zpoždění sítě [25].



Nechť $D_i \sim \mathbb{N}(0, \sigma)$ pak platí:

$$\frac{1}{n}\sum_{k=0}^{n}(Tr_i) = \frac{1}{n}\sum_{i=1}^{n}(Ts_i + D_i) = Ts + \frac{1}{n}\sum_{i=1}^{n} D_i \sim T_s \tag{2.3}$$

Pokud přijímací strana úspěšně odhadla mezipaketový interval nastavený na vysílači, pak jednoduchou úpravou vztahu 2.1 můžeme vypočítat změnu zpoždění sítě, tedy jitter.

2.4.2.2 Přesnost metody

Přesnost této metody se výrazně liší mezi implementacemi, jelikož záleží nejen na spolehlivosti odhadu doby mezi pakety, ale také na rozlišení a stabilitě časovače používaného ke zjištění doby příchodu.

Rozlišení časovače je hlavní limitující prvek. Pokud bychom značkovali dobu příchodu paketů v softwaru v uživatelském prostoru operačního systému, byla by přesnost značky velice nízká, protože doba mezi faktickým příchodem paketu a předáním dat programu může být i řádově desítky milisekund, kvůli plánování a přepínání kontextu v operačním systému. Lepší metodou je implementace časové značky přímo v prostoru jádra operačního systému, nebo využití podpory síťové karty. V [25] bylo pomocí časových značek prováděných přímo na síťové kartě dosaženo přesnosti 1 µs.

Spolehlivost odhadu je dalším důležitým prvkem limitujícím přesnost. Pokud předpokládáme, že jitter je náhodná veličina s normálním rozdělením $\mathbb{N}(0, \sigma)$ [25], můžeme použít jako chybovou funkci konfidenční interval. Vztah 2.4 zobrazuje chybovou funkci pro konfidenční interval 99.7 %, pokud

jej aplikujeme na dostatečně velký vzorek naměřených dat.

$$\epsilon = 3 \cdot \frac{\overline{S}}{\sqrt{N-1}} \tag{2.4}$$

Kde je ϵ odhad chyby, $\overline{S}$ je výběrová směrodatná odchylka a N počet vzorků [25].

Výběrovou směrodatnou odchylku lze získat pomocí výpočtu vyjádřeného ve vztahu 2.5.

V případě, že bychom chtěli zjistit potřebnou velikost vzorku (N) pro zajištění dostatečné nízké chyby, je nutné nejdříve vypočítat hodnotu výběrové směrodatné odchylky pomocí vztahu 2.5.

$$\overline{S} = \sqrt{\frac{1}{n-1} \cdot \sum_{\forall i} (Tr_i - Ts)^2} \tag{2.5}$$

Výběrová směrodatná odchylka je úzce spjatá s jittrem v síti. Pro zjednodušení výpočtu je možné ji také odhadnout pomocí pravidla 3 σ [26].

Pravidlo říká, že u přibližně normálně rozděleného statistického souboru by se měly téměř všechny relevantní hodnoty nacházet do vzdálenosti tří směrodatných odchylek.

Výběrová směrodatná odchylka $\overline{S}$ může být ve vztahu 2.4 nahrazena za maximální možný síťový jitter. Tímto způsobem omezíme chybu měření shora pomocí pravidla 3 σ, což vede ke zjednodušenému vztahu:

$$\epsilon \leq \frac{1}{2} \cdot \frac{J}{\sqrt{N-1}} \tag{2.6}$$

kde je J maximální možný jitter v síti, který může být získán méně přesnou metodou nebo kvalifikovaným odhadem.

2.5 Operační systém GNU/Linux

Počátek tohoto operačního systému se datuje do roku 1991, kdy jej začal vyvíjet finský student Linux Tovards na základě unixového operačního systému MINIX. Název GNU/Linux vznikl až spojením s projektem GNU založeným v roce 1983 Richardem Stallmanem, který si kladl za cíl vytvořit operační systém složený pouze z otevřeného softwaru [27].

Spojením obou projektů vznikla řada operačních systémů, kterým se dnes zkráceně říká Linuxové distribuce a pokrývají široké spektrum zařízení. Od malých jednoprocesorových vestavných zařízení po multiprocesorové výpočetní clustery [28].

Naprostá většina zdrojového kódu operačního systému Linux je napsána v jazyce C, avšak určité funkcionality, na které je kladen požadavek rychlosti a maximální efektivity (např. plánovač), jsou napsány v assembleru [29].

Operační systém je možné zkompilovat na velké množství cílových architektur. Pro tuto práci je ovšem nejdůležitější procesorová architektura MicroBlaze, která je při splnění následujících požadavků podporována [30]:

- Kontrolér vnější operační paměti, která má kapacitu alespoň 32 Mbit.
- Časovač s dvěma kanály a připojeným přerušením
- Sériový port
- MicroBlaze s Memory Managment Unit (MMU)

Po splnění výše popsaných požadavků bude možné linuxové jádro přeložit. Pro zavedení z nevolatilní paměti jsou potřeba ještě následující periferie, které Xilinx označuje jako volitelné [30].

- Externí nevolatilní paměť
- Minimální velikost lokální paměti procesoru 4 Kb (BRAM) pro First Stage bootloader, která se inicializuje s nahraným bitstreamem.

Operačnímu systému se během zavádění na platformě ARM i MicroBlaze předává soubor s názvem device-tree. Jedná se o stromovou strukturu popisující typ procesoru, sběrnici a dostupné periferie. Na základě informací obsažených v tomto souboru může operační systém zavádět ovladače a celkově zařízení spravovat.

2.6 Vývojové desky

Vedoucím práce bylo rozhodnuto, že paketové zrcadlo bude implementováno na dvou platformách a to Numato Mimas A7 a Avnet AES KU-040-DB-G.

2.6.1 Numato Mimas

Tato vývojová deska byla vybrána z důvodu její nízké pořizovací ceny a malé velikosti. Je osazena Xilinx Artix 7 FPGA (XC7A50-T1FGG484C) a mimo jiné disponuje pro tuto práci nejdůležitějšími funkcemi [31]:

- 2Gbit DDR3 paměti
- 128Mbit flash paměť pro konfiguraci FPGA a uživatelská data
- 100MHz CMOS oscilátor
- Převodník z sérové linky na USB
- 1Gbps Ethernet
- 8 LED diod, 4 tlačítka a 8 přepínačů

2.6.2 Avnet AES KU-040

Druhá deska byla vybrána hlavně z důvodu absence rychlých portů na levnějších FPGA Artix 7, což by znemožňovalo budoucí rozšíření zrcadla na 10 Gbps Ethernet. Vývojová deska Avnet AES KU-040-DB-G je osazena Xilinx Kintex UltraScale FPGA (XCKU040-1FBVA676). Deska poskytuje pro tuto práci nejdůležitější funkcionality [32]:

- 8Gbit DDR4 paměti
- 256Mbit flash paměť pro konfiguraci FPGA
- 256Mbit flash paměť pro uživatelská data
- Dvě rozhraní 1Gbps Ethernet
- Převodník z sériové linky na USB
- Dvě SFP+ rozhraní
- 8 LED diod, 5 tlačítek a 8 přepínačů
- Digilent USB-JTAG programátor FPGA

2.7 CESNET Modular Video Transfer Platform (MVTP)

Jedná se o zařízení používané pro nízkolatenční přenos videa navržené sdružením CESNET. Zařízení je tvořeno FPGA čipem, ve kterém je implementováno veškeré zpracování signálů. Navržená architektura dokáže přenést video mezi vysílačem a přijímačem s latencí pod 1 ms [33]. Platforma MVTP umožňuje přenést až dva signály ve 4K rozlišení (4096×3112) nebo osm signálů v HD (1280×720) rozlišení přes síť 10Gbps Ethernet .

Vstupem do zařízení je SDI (Serial Digital Interface) signál z kamery a výstupem jsou datové pakety na 1Gbps, nebo 10Gbps Ethernetovém rozhraní. Zařízení dokáže také pakety přijímat a převést data zpět do SDI signálu.

Pakety splňují časové požadavky definované normou *SMPTE ST 2110-21:201*. Pro tuto práci je nejdůležitější fakt, že podle této normy mají pakety obsahující video vždy stejnou délku a jsou vysílány se stejným časovým odstupem [34].

Platforma MVTP je používána například pro přenos videa pro výukové účely v lékařství a umění.

Kapitola 3

Analýza a návrh

Návrh celého zařízení se podřizoval specifikaci od vedoucího práce popsané v sekci 1.1. Hlavní požadavky, které formovaly celkový návrh jsou vzdálená administrace a automatická konfigurace zařízení. To znamená nutnost podpory IP, ARP, UDP a DHCP protokolu. A vzhledem ke vzdálené konfiguraci je rovněž potřeba implementovat autentizaci a šifrování.

Ačkoliv vše výše popsané je možné navrhnout a vytvořit pomocí automatů v hardwaru, jednalo by se o příliš pracné řešení. Z toho důvodu byl po konzultaci s vedoucím práce zvolen návrh obsahující mikroprocesor. Jako jediný vhodný kandidát se v tomto případě jevil MicroBlaze, vzhledem k jeho podpoře v Xilinx FPGA.

Dále bylo po konzultaci s vedoucím práce rozhodnuto pro tento procesor zkompilovat Linuxový operační systém. Hlavním důvodem, který vedl k tomuto rozhodnutí, je snížení množství kódu nutného k implementaci, protože většina požadavků, které by byly implementovány v softwaru, jsou již v Linuxovém systému vyřešeny. Druhým důvodem je jednodušší rozšiřitelnost zrcadla v budoucnu. A třetím, velice důležitým důvodem, je bezpečnost, která nesmí být hlavně u síťových zařízení podceňována. Při množství kódu nutného k implementaci by bylo nemožné zajistit stejnou úroveň zabezpečení, jako mají již léta prověřené implementace v Linuxu.

3.1 Hardwarová platforma

Požadavky vedoucího práce vyžadují implementaci zařízení na dva odlišné FPGA čipy. Z toho důvodu bylo třeba vytvořit dvě hardwarové platformy, které se ovšem liší pouze ve verzi použitých IP jader a podpoře měření síťového jitteru.

V následujícím textu je popsáno řešení hlavních problémů pro obě hardwarové platformy.

3.1.1 Ethernet

Vzhledem k tomu, že obě desky vybrané vedoucím práce (viz sekce 2.6) obsahují fyzickou vrstvu (viz sekce 2.1.2) ve formě externího čipu, je třeba implementovat v FPGA pouze vrstvu MAC.

Pro realizaci této vrstvy existuje značné množství IP jader. Jen Xilinx nabízí celkem čtyři různé jádra. Po důkladné analýze všech možností byl výběr zúžen na následující tři řešení:

- Použití Xilinx Tri-mode Ethernet MAC
- Použití Xilinx 1G/2.5G Ethernet Subsystem
- Implementace vlastní jednotky

Funkcionalita prvních dvou popsaných IP jader je velice podobná. Jádra tedy překládají data mezi fyzickou vrstvou, se kterou komunikují pomocí rozhraní RGMII, a sběrnicí AXI4–Stream. Dále také kontrolují/počítají FCS a přidávají/odebírají preambuli s oddělovačem začátku paketu (viz sekce 2.1.1).

Jedná se o velice jednoduchou funkcionalitu, kterou není příliš těžké implementovat vlastními silami. Nevýhodou vlastního řešení je ale absence ovladačů této jednotky pro Linuxový operační systém. Ovladače by bylo nutné do jádra vytvořit, čímž by se zvýšila pracnost.

Jádro Xilinx Tri-mode Ethernet MAC je placené jádro podporující rychlosti 10 Mbps, 100 Mbps a 1000 Mbps včetně plně i poloviční duplexního spojení [35]. Součástí jádra je také podpora přenosu protokolu přesného času PTP [24]. Linuxové jádro obsahuje přímo ovladače a v případě jeho použití by nebylo nutné cokoliv implementovat. Jeho hlavní nevýhodou je ovšem nutnost zakoupení licence.

Poslední možností, která byla zvažována, je použití Xilinx 1G/2.5G Ethernet Subsystem. Ten se skládá z dvojice jader, která jsou dodávána jako jeden systém. Jedná se o výše popsané jádro Tri-mode Ethernet MAC a AXI Ethernet Buffer, který se chová jako vyrovnávací paměť a dokáže také provádět výpočet kontrolních součtů IP a UDP vrstvy. Mimo výhody popsané u Tri-mode Ethernet MAC lze toto jádro jednoduše připojit přímo k jádru AXI DMA, které bude data předávat rovnou do operační paměti.

Hlavními nevýhodami jsou nutné licenční poplatky a také zcela chybějící dokumentace k části AXI Ethernet Buffer. Xilinx v [36] popisuje pouze vnější rozhraní, ale nepopisuje komunikaci mezi Bufferem a Tri-mode Ethernet MAC.

I přes výše popsané nevýhody bylo rozhodnuto použít jádro AXI Ethernet Subsystem, jelikož sdružení CESNET má zakoupenou licenci a odhadnutá pracnost řešení související s chybějící dokumentací byla nižší, než implementace vlastní jednotky.

3.1.2 Jednotka zrcadla paketů

Tato jednotka implementuje funkci paketového zrcadla. Ze zadání na ni jsou kladeny následující funkční požadavky:

- Podpora protokolu IPv4 a IPv6.
- Nastavovací registrové pole.
- Detektor paketů k zrcadlení. Detekují se pouze UDP porty podle těchto parametrů:
  - Zdrojová IP adresa
  - Zdrojový UDP port
  - Cílový UDP port

  Přičemž zadání zdrojového a cílového portu je volitelné.
- Volitelné prohazování zdrojového a cílového portu.
- Přeposlání paketu zpět k odesílateli, nebo na zadanou cílovou IP adresu.
- Vestavěné čítače statistik zrcadlených paketů.

3.1.2.1 Umístění

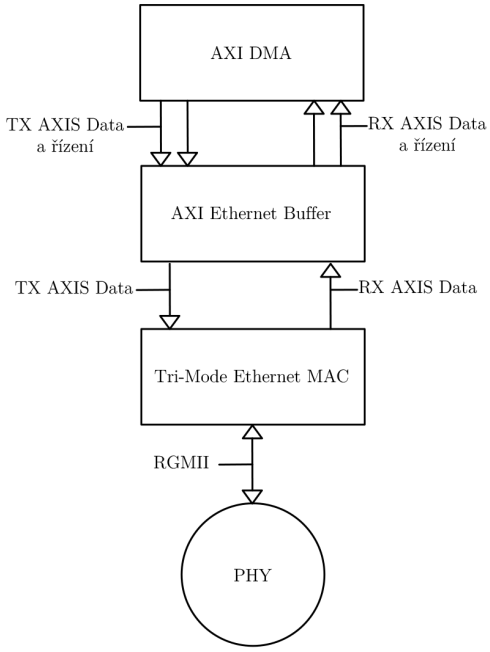
Jednotku lze umístit kamkoliv mezi vnější port FPGA čipu a DMA jádro. Zapojení datové cesty paketů je zobrazeno na obrázku 3.1.

Prvním možným umístěním je mezi jádro AXI Ethernet Buffer a AXI DMA. Pokud bychom jednotku situovali do těchto míst, nemusela by počítat UDP ani IP kontrolní součty, jelikož jejich výpočet je plně podporován v AXI Ethernet Buffer [36]. To ale přináší určité nevýhody, které souvisejí s latencí. Vyrovnávací paměť může mít podle [36] velikost 16 Kb, 32 Kb, 64 Kb a 128 Kb, nicméně její zpoždění už zdokumentováno není.

Další nevýhoda tohoto umístění pramení z propojení mezi DMA a vyrovnávací pamětí. Ty jsou totiž spojeny celkem čtyřmi AXI4–Stream rozhraními. Dvě jsou určena pro odesílací a dvě pro přijímací datový proud. Podle [36] je v každém páru jeden proud datový a druhý řídící. Jednotka umístěná na tomto komunikačním kanálu by musela mít osm rozhraní AXI4–Stream, jelikož by bylo nutné upravovat také řídící sběrnice, zatímco ostatní umístění vyžadují rozhraní pouze čtyři.

Druhou možností je vložit jednotku přímo na RGMII rozhraní před vrstvu MAC. Výhodou tohoto umístění je nejnižší možná latence zrcadla. Naopak hlavní nevýhodou je duplicitní počítání FCS. Kontrolní součet počítaný na vrstvě MAC přestane být platný kvůli úpravám hodnot v hlavičce Ethernetového rámce (prohození IP adresy atd.). Jádro Tri-Mode Ethernet MAC mimo počítání FCS obsahuje i další mechanizmy překladu mezi AXI4–Stream

Obrázek 3.1: Blokové schéma datové cesty paketů mezi DMA a Ethernet PHY.



a RGMII jako chyby v příjmu a odesílání, které by bylo nutné v jednotce zrcadla duplicitně implementovat.

Bylo rozhodnuto použít třetí možnost tedy vložit jednotku mezi jádro Tri-Mode Ethernet MAC a AXI Ethernet Buffer. Jednotka zrcadla nebude muset vůbec implementovat MAC vrstvu, jelikož od ní bude odstíněna pomocí jádra Tri-Mode Ethernet MAC a nebude třeba duplicitně přepočítávat kontrolní součet FCS.

Jednotka zrcadla ovšem bude muset počítat kontrolní součty IP a UDP vrstvy. V tomto případě se ale nebude jednat o duplicitní výpočet, jelikož paketová data vůbec nedorazí do AXI Ethernet Buffer.

Toto umístění také nabízí téměř plnou kontrolu nad latencí, jelikož Tri-Mode Ethernet MAC neobsahuje žádné buffery [35] a případné přidané zpoždění je tedy minimální.

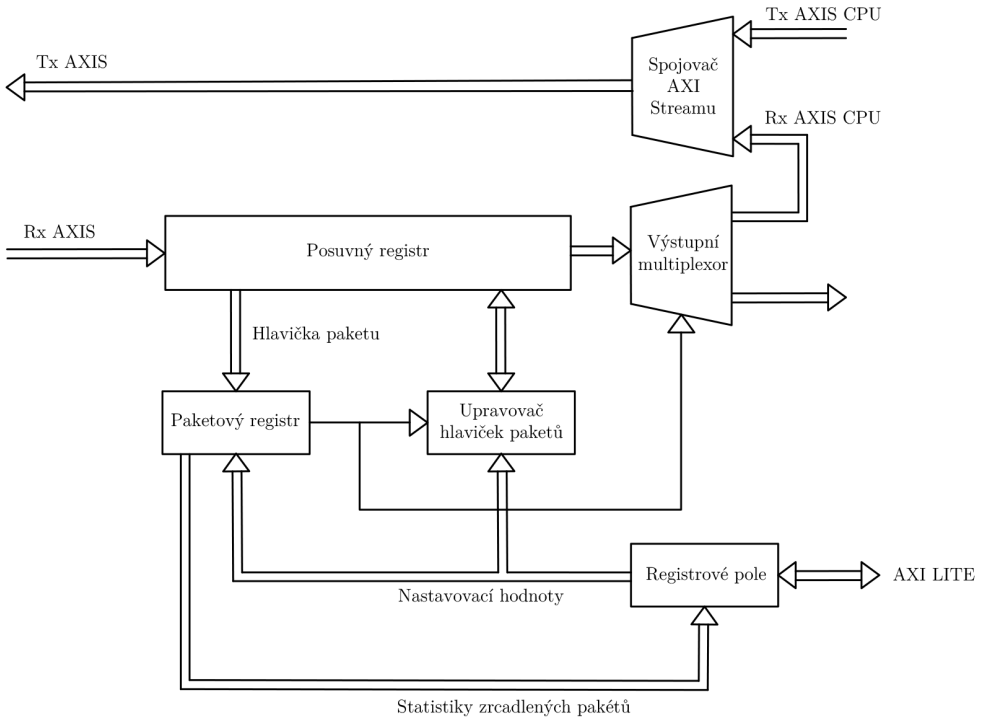
3.1.2.2 Architektura Jednotky

Jednotka pro zrcadlení používá dlouhý posuvný registr. Toto řešení se jevilo jako jediné možné, aby byla zaručena propustnost 1 Gbps a případná jednoduchá úprava na 10Gbps Ethernet.

Po načtení hlaviček jednotlivých síťových vrstev (MAC, IP a UDP) je zkontrolován paket vůči hodnotám nastavených uživatelem. Podle toho je rozhodnuto, zda paket bude propuštěn dále směrem do procesoru, nebo

bude zrcadlen. Blokové schéma datové cesty paketu v jednotce je zobrazeno v obrázku 3.2.

Obrázek 3.2: Blokové schéma jednotky zrcadla paketů.



3.1.2.3 Paketový filtr

Tento blok kontroluje každý příchozí paket a rozhoduje, zda bude propuštěn do procesoru, nebo bude přeposlán na základě parametrů určených uživatelem. Ty jsou popsány v sekci 3.1.2.

Po domluvě s vedoucím práce bylo rozhodnuto, že pro správnou funkčnost jednotky bude povinné zadat pouze parametr zdrojové IP adresy. Pokud uživatel nenastaví zbylé parametry jako zdrojový a cílový port, bude paketový filtr přeposílat veškeré UDP pakety ze zadané IP adresy.

3.1.2.4 Registrové pole

Dle požadavků vedoucího práce bylo nutné opatřit jednotku registry, které budou obsahovat nastavující hodnoty zrcadla. Jako vhodné přístupové rozhraní k registrům bylo zvoleno AXI4–Lite (viz sekce 2.3.2.2), které je používáno v IP infrastruktuře společnosti Xilinx právě pro vyčítaní a zápis kontrolních registrů. Blok je opatřen celkem 32 čtyřoktetovými registry, z nichž je část re-

zervovaná pro případné budoucí rozšíření. Velikost registrů je dána standardní datovou šířkou AXI4–Lite rozhraní.

Struktura registrového pole je zobrazena v tabulce 3.1.

Tabulka 3.1: Registry v jednotce zrcadla. I/O znamená vstupně-výstupní, O znamená pouze výstupní registr

Registr	bity 31 – 16	bity 15 – 0	Typ
0	Konfigurační registr		I/O
1	IPv4 adresa zdroje		I/O
2	IPv4 adresa cíle		I/O
3	Zdrojový UDP port	Cílový UDP port	I/O
4	Počet zrcadlených paketů		O
5	Počet zrcadlených bitů, bit (63 – 32)		O
6	Počet zrcadlených bitů, bit (31 – 0)		O
7	Zdrojová IPv6 adresa, bit (127 – 96)		I/O
8	Zdrojová IPv6 adresa, bit (95 – 64)		I/O
9	Zdrojová IPv6 adresa, bit (63 – 32)		I/O
10	Zdrojová IPv6 adresa, bit (31 – 0)		I/O
11	Cílová MAC adresa, bit (31 – 0)		I/O
12	Rezervováno	Cílová MAC adresa, bit (47 – 32)	I/O
13	Cílová IPv6 adresa, bit (127 – 96)		I/O
14	Cílová IPv6 adresa, bit (95 – 64)		I/O
15	Cílová IPv6 adresa, bit (63 – 32)		I/O
16	Cílová IPv6 adresa, bit (31 – 0)		I/O
17 – 31	Rezervováno		

Konfigurační registr se skládá z:

Start zrcadla (bit 0) Zapsáním logické jedničky se spustí zrcadlení paketů, které vyhovují nastavení filtru

Reset čítačů (bit 1) Zapsáním logické jedničky se vyresetují vestavěné čítače počtu zrcadlených paketů. Na další náběžnou hranu přepíše zařízení daný bit zpět do logické nuly.

Start prohazování portů (bit 2) Zapsáním logické jedničky bude zrcadlo prohazovat zdrojový port s cílovým.

Rezervováno (bit 31-3)

3.1.2.5 Upravovač hlaviček paketu

Tento blok slouží k úpravě dat v hlavičkách přeposílaných paketů. Jednotka počítá kontrolní IP a UDP součty, které jsou následně vloženy do paketu.

Před odesláním je také upraveno pole TTL na hodnotu *64* (viz sekce 2.2.1.1). Blok vyplní MAC a IP adresu odesílatele hodnotami adres příjemce ze zpracovávaného paketu. Dále je upravena cílová MAC a IP adresa. Chování jednotky v tomto případě závisí na hodnotách specifikovaných uživatelem. Nulová MAC a IP adresa příjemce zapsaná v registrovém poli má za následek zrcadlení paketu zpět k odesílateli. To znamená, že jednotka pouze prohodí zdrojovou a cílovou MAC a IP adresu.

V opačném případě jsou na příslušná pole v Ethernetovém rámci vloženy hodnoty z registrového pole.

3.1.2.6 Výpočet kontrolních součtů

Po konzultaci s vedoucím práce bylo rozhodnuto, že jednotka zrcadla nebude muset kontrolovat integritu paketu výpočtem a kontrolou kontrolního součtu. Ovšem při odesílání zrcadlených paketů je nutné tento výpočet provádět, jelikož jednotka upravuje hodnoty v hlavičkách paketů.

Kontrolní součet IPv4 hlavičky je možné počítat přímo v posuvném registru jednotky, jelikož je umístěn těsně před jejím koncem. Po zpracování posledního bajtu hlavičky je kontrolní součet vložen do paketu bez nutnosti použití vyrovnávací paměti. Dochází tak k výpočtu s minimální přidanou latencí. To bohužel není případ kontrolního součtu na protokolu UDP, který pokrývá i datové pole a je umístěn na začátku paketu. Ačkoliv je při použití IPv4 tento součet nepovinný a jednotka by jej mohla vyplnit samými nulami, při použití protokolu IPv6 je jeho výpočet nutný (viz sekce 2.2.2).

Předpokládaný postup je tedy použití vyrovnávací paměti, do které budeme ukládat paket před odesláním, přičemž v průběhu ukládání se bude počítat kontrolní UDP součet podle algoritmu popsaného v sekci 2.2.1.2. Po zpracování celého paketu by byl výsledný kontrolní součet vložen na příslušné místo na začátku UDP hlavičky a paket by mohl být odeslán. Tímto způsobem ale značně zvyšujeme latenci jednotky.

Algoritmus výpočtu kontrolního součtu je skutečně jenom součet posloupnosti 16bitových čísel obsažených v paketu. Pouze v případě přetečení dochází k přičtení další jedničky k výsledku. Vzhledem k tomu, že sčítání je komutativní operace, tak pouhým prohozením posloupnosti 16bitových slov se výsledek kontrolního součtu nezmění. Při prohazování UDP portu nebo při přeposílání paketu zpět k odesílateli není tedy potřeba součet vůbec počítat.

Problém nastává ve chvíli, kdy přeposíláme paket jinam než odesílateli. V tomto případě dochází k celkové změně dat a zneplatnění předešlého kontrolního součtu. Při bližším pohledu na algoritmus výpočtu můžeme přesně definovat vztah mezi původním a novým kontrolním součtem, známe-li předešlou cílovou IP adresu. Jedná se o pouhé odečtení původní IP adresy a následné přičtení adresy nové. Algoritmus výpočtu nového kontrolního součtu při změně adresy je zanesen v pseudokódu 3.1.

Pseudokód 3.1: Zrychlený algoritmus výpočtu kontrolního součtu při znalosti původní IP adresy

```
ui16 calcheck_fast(ui16[] old_ip, ui16[] new_ip,
                   ui16[] old_checksum, int length)
{
    int i = 0;
    u16 sum = not old_checksum
    while (i < length){
        sum = sum - old_ip[0];
        if(undewflow(old_checksum)){
            sum = sum - 1;
        }
        i = i+1;
    }
    i = 0;
        while (i< length){
        sum = sum + new_ip[0];
        if(overflow(checksum)){
            sum = sum + 1;
        }
        i = i+1;
    }

    return not(sum);
}
```

Tímto způsobem lze vypočítat kontrolní součet pouze z hlavičky a není nutné data nikde ukládat, čímž se sníží latence.

Toto řešení bohužel závisí na správnosti přijatého kontrolního součtu, což je jeho hlavní nevýhoda. Podle [37] je pravděpodobnost nesprávného kontrolního součtu v UDP 0.001 %. Přičemž z těchto chyb je až 90 % zaviněno špatným výpočtem už na straně odesílatele.

Za předpokladu, že odesílatel vypočítá správnou hodnotu kontrolního součtu je pravděpodobnost chyby minimální. Z toho důvodu převážily výhody této metody nad nevýhodami a bylo rozhodnuto ji použít.

3.1.2.7 Fragmentované IP pakety

Při nižších hodnotách velikosti rámce na fyzické vrstvě je možné vetší IP pakety rozdělit na menší části. Tento jev se nazývá fragmentace (viz sekce 2.2.1.3).

Fragmentace přináší problém, protože hlavička UDP paketu je obsažena pouze v prvním fragmentu. Zrcadlo by pro správnou funkci muselo v tomto případě sestavovat pakety. To by vyžadovalo připojení externí RAM paměti,

do které by se ukládaly jednotlivé fragmenty. Po přijetí posledního fragmentu by došlo k vyhodnocení, zda se má paket přeposlat, nebo má být propuštěn do procesoru.

Po konzultaci s vedoucím práce bylo rozhodnuto vynechat podporu fragmentovaných paketů, jelikož se jako zdroj předpokládá platforma MVTP (viz sekce 2.7), která posílá pakety o správné délce, a tudíž k fragmentaci nedochází. Jednotka tedy bude všechny fragmentované pakety propouštět do procesoru.

3.1.2.8 Podpora nižších rychlostí

Jádro Xilinx Tri-Mode Ethernet MAC podporuje 1 Gbps, 100 Mbps a 10 Mbps rychlost Ethernetového rozhraní. Ačkoliv bylo vedoucím práce řečeno, že použití zařízení na nižších rychlostech, než je 1 Gbps je nepravděpodobné, bylo rozhodnuto podporovat i rychlosti 100 Mbps a 10 Mbps.

Ethernetové jádro při nižších rychlostech posílá data se stejnou hodinovou frekvencí 125 MHz a se šířkou sběrnice 8 bitů. Datová propustnost AXI4–Stream rozhraní je tedy přesně 1 Gbps. V případě nižších rychlostí Ethernetu je odeslán jeden oktet každých 10 (při rychlosti 100 Mbps), nebo 100 taktů (při rychlosti 10 Mbps).

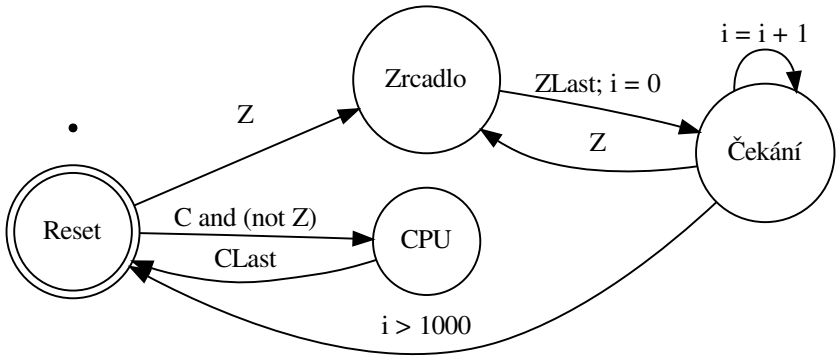
To přináší otázku, jak realizovat posuvný registr. Nejjednodušší možností je ukládání AXI4–Lite signálů do dostatečně dlouhé paměti. Toto řešení ovšem vyžaduje pro uložení hlavičky posuvný registr o délce 8 000 stupňů, jelikož všechny hlavičky, včetně rozšířených nepovinných nastavení u IPv4 2.3, jsou dlouhé 80 oktetů a validní data jsou u 10 Mbps každých 100 taktů. Jedná se o zbytečné plýtvání, a proto nebylo toto řešení použito.

Druhým možným řešením by bylo vyčtení připojené rychlosti Ethernetu z jádra Tri Mode Ethernet MAC a ukládání pouze validních dat. Tento způsob zachová malou velikost posuvného registru a zároveň sníží latenci na úplné minimum.

Třetím řešením je použití paměti FIFO v paketovém módu, která ukládá pouze validní data a začne vysílat až po přijetí celého paketu. Toto řešení má hlavní výhodu v jednoduchosti celé implementace, ovšem nevýhoda je přidání latence v datové cestě paketu. Ta je v tomto případě stejná, jako doba příjmu paketu, což je v případě 1Gbps Ethernetu 15 µs.

I přes velkou nevýhodu latence bylo rozhodnuto, že bude přidána jedna vyrovnávací paměť přímo za jádro Tri Mode Ethernet MAC. Tento buffer bude navíc sloužit k zahazování dat při detekci chyby v příjmu, kterou může Ethernetové jádro detekovat až při kontrole kontrolního součtu, který je umístěn na konci rámce 2.1.1.

Obrázek 3.3: Konečný automat arbitračního algoritmu implementovaného v jádru AXI4–Stream combiner. *Z* – Zrcadlo by chtělo odeslat přijatý paket, *ZLast* – Poslední oktet odesílaného paketu ze zrcadla. *C* – CPU by chtělo odeslat paket. *CLast* – Poslední oktet odesílaného paketu z CPU. *i* - Čekací iterátor.



3.1.2.9 Spojení dvou AXI4–Stream rozhraní do jednoho

Ve chvíli, kdy bude paket detekován k přeposlání, bude třeba spojit do jednoho rozhraní AXI4–Stream zrcadlící pakety s rozhraním paketů vycházejícím z procesoru. Xilinx poskytuje IP jádro AXI Stream Switch [38], pomocí kterého by bylo možné tuto funkcionalitu implementovat. Toto jádro podporuje komunikaci až 16 masterů k 16 slaveům, přičemž cíl komunikace je určen signálem *TDEST*. V případě zrcadla se jedná o komunikaci dvou masterů k jednomu slaveovi, tím pádem by signál *TDEST* mohl být konstantní. Hlavním problémem je v tomto případě arbitrační strategie, která nedokáže dostatečně zvýhodnit jedno master rozhraní. Z toho důvodu bylo rozhodnuto implementovat vlastní jádro s názvem AXI4 Stream Combiner kombinující datový proud od dvou masterů k jednomu slave.

Implementovaný arbitrační algoritmus bude, pro co největší snížení latence, upřednostňovat jednotku zrcadla.

Sběrnice je přidělena rozhraní, které si o ni zažádá jako první. Hlavním rozdílem mezi rozhraními je fakt, že když CPU dokončí přenos, je sběrnice okamžitě uvolněna, zatímco po ukončení přenosu ze zrcadla čeká arbitr ještě 1 000 taktů na případný další přenos a nepřidělí sběrnici procesoru. Toto čekání arbitra na případnou další zprávu není v jádře AXI Stream Switch podporováno [39].

Konečný automat popisující arbitrační strategii je popsán v obrázku 3.3.

V případě obsazení odesílajícího rozhraní datovým provozem z Mikropro-

cesoru je nutné data připravená k přeposlání dočasně uložit. Z toho důvodu bude implementované jádro obsahovat paměť FIFO. Ta bude také realizovat přechod mezi přijímací a odesílací hodinovou doménou. Přidaná latence závisí na obsazení odesílací sběrnice. Pokud bude volná, fronta přidá pouze 32 ns (4 takty) latence při přechodu hodinových domén.

3.1.2.10 Výsledná latence jednotky zrcadla

Pro výpočet latence v jednotce zrcadla je nutné vzít v úvahu navržené buffery a dobu strávenou v posuvném registru. Jednotka je navržená s jednou pamětí FIFO v paketovém módu. To znamená, že potřebuje přijmout celý paket, než propustí data na výstup. Na 1Gbps Ethernetu přicházejí bajty po AXI4–Stream sběrnici s periodou 8 ns. Přijetí jednoho datagramu o délce 1 518 bajtů do této fronty trvá tedy $1\,518 \cdot 8$ ns = 12 144 ns

Dále je v datové cestě 128 stupňů dlouhý posuvný registr. Přicházející paket je tedy zpožděn o $128 \cdot 8$ ns = 1 024 ns. Dále je přítomna další FIFO realizující přechod mezi hodinovými doménami. Tato FIFO přidá 4 takty na výstup v případě, že Ethernetové rozhraní není obsazené procesorem. Přidané zpoždění je tedy $4 \cdot 8$ ns = 32 ns. V opačném případě musí paket počkat na uvolnění rozhraní. To může trvat až $1\,518 \cdot 8$ ns = 12 144 ns, tedy dobu odeslání jednoho paketu o velikosti 1 500 bajtů.

Z výše popsaných mezivýsledků můžeme říci, že maximální zpoždění zrcadla je na 1Gbps Ethernetu přibližně 12 144 ns + 1 024 ns + 12 144 ns = 25 312 ns = 25,312 µs, přičemž hlavní část latence je tvořena příjmem samotných dat.

3.1.3 Jednotka měření síťového jitteru

Tato jednotka měří síťový jitter příchozího paketovém proudu a jsou na ni kladeny následující požadavky:

- Vysoká přesnost měření
- Podpora výpočtu jednoduchých statistických metod nad daty
- Registrové pole s výsledky a nastavením jednotky

3.1.3.1 Metoda měření jitteru

V sekci 2.4 jsou popsány dvě metody přesného měření síťového jitteru.

První metoda vkládá časové značky do paketů. Její hlavní nevýhodou je synchronizace času. Ačkoliv jádro Tri-Mode Ethernet MAC [35] podporuje přesnější protokol PTP, nelze se spolehnout, že jej budou podporovat i vnitřní síťové prvky.

Druhou metodou je Inter-Arrival Histogram. Tato metoda vyžaduje datový proud stejně dlouhých paketů, které bude odesílatel posílat vždy po stejné

době. Pokud splníme tento požadavek na datový proud, pak je možné dobu mezi pakety odhadnout a poměrně přesně počítat síťový jitter (viz sekce 2.4.2).

Požadavky na takový datový proud splňuje platforma MVTP (viz sekce 2.7), ale jakékoliv jiné zařízení, které vyhovuje standardu *SMPTE ST 2110-21:201*, lze také použít [34].

Pokud předpokládáme, že zdrojem proudu dat je platforma MVTP, pak použití metody Inter-Arrival Histogram umožňuje měřit jitter z normálního přenosu videa a nevyžaduje žádné speciální úpravy paketů. Jedná se o velkou výhodu oproti druhé metodě, která vyžaduje úpravu paketu přidáním časové značky. Z toho důvodu bylo rozhodnuto o použití metody Inter-Arrival Histogram.

3.1.3.2 Statistická analýza nad naměřenými daty

Jedním z požadavků vedoucího práce byla možnost určité statistické analýzy nad naměřenými daty. Aby bylo možné v budoucnu jednoduše rozšiřovat statistické funkce, bylo rozhodnuto vložit do jednotky paměť naměřených dat, která bude přístupná přes rozhraní AXI4 popsané v sekci 2.3.2.1. Tímto rozhodnutím je možné rozdělit funkce, které je třeba implementovat v hardwaru a které budou implementovány v softwaru.

Software bude moci pracovat pouze s určitým množstvím dat uložených v přidané paměti. V hardwaru budou naopak implementovány statistické funkce provádějící analýzu nad všemi daty, které byly získané od počátku daného měření.

Po konzultaci s vedoucím práce bylo rozhodnuto v hardwaru implementovat maximální a průměrnou hodnotu jitteru. Maximální hodnota jitteru bude počítána jako rozdíl mezi nejdelším a nejkratším intervalem mezi pakety.

V softwaru bude implementován průměr, rozptyl a směrodatná odchylka z posledních n hodnot uložených v paměti. Tím bude možné sledovat změnu jitteru a jeho případný trend.

3.1.3.3 Zajištění integrity dat v paměti výsledků

Paměť výsledků je navržena jako kruhový buffer, ve kterém budou nově získané hodnoty přepisovat ty nejstarší. Aby bylo možné zrekonstruovat posloupnost dat, bude v registrovém poli uložena adresa nejstaršího a nejnovějšího záznamu. Stáří těchto naměřených intervalů bude klesat se zvyšující se adresou v paměti.

Do této paměti bude měřící jednotka provádět zápis při každém příchodu paketu. Při plném vytížení Ethernetového rozhraní bude zápis probíhat přibližně každých 15 µs. Jedná se tedy o téměř neustálý přístup. Aby při vyčítání naměřených intervalů nedocházelo ke ztrátě integrity dat z důvodu souběžného přístupu CPU a měřící jednotky, bude implementováno pozastavení měření.

Mikroprocesor tedy zápisem příslušného bitu v registrovém poli pozastaví zápis a po vyčtení dat jej znovu zapne.

3.1.3.4 Umístění jednotky

Podobně jako u jednotky paketového zrcadla je možné umístit tuto jednotku do tří míst. Blokové schéma datové cesty paketu je zobrazeno v obrázku 3.1.

Prvním možným umístěním je mezi jádro AXI Ethernet Buffer a AXI DMA. V tomto případě nám větší latence paketů nevadí. Pro správnou funkci je pouze potřeba, aby zpoždění bylo konzistentní a neměnilo se v průběhu měření. Jádro AXI DMA ovšem vyčítá AXI Ethernet Buffer pouze pokud dostane přidělenou sběrnici do operační paměti. V opačném případě se data ukládají do AXI Ethernet Buffer a čekají na přenos. Zpoždění procházejících paketů skrz rozhraní může být velice odlišné. Tento problém by šel vyřešit přidáním paměti FIFO před DMA řadič, ale ani v tomto případě není stabilita zpoždění zaručena, jelikož manuál k jádru AXI Ethernet Buffer tuto informaci neobsahuje [36].

Druhé možné místo je mezi jádrem Tri-Mode Ethernet MAC a AXI Ethernet Buffer. Vzhledem k tomu, že Tri-Mode Ethernet MAC neobsahuje žádné buffery, je zaručeno, že je latence přenosu paketu mezi RGMII rozhraním a měřící jednotkou vždy stejná. Hlavní výhodou je použití paketového filtru, který je implementován v jednotce zrcadla paketů (viz sekce 3.1.2).

Poslední možné místo je přímo na rozhraní RGMII. V tomto případě je latence také vždy stejná. Hlavní nevýhodou je implementace nového paketového filtru na RGMII. Z toho důvodu bylo rozhodnuto o použití druhé možnosti, tedy umístit jednotku měření jitteru na AXI4–Stream rozhraní mezi jádra Tri-Mode Ethernet MAC a AXI Ethernet Buffer.

Jednotka bude na tomto místě provoz pouze odposlouchávat, a nebude tak v přímé datové cestě mezi Ethernetovým rozhraním a mikroprocesorem. Toto umístění tedy nebude mít žádný negativní dopad na zpoždění zpracovávaných paketů, které zůstane nezměněno.

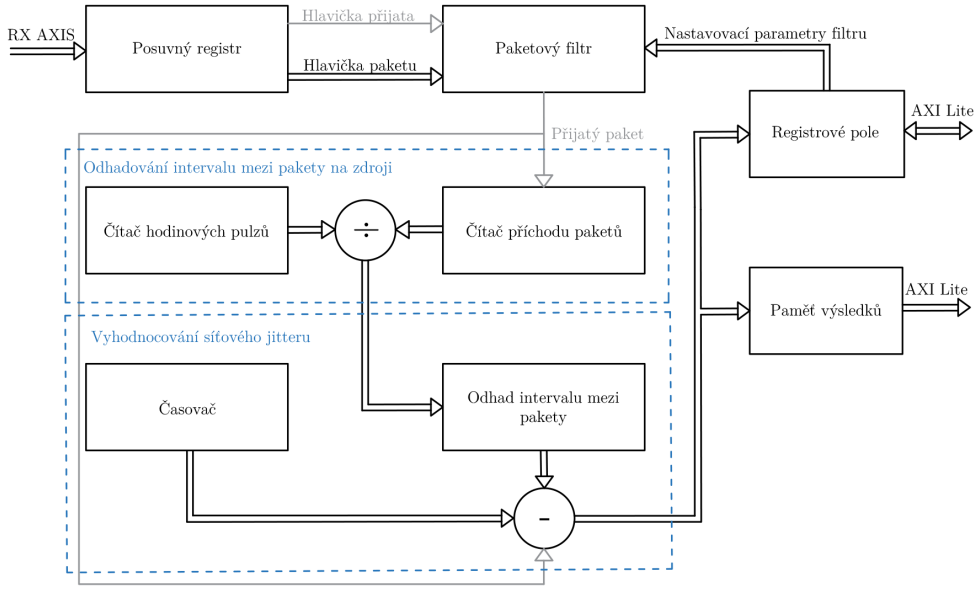
3.1.3.5 Popis architektury jednotky

Architektura jednotky je rozdělena do několika bloků (obrázek 3.4). Prvním z nich je paketový filtr, který slouží k rozpoznání proudu dat z platformy MVTP nad kterým je následně prováděno měření.

Tento filtr zkoumá hlavičky paketů a hledá takové, které vyhovují nastavení od uživatele. V případě nálezu je vygenerován signál, který upozorní měřící blok na příchod paketu. Totožný filtr je implementován v jednotce zrcadlení paketů. Popis jeho chování je zanesen v sekci 3.1.2.3.

Funkce měřícího bloku je rozdělena do dvou fází. První fáze slouží k odhadnutí mezi-paketového intervalu, se kterým jsou pakety odesílány na zdroji. V časovači jsou akumulovány hodinové pulzy a čítač počítá příchody jed-

Obrázek 3.4: Blokové schéma jednotky pro měření jitteru.



notlivých paketů. Jakmile je dosaženo velikosti učícího vzorku definovaného uživatelem, tak se hodnota v časovači celočíselně vydělí počtem vzorků. Tato operace tedy vypočítá odhad intervalu mezi pakety na zdroji.

Po vypočtení odhadu přechází jednotka do měřící fáze. V té dochází k samotnému měření jitteru. Časovač je s každým detekovaným příchodem paketu přečten a vynulován. Přečtená hodnota je následně odečtena v absolutní hodnotě od odhadu vytvořeného v učící fázi. Tento rozdíl je pak uložen do paměti výsledků a je průměrován a zpracováván pro určení maximální hodnoty jitteru.

Jednotka je také vybavena registrovým polem, které slouží k zapsání nastavovacích parametrů a vyčítání změřených parametrů sítě.

3.1.3.6 Registrové pole

Aby bylo možné vyčítat naměřené hodnoty a nastavovat měřící jednotku, bylo třeba implementovat registrové pole. Komunikační rozhraní bylo zvoleno stejné jako u jednotky zrcadlení paketů (viz sekce 3.1.2.4), a to AXI4–Lite. Jednotlivé registry jsou zobrazeny v tabulce 3.2.

Konfigurační registr je složen z několika jednobitových hodnot. Jejich struktura je:

Start měření (bit 0) Zapsáním logické jedničky se spustí měření síťového jitteru.

Tabulka 3.2: Registry v jednotce počítání jitteru. I/O znamená vstupně-výstupní, O znamená pouze výstupní registr

Registr	bity 31 – 16	bity 15 – 0	Typ
0	Konfigurační registr		I/O
1	Velikost učící fáze		I/O
2	Adresa nejstaršího naměřeného intervalu		O
3	Adresa nejnovějšího naměřeného intervalu		O
4	Naměřený maximální jitter		O
5	Naměřený průměrný jitter		O
6	Zdrojová IPv4 adresa		I/O
7	Zdrojová IPv6 adresa, bit (127 – 96)		I/O
8	Zdrojová IPv6 adresa, bit (95 – 64)		I/O
9	Zdrojová IPv6 adresa, bit (63 – 32)		I/O
10	Zdrojová IPv6 adresa, bit (31 – 0)		I/O
11	Zdrojový UDP port	Cílový UDP port	I/O
12	Odhadnutý čas mezi pakety, bit (63 – 32)		O
13	Odhadnutý čas mezi pakety, bit (31 – 0)		O
14 – 31	Rezervováno		

Reset naměřené hodnoty maximálního jitteru (bit 1) Zapsáním logické jedničky se vyresetuje hodnota doposud naměřeného maximálního jitteru.

Reset naměřené hodnoty průměrného jitteru (bit 2) Zapsáním logické jedničky se vyresetuje hodnota doposud naměřeného průměrného jitteru.

Pozastavení (bit 3) Zapsáním logické jedničky se pozastaví měření.

Pamět výsledků prázdná (bit 4, pouze výstupní) Logická jednička značí, že v paměti výsledků jsou přítomny validní hodnoty.

Validní jitter (bit 5, pouze výstupní) Logická jednička značí, že je hodnota naměřeného průměrného jitteru validní.

Reset měření (bit 6) Zapsáním logické jedničky se provede reset měření. Jednotka začne znovu učící fází.

3.1.3.7 Přesnost měření

Přesnost měření je závislá na rozlišení časovače a přesné znalosti intervalu, se kterým jsou jednotlivé pakety odesílány na zdroji (viz sekce 2.4.2.2).

Jednotlivá data přicházejí po AXI4–Stream rozhraní z jádra Tri-Mode Ethernet MAC taktovaného frekvencí 125 MHz [35]. Rozlišení vnitřního

časovače měřící jednotky je tedy 8 ns. Hlavním limitujícím faktorem bude v našem případě odhad původního intervalu mezi pakety.

Přesnost odhadu je dána maximální velikostí očekávaného jitteru sítě a velikostí vzorku v učící fázi algoritmu (viz sekce 2.4.2). V tabulce 3.3 je zanesena přesnost měření pro vybrané hodnoty jitteru a velikosti vzorku v učící fázi vypočítaná podle vztahu 2.6.

Tabulka 3.3: Vybrané hodnoty velikosti vzoru závislé na maximální hodnotě měřící chyby a očekávaném jitteru sítě.

Očekávaný jitter	100 ns	1 µs	10 µs	100 µs	1 ms
Maximální chyba	8 ns	8 ns	8 ns	32 ns	256 ns
Velikost učícího vzorku	50	3 907	390 625	2 441 410	3 814 697

3.2 Operační systém

Vedoucím práce bylo rozhodnuto, že množství softwarových služeb vyžaduje kompilaci Linuxového operačního systému.

Hlavní limitující faktor softwaru je velikost vestavěné flash na jednotlivých vývojových deskách (viz sekce 2.6). Deska Numato Mimas obsahuje jednu 128Mib flash, do které musí být umístěn bitstream i obraz s Linuxem, přičemž bitstream má fixní velikost 20 Mib. Avnet KU-040-DB-G obsahuje dvě 256Mib flash. Jedna slouží k uložení bitstreamu a druhá datům, přičemž bitstream má fixní velikost 128 Mib.

Vedoucím práce byly kladeny na software následující požadavky:

- Nastavení Ethernetového rozhraní přes DHCP
- Ruční konfigurace Ethernetového rozhraní
- Odesílání nastavené konfigurace na servery sdružení CESNET
- Vzdálená konfigurace

3.2.1 Linuxová distribuce

Již od samotného počátku bylo zřejmé, že nalézt distribuci vhodnou pro takto specifické zařízení, bude obtížné. V průběhu analýzy bylo ovšem zjištěno, že pro procesor MicroBlaze žádné distribuce neexistují. Přeložení vlastního operačního systému na míru je tedy jediným řešením.

Dvě nejpoužívanější překladová prostředí se nazývají Buildroot a Yocto project, přičemž obě dvě je možné použít. Překladové prostředí Buildroot je zaměřeno na jednoduchost a minimalismus. V základním nastavení jsou vypnuté kompilace téměř všech balíčků, což vede k naprosto minimální velikosti výsledného obrazu. Velkou výhodou celého prostředí je právě jednoduchost,

jelikož je používán normální makefile. Na druhou stranu nevýhodou je absence jakéhokoliv přednastavení. Překlad distribuce tedy vyžaduje velké množství nastavování. Další velkou nevýhodou je, že změna v jakémkoliv konfiguračním souboru vyžaduje následný překlad celého systému, což vede k dlouhé době ladění distribuce [40].

Prostředí Yocto naopak přichází s konceptem překladových vrstev. Každá vrstva přidá jednu, nebo více funkcionalit. Konfigurace distribuce se tímto způsobem velice urychlí. Hlavní nevýhodou je ale robustnost a velikost celého překladového systému a s tím související doba, za kterou se jej dokáže člověk naučit [40].

Tuto nevýhodu ovšem odstraňuje překladové prostředí Petalinux, které je vytvořené společností Xilinx. Jedná se nadstavbu nad projektem Yocto upravenou pro překlad na platformy MicroBlaze a Xilinx Zynq.

Součástí prostředí Petalinux jsou i programy generující konfigurační soubory Linuxu z vytvořené hardwarové platformy. To znamená, že prostředí automaticky konfiguruje překlad systému tak, aby se zakompilovaly ovladače na používané IP jádra a automaticky generuje device-tree (konfigurační soubor definující dostupný hardware pro jádro operačního systému) [30].

V neposlední řadě je také velkou výhodou podpora společností Xilinx. Z výše popsaných důvodů bylo rozhodnuto o použití překladového prostředí Petalinux.

3.2.2 Zavádění operačního systému

Zavádění operačního systému je rozděleno do dvou fází.

První fází je zavaděč FSBL. Jeho hlavní úlohou je nahrání zavaděče druhého stupně z SPI flash do paměti a jeho spuštění. Tímto programem je při nahrání bitstreamu do FPGA inicializovaná lokální paměť mikroprocesoru MicroBlaze.

Zavaděč druhého stupně je složitější program, jehož hlavní úlohou je samotné zavedení operačního systému. Prostředí Petalinux používá zavaděč U–Boot. Ten dokáže nahrát systém nejen z SPI flash, ale také z Ethernetového rozhraní.

Pro tuto práci je nejdůležitější podpora komprimovaných obrazů jádra a souborového systému, které budou vzhledem k malé velikosti flash paměti na desce Numato Mimas použity (viz sekce 2.6).

3.2.3 Uložení obrazu Linuxu do vestavěné flash

Aby nebylo potřeba při každém restartu zařízení provádět novou konfiguraci, je nutné umístit na vestavěnou flash paměť perzistentní úložiště do kterého bude možné zapisovat z operačního systému. Nabízí se tedy dva možné způsoby. Prvním je uložení celého kořenového souborového systému na flash. Souborový systém by byl JFFS2, který je určený přímo pro nativní přístup

k flash pamětem [41]. Výhodou, ale i nevýhodou tohoto řešení, je uložení veškerých souborů perzistentně a jakákoliv změna vydrží i výpadek napájení. Nevýhoda spočívá také v náchylnosti k chybám, které mohou být fatální. Například špatný zápis do flash paměti může vyústit v poškození souborového systému, který zapříčiní okamžitý pád, nebo chybu při příštím zavádění operačního systému.

Z toho důvodu bylo rozhodnuto o nahrání kořenového souborového systému do DDR operační paměti zařízení (RAMFS), které mají zvolené vývojové desky dostatek (viz sekce 2.6). Veškeré změny provedené v souborovém systému budou následně při restartu nebo vypnutí zařízení zahozeny. Tím je minimalizováno riziko poškození souborového systému.

V tomto případě se nejedná o perzistentní úložiště dat, a z toho důvodu bude na flash vytvořený malý oddíl se souborovým systémem JFFS2, který se při zavádění systému připojí. Tento oddíl bude sloužit k perzistentnímu uchování konfiguračních souborů. V případě chyby, která zapříčiní nečitelnost konfiguračních souborů, budou uložená nastavení přepsána výchozí hodnotou.

3.2.4 Obsah perzistentního úložiště

V perzistentním úložišti bude uloženo nastavení celého zařízení. Toto nastavení bude při každém startu zařízení přečteno a aplikováno. Úložiště bude tedy obsahovat následující informace:

- IP adresa rozhraní
- Maska podsítě
- Výchozí brána
- Heslo k webové administraci
- MAC adresa zařízení

Speciální položkou je MAC adresa zařízení, kterou bude možné pouze číst. Tato položka slouží k počátečnímu nakonfigurování příslušné MAC adresy pro dané zařízení.

3.2.5 Komunikace s vytvořenými jednotkami

V části návrhu hardwarové platformy bylo rozhodnuto použít pro komunikaci s jednotkami rozhraní AXI4–Lite. Jedná se tedy o zařízení mapovaná do paměti. Zápis na určitou paměťovou adresu se projeví v registrovém poli implementované jednotky. Z toho důvodu je možné implementovat obslužný program jednotek třemi způsoby.

Nejjednodušším způsobem je implementovat program v uživatelském prostoru operačního systému. Komunikace by v tomto případě byla realizována

přímým zápisem do paměti. V obslužném programu jednotky by byl začátek adresního prostoru přiděleného jednotce uložen jako konstanta.

Výhodou výše popsaného řešení je velice rychlá a jednoduchá implementace. Naopak nevýhodou je potřeba překompilování a úprava kódu při změně přiděleného adresního prostoru a špatná přenositelnost takového programu na jiné zařízení. Tuto nevýhodu by bylo možné vyřešit implementací obslužného programu v prostoru jádra operačního systému.

Výhodou implementace jaderného ovladače je automatické přidělení adresového prostoru jednotky podle device-tree. Nevýhodou jsou vyšší časové nároky na vývoj a testování, jelikož jakákoliv chyba v ovladači zapříčiní pád celého operačního systému.

Posledním možným způsobem je implementovat obslužný program jako ovladač v uživatelském prostoru. V tomto případě jádro každé jednotce přidělí speciální ovladač s názvem Userspace IO. Tento program dostane přidělený adresový prostor implementované jednotky a vytvoří souborový deskriptor. Ovladač v uživatelském prostoru používá tento deskriptor pro komunikaci s jednotkou bez znalosti přesného adresního mapování .

Tento způsob efektivně řeší nevýhodu prvního řešení, jelikož je program odstíněn od přesných adresových rozsahů, které jsou přiděleny ovladači Userspace IO. Rovněž při chybě nedojde k pádu celého operačního systému, ale pouze k pádu programu. Z toho důvodu bylo rozhodnuto použít toto řešení.

3.2.6 Konfigurace Ethernetového rozhraní

Jedním z požadavků vedoucího práce je automatická konfigurace Ethernetového rozhraní. Plně automatická konfigurace je možné pouze v případě, že je v síti přítomný DHCP server (viz sekce 2.2.4).

V opačném případě je určitá uživatelská akce nutná. Po dohodě s vedoucím práce byla konfigurace rozdělena na tři možnosti:

- Konfigurace pomocí DHCP
- Konfigurace pomocí nastavených parametrů uživatelem. Tyto parametry jsou uloženy v perzistentním úložišti konfiguračních souborů (viz sekce 3.2.3).
- Výchozí konfigurace pevně uložená v zařízení, která se uplatní pokud není přítomné DHCP a ani není uložena žádná konfigurace.

Poslední konfigurace se uplatní hlavně během prvního použití k nastavení sítě. Předpokládá se, že uživatel připojí zařízení pomocí Ethernetu do počítače a nastaví jej pomocí webového rozhraní. Hodnoty výchozí konfigurace jsou zaneseny v tabulce 3.4.

Tabulka 3.4: Hodnoty výchozí konfigurace Ethernetového rozhraní

Ip adresa	192.168.1.254
Maska podsítě	255.255.255.0
Výchozí brána	192.168.1.1

3.2.7 Odesílání konfigurace

Požadavkem vedoucího práce bylo, aby zařízení po nakonfigurování z Ethernetového rozhraní začalo odesílat svou IP adresu na server sdružení CESNET

Po dohodě s vedoucím práce bylo rozhodnuto, že sdružení CESNET vytvoří jednoduché API, přijímající data ve formátu JavaScript Object Notation (JSON). Jedná se o jednoduchou strukturu, která se používá pro přenos informací mezi různými platformami. Jeho struktura je popsaná v [42]. Na sever bude zařízení každých pět minut odesílat HTTP paket obsahující JSON, ve kterém bude zapsána MAC adresa jako identifikátor a nakonfigurovaná IP adresa.

3.2.8 Vzdálená administrace

Pro vzdálenou administraci systému se nabízelo jednoduché řešení, a to přístup do konzole pomocí protokolu SSH. To vyžaduje zakompilování SSH serveru. Překladový systém Petalinux nabízí dva možné servery, a to Openssh a Dropbear.

Byl zvolen server Dropbear, kvůli nižší velikosti přeloženého binárního souboru než má server Openssh. Vzhledem k velice omezeným možnostem velikosti finálního obrazu se jedná o velikou výhodu.

3.2.9 Bezpečnost

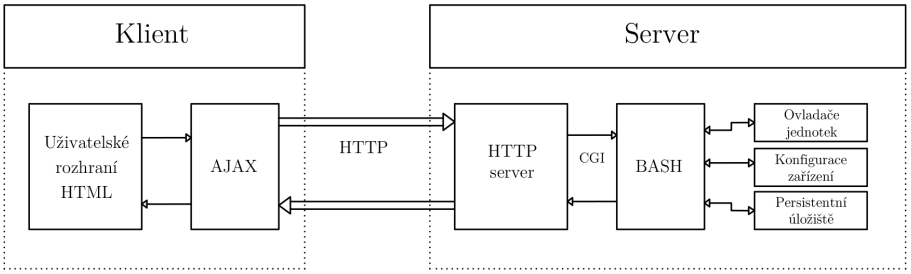
Vzhledem k tomu, že se jedná o zařízení neustále připojené do počítačové sítě, nesměla být bezpečnost v žádném případě podceněna.

Základním pilířem bezpečnosti je silné administrátorské heslo. K možnosti vytváření jiného uživatele než administrátora nebylo přistoupeno, protože jakýkoliv servisní zásah potřebuje administrátorská práva.

Největším problémem bylo zabezpečení webového rozhraní. Jednoduché a malé HTTP servery nepodporují šifrování pomocí protokolu HTTPS (miniweb, busybox). Navíc bez znalostí případné adresy zrcadla ani není možné vytvořit důvěryhodný certifikát. Další nevýhodou použití šifrování by bylo značné prodloužení doby načítání webového konfiguračního rozhraní, která se při testování možných řešení pohybovala v řádu nižších desítek sekund.

Z toho důvodu bylo po poradě s vedoucím práce rozhodnuto nepoužít protokol HTTPS a navrhnout webové rozhraní tak, aby případný útočník po získání přístupu nemohl zařízení nijak poškodit či zneužít. I přesto bude

Obrázek 3.5: Blokové schéma architektury webové aplikace.



přístup do webové administrace zařízení chráněn pomocí jednoduchého ověření přístupu v protokolu HTTP.

Dále bylo rozhodnuto, že zařízení bude blokovat veškerou příchozí komunikaci vyjma HTTP, SSH a ICMP. Aby byla minimalizována pravděpodobnost zneužití vzdáleného přístupu přes SSH, bude možné jej využít pouze z infrastruktury CESNET. Tím je možnost útoku na toto rozhraní výrazně omezena.

3.3 Webové rozhraní

Návrh webového rozhraní musí zohlednit rychlost platformy MicroBlaze. Z toho důvodu bylo rozhodnuto většinu funkcionalit implementovat na straně klienta, který si bude pomocí dynamického načítání žádat pouze o ta data, která si uživatel vyžádal.

3.3.1 Architektura

Architektura webové aplikace byla navržena tak, aby bylo možné požadavky zpracovávat co nejrychleji. Server poskytuje jednoduché API, pomocí kterého je možné vyčítat a zapisovat veškerá nastavení.

Dynamické generování obsahu bude implementováno pomocí jednoduchých skriptů v shellu bash. Vstupem skriptů bude HTTP post požadavek a výstupem data ve formátu JSON. Komunikace mezi skriptem a webovým serverem bude zařízena pomocí Common Gateway Interface (CGI).

Na uživatelské straně bude využito technologie AJAX, která bude asynchronně využívat vytvořené API. Schéma architektury je zobrazena v obrázku 3.5.

Výše popsaný způsob se jevil jako jediné možné řešení. Zkompilování robustních systémů jako PHP, Python nepřicházelo v úvahu vzhledem k velikosti flash a procesorovému výkonu zařízení.

3.3.2 Návrh uživatelského rozhraní

Pro implementaci uživatelského rozhraní bylo rozhodnuto o použití frameworku Bootstrap, pomocí kterého se dají velice rychle vytvářet responzivní webové stránky.

Je také potřeba počítat s pomalejší odezvou webu. Z toho důvodu bylo rozhodnuto přidat do uživatelského rozhraní točící se kolečko pro znázornění načítání. Ve chvíli načítaní je uživatelské rozhraní zablokováno, aby nedocházelo k panickému klikání uživatele, které by dobu odezvy stránky výrazně prodloužilo.

Uživatelské rozhraní bude rozděleno do do čtyř webových stránek, přičemž nastavení jednotky měření síťového jitteru nebude zobrazováno na desce Numato Mimas A7:

Přehled Na stránce bude zobrazeno:

- Základní přehled nastavení zařízení IP adresa, maska sítě, výchozí brána
- Nastavení jednotky zrcadla paketů
- Statistiky zrcadlených paketů
- Tlačítko resetu statistik paketů
- Verze firmwaru a datum překladu

Nastavení zrcadla Hlavním prvkem této stránky bude formulář obsahující nastavení jednotky, který bude předvyplněn na základě aktuálního nastavení:

- Zapnutí/Vypnutí jednotky
- Výběr mezi protokoly IPv4 a IPv6
- IP adresa zdroje
- IP adresa cíle (může být nevyplněná)
- Zdrojový UDP port (může být nevyplněný)
- Cílový UDP port (může být nevyplněný)
- Nastavení prohazování zdrojového a cílového portu

Nastavení zařízení Tato stránka bude sloužit k nastavení aktuální IP adresy zařízení a změně hesla do webové administrace. Skládá se tedy ze dvou formulářů. Formulář pro IP adresu zařízení bude předvyplněn na základě aktuálního nastavení.

Nastavení měření jitteru Jedná se o stránku zobrazenou pouze na desce Avnet KU-040-DB-G. Hlavním prvkem této stránky bude formulář s jednotlivými možnostmi nastavení, který bude předvyplněn aktuálním nastavením:

- Zapnutí/vypnutí jednotky
- Výběr mezi protokoly IPv4 a IPv6
- IP adresa zdroje
- zdrojový UDP port (může být nevyplněný)
- Cílový UDP port (může být nevyplněný)

Pod tímto formulářem budou přehledně zobrazena naměřená data a histogram hodnot uložených v paměti výsledků (viz sekce 3.1.3.5). Naměřená data budou aktualizována každých 10 sekeund a obsahují:

- Odhadnutý interval mezi pakety na zdroji
- Průměrný jitter
- Maximální naměřený jitter
- Průměrný jitter z hodnot uložených v paměti výsledků
- Směrodatná odchylka hodnot uložených v paměti výsledků

Kapitola 4

Realizace

4.1 Hardwarová platforma

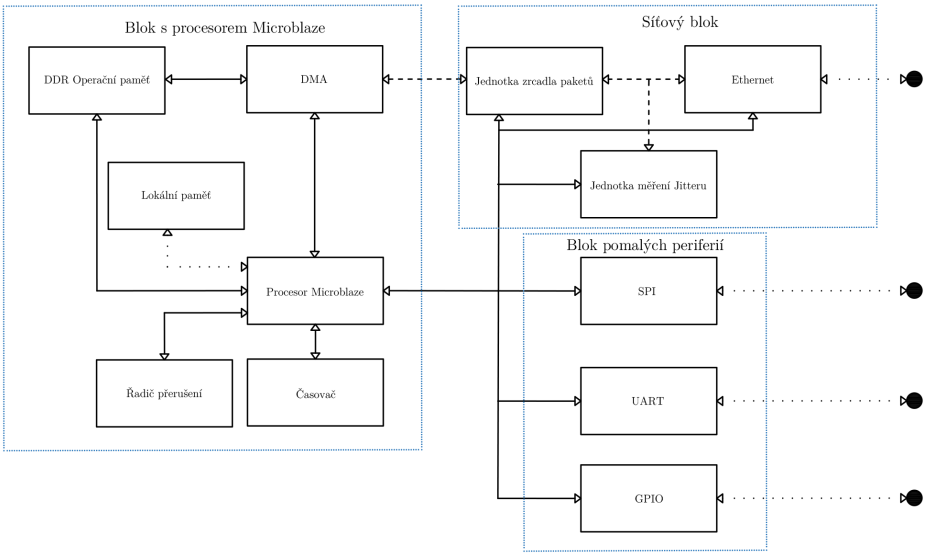
Rozdíly v architektuře v FPGA osazených na deskách vybraných vedoucím práce (viz sekce 2.6) zapříčinily vytvoření dvou hardwarových platforem. Při realizaci byl kladen důraz na to, aby byly obě platformy co nejvíce podobné a rozcházely se pouze v několika detailech.

Hardwarová platforma byla rozdělena do tří hlavních bloků. Jedná se o:

- Blok procesorového jádra MicroBlaze
- Síťový blok
- Blok pomalých periferií

Obrázek 4.1: Finální podoba vytvořeného zařízení.

Obrázek 4.2: Blokové schéma hardwarové platformy. Plnou čarou je značno propojení AXI4 a AXI4 Lite, čárkovaně propojení AXI4–Stream a tečkovaně je naznačené jiné propojení.



4.1.1 Datové propojení

Hardwarové platformy obsahují celkem dvě datová propojení. Pomalé periferie jsou připojené pomocí AXI4–Lite. Jedná se převážně o registrová rozhraní, pomocí kterých je do jader zapisováno nastavení. Toto propojení je realizováno pomocí jednoho křížového přepínače implementovaného v IP jádře AXI Interconnect, které je popsané v [43].

Druhé datové propojení obsahuje rychlé a časově náročné přenosy dat z DDR operační paměti pomocí AXI4. Používá jej mikroprocesor MicroBlaze a Ethernetové DMA. Pro realizaci tohoto propojení bylo použito jádro AXI Smartconnect popsané v [44].

4.1.2 Rozvody hodin

Vedení hodinového rozvodu je realizované na každé desce odlišně.

Hlavním rozdílem je rychlost hodin. Zatímco sběrnice a procesor MicroBlaze je na Kintex Ultrascale taktován frekvencí 100 MHz, tak v Artixu je frekvence hodin nakonfigurována na 60 MHz. Tato frekvence byla zvolena s ohledem na kritickou cestu uvnitř uvnitř procesoru MicroBlaze a také uvnitř křížového přepínače vedoucího AXI4–Stream do DDR paměti.

Hodiny o této frekvenci jsou zavedeny do jednotlivých bloků a je jimi taktována sběrnice AXI4 a AXI4-lite.

Tabulka 4.1: Nastavení procesoru MicroBlaze na jednotlivých deskách.

	Avnet KU040-DB-G	Numato Mimas A7
Konfigurace	Max. výkon	Max. frekvence
Instrukční a datová cache	ANO	ANO
Jednotka správy paměti	ANO	ANO
Výjimky	ANO	ANO
Násobička	ANO	ANO
HW podpora plov. des. čárky	ANO	NE
Predikce skoků	ANO	NE
Velikost instrukční cache	64 kB	32 kB
Velikost datové cache	64 kB	32 kB

4.1.3 Blok procesorového jádra MicroBlaze

4.1.3.1 Microblaze

Hlavním prvkem obou hardwarových platforem je procesorové IP jádro MicroBlaze. Na něj jsou pomocí lokální paměťové sběrnice (LMB) připojené paměti BRAM. Ty jsou inicializované při nahrání bitstreamu FPGA programem FSBL. To umožňuje zavedení operačního systému Linux (viz sekce 2.5). Velikost lokálních pamětí byla nakonfigurována na 16 KiB.

Dále je v procesoru MicroBlaze přítomné datová rozhraní AXI4 a paměťová rozhraní AXI4 s koherentním rozšířením pro přístup do DDR paměti.

Nastavení procesorového jádra se na jednotlivých FPGA liší. Pro větší Kintex Ultrascale byla zvolena konfigurace *Maximální výkon* popsaná v sekci 2.3.1.1, zatímco Artix byl nakonfigurován na *Maximální frekvenci*. Podrobnější nastavení obou procesorů je zobrazeno v tabulce 4.1.

4.1.3.2 Časovač

Pro časovač bylo použito IP jádro AXI Timer, které je popsané v [45]. Jedná se o dvoukanálový časovač potřebný k běhu operačního systému Linux (viz sekce 2.5).

AXI4–Lite rozhraní bylo připojeno na AXI Interconnect a přerušení od časovače bylo zapojeno na řadič přerušení.

4.1.3.3 Řadič přerušení

Jako řadič přerušení bylo použito IP jádro AXI Interrupt Controller popsané v [46]. Řadič byl připojen k procesorovému jádru MicroBlaze pomocí dedikovaného rozhraní pro přerušení. Nastavení řadiče je prováděno přes AXI4–Lite rozhraní přípojeného k AXI Interconnect.

4.1.3.4 Řadič externí paměti DDR

Deska Numato Mimas A7 je osazená 2Gbit externí paměti DDR3. Jako řadič této paměti bylo použito jádro Memory Interface Generator (7 series) popsané v [47]. Rychlost DDR3 rozhraní byla nakonfigurována na 800 MHz s šířkou slova 16 bitů. Šířka paměťové sběrnice AXI4 byla nastavena na nejvyšší možnou hodnotu a to 128 bitů, přičemž priorita požadavků byla nastavena tak, aby čtení mělo vždy přednost před zápisem.

Deska Avnet KU040-DB-G je osazená dvěma čipy paměti DDR4 s celkovou velikostí 8 Gbit. V tomto případě bylo jako řadič paměti zvoleno jádro DDR4 SDRAM (MIG) [48]. To bylo nakonfigurováno pro rychlost rozhraní 800 MHz se šířkou slova 32 bitů. Šířka paměťové sběrnice AXI4 byla nastavena na nejvyšši možnou hodnotu a to 256 bitů. Další nastavení týkající se přímo osazených čipů na desce byla provedena podle [32].

4.1.4 Blok pomalých periferií

Do tohoto bloku byly zařazeny periferie připojené přímo na pin FPGA a zároveň je komunikace s nimi realizována pomocí pomalejšího AXI4–Lite rozhraní. Všechny zde popsané periferie jsou připojené k AXI Interconnect jádru a mají zapojené přerušení do řadiče přerušení.

4.1.4.1 GPIO

Pro ovládání pinů všeobecného použití bylo použito IP jádro AXI GPIO popsané v [49]. Jednotlivé instance ovládají uživatelské přepínače, tlačítka a LED diody.

4.1.4.2 SPI rozhraní

Pro komunikaci s osazenou flash bylo použito jádro AXI Quad SPI popsané v [50]. Jelikož je na obou vývojových deskách (viz sekce 2.6) osazena flash čip od společnosti Micron, muselo být jádro nastaveno tak, aby bylo kompatibilní s tímto výrobcem.

Na desce Numato je osazena jediná paměť flash, ve které je uložen také bitstream. Tato flash je připojena na konfigurační banku FPGA, která není přístupná přímo z programovatelné logiky. Aby bylo možné k flash přistupovat i po nahrání bitstreamu, bylo SPI jádro nastavené na používání bloku STARTUPE2. Ten je implementován přímo na křemíku a zajišťuje přeposílání dat z konfigurační do uživatelské logiky [50] [51].

4.1.4.3 UART

Pro komunikaci na sériové lince bylo použito IP jádro AXI UART Lite popsané v [52]. Jádro bylo nakonfigurováno pro baudrate 115 200.

4.1.5 Síťový blok

V tomto bloku jsou implementované funkce nezbytné pro komunikaci po Ethernetovém rozhraním, jednotka zrcadla paketů a na desce Avnet KU040-DB-G také jednotka měření síťového jitteru.

4.1.5.1 Ethernetové rozhraní

Pro komunikaci na Ethernetové vrstvě MAC bylo použito jádro Tri-Mode Ethernet MAC (viz sekce 3.1.1). Toto jádro bylo nastaveno do módu používající RGMII pro komunikaci s fyzickou vrstvou. Rovněž bylo nastaveno používaní MDIO pro podporu konfigurace vrstev PHY ze softwaru. Tato dvě rozhraní byla následně připojena na pin FPGA.

Jádro bylo nastaveno tak, aby bylo možné provádět konfiguraci ze softwaru pomocí AXI4–Lite rozhraní, které bylo připojeno na AXI4 Interconnect.

Jádro Tri-Mode Ethernet MAC bylo připojeno podle [35] na 300 MHz (referenční signál) a 125 MHz (signál pro odesílací logiku) zdroj hodinového signálu.

4.1.5.2 Jednotka zrcadla paketů

Jednotka zrcadla paketů byla implementována podle návrhu v sekci 3.1.2 v jazyce VHDL. Přijímací a odesílací strana byla taktována hodinami generovanými v jádře Tri-Mode Ethernet MAC.

Přijatá paketová data jsou ukládána do 128 stupňů dlouhého posuvného registru. Jakmile jsou v něm přítomné MAC, IP a UDP hlavičky je zavolána funkce *swap_header_data*, která na základě čísla protokolu v MAC hlavičce rozhodne, zda se má zavolat *ipv4_mirror_match*, nebo *ipv6_mirror_match*.

V těchto dvou funkcích je implementován paketový filtr. Na základě hodnot uložených v registrovém poli zkontrolují přijatý paket a navrátí příslušnou logickou hodnotu.

Pokud paket zadaným hodnotám nevyhovuje, je nastaven výstupní multiplexor na propouštění dat do procesoru. V opačném případě jsou zavolány příslušné funkce *ipv4_swap*, nebo *ipv6_swap*, které podle hodnot nastavených v registrovém poli upraví paket a výstupní multiplexor je nastaven na odeslání dat zpět k Ethernetovému IP jádru.

Kontrolní součty jsou následně spočítány a vloženy do zrcadlených paketů na příslušná místa těsně před tím, než první bajt paketu opustí posuvný registr.

4.1.5.3 Jednotka měření síťového jitteru

Jednotka měření jitteru byla implementována pouze na desce KU040-DB-G (viz sekce 2.6) podle návrhu v sekci 3.1.3 v jazyce VHDL. Hodinový zdroj pro tuto jednotku je tvořen jádrem Tri-Mode Ethernet MAC, které jej generuje

přímo z Ethernetového rozhraní. Těmito hodinami jsou taktována data, která vstupují do posuvného registru o hloubce 128 stupňů.

Paketový filtr byl vytvořen stejným způsobem jako v sekci 4.1.5.2. Ve chvíli, kdy tento filtr detekuje shodu s hodnotami v registrovém poli, je vygenerován pulz na signálu *packet_match*, který je následně zpracováván měřícím automatem.

Tento automat ovládá měřící 64bitový časovač, který má přesnost 8 ns, děličku a výstupní registry.

Pro operaci dělení bylo využito IP jádro Divider Generator popsané v [53]. Toto jádro bylo nakonfigurováno pro dělení 64bitových čísel s největší možnou latencí 67 hodinových pulsů.

Naměřená hodnota jitteru je následně zmenšena na 32bitové číslo, které je uloženo do paměti BRAM o velikosti 32 KiB. Celkem paměť naměřených hodnot pojme až 8 192 hodnot.

Pro přístup do této paměti z procesoru MicroBlaze bylo použito jádro AXI BRAM Controller popsané v [54]. To bylo připojené na jádro AXI Interconnect. Na tento Interconnect bylo také připojeno konfigurační rozhraní registrového pole AXI4–Lite.

4.1.5.4 AXI Ethernet Buffer

Jádro je používáno jako buffer mezi procesorem a Ethernetovou MAC vrstvou. Buffer byl nastaven na velikost 8 Kib s podporou hardwarové akcelerace výpočtu IP, UDP a TCP kontrolních součtů.

Datový výstup této jednotky byl připojen na Jádro AXI DMA, které přijatá data ukládá do operační paměti procesoru MicroBlaze a konfigurační rozhraní AXI4–Lite bylo připojeno na AXI Interconnect.

4.2 Software

Stejně jako v hardwarové platformě bylo vytvořen prakticky identický software pro obě desky. Hlavním rozdílem je absence programového vybavení sloužící k ovládání jednotky měření jitteru na desce Numato Mimas, protože tato jednotka není v hardwarové platformě vytvořené pro tuto desku přítomná. Další výraznou změnou jsou také rozdílné Linuxové ovladače.

4.2.1 Ovladače vytvořených jednotek

Ovladače jednotek byly implementovány v uživatelském prostoru operačního systému 3.2.5 pomocí jazyka C. Po spuštění ovladače je otevřen příslušný soubor ve složce */dev/UIOx*, pomocí kterého je následně prováděna komunikace s jednotkou.

V této diplomové práci není prostor na zdokumentování veškerých implementovaných funkcí. Navíc většina z nich provádí pouze kontrolu uživatelského

vstupu nebo zápis do paměti. Z toho důvodu bylo rozhodnuto pouze podrobně popsat ovládání jednotlivých programů. Kompletní dokumentace v programu Doxygen je v elektronické příloze této práce.

4.2.1.1 Ovladač jednotky zrcadla

Jednotce zrcadla byl přidělen souborový deskriptor */dev/uio0*. Při spuštění programu dochází k otevření tohoto deskriptoru a následně s jeho pomocí přistupováno do registrového pole jednotky.

Ovladač je rozdělen na dvě části. Nastavování a čtení registrů a konfigurace Ethernetového rozhraní. Program se ovládá z příkazové řádky pomocí parametrů.

-set Značí, že uživatel chce provést zápis. Za tímto parametrem musí následovat volba a hodnota, kterou chce uživatel zapsat. Přičemž jednotlivé volby jsou:

- **-sip** Zdrojová IPv4 adresa, ze které bude přicházet paketový proud k obracení. Tato volba je povinná, v případě že není vyplněna IPv6 adresa zdroje. Bez správného vyplnění nebude zrcadlo spuštěno.
- **-dip** Cílová IPv4 adresa, na kterou bude přicházející paketový proud přeposílán. Tato volba je nepovinná. Pokud nebude vyplněná, bude paketový proud zrcadlen zpět k odesílateli.

  V případě že cílová IP adresa leží ve stejné síti jako zařízení, je vygenerován ARP dotaz pro zjištění MAC adresy cílového zařízení. V opačném případě se používá MAC adresa, ze které přichází paketový proud.
- **-sipv6** Zdrojová IPv6 adresa (viz -sip). Tato volba je povinná, pokud není vyplněna zdrojová IPv4 adresa.
- **-dipv6** Cílová IPv6 adresa (viz -dip).
- **-dmac** Nastavení cílové MAC adresy v případě že cílové zařízení neodpovídá na ARP.
- **-sport** Nastavení zdrojového UDP portu, ze kterého přichází paketový proud. V případě nenastavení přeposílá zrcadlo pakety z jakéhokoliv portu.
- **-dport** Nastavení cílového UDP portu, na který přichází paketový proud. V případě nenastavení přeposílá zrcadlo pakety mířící do jakéhokoliv portu.
- **-pswap** Zápisem 1 nastavíme zrcadlo do režimu prohazování cílového a zdrojového portu. Zápisem 0 toto prohazování vypneme

-read Značí, že uživatel chce přečíst nastavení zrcadla, nebo nastavení Ethernetového rozhraní. Za tímto parametrem musí následovat volba, kterou

chce uživatel přečíst. Volby jsou stejné jako u parametru *-set* a navíc jsou k nim přidány tyto:

-status Značí status zrcadlící jednotky. Pokud je vyčtena hodnota 1, znamená to, že je zrcadlo zapnuté. Hodnota 0 značí opak

-stat Touto volbou budou vyčtené statistiky. Počet přeposlaných paketů a bajtů (formátovaně).

-bytes Vyčte množství přeposlaných bajtů.

-packets Vyčte množství přeposlaných paketů.

-curip Vyčte aktuální nastavenou IPv4 adresu rozhraní.

-cruipv6 Vyčte aktuální nastavenou IPv6 adresu rozhraní

-curnmsk Vyčte aktuální nastavenou IPv4 masku podsítě.

-curnmskv6 Vyčte aktuální nastavenou IPv6 masku podsítě.

-start Tento argument neočekává další volbu. Po jeho zavolání dojde k zapnutí jednotky zrcadla.

-stop Po zavolání dojde k vypnutí jednotky zrcadla

-rstcnt Vyresetuje čítače statistik.

-reset Vyresetuje nastavení zrcadla i čítače statistik.

4.2.1.2 Ovladač jednotky měření jitteru

Ovladač měření jitteru bude přítomný pouze na desce Avnet KU040-DB-G a byl mu přidělen souborový deskriptor */dev/uio1*. V programu jsou použity stejné části kódu jako v ovladači pro ovládání jednotky zrcadla. Jedná se zejména o ověřování validního vstupu od uživatele. Parametry sloužící k ovládání jsou rovněž velice podobné.

-start Zapnutí zrcadla. Je očekáván ještě další parametr a to velikost učícího vzorku pro odhadnutí zdrojového intervalu mezi pakety (viz sekce 2.4.2).

-disable Pozastaví vykonávání měření a zápis do paměti výsledků (viz sekce 3.1.3.3).

-enable Spustí měření po zavolání parametru *-disable*. V tomto případě znovu nedochází k odhadu zdrojového intervalu mezi pakety.

-stop Zastaví jednotku a smaže veškerá uložená data.

-reset Smaže veškerá uložená data.

-set Značí, že uživatel chce provést zápis. Za tímto parametrem musí následovat volba a hodnota, kterou chce uživatel zapsat. Volbou může být:

- **-sip** Zdrojová IPv4 adresa, ze které bude přicházet paketový proud vyhovující požadavkům metody Interarrival Histogram. (viz sekce 2.4.2). Tato volba je povinná v případě že není vyplněna IPv6 adresa zdroje.
- **-sipv6** Zdrojová IPv6 adresa. Tato volba je povinná v případě, že není vyplněna IPv4 adresa zdroje.
- **-sport** Zdrojový port paketového proudu.
- **-dport** Cílový port paketového proudu.

-read Značí vyčítání hodnot z registrového pole. Za tímto parametrem musí následovat volba, kterou chce uživatel přečíst. Volby jsou stejné jako v případě argumentu *-set* a jsou k nim ještě přidány tyto:

- **-status** Značí status měření. Pokud je hodnota 1, znamená to, že probíhá měření. V opačném případě je vyčtena hodnota 0.
- **-estime** Vyčítá odhadnutou dobu mezi pakety s jakou vysílá zdroj. Hodnota je udávaná v milisekundách.
- **-avg** Vyčítá průměrnou hodnotu síťového jitteru od začátku měření. Hodnota je udávaná v milisekundách.
- **-ptp** Vyčítá naměřenou hodnotu maximálního síťového jitteru.
- **-lvalues** Vyčítá směrodatnou odchylku, střední hodnotu a rozptyl z posledních 8 192 naměřených vzorků.

-getsamples Zapíše do souboru obsah paměti výsledků (viz sekce 3.1.3.2). Jako volbu očekává název souboru, do kterého bude zapsáno.

4.2.2 Služby po spuštění

Do výsledného obrazu Linuxu bylo třeba přidat také dva programy, které souvisejí s automatickou konfigurací a jejím odesíláním. Oba běží jako služby na pozadí po celou dobu spuštění systému a byly implementovány pomocí skriptu v příkazovém procesoru bash.

4.2.2.1 Konfigurace Ethernetového rozhraní a nastavení zařízení

Tato služba má za úkol po spuštění připojit oddíl s perzistentním úložištěm a podle uložené konfigurace nastavit celé zařízení.

Po připojení se nejprve zkontroluje validita hodnot na perzistentním úložišti. Následně se nakonfigurují GPIO výstupy (LED diody a uživatelské přepínače) V dalším kroku je zkontrolováno, zda uživatel nechtěl vymazat

Tabulka 4.2: Význam led diod a uživatelských přepínačů na jednotlivých deskách

Avnet AES-KU-040	Numato Artix	Funkce
LED D24	LED D1	Konfigurace pomocí DHCP
LED D25	LED D2	Konfigurace z uloženého nastavení
LED D26	LED D3	Výchozí konfigurace
SW 2	SW 2	Reset do výchozího nastavení

nastavené hodnoty na perzistentním úložišti pomocí přepnutého SW 2. Pokud ano, pak jsou tyto soubory vyresetovány. Výchozí hodnota nastavení je uvedena v tabulce 3.4.

Dále je nastaveno heslo do webového serveru, nakonfigurována MAC adresa rozhraní a paketový filtr IP-tables.

Po této konfiguraci, je přistoupeno k zapnutí Ethernetového rozhraní a na základě uloženého nastavení je provedena konfigurace pomocí DHCP, nebo pevnou IP adresou (viz sekce 3.2.6).

Typ konfigurace je indikován také pomocí LED diod osazených na desce. Význam led diod a uživatelského přepínače je zanesen v tabulce 4.2.

4.2.2.2 Odesílání konfigurace

Konfigurace se odesílá pomocí HTTP post paketu s nakonfigurovanými daty ve formátu JSON (viz sekce 3.2.7).

Konfigurace se každých pět minut odesílá pomocí programu NetCat, který vytváří TCP spojení k cílovému serveru. Do tohoto spojení jsou následně skriptem vepsaná HTTP data a samotný JSON s MAC a IP adresou.

4.2.3 Webová aplikace

Webová aplikace musela být realizována s ohledem na typ a výkonnost zařízení. To spočívalo v co největší minimalizaci vkládaných souborů, jelikož protokol HTTP si pro každé načtení obrázku a souboru vytvoří nové TCP spojení, které procesor poměrně zatěžuje.

4.2.4 Uživatelské rozhraní

Uživatelské rozhraní bylo implementováno v jazyce HTML a CSS pomocí frameworku Bootstrap. Tím byla zaručena responzivita celého webu a možnost použití i na mobilním zařízení. Členění webu do jednotlivých stránek a funkcionality na stránkách byly implementovány podle sekce 3.3.2.

4.2.4.1 Serverové skripty

Na serverové straně bylo implementováno několik skriptů reprezentujících API. Těmto skriptům předává HTTP server na standardním vstupu HTTP obsah. Ten je v každém skriptu ošetřen proti speciálním znakům pomocí programu AWK a následně jsou data zpracována. Na adrese */cgi-bin* jsou dostupné následující skripty:

device-data vypíše JSON s aktuálním nastavením Ethernetového rozhraní.

device-set očekává na standardním vstupu nastavení Ethernetového rozhraní. Po nutné kontrole validity vstupu jsou data uložena do perzistentní paměti a je nastaveno Ethernetové rozhraní. Následně vypíše JSON s případným chybovým výstupem.

ipdv-data je přítomný pouze v desce Avnet AESKU040 a vypíše JSON s aktuálním nastavením jednotky měření jitteru.

ipdv-raw-data vypíše hodnoty naměřeného jitteru z paměti výsledků ve formátu JSON.

ipdv-set očekává na standardním vstupu nastavení jednotky měření jitteru. Po nastavení vypíše JSON s případným chybovým výstupem.

mirror-data vypíše JSON s hodnotami nastavení jednotky zrcadla.

mirror-set očekává na standardním vstupu nastavení jednotky zrcadla. Po nastavení vypíše JSON s chybovým výstupem.

overviw-data vypíše JSON s nastavením celého zařízení. Jedná se o konfiguraci Ethernetového rozhraní, jednotky zrcadla a verzi softwaru.

overview-stat vypíše JSON se statistikami zrcadlených paketů.

reset-stat vyresetuje statistiky zrcadla.

4.2.4.2 Javascript

Uživatelské nastavení se ze značné části skládá z podprogramů v JavaScriptu. Pro urychlení implementace bylo využito knihovny jQuery.

Při přístupu na jakoukoliv stránku je zavolána funkce z knihovny jQuery *getJSON*, která načte příslušná data pro stránku. V případě stránky s nastavením jednotky zrcadla paketů se jedná o aktuální nastavení. Tato data jsou následně rozparsována a vložena do příslušných polí na stránce. Jakmile je tato operace dokončena, dojde ke skrytí kolečka indikujícího načítání.

Naopak při nastavování je nejprve zobrazeno kolečko indikující načítání a následně je uživatelský vstup zkontrolován pomocí funkce *validate_form*. Pokud je v pořádku, jsou data odeslána na příslušnou API stránku pomocí

funkce post. V opačném případě je uživateli barevně vyznačen nevalidní vstup a zobrazena nápověda se správným formátem.

Po odeslání nastavení je znovu zparsován JSON s případnou chybou přímo z ovladače zařízení a skryto kolečko indikující načítání.

JavaScript je rovněž zodpovědný za dynamickou úpravu hodnot na stránce se statistkou zrcadlených paketů a výsledky měření síťového jitteru. Na těchto webových stránkách je každých deset sekund generován požadavek na datové API a dochází tak k průběžné aktualizaci hodnot.

4.2.5 Linux

Pro každou desku musel být vytvořen zvláštní projekt v překladovém prostředí Petalinux. Projekty musely být dva, jelikož jsou jednotlivé desky osazeny odlišnými komponentami a jsou tedy potřeba různé ovladače.

V této diplomové práci není prostor na podrobné popsání celé konfigurace překladového prostředí Petalinux, jelikož konfigurační soubory obsahují stovky záznamů. Z toho důvodu bylo rozhodnuto popsat pouze důležité prvky nastavení.

4.2.5.1 Nastavení překladu

Nastavení překladu se v prostředí Petalinux vytváří pomocí příkazu *petalinux-config*. Po jeho zavolání je zobrazeno grafické prostředí menu-config. V něm bylo zaškrtnuto používání zdrojových kódů z repositáře společnosti Xilinx a automatická konfigurace zavaděče U-Boot a FSBL. Tato automatická konfigurace nastaví adresové konstanty ve zdrojových kódech zavaděčů na správné hodnoty.

Nastavení hardwaru provedlo prostředí automaticky podle vygenerované hardwarové platformy. Byly pouze upraveny oddíly na flash paměti.

fpga slouží k uložení bitstreamu. Jeho přítomnost je nutná i v případě desky Avnet, kde je bitstream uložený na druhé flash. Bez tohoto oddílu nelze v prostředí Petalinux enviroment Linux vůbec přeložit.

boot zde je uložen zavaděč U-Boot.

bootenv umístění perzistentního nastavení zavaděče U-Boot

kernel slouží k uložení jádra a kořenového souborového systému

jffs2 oddíl k uložení perzistentního nastavení zařízení

Velikosti jednotlivých oddílů musí být zarovnány do bloků flash paměti, která na obou deskách činí 64 KiB. Výsledné hodnoty velikostí jsou zaneseny v tabulce 4.3.

Tabulka 4.3: Velikosti oddílu v paměti flash pro jednotlivé desky hexadecimálně v oktetech

	Avnet KU040	Numato Mimas
fpga	0x1 00 00	0x22 00 00
boot	0x7 00 00	0x8 00 00
bootenv	0x1 00 00	0x2 00 00
kernel	0xD0 00 00	0xAA 00 00
jffs2	0xC 00 00 00	0x1E 00 00

Následně bylo nastaveno zavádění ze souborového systému umístěného v operační paměti. Tento souborový systém bude uložen v paměti flash na stejném oddílu jako jádro.

4.2.5.2 Konfigurace device-tree

Prostředí Petalinux dokáže ve značné míře vygenerovat konfiguraci samo z implementované hardwarové platformy. Vygenerované device-tree obsahuje ale pouze IP jádra společnosti Xilinx. Z toho důvodu bylo třeba device-tree doplnit o další informace. Jedná se například o jednotku zrcadla paketů a v případě desky Avnet KU040-DB-G také jednotku měření jitteru.

Do device-tree byly doplněny adresy, na které jsou v hardwarové platformě jednotky mapovány, a řetězec kompatibility, podle kterého operační systém zavolá příslušný ovladač. V tomto případě byl zvolen řetězec *generic-uio*, který zajistí, že k jednotce bude možné přistupovat z ovladače v uživatelském prostoru 3.2.5. V případě desky Avnet bylo ještě třeba přidat paměť naměřených výsledků síťového jitteru a označit ji jako rezervovanou, aby ji operační systém nevyužíval.

Dále bylo potřeba upravit typ Ethernetové fyzické vrstvy. Deska Avnet KU040-DB-G má osazený čip od Texas Instrumens *DP83867*. V tomto případě byl jako řetězec kompatibility vyplněn *ti,dp83867* a adresa zařízení byla nastavena na hodnotu *0x03*. Ovladač tohoto čipu potřebuje zároveň znát i hodnotu zpoždění propagace jednotlivých signálů, která byla podle [32] nastavena na 1 ns a velikost vnitřní fronty byla zvolena 4 oktety.

Deska Numato Artix A7 je osazena čipem od společnosti Realtek *RTL8211E*. Řetězec kompatibility byl tedy vyplněn hodnotou *realtek,RTL8211E* a adresa zařízení byla nastaven na hodnotu *0x03*.

Dále bylo třeba nastavit v device-tree uživatelskou flash. V případě desky Avnet byla zvolena hodnota řetězce kompatibility *micron,n25q256a* a v případě desky Numato byla zadána hodnota *micron,n25q128a13*.

4.2.5.3 Konfigurace jádra Linuxu

Jádro bylo na obě desky nakonfigurováno tak, aby bylo co nejmenší. Z výchozí konfigurace byly odstraněny všechny nepotřebné ovladače. Na obou deskách byly vybrány ovladače pro příslušný čip Ethernetové fyzické vrstvy. Dále byly přidány ovladače pro SPI jádro (viz sekce 4.1.4.2) a osazenou flash.

Do jádra byly přidány ovladače implementující abstrakci nad SPI flash aby bylo možné do dané flash zapisovat přímo z Linuxu. Dále byly přidány ovladače *NETFILTER* nutné ke správné funkci IP-tables.

Následně byla přidána podpora souborového systému JFFS2 který bude použit pro perzistentní úložiště dat. Pro správnou funkci Ethernetového rozhraní musel být také zakompilován ovladač Ethernetového jádra Tri-Mode Ethernet MAC (viz sekce 3.1.1). Podpora síťových protokolů jako IPv4 a IPv6 byla rovněž přidána.

4.2.5.4 Konfigurace kořenového filesystému

Filesystém byl v obou deskách nastaven shodně, kromě ovladače jednotky jitteru, který nebyl přidán na desku Numato.

Nejdůležitějším balíkem programů, který byl zkompilován je Busybox. Jeho podnázev Švýcarský nůž vestavného Linuxu nemůže být výstižnější. Obsahuje zmenšené a zjednodušené verze nejdůležitějších Linuxových programů. Byl napsán pro zařízení s malými prostředky a je extrémně modulární [55].

Pomocí konfiguračního souboru lze jednoduše vybrat velikost a obsáhlost tohoto balíku. V implementovaném zařízení obsahuje Busybox přibližně 50 programů. Naprostá většina jsou základní programy pro práci se soubory *(cp, mv)*, nastavování práv *(chmod, chown)* a čtení souborů *(cat)*. Balík obsahuje také webový server *(httpd)*, dhcp klienta *(udhcpc)* a síťové klienty *(wget a netcat)*.

Dále byly do kořenového filesystému přidán jako výchozí příkazový procesor bash, ssh-server *(dropbear)* a implementované služby po spuštění. Do adresáře */srv/www* byla vložena webová aplikace.

Do adresáře */usr/bin* byly vloženy implementované ovladače jednotek. Dále byl také zkompilován balík *MTD utils* obsahující programy pro čtení a zápis do flash přímo z Linuxu.

Jiné balíky nebyly přidány z důvodu malé flash. Linux tedy neobsahuje interpretr žádného skriptovacího jazyku kromě bash.

4.2.5.5 Konfigurace zavaděče U-Boot

V zavaděči bylo nastaveno používání ovladačového subsystému. Tím se zakompiloval kód, který zavádí ovladačovou abstrakci a je možné používat přítomné periferie. Toto nastavení bylo zvoleno, aby byl v U-Bootu funkční Ethernet a bylo případně možné zavádět operační systém ze síťového úložiště.

Dále byl přikompilován ovladač Ethernetové fyzické vrstvy pro příslušnou desku a ovladače pro jádro Xilinx Tri Mode Ethernet MAC (viz sekce 3.1.1).

Pro správné zavedení z SPI flash byly přidány ovladače pro použité jádro Quad SPI (viz sekce 4.1.4.2) a generické ovladače pro flash.

Kapitola 5

Testování

Testování všech komponent zařízení probíhalo průběžně během procesu návrhu i realizace. Během tvorby této diplomové práce bylo vytvořeno velké množství funkčních vzorků, na kterých byla funkčnost a adekvátnost zvolených řešení důkladně testována. Tato kapitola popisuje metody použité k ověření funkčnosti jednotlivých komponent a následné otestování zařízení jako celku.

5.1 Hardwarová platforma

5.1.1 Jednotka zrcadla paketů

5.1.1.1 Simulace jednotky

Jednotka byla v počátku vývoje testována pomocí simulace. Byl vytvořený testbench v jazyce VHDL, který vkládal paketová data do jednotky zrcadla po AXI4–Stream rozhraní se stejným časováním jako jádro Tri-Mode Ethernet MAC.

Paketová data byla získána pomocí programu Wireshark z reálného síťového provozu. Během simulace nebyly zjištěny žádné chyby a jednotka správně zrcadlila a přeposílala UDP pakety. Simulace tedy úspěšně ověřila funkčnost VHDL modelu jednotky.

5.1.1.2 Testování na reálné síti

Testování na reálné síti probíhalo v několika krocích na obou deskách shodně. Jednotka byla implementována a nahrána do FPGA. Dále byl paketový filtr nastaven na IP adresu zdrojového počítače a cílový port byl nastaven na 9999, přičemž docházelo k prohazování zdrojového a cílového portu.

Na tento port byl pomocí programu NetCat odeslán textový soubor o velikosti 1 MB. Vzhledem k prohazování portů program rovnou přijímal odeslaná data. Shodnost souborů byla ověřena pomocí hashovacího algoritmu MD5.

Shodným způsobem bylo testováno i přeposílání paketů k jinému příjemci. Jediným rozdílem bylo nastavení jednotky, kde byla vyplněna také cílová MAC i IP adresa. Oba tyto testy byly úspěšné.

Dále byla jednotka otestována pomocí paketového generátoru a analyzátoru IXIA, který generoval a zároveň vyhodnocoval přicházející proud paketů. Test generátoru potvrdil, že implementovaná jednotka je schopná na obou deskách bezchybně zrcadlit UDP pakety na síti Ethernet až do plné rychlosti rozhraní.

Poslední test jednotky zrcadla paketů byl proveden pomocí platformy MVTP. Zařízení s připojenou kamerou odesílalo datový proud na paketové zrcadlo. To bylo nakonfigurováno, tak aby přeposílalo přijaté pakety k cílové platformě MVTP s připojeným displejem. Na tomto displeji byl bezchybně zobrazený obraz z kamery. Celý test trval kolem pěti minut během kterých nebyla zaznamenána žádná chyba v přenosu.

5.1.2 Jednotka měření jitteru

5.1.3 Simulace

Pro otestování funkčnosti během návrhu a implementace jednotky byl vytvořen VHDL testbench. Tato simulace testovala pouze výpočet, jelikož paketový filtr byl otestován již u jednotky zrcadla.

Simulace nejprve testovala správné odhadnutí doby mezi pakety pomocí generování pulsu příchozího paketu s přesnou periodou. Jednotka v tomto případě správně odhadla dobu mezi pakety a měřila nulový jitter.

V následujícím kroku simulační prostředí generovalo časy příchodu paketů náhodně. Prostředí dále vyhodnocovalo výstup jednotky. Hodnoty vypočítané jednotkou a simulací byly vždy stejné. Tímto způsobem byla tedy ověřena správnost VHDL modelu v simulaci.

5.1.4 Testování s přesným generátorem paketů

Pro správné otestování jednotky bylo třeba vytvořit síťové zařízení, které bude vysílat pakety s přesným časováním. Z toho důvodu byl vytvořen na desce Avnet KU040-DB-G generátor paketů s přesnou dobou mezi pakety nastavenou na 1 ms.

Jednotka měření síťového jitteru byla implementovaná a nahrána na druhou desku Avnet, přičemž tyto dvě desky byly propojeny napřímo Ethernetovým kabelem. Použití stejných desek a přímé propojení bylo zvoleno z toho důvodu, aby byly minimalizovány časové nepřesnosti v Ethernetových fyzických vrstvách.

Jednotka odhadla dobu mezi pakety na zdroji pomocí 50 vzorků správně a naměřila průměrný jitter 0 ns. Přičemž maximální jitter byl naměřen 8 ns. Tato hodnota je přesně jeden hodinový pulz a je nejspíše způsobena

časovou nepřesností na Ethernetových fyzických vrstvách. Tento testy byl tedy úspěšný.

5.1.5 Integrace všech IP jader dohromady

Celou hardwarovou platformu bylo možné otestovat pouze funkčně pomocí softwaru. Test hardwarové platformy byl tedy proveden úspěšným zkompilováním a zavedením linuxového operačního systému. Systém se úspěšně zavedl z SPI flash, na kterou bylo možné i zapisovat.

Fungující Ethernetové rozhraní bylo otestováno stažením souboru pomocí programu wget a kontrolu jeho integrity hashovacím algoritmem MD5.

5.2 Software

Testování softwaru probíhalo na obou dvou deskách úplně totožně. V této sekci je proto popsáno testování pouze desky Avnet KU040-DB-G, které oproti desce Numato obsahuje navíc i části týkající se měření jitteru.

5.2.1 Ovladače implementovaných jednotek

Ovladače implementovaných jednotek byly otestovány nejprve zadáváním nevalidních vstupů. Ovladač úspěšně upozorňoval uživatele a zápis neprovedl.

Dále bylo testováno, zda se zapsané hodnoty skutečně objevují v registrovém poli dané jednotky. Pomocí hardwarového ladícího nástroje bylo provedeno vyčtení hodnot z registrů, které byly shodné s hodnotami zadanými v ovladači.

Dále byly ověřeny funkcionality pro jednotlivé ovladače jako zápis a následné vyčtení hodnot.

Pro jednotku měřící síťový jitter byly vyčteny hodnoty z paměti výsledků. Následně byla úspěšně porovnána adekvátnost vyčtených hodnot s hardwarově naměřeným průměrným jitterem.

5.2.2 Webové rozhraní

Ve webovém rozhraní byla nejprve zkontrolována bezpečnost. Při zadání nesprávného hesla systém odmítl přihlášení. Následně bylo úspěšně otestováno nakonfigurování jednotky zrcadla. Správné nastavení bylo otestováno funkčně pomocí datového proudu z paketového generátoru IXIA, který vyhovoval nastaveným parametrům.

Testování jednotky měření jitteru probíhalo rovněž pomocí paketového generátoru IXIA. Webové rozhraní zobrazovalo naměřené hodnoty a histogram hodnot vyčtených z paměti naměřených dat.

Dále bylo úspěšně otestováno nakonfigurování Ethernetového rozhraní, změna hesla i průběžné aktualizování statistik zrcadlených paketů a hodnot síťového jitteru.

5.2.3 Bezpečnost

Ověřování bezpečnosti zařízení bylo omezeno na dostupné prostředky. Nejprve bylo otestováno, že při zadání nesprávného hesla není možné se do zařízení dostat ani přes sériovou linku, ani přes ssh. V žádném z těchto pokusů nebyl povolen přístup do zařízení.

Dále byly na zařízení otestovány otevřené porty pomocí programu Netstat. Zařízení neblokovalo pouze HTTP a SSH porty na protokolu TCP. Jiný pokus o komunikaci není tedy možný.

Dále byl proveden pokus o připojení se na zařízení pomocí SSH ze sítě nepatřící do infrastruktury CESNET. Tento test proběhl úspěšně, jelikož nedošlo k připojení.

5.2.4 Konfigurace

Zařízení bylo připojeno do sítě s aktivním DHCP klientem a spuštěno. Došlo k úspěšnému nakonfigurování pomocí DHCP indikované rozsvícením příslušné LED diody.

Následně bylo zařízení vypnuto a připojeno do sítě bez DHCP serveru a bylo úspěšně otestováno, že po zapnutí dojde k výchozímu nastavení Ethernetového rozhraní a rozsvícení příslušné LED diody.

V dalším kroku bylo pomocí webového rozhraní nakonfigurováno Ethernetové rozhraní pevnou adresou. Přepnutí konfigurace bylo indikováno rozsvícením příslušné LED diody a zhasnutím diody indikující výchozí konfiguraci.

Pomocí programu ifconfig byla zkontrolována IP adresa Ethernetového rozhraní, která souhlasila s nastavenou. Po restartu zařízení došlo ke správnému nakonfigurování z předchozího nastavení.

Výše popsané chování je přesně v souladu s návrhem popsaným v sekci 3.2.6, test byl tedy úspěšný.

5.2.5 Odesílání konfigurace

Zařízení bylo ponecháno zapnuté a připojené do Ethernetové sítě po dobu jednoho dne. Na cílovém serveru sdružení CESNET bylo následně zjištěno, že zařízení po celý den každých pět minut odeslalo svoji konfiguraci. Test byl z toho důvodu úspěšný.

5.3 Celé zařízení

Otestování zařízení jako celku bylo provedeno pomocí paketového generátoru IXIA. Zařízení bylo propojeno přímo s generátorem pomocí přepínače. Do tohoto přepínače byl připojený i laptop.

Vytvořené zařízení bylo zapnuto. Jakmile LED dioda indikovala výchozí konfiguraci Ethernetového rozhraní, byla pomocí laptopu načtena webová stránka s konfigurací. Na ní bylo nastaveno zrcadlení paketů a měření síťového jitteru z IP adresy 192.168.1.100 ze zdrojového portu 10 000 na cílový port 20 000. Přičemž paketový generátor byl nakonfigurován tak, aby odesílal tyto hodnoty.

Z webového rozhraní byly následně úspěšně vyčítány statistiky zrcadlení, které byly shodné se statistikami na generátoru IXIA. Naměřený jitter byl rovněž zanedbatelný. Test byl tedy úspěšný.

Kapitola 6

Měření výsledků a srovnání

V této kapitole jsou popsány metody sloužící k porovnání výkonu implementovaných jednotek v FPGA. Naměřené výsledky jsou následně porovnány s jiným způsobem řešení daného problému.

6.1 Zrcadlení paketů

Propustnost paketového zrcadla implementovaného v FPGA byla porovnávána s implementací pomocí IP-tables a nástrojem hd-rum z balíku Ultragrid, což je softwarová implementace paketového zrcadla.

Tato paketová zrcadla byla připojena přímo k paketovému generátoru a analyzátoru Ixia. Testování propustnosti bylo provedeno na Ethernetových rámcích dlouhých 64, 1 518 bajtů a náhodnou velikostí mezi 64 a 1 518 bajtů s rovnoměrným rozdělením. Tyto pakety byly generovány na 25, 50, 75 a 100% propustnosti linky. Každé měření bylo spuštěné jednu minutu. Výsledky měření ukazují procentuální poměr mezi odeslanými a zrcadlenými pakety a jsou zobrazeny v tabulkách 6.1 až 6.3.

Audiovizuální přenosy obvykle obsahuji pakety různé délky. Například standard *AES67* popisující přenos zvuku přes počítačovou síť vyžaduje pakety, které obsahují 1 ms zvukové stopy a doporučuje, aby byly podporovány také pakety přenášející pouze 250 µs zvuku [56]. To při při použití klasické

Tabulka 6.1: Naměřené výsledky pro paketové zrcadlo implementované pomocí IP tables

Zatížení linky	64B rámce	1518B rámce	Náhodná vel. rámců
25%	97,23%	100%	100%
50%	33,14%	100%	100%
75%	38,69%	100%	100%
100%	24,56%	99.98%	99.99

Tabulka 6.2: Naměřené výsledky pro paketové zrcadlo implementované pomocí nástroje hd-rum

Zatížení linky	64B rámce	1518B rámce	Náhodná vel. rámců
25%	43,63%	100%	100%
50%	22,35%	100%	100%
75%	15,18%	100%	99.94%
100%	11,50%	100%	99.95

Tabulka 6.3: Naměřené výsledky pro paketové zrcadlo implementované v FPGA

Zatížení linky	64B rámce	1518B rámce	Náhodná vel. rámců
25%	100%	100%	100%
50%	100%	100%	100%
75%	100%	100%	100%
100%	100%	100%	100%

vzorkovací frekvence 48 kHz a 24bit vzorků znamená, že 250µs paket obsahuje $12 \cdot 3$ bajtů v datové části UDP paketu. Ethernetový rámec má tedy celkem 82 bajtů. Z toho důvodu je velice důležité, aby zařízení dokázalo zrcadlit i krátké pakety.

Paketové zrcadlo implementované v FPGA dosahuje 100% propustnosti 1Gbps Ethernetu na všech délkách paketů, zatímco zbylá dvě řešení jsou schopná uspokojivě zrcadlit pouze dlouhé rámce. Na krátkých rámcích dosahují lepších výsledků IP-tables, které dokáží zrcadlit přibližně dvakrát více paketů než nástroj hd-rum.

6.2 Měření jitteru

Přesnost měření jitteru byla posuzována pomocí generátoru paketů s přesnými časovými charakteristikami popsaného v sekci 5.1.4.

V generátoru bylo nastaveno odesílání paketů s určitým síťovým jitterem. Parametry tohoto jitteru byly vybrány z normálního rozdělení $\mathbb{N}(0, 100)$ a uloženy do generátoru.

Tabulka 6.4: Parametry síťového jitteru odesílaného generátorem s přesnými časovými charakteristikami

Name	Value
Průměrný jitter	653.584 ns
Směrodatná odchylka	493.314 ns
Maximální jitter	5,216 ns

Generátor byl propojen přímo s deskou Avnet KU040-DB-G pomocí Ethernetového kabelu. Doba učící fáze byla v jednotce nastavena na 4 000 vzorků. Výsledky měření jsou zobrazeny v tabulce. Sloupeček rozdílu ukazuje procentuální rozdíl mezi jittrem na generátoru a měřeným jittrem. Všechny hodnoty jsou pod 1 %, což ukazuje na vysokou přesnost měření i na síti s větší hodnotou síťového jitteru.

Tabulka 6.5: Naměřené hodnoty síťového jitteru

Název	Hodnota	Hod. takty	Generátor	Rozdíl
Doba mezi pakety na gen.	1 ms	–	1 ms	–
Průměrný jitter	648 ns	81	653.584 ns	0.854 %
Maximální jitter	5,224 ns	653	5,216 ns	0.152 %
Posledních 8 000 prům.	654 ns	–	653.584 ns	0.063 %
Posledních 8 000 Směr. odch.	493 ns	–	493.314 ns	0.063 %

Jednotka tedy dosahuje přesnosti 8 ns. Jedná se o řádově vyšší hodnotu, než udávají metody popsané v odborné literatuře. Z velké části je tento výsledek dán použitím FPGA místo klasického mikroprocesoru. Jednotka ovšem překonává i předchozí pokusy měření síťového jitteru pomocí specializovaných hardwarových obvodů. V [57] byla implementovaná Wi–Fi karta v FPGA, která dokáže měřit zpoždění sítě pouze v mikrosekundách.

Závěr

Diplomová práce se zabývá návrhem a realizací zařízení pro testování Ethernetových sítí. Zařízení je dostupné ve dvou verzích. Deska Numato Mimas A7 osazená FPGA čipem Artix je schopná zrcadlení a přeposílání paketového proudu v plné rychlosti rozhraní 1Gbps Ethernet. Parametry tohoto zařízení byly porovnány s jinými softwarovými implementacemi, které na krátkých paketech dosahovaly propustnosti maximálně 25 %.

Pro automatickou konfiguraci a vzdálenou administraci byl v FPGA čipu spolu se speciální jednotkou zrcadlení paketů implementován i softprocesor MicroBlaze, pro který byl zkompilován vlastní Linuxový operační systém. Deska je po připojení do sítě schopna automatické konfigurace přes DHCP a po nakonfigurování odesílá na servery sdružení CESNET své aktuální nastavení.

Zařízení je možné spravovat přes standardní protokol SSH nebo přes jednoduchou webovou aplikaci.

Na desce Avnet KU040-DB-G osazené dražším čipem FPGA Kintex je mimo výše popsaného pro desku Numato implementováno také měření síťového jitteru. To je realizováno pomocí speciální jednotky uvnitř FPGA a dosahuje přesnosti až 8 ns. Naměřená data je možné vyčítat pomocí webového rozhraní.

Výše popsaná zařízení byla navržena, zkonstruována a otestována jako funkční celek. Sdružení CESNET předpokládá, že umístí desítku těchto zařízení implementovaných na desce Numato Mimas A7 u jejich zahraničních partnerů. Vytvoří se tak celoevropská testovací infrastruktura pro nízkolatenční přenosy. V budoucnu se předpokládá také nasazení desek Avnet, na kterých bude implementováno paketové zrcadlo s rozhraním 10Gbps Ethernet.

Literatura

[1] Half of all internet traffic goes to Netflix and YouTube. 2018. Dostupné z: `https://testinternetspeed.org/blog/half-of-all-internet-traffic-goes-to-netflix-and-youtube/`

[2] Janeček, J.; Bílý, M.: *Lokální sítě*. ČVUT, vyd. 3 vydání, 2008, ISBN 978-80-01-04014-0.

[3] Li, Y.; Li, D.; Cui, W.; aj.: Research based on OSI model. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, May 2011, s. 554–557, doi:10.1109/ICCSN.2011.6014631.

[4] IEEE Standard for Ethernet - Redline. *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015) - Redline*, Aug 2018: s. 1–8275.

[5] Prodanoff, Z. G.; King, R.: CRC32 Based Signature Generation Methods for URL Routing. In *IEEE SoutheastCon, 2004. Proceedings.*, March 2004, s. 153–158, doi:10.1109/SECON.2004.1287910.

[6] Pai, Y.; Cheng, F.; Lu, S.; aj.: Sub-Trees Modification of Huffman Coding for Stuffing Bits Reduction and Efficient NRZI Data Transmission. *IEEE Transactions on Broadcasting*, ročník 58, č. 2, June 2012: s. 221–227, ISSN 0018-9316, doi:10.1109/TBC.2012.2189610.

[7] Internet Protocol. RFC 791 [online], Září 1981, doi:10.17487/RFC0791. Dostupné z: `https://rfc-editor.org/rfc/rfc791.txt`

[8] Whatever happened to the IPv4 address crisis? Dostupné z: `https://www.networkworld.com/article/2174297/whatever-happened-to-the-ipv4-address-crisis-.html`

[9] Deering, D. S. E.; Hinden, B.: Internet Protocol, Version 6 (IPv6) Specification. RFC 8200 [online], Červenec 2017, doi:10.17487/RFC8200. Dostupné z: `https://rfc-editor.org/rfc/rfc8200.txt`

[10] User Datagram Protocol. RFC 768 [online], Srpen 1980, doi:10.17487/RFC0768. Dostupné z: https://rfc-editor.org/rfc/rfc768.txt

[11] Braden, R. T.: Requirements for Internet Hosts - Communication Layers. RFC 1122, Říjen 1989, doi:10.17487/RFC1122. Dostupné z: https://rfc-editor.org/rfc/rfc1122.txt

[12] Fairhurst, G.; Westerlund, M.: Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums. RFC 6936, Duben 2013, doi:10.17487/RFC6936. Dostupné z: https://rfc-editor.org/rfc/rfc6936.txt

[13] An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826, Listopad 1982, doi:10.17487/RFC0826. Dostupné z: https://rfc-editor.org/rfc/rfc826.txt

[14] Moon, D.; Lee, J. D.; Jeong, Y.-S.; aj.: RTNSS: a routing trace-based network security system for preventing ARP spoofing attacks. *The Journal of Supercomputing*, ročník 72, č. 5, May 2016: s. 1740–1756, ISSN 1573-0484, doi:10.1007/s11227-014-1353-0. Dostupné z: https://doi.org/10.1007/s11227-014-1353-0

[15] Droms, R.: Dynamic Host Configuration Protocol. RFC 2131, Březen 1997, doi:10.17487/RFC2131. Dostupné z: https://rfc-editor.org/rfc/rfc2131.txt

[16] Xilinx, Inc.: *MicroBlaze Processor: Reference Guide [online]*. [cit. 2019-02-18]. Dostupné z: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_2/ug984-vivado-microblaze-ref.pdf

[17] Xilinx, Inc.: *Vivado Design Suite - Vivado AXI Reference [online]*. 2017, [cit. 2019-02-18]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_ref_guide/latest/ug1037-vivado-axi-reference-guide.pdf

[18] ARM: *AMBA AXI and ACE Protocol Specification [online]*. 2011, [cit. 2019-02-18]. Dostupné z: https://silver.arm.com/download/download.tm?pv=1198016

[19] ARM: *AMBA 4 AXI4-Stream Protocol [online]*. 2010, [cit. 2019-02-18]. Dostupné z: https://silver.arm.com/download/download.tm?pv=1074010

[20] Demichelis, C.; Chimento, P.: IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). 2002, doi:DOI10.17487/RFC3393. Dostupné z: https://www.rfc-editor.org/refs/ref3393.txt

[21] Understanding Jitter in Packet Voice Networks (Cisco IOS Platforms)[online]. 2006, [cit. 2019-02-18]. Dostupné z: https://www.cisco.com/c/en/us/support/docs/voice/voice-quality/18902-jitter-packet-voice.html

[22] Freiberger, M.; Xia, T. J.; Peterson, D. L.; aj.: Demonstration of a novel method for real-time network latency measurements in the optical transport network using G.709 overhead. In *2009 Conference on Optical Fiber Communication - incudes post deadline papers*, March 2009, s. 1–3, doi:10.1364/NFOEC.2009.NTuC4.

[23] Watt, S. T.; Achanta, S.; Abubakari, H.; aj.: Understanding and applying precision time protocol. In *2015 Saudi Arabia Smart Grid (SASG)*, Dec 2015, s. 1–7, doi:10.1109/SASG.2015.7449285.

[24] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems - Redline. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, July 2008: s. 1–300.

[25] Berger-Sabbatel, G.; Grunenberger, Y.; Heusse, M.; aj.: Interarrival Histograms: A Method for Measuring Transmission Delays in 802.11 WLANs. 01 2007.

[26] Smirnov, N.; Dunin-Barkovskii, I.: *Mathematische statistik in der technik*. Deutscher Verlag der Wissenschaften, 1969.

[27] Vychodil, V.: *Operační systém Linux*. Computer Press, vyd. 1 vydání, 2003, ISBN 80-7226-333-1.

[28] Where Linux rules: Supercomputers. 2013. Dostupné z: https://www.zdnet.com/article/20-great-years-of-linux-and-supercomputers/

[29] Ph.D., I. M. S.: *Jádro operačního systému Linux [online]*. 2015, [cit. 2019-02-22]. Dostupné z: https://moodle.fit.cvut.cz/course/format/wiki/mediafile.php?id=842&path=%2flectures%2fmi-oli-prednasky.pdf

[30] Xilinx, Inc.: *PetaLinux Tools Documentation [online]*. 2017, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug1156-petalinux-tools-workflow-tutorial.pdf

[31] Mimas Artix 7 FPGA Development Board with DDR SDRAM and Gigabit Ethernet. 2018. Dostupné z: https://numato.com/docs/mimas-artix-7-fpga-development-board-with-ddr-sdram-and-gigabit-ethernet/

[32] Avnet, Inc.: *Kintex UltraScale KU040 Development Board [online]*. 2015, [cit. 2019-02-21]. Dostupné z: https://www.avnet.com/opasdata/d120001/medias/docus/13/aes-AES-KU040-DB-G-User-Guide.pdf

[33] Navratil, J.; Ubik, S.: Surgery telepresence for universities and symposia. In *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Oct 2012, s. 218–222, doi:10.1109/HealthCom.2012.6379411.

[34] ST 2110-21:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: Traffic Shaping and Delivery Timing for Video. *ST 2110-21:2017*, Nov 2017: s. 1–17, doi:10.5594/SMPTE.ST2110-21.2017.

[35] Xilinx, Inc.: *Tri-Mode Ethernet MAC v9.0 [online]*. 2018, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/tri_mode_ethernet_mac/v9_0/pg051-tri-mode-eth-mac.pdf

[36] Xilinx, Inc.: *AXI 1G/2.5G Ethernet Subsystem v7.0 [online]*. 2017, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_ethernet/v7_0/pg138-axi-ethernet.pdf

[37] Wessels, D.: *Observations on Checksum Errors in DNS UDP Messages [online]*. 2011, [cit. 2019-02-22]. Dostupné z: https://www.dns-oarc.net/files/workshop-201110/observations-on-checksum-errors.pdf

[38] Xilinx, Inc.: *AXI4-Stream Clock Converter [online]*. [cit. 2019-02-18]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axis_infrastructure_ip_suite/v1_1/pg085-axi4stream-infrastructure.pdf

[39] Xilinx, Inc.: *AXI4-Stream Interconnect v1.1 [online]*. [cit. 2019-02-18]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axis_interconnect/v1_1/pg035_axis_interconnect.pdf

[40] Alexandre Belloni, T. P.: *Buildroot vs. OpenEmbedded/Yocto Project: A Four Hands Discussion [online]*. 2016, [cit. 2019-02-22]. Dostupné z: https://events.static.linuxfound.org/sites/events/files/slides/belloni-petazzoni-buildroot-oe_0.pdf

[41] Woodhouse, D.: *JFFS : The Journalling Flash File System [online]*. Red Hat, Inc., [cit. 2019-02-21]. Dostupné z: https://sourceware.org/jffs2/jffs2.pdf

[42] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. 2014, doi:DOI10.17487/RFC7159. Dostupné z: https://www.rfc-editor.org/refs/ref7159.txt

[43] Xilinx, Inc.: *AXI Interconnect v2.1 [online]*. 2017, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_interconnect/v2_1/pg059-axi-interconnect.pdf

[44] Xilinx, Inc.: *SmartConnect v1.0 [online]*. 2019, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/smartconnect/v1_0/pg247-smartconnect.pdf

[45] Xilinx, Inc.: *AXI Timer v2.0 [online]*. 2016, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_timer/v2_0/pg079-axi-timer.pdf

[46] Xilinx, Inc.: *AXI Interrupt Controller (INTC) v4.1 [online]*. 2018, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_intc/v4_1/pg099-axi-intc.pdf

[47] Xilinx, Inc.: *7 Series FPGAs Memory Interface Solutions [online]*. 2011, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/ug586_7Series_MIS.pdf

[48] Xilinx, Inc.: *UltraScaleArchitecture-Based FPGAs Memory IP v1.4 [online]*. 2018, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/ultrascale_memory_ip/v1_4/pg150-ultrascale-memory-ip.pdf

[49] Xilinx, Inc.: *AXI GPIO v2.0 [online]*. 2016, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_gpio/v2_0/pg144-axi-gpio.pdf

[50] Xilinx, Inc.: *AXI Quad SPI v3.2 [online]*. 2018, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_quad_spi/v3_2/pg153-axi-quad-spi.pdf

[51] Xilinx, Inc.: *7 Series FPGAs Configuration [online]*. 2018, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

[52] Xilinx, Inc.: *AXI UART Lite v2.0 [online]*. 2017, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_uartlite/v2_0/pg142-axi-uartlite.pdf

[53] Xilinx, Inc.: *Divider Generator [online]*. 2016, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/div_gen/v5_1/pg151-div-gen.pdf

[54] Xilinx, Inc.: *AXI Block RAM (BRAM) Controller v4.0 [online]*. 2016, [cit. 2019-02-21]. Dostupné z: https://www.xilinx.com/support/documentation/ip_documentation/axi_bram_ctrl/v4_0/pg078-axi-bram-ctrl.pdf

[55] BusyBox: The Swiss Army Knife of Embedded Linux. Dostupné z: https://busybox.net/about.html

[56] Society, A. E.: AES67, AES Standard for Audio Applications for Networks - High-Performance Streaming Audio-Over-IP Interoperability. 2015.

[57] Di Stefano, A.; Terrazzino, G.; Giaconia, C.: FPGA implementation of a reconfigurable 802.11 medium access control. *International Conference on Wireless Reconfigurable Terminals and Platforms (WIRTEP)*, 2006.

Příloha **A**

Seznam použitých zkratek

ACE AXI Coherency Extensions

AMBA Advanced Microcontroler Bus Architecture

ARP Adress Resolution Protocol

AXI Advanced eXtensible Interface

BRAM Block RAM

CMOS Complementary Metal–Oxide–Semiconductor

CRC Cyclic redundancy check

CSMA/CD Carrier Sense Multiple Access with Collision Detection

DDR Double Data Rate

DF Do not Fragment

DHCP Dynamic Host Configuration Protocol

DMA Direct Memory Access

FCS Frame Check Sequence

FPGA Field Programable Gate Array

GNU GNU is Not Unix!

Gbps Gigabits per second

HTTPS Hypertext Transfer Protocol Secure

HTTP Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

ISO International Organization for Standardization

JSON JavaScript Object Notation

JTAG Joint Test Action Group

Kbit Kilobit

LMB Local Memory Bus

MAC Media Access Control

MF More Fragments

MHz Megahertz

MII Media Independent interface

MMU Memory Management Unit

MS milisekunda

MVTP Modular Video Transfer Platform

MVTP Modular Video Transfer Platform

Mbit Megabit

Mbps Megabitů za sekundu

NS Nanosekunda

NTP Network Time Protocol

OSI Open Systems Interconnection

PCS Physical Coding Sublayer

PMA Physical Medium Attachment Sublayer

PMD Physical Medium Dependent Sublayer

PTP Precision Time Protocol

QOS Quality of Service

RAM Random Access Memory

RFC Request For Comments

RISC Reduced Instruction Set Computer

SDI Serial Digital Interface

SFD Start of Frame Delimiter

SFP Small Form-factor Pluggable

SMPTE Society of Motion Picture and Television Engineers

SSH Secure Shell

TCP Transmission Control Protocol

TTL Time To Live

UDP User Datagram Protocol

USB Universal Serial Bus

U16 16bitové celé číslo bez znaménka

VHDL Very High Speed Integrated Cirucuit Hardware Description Language

Příloha **B**

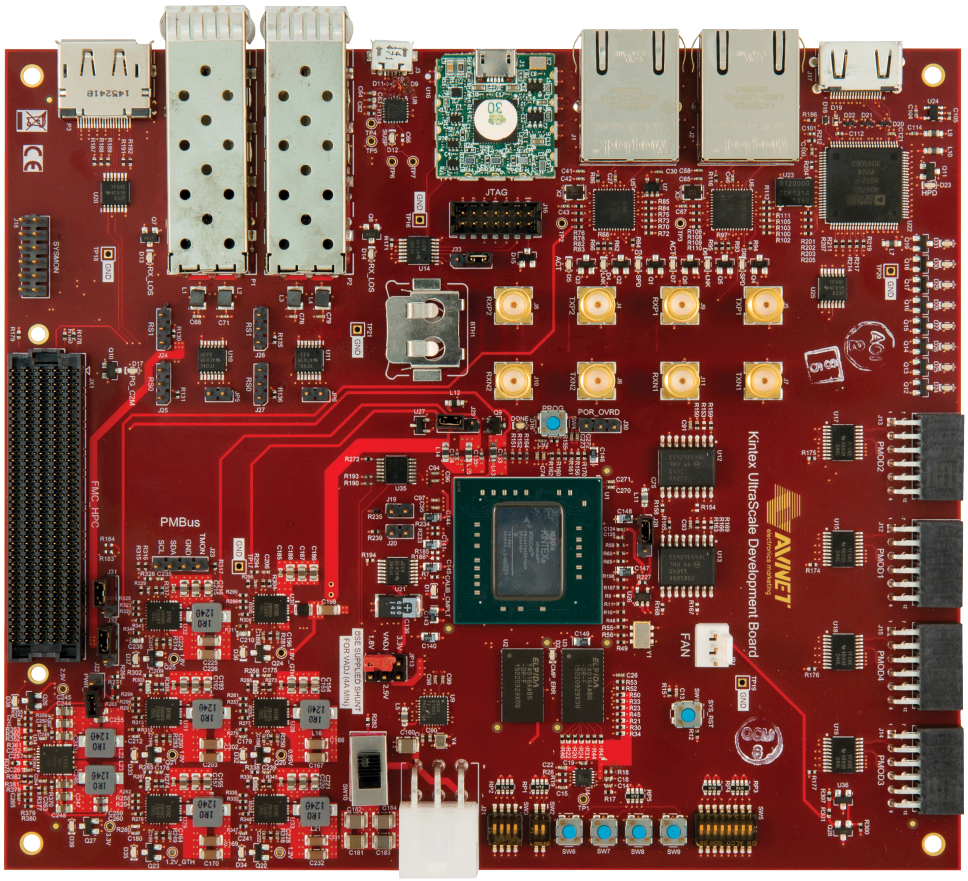
Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `doc` ..... dokumentace ovladačů jednotek v programu Doxygen (html)
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF

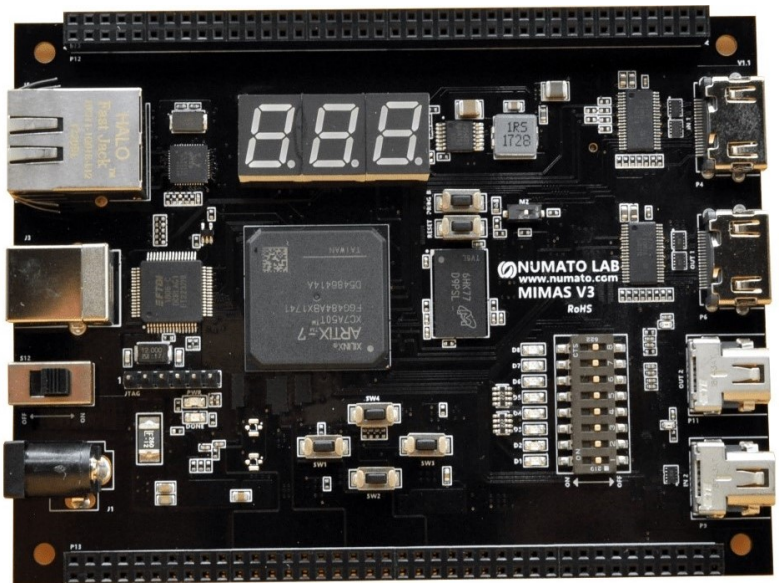
Příloha C

Vývojové desky

Obrázek C.1: Vývojová deska Avnet KU-040-DB-G



Obrázek C.2: Vývojová deska Numato Mimas A7



Příloha D

Spotřeba zdrojů

D.1 Artix A7 na desce Numato Mimas A7

Tabulka D.1: Celkové využití zdrojů hw platformou na FPGA Artix A7

Typ	Použito	Celkem k dispozici	Využití
LUT	26 920	32 600	82,58 %
LUTRAM	3 742	9 600	38,98 %
FF	31 174	65 200	47,81 %
BRAM	48,5	75	64,67 %
DSP	4	120	3,33 %
IO	88	250	35,20 %
MMCM	4	5	80 %
PLL	1	5	20 %

Tabulka D.2: Využití zdrojů jednotkou zrcadla na FPGA Artix A7

Typ	Použito	Celkem k dispozici	Využití
LUT	2 943	32 600	9,02 %
LUTRAM	517	9 600	5,38 %
FF	3 577	65 200	5,48 %
BRAM	1	75	1,33 %
DSP	0	120	0 %
IO	0	250	0 %
MMCM	0	5	0 %
PLL	0	5	0 %

D.2 Kintex Ultrascale na desce KU-040-DB-G

Tabulka D.3: Celkové využití zdrojů hw platformou na FPGA Kintex Ultrascale xcku–040

Typ	Použito	Celkem k dispozici	Využití
LUT	44 731	242 400	18,45 %
LUTRAM	5 374	112 800	4,76 %
FF	66 163	484 800	13,65 %
BRAM	123	600	20,5 %
DSP	7	1 920	0,36 %
IO	117	312	37,5 %
MMCM	3	10	30 %
PLL	2	20	10 %

Tabulka D.4: Využití zdrojů jednotkou měření jitteru na FPGA kintex ultrascale

Typ	Použito	Celkem k dispozici	Využití
LUT	6 227	242 400	2,56 %
LUTRAM	16	112 800	0,01 %
FF	14 892	484 800	3,07 %
BRAM	8	600	1,33 %
DSP	0	1 920	0 %
IO	0	312	0 %
MMCM	0	10	0 %
PLL	0	20	0 %

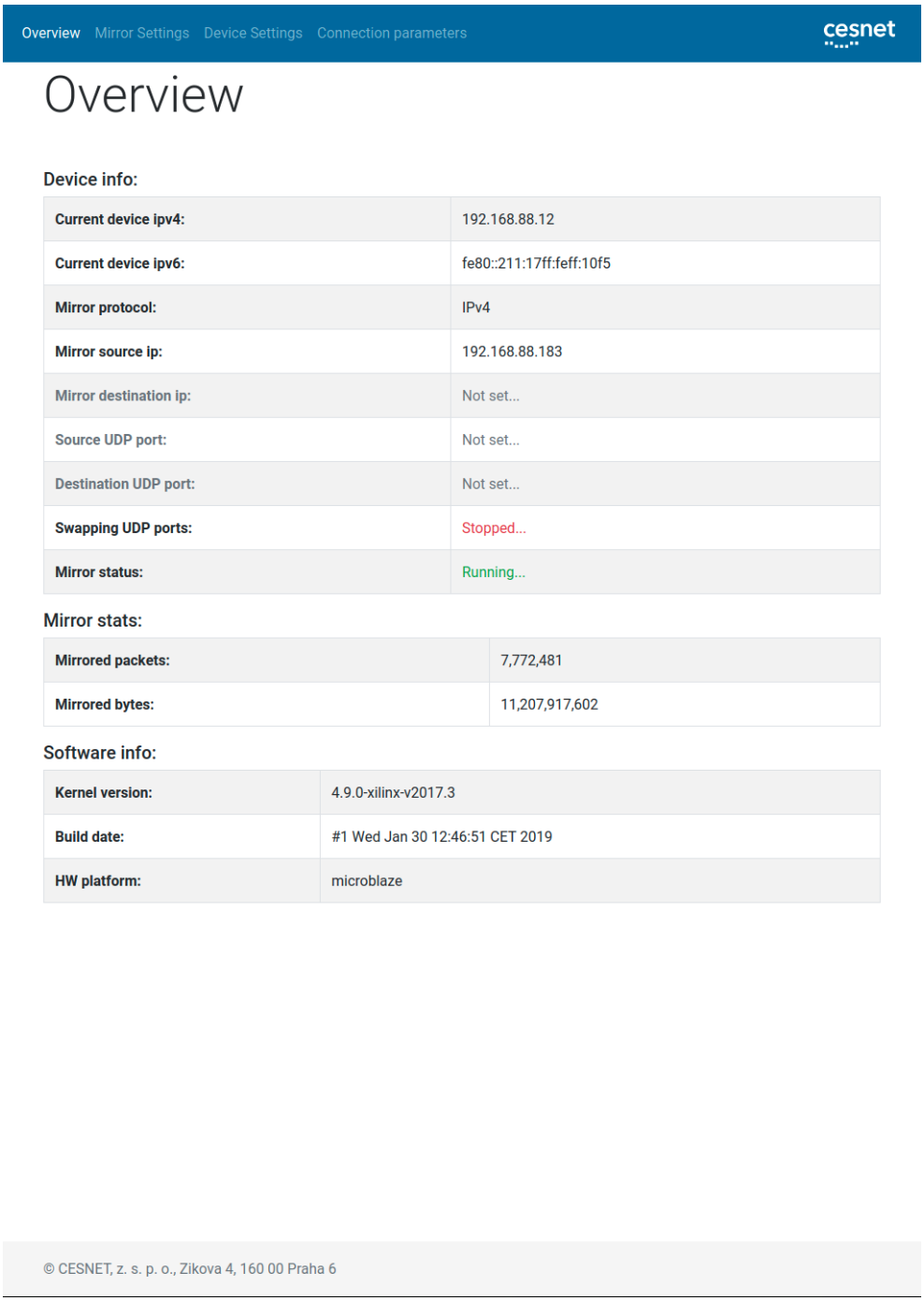
Tabulka D.5: Využití zdrojů jednotkou zrcadla na FPGA kintex ultrascale

Typ	Použito	Celkem k dispozici	Využití
LUT	2618	242400	1,08 %
LUTRAM	55	112800	0,04 %
FF	2897	484800	0,59 %
BRAM	2	600	0,33 %
DSP	0	1920	0 %
IO	0	312	0 %
MMCM	0	10	0 %
PLL	0	20	0 %

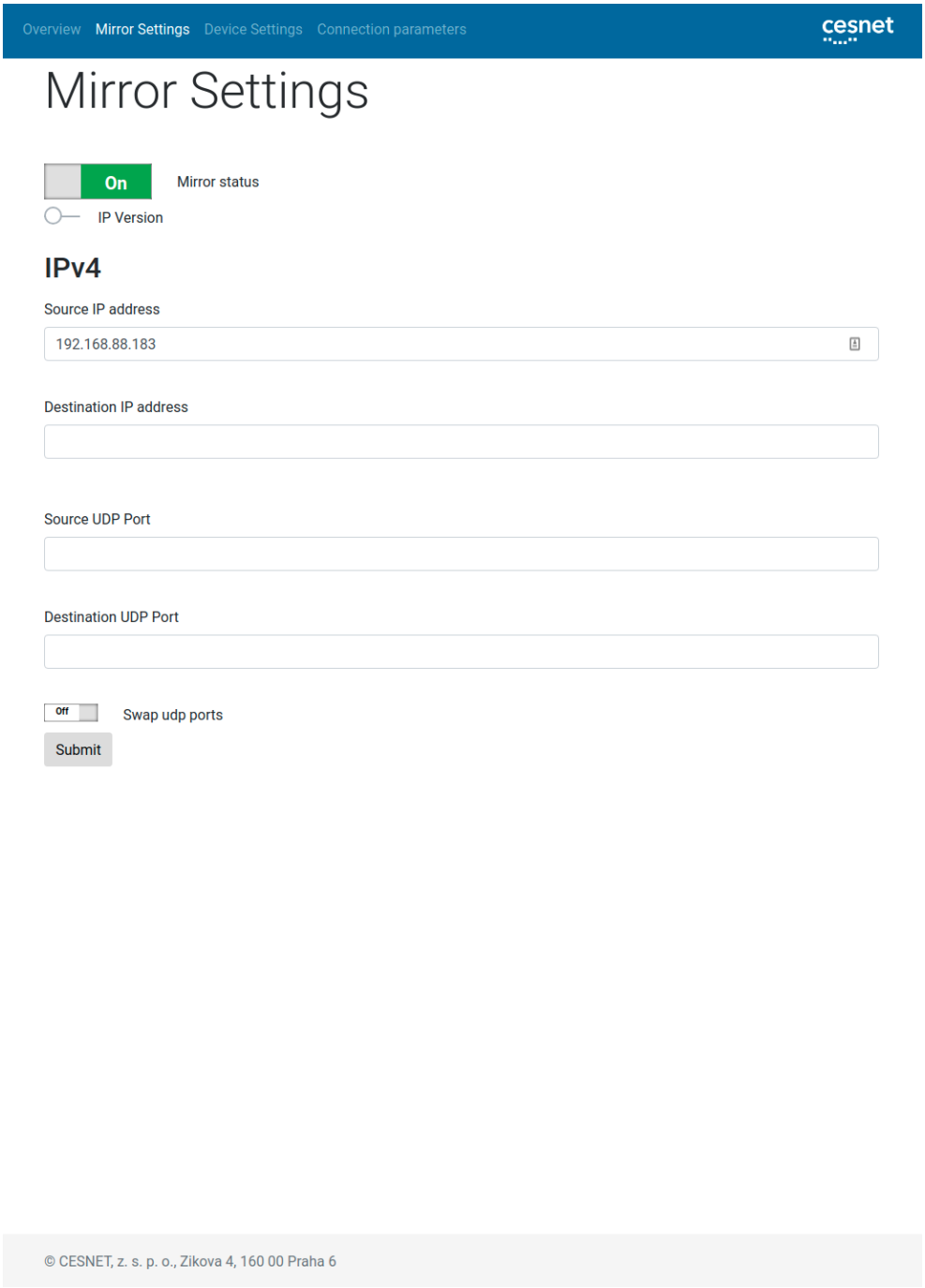
Příloha E

Vytvořené webové rozhraní

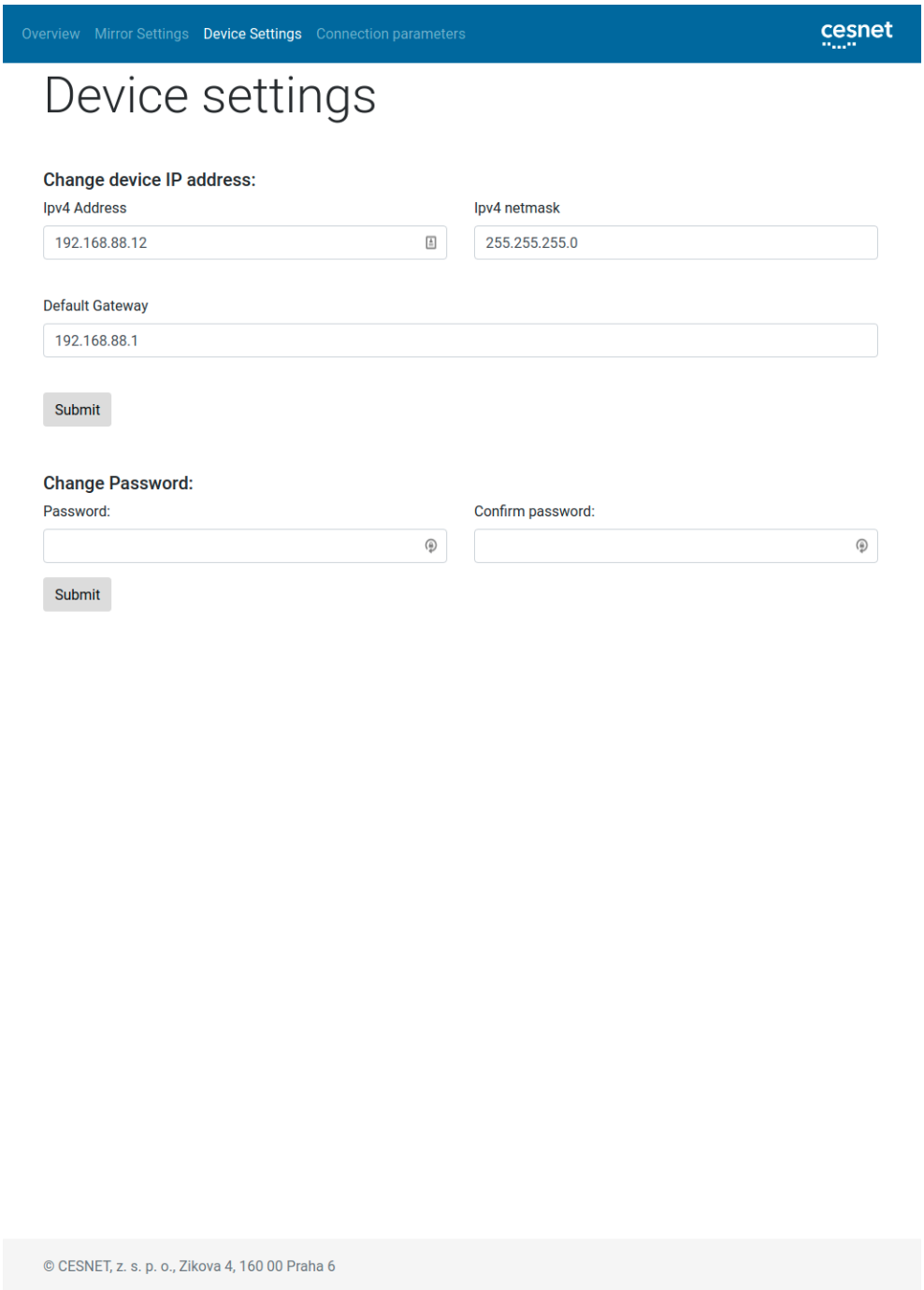
Obrázek E.1: Design webové stránky s celkovým přehledm informací o zařízení



Obrázek E.2: Design webové stránky s nastavením jednotky zrcadla



Obrázek E.3: Design webové stránky s nastavením zařízení



Obrázek E.4: Design webové stránky s nastavením a výsledky měření síťového jitteru