VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

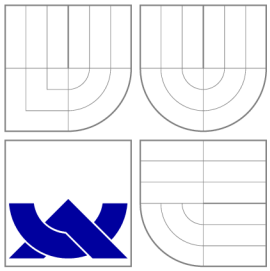
PŘENOSY RASTROVÝCH DAT V FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

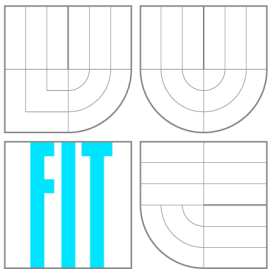
AUTOR PRÁCE
AUTHOR

Bc. MARTIN MUSIL

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PŘENOSY RASTROVÝCH DAT V FPGA

RASTER IMAGE DATA TRANSFERS IN FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN MUSIL

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. PAVEL ZEMČÍK

BRNO 2012

Abstrakt

Tato práce se zabývá návrhem a realizací vysokorychlostních komunikačních rozhraní na čipu FPGA a jejich využitím pro zpracování a přenos rastrových dat. V implementační části bylo vytvořeno koncové zařízení PCI Express, které zajišťuje přenos dat mezi čipem PFGA a RAM pamětí počítače. Jako zdroj obrazových dat pro zpracování byla k FPGA připojena videokamera Unicam M621 s rozhraním Ethernet. Projekt byl realizován na vývojovém kitu Xilinx SP605. Využití obou rozhraní bylo demonstrováno na aplikaci detekce hran pomocí Sobelova operátoru. V rámci práce byl vytvořen ovladač PCI Express zařízení pro operační systém Linux a jednoduché aplikační rozhraní v jazyce C.

Abstract

This work deals with the design and implementation of high-speed communication interfaces into FPGA chip and their utilizing for image transmission and processing. In the implementation part has been created PCI Express endpoint device, which provides data transfers between the FPGA chip and computer RAM memory. As a source of image data for further processing was connected the Unicam M621 camera throught the Ethernet interface to FPGA chip. The project was implemented on the Xilinx SP605 development board. Using both of the the interfaces were demonstrated on the application of edge detection using Sobel operator. The PCI Express endpoint device driver for the Linux operating system and a simple application interface in C language was also created within this project.

Klíčová slova

FPGA, VHDL, PCI Express, přenos dat, ovladač zařízení, Ethernet, Detekce hran, Sobelův operátor, Xilinx SP605, Unicam M621

Keywords

FPGA, VHDL, PCI Express, data transfer, device driver, Ethernet, edge detection, Sobel operator, Xilinx SP605, Unicam M621

Citace

Martin Musil: Přenosy rastrových dat v FPGA, diplomová práce, Brno, FIT VUT v Brně, 2012

Přenosy rastrových dat v FPGA

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Musil
22. května 2012

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Doc. Dr. Ing. Pavlu Zemčíkovi za odborné vedení, konzultace a připomínky, které mi pomohly při řešení diplomové práce.

Obsah

Kapitola 1

Úvod

V posledních desetiletích dochází k prudkému rozvoji elektroniky a výpočetní techniky. Elektronická zařízení jsou již téměř nedílnou součástí našeho života a jejich význam pro člověka v budoucnu jistě poroste. Nejedná se ale jen o počítače, ať už stolní nebo přenosné, se kterými si nejčastěji spojujeme pojem výpočetní technika. Daleko častěji se setkáváme se zařízeními, která obsahují specializované integrované obvody a jsou určena pro konkrétní aplikaci, jako DVD nebo MP3 přehrávače, mobilní telefony, ale i síťové routery a switche, řídící jednotky spalovacích motorů, řízení výrobních linek a mnoho dalších. Tato práce bude zaměřena zejména na problematiku vysokorychlostních přenosů dat, které jsou mnohdy pro fungování takových zařízení nezbytné.

Právě rozvoj v oblasti specializovaných zařízení klade požadavky na různé parametry integrovaných obvodů, ať už z hlediska výpočetního výkonu, příkonu, rozměrů a v neposlední řadě i ceny. Obvody navržené na míru konkrétní aplikaci(ASIC) nejlépe splní většinu požadovaných parametrů, avšak jejich výrobní cena je velmi vysoká a je jich nutné vyrábět velké série, aby se výroba vyplatila. Použitím univerzálních procesorů zase nelze snadno dosáhnout požadované výkonnosti a efektivity. Fenoménem dneška se tak pomalu stávají obvody FPGA(*Field Programmable Gate Array*).

FPGA je integrovaný obvod, který je navržen tak, aby ho mohl zákazník po výrobě nakonfigurovat podle svých požadavků. Skládá se z bloků CLB, které je možné naprogramovat na provádění složitých logických funkcí, a rozsáhlé propojovací sítě, která slouží právě pro propojování CLB bloků mezi sebou. Pro popis konfigurace FPGA se používají nejčastěji jazyky VHDL a Verilog.

V současné době existuje celá řada aplikací pro zpracování obrazu vhodných k implementaci na čipu FPGA, od jednoduchých filtrů, detektorů hran, až po složité a výpočetně náročné jako je detekce objektů v obraze. Programovatelný hardware přináší pro zpracování obrazu spoustu výhod, je možné v něm relativně lehce dosáhnout vysoce paralelního zpracování, navrhnout specializované obvody pro urychlení výpočtu a mnoho dalších. Tím, že navrhneme speciální obvod také v neposlední řadě získáme výhodu nízké spotřeby a malých rozměrů výsledného zařízení, což usnadňuje nasazení zařízení například v kamerových bazpečnostních systémech či automatizaci výroby.

Právě okolo algoritmů pro zpracování obrazu bude částečně pojednávat tato práce. Dále popsaná a v rámci práce implementovaná datová rozhraní budou sloužit právě jako základ pro fungování různých obvodů pro zpracování obrazového vstupu.

Po tomto úvodu následuje kapitola pojednávající o problematice přenosů dat. Jsou zde popsány principy fungování sběrnice PCI Express, a to nejen z pohledu realizace komunikace po sběrnici, ale také spolupráce s operačním systémem a paměťovým subsystémem počítače. Dále je zde představeno síťové rozhraní Ethernet, jeho fyzická realizace a vlastnosti na něm postavených komunikačních protokolů. Třetí kapitolu tvoří specifikace zadání, kde jsou zpřesněny požadavky na výsledné řešení. Kapitola čtyři se pak zabývá podrobným návrhem obvodů komunikačních zařízení tak, aby vyhověly specifikacím. V páté kapitole se nachází popis již implementovaných obvodů, jsou zde shrnuty důležité informace důležité pro jejich použití v různých projektech. Nechybí ani souhrn výsledných vlastností obvodů. Kapitola je zakončena ukázkou použití zde realizovaných komunikačních rozhraní v úloze zpracování obrazu.

Kapitola 2

Vysokorychlostní přenosy dat s FPGA

V dnešní době se stále více prosazují různá zařízení a akcelerační karty, které staví na flexibilitě, nízké spotřebě a vysokém potencionálním výpočetním výkonu čipů FPGA. Zejména libovolně škálovatelná a paralelizovatelná architektura předurčuje čipy FPGA pro masivně paralelní výpočty, ale samozřejmě nejen pro ně. Aby bylo možné využít vysokého potenciálu zpracování velkého objemu dat, musí být čip propojen vysokorychlostními sběrnicemi s jejich zdrojem, například počítačem, kamerou či počítačovou sítí.

V této kapitole se nachází úvod do architektury programovatelných hradlových polí. Je zde popsána jejich vnitřní struktura, zejména vlastnosti programovatelných i pevných logických bloků a způsob jejich propojení. Nechybí ani krátké seznámení s jazykem VHDL určeným pro popis hardwarových obvodů. Kapitola pokračuje specifikacemi dvou komunikačních rozhraní, PCI Express[9] a Ethernet[8], která jsou dostupná na vývojovém kitu Xilinx SP605[4], který bude prezentován v rámci kapitoly 3.

Závěr kapitoly obsahuje lehký úvod do problematiky zpracování obrazu. Je zde prezentován známý algoritmus detekce hran pomocí Sobelova operátoru[6]. Zajímavým algoritmem, jehož princip je zde rozepsán, je algoritmus Adaboost od autorů Viola a Jones[11]. Tento algoritmus je základem detekčního obvodu publikovaného v článku *Hardware Detection of Scalable Objects Based on Adaboost Classifier*[7].

2.1 Programovatelná hradlová pole

Programovatelná hradlová pole jsou velmi populární alternativou k zákaznickým integrovaným obvodům, tzv. ASIC (*Application Specific Integration Circuit*). Rozdíly těchto dvou architektur, struktura obvodů FPGA a její výhody budou dále popsány v této kapitole.

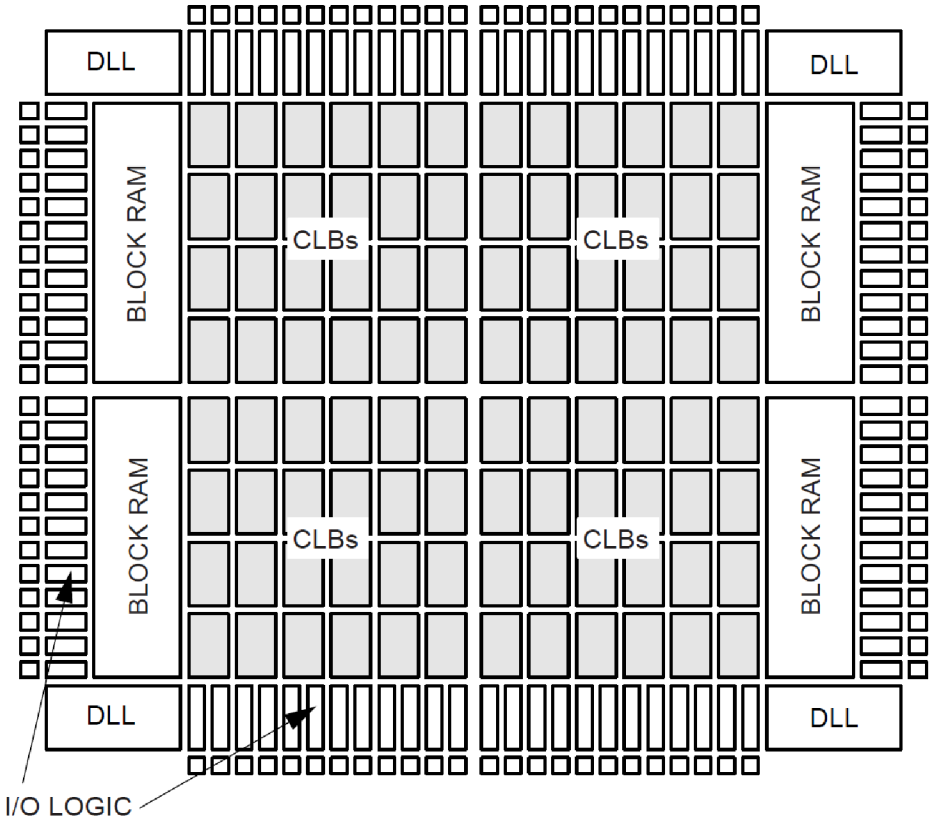
Obvody FPGA

Programovatelné hradlové pole (FPGA, *Field Programmable Gate Array*) je speciální integrovaný obvod, který je možné po výrobě libovolně naprogramovat a dosáhnout tak požadované funkcionality. Jeho záhladem je velký počet programovatelných bloků CLB(*Configurable Logic Block*) a složitá propojovací síť, kterou je možné tyto bloky libovolně propojovat (viz Obrázek 2.1). Nejnovější čipy obsahují také spoustu podpůrných obvodů, od blokových pamětí, násobiček, až po celá procesorová jádra. Pro popis konfigurace obvodu se používají

tzv. HDL jazyky, což je zkratka z anglického *Hardware Description Language*, neboli česky *Jazyk pro popis hardware*.

FPGA obvody dnes nacházejí uplatnění v široké škále aplikací a to hlavně díky své programovatelnosti, výkonnosti a flexibilitě. Výhodou je také jejich neustále klesající spotřeba energie a v neposlední řadě cena samotného čipu. Jejich typické použití je v oblasti menších sérií zařízení, kde se nevyplácí návrh zákaznického integravaného obvodu (ASIC) a současně nedostačuje konvenční řešení s univerzálním procesorem. Další využití nacházejí v prototypování složitějších zákaznických integrovaných obvodů, kde nabízí možnost implementovat a testovat navrhovaný obvod ještě před samotnou výrobou. Velké FPGA čipy dnes umožnují i realizaci složitých a výkonných procesorů.

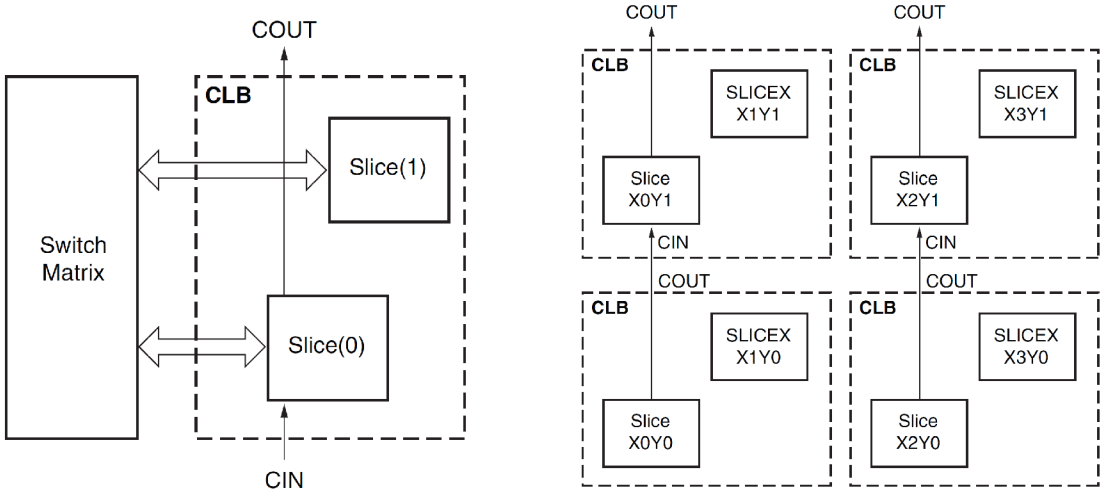
Pokud není uvedeno jinak, technické parametry uvedené v této kapitole jsou převzaty z architektury čipů rodiny Spartan-6 od firmy Xilinx.



Obrázek 2.1: Obvody FPGA: Zjednodušené schéma FPGA obvodu Spartan-II firmy Xilinx. Převzato z [13].

Blok CLB

Konfigurovatelný blok CLB je hlavní prostředek pro implementaci sekvenčních a kombinačních logických obvodů. Každý CLB blok je složen ze dvou menších částí, bloků *Slice* napojených na přepínací síť, která zprostředkovává připojení v rámci globální propojovací sítě (obr. 2.2 vlevo). Bloky CLB jsou uspořádané v pravidelné matici. Pro efektivní implementaci složitějších funkcí, které vyžadují více bloků *Slice*, jsou některé sousední bloky



Obrázek 2.2: Obvody FPGA: Uspořádání bloků *Slice* v rámci CLB (vlevo), organizace CLB a vzájemné propojení sousedních *Slices* (vpravo). Převzato z [3].

propojeny přímými spoji. Ušetří se tak nejen na velikosti propojovací sítě, ale dosáhneme i nižšího zpoždění datové cesty (obr. 2.2 vpravo). Podrobnější informace jsou jsou dostupné ve specifikacích rodiny Spartan-6[3].

Slice

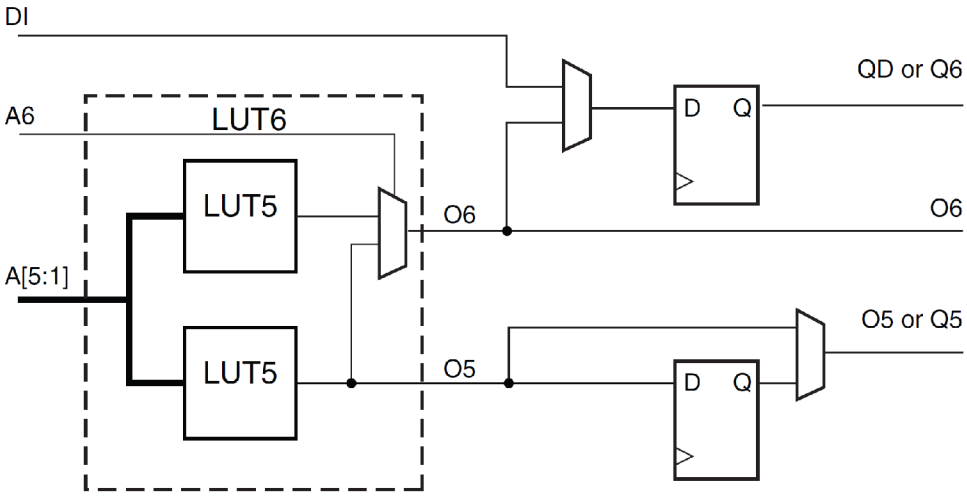
Každý blok Slice obsahuje čtyři generátory logických funkcí, tzv. LUT tabulky(*Look-up table*) a 8 paměťových elementů, které jsou tvořeny klopnými obvody typu D. Architektura Spartan-6 má ještě několik druhů Slice lišících se podle detailů implementace. Bloky *Slice-L* obsahují více propojovacích cest se sousedními bloky Slice a hodí se pro implementaci více-vstupých multiplexorů nebo vícebitových čítaček. *Slice-M* mají navíc obvodovou podporu pro implementaci posuvných registrů a distribuovaných RAM pamětí.

LUT tabulka má v architektuře Spartan-6 šest vstupů a je implementována jako asociativní paměť jednoho bitu (32x1b). Pomocí ní lze realizovat libovolnou logickou funkci tvořenou až šesti proměnnými (vstupy A1-A6, viz Obrázek 2.3). Výstupem LUT jsou dva nezávislé výstupy O5 a O6. Je tak možné realizovat jednu funkci šesti vstupů nebo dokonce dvě s pěti vstupy. Výstupy LUT jsou připojené na klopné obvody synchronizované hranou hodin, lze tak jednoduše vytvořit z bloku Slice synchronní obvod, například registr. Pokud je třeba realizovat složitější funkci, lze využít zřetězení několika sousedních LUT nebo i bloků Slice, jak je tomu na obrázku 2.2 vpravo.

IP jádra

Některé složitější logické funkce a obvody, které se opakují v různých implementacích lze s výhodou používat ve formě podobné knihovnám standardních programovacích jazyků. Tyto obvody se nazývají tzv. *IP jádra*(z anglické zkratky *Intelectual Property*) a jsou často tvořeny přímo výrobci FPGA obvodů nebo menšími nezávislými společnostmi, které IP jádra vyvíjí a prodávají příslušné licence k používání.

IP jádra se dělí na *soft* a *hard* jádra. *Soft IP* jádro je tak napsáno v některém z HDL jazyků, jazyků pro popis hardware. To zajišťuje kompatibilitu mezi jednotlivými čipy, takže



Obrázek 2.3: Obvody FPGA: Schéma bloku Slice architektury Spartan-6. Převzato z [3].

je relativně jednoduše přenositelné. Limitující může být hlavně velikost čipu případně nepřítomnost některých speciálních hardwarových bloků na FPGA, mezi nimi i *hard IP* jader.

Pojem *hard IP* jádro je obdobný *soft IP* jádru jen s tím rozdílem, že *hard* jádro je již výrobcem implementováno přímo na čipu. V FPGA jednotkách se dnes stále více začínají prosazovat, a to nejen pro zjevnou úsporu CLB bloků a propojovací sítě. V čipech FPGA se tak nacházejí například paměti Block-RAM, řadiče různých periferií(například DDR pamětí), obvody komunikačních rozhraní jako PCI Express a Ethernet, jednoduché programovatelné DSP jednotky či přímo celá jádra procesorů ARM.

Jazyk VHDL

VHDL je programovací jazyk pro popis hardwarových obvodů. Používá se pro návrh a simulaci digitálních integrovaných obvodů pro programovatelná hradlová pole (FPGA, CPLD a další). Dále se uplatňuje při návrhu a výrobě zákaznických obvodů ASIC. Zkratka VHDL znamená *VHSIC Hardware Description Language*(VHSIC jazyk pro popis hardware), kde VHSIC je zkratka pro *Very-High-Speed Integrated Circuit*(velmi rychlý integrovaný obvod)[10].

VHDL je standardem IEEE od roku 1987, byl revidován v roce 1997. Jazyk byl navržen pro syntézu a simulaci digitálních obvodů z textového popisu architektury. Ačkoli je možné všechny konstrukce jazyka simulovat, ne všechny mohou být syntetizovatelné.

Syntéza je proces, při kterém se popis jazyka konvertuje na zapojení a nastavení jednotlivých logických bloků(CLB) hradlového pole. Mimo jiné je součástí syntézy minimalizace logických funkcí, která ma za úkol optimalizovat množství použitých zdrojů a funkčních jednotek. Finálním krokem syntézy je mapování syntetizovaného obvodu na prvky cílové technologie. V této fázi je nutné brát v úvahu nejen odlišné implementace CLB bloků čipů jiných výrobců nebo sérií, ale také na možnosti propojovací sítě, která nemusí postačovat z hlediska kapacity nebo zpoždění.

Výhodou používání jazyka VHDL je fakt, že byl přijat za standard napříč výrobci programovatelných hradlových polí (Xilinx, Altera) a jednou napsaný kód je tak znovupoužitelný i pro čipy jiného výrobce. Další jeho výhodou je, že pomocí něj popsané obvody

mohou být použity přímo i jako podklady pro výrobu zákaznických čipů ASIC. V současné době je tak velmi rozšířený přístup, kdy je vyráběný obvod ASIC nejprve popsán v jazyce VHDL a jeho chování je otestováno v simulátoru. Poté následuje syntéza obvodu a testování přímo na čipu FPGA. Po úspěšném testování je zápis v jazyku VHDL použit na generování masky pro výrobu obvodu.

2.2 Sběrnice PCI Express

Sériové sběrnice

Sběrnice PCI Express je náhradou za dnes již zastaralé sběrnice PCI a AGP. Na rozdíl od nich se již nejedná o sběrnici paralelní, ale sériovou. Velkou nevýhodou paralelních sběrnic byl fakt, že mezi jednotlivými bity na sběrnici docházelo k tzv. *Clock skew* efektu. Podle fyzických vlastností každého vodiče se jednotlivé bity oproti ostatním buď zpožďovaly nebo předbíhaly. Tomu musela odpovídat doba vystavení dat na sběrnici a z ní plynoucí zpoždění před vysláním dalších dat. Paralelní sběrnice tak brzy dosáhly stropu možné taktovací frekvence. Sériové sběrnice jsou oproti paralelním na stejných frekvencích značně pomalejší, ale na druhou stranu na nich z pochopitelných důvodů nemůže docházet ke *Clock skew* efektu, a tak je možné je taktovat na výrazně vyšších frekvencích než sběrnice paralelní. Mezi nesporné výhody patří i zmenšení propojovacích kabelů i místa na desce plošných spojů.

Architektura PCI Express

PCI Express sběrnice je vysokorychlostní sériovou náhradou staré PCI sběrnice. Hlavním rozdílem oproti PCI je její architektura. Zatímco PCI zařízení sdílela stejnou paralelní sběrnici, PCI Express je založena na *point-to-point* komunikaci s řadičem(*PCI Express host*). Vodiče nejsou sdílené, není tak potřeba jako u starého PCI řešit přidělování sběrnice. PCI Express také podporuje plně duplexní komunikaci mezi libovolnými koncovými zařízeními.

Vyšší přenosové rychlosti PCIe se dosahuje přiřazením více samostatných linek koncovému zařízení. Posílaná data jsou pak rozdělena mezi více linek a na nich paralelně odeslána. Počet přidělených linek si zařízení vyjedná s PCI Express-hostem při inicializaci. PCI Express-host může zařízení přidělit méně linek, než zažádalo. Sníží se tím přenosová rychlost, ale funkčnost není omezena. V současné době je možné pracovat s počty linek: 1, 2, 4, 8, 16 a 32.

Pro přenos po sériové lince je třeba znát frekvenci přenosu, aby bylo možné data ze sběrnice správně vzorkovat. Hodinový signál sběrnice je součástí konektoru a je stejně jako datové vodiče diferenciální. Slouží ale pouze jako referenční kmitočet, nemůže být použit pro vzorkování signálu na datových vodičích, protože může docházet k fázovému posunu, tzv. *clock skew* mezi datovými a hodinovým vodičem. Vzorkovací obvod je proto synchronizován přímo datovými vodiči. Data jsou kódována kódem 8/10, který znamená, že 8 bitů dat je zakódováno na 10 přenášených bitů. Tím, že přidáme redundanci vzniká více možných přenášených kombinací a jsou z nich vybrány pouze ty, které nejlépe splňují kritéria pro přenos synchronizace hodin přímo v datech. Jde například o kódová slova s častou a rovnoměrnou změnou bitů 1 na 0 a naopak.

Přenosy dat

Datové přenosy po PCI Express sběrnici probíhají zasíláním paketů. Ty jsou zabezpečeny proti přenosovým chybám. Standard také stanovuje řízení datového toku založené na kreditovém systému a několik tříd priorit doručování paketů.

Zabezpečení dat proti chybám při přenosu je volitelné. Pokud chce zařízení využívat zabezpečený přenos, musí být schopno ve své režii spočítat podle standardizovaného algoritmu 32-bitový kód ECRC a připojit ho ke konci každého odeslaného paketu. Kontrola příchozích paketů je talé zcela v režii příjemce. Podle PCI Express specifikace nesmí v zařízeních ležících mezi odesílatelem a příjemcem, například přepínačích, docházet k zahazování paketů. Přepínače sice mohou provádět kontrolu chyb, ale nesmí se tak dít na úkor zpoždění přeposílání paketu. Případné chyby mohou být příjemci následovně nahlášeny speciálním paketem.

Specifikace PCI Express definuje čtyři základní typy komunikace rozdělené podle adresového prostoru, do kterého jsou mapovány. Každý typ používá jiné hlavičky paketů a definuje odlišné transakce:

- *memory mapped* - přenos z/do paměťově mapovaného prostoru
- *IO mapped* - přenos v rámci IO mapovaného prostoru
- *Configuration* - pro konfiguraci zařízení
- *Message* - systém zasílání zpráv

Paměťové transakce Přenosy dat v rámci adresového prostoru hlavní paměti. V tomto módu je možné číst/zapisovat data do hlavní paměti. Je také možné přistupovat k paměťově mapovaným lokacím, což se může hodit například pro zasílání dat mezi dvěma koncovými zařízeními. Tento mód také podporuje, narozdíl od IO operací, blokové přenosy dat.

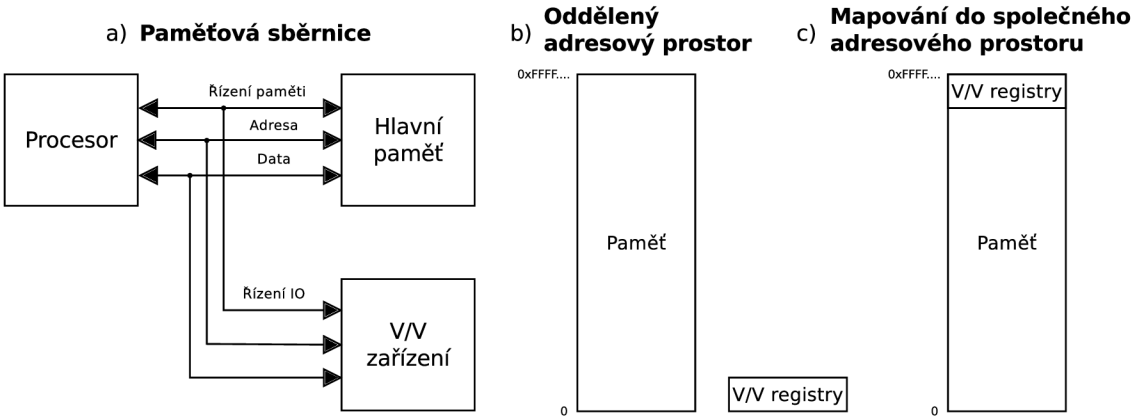
IO transakce Jedná se o přenosy dat v rámci vstupně-výstupního adresového prostoru, není tak možný přímý přístup k hlavní paměti počítače. Přenos dat v tomto módu je plně v řízení procesoru a je možné přenášet pouze dvojslova(32B) v rámci jedné operace čtení/zápisu. Tento mód podporuje PCI Express sběrnice pouze v rámci zpětné kompatibility a nedoporučuje se ho již používat.

Konfigurační transakce Používají se k přístupu do konfiguračních registrů PCI Express zařízení.

Systém zasílání zpráv Zde je možné implementovat libovolné aplikačně specifické komunikační mechanismy od oznamování událostí až po komunikaci mezi zařízeními. Tento mechanismus je používán například pro vyvolání přerušení.

Paměťově mapované operace

Každé zařízení připojené pomocí sběrnice PCI Express potřebuje komunikovat se zbytkem systému. Obvykle se tak děje přes řídící registry, ke kterým má procesor přístup a zápisem/čtením z nich provádí výměnu dat/řízení. Aby mohl procesor tyto registy používat,



Obrázek 2.4: a) Blokové schéma paměťové sběrnice; b), c) Rozdíly mezi odděleným a společným adresovým prostorem.

musí jim být přiřazena adresa v rámci jednoho ze dvou adresových prostorů, vstupně-výstupního nebo paměťového (viz obrázek 2.4).

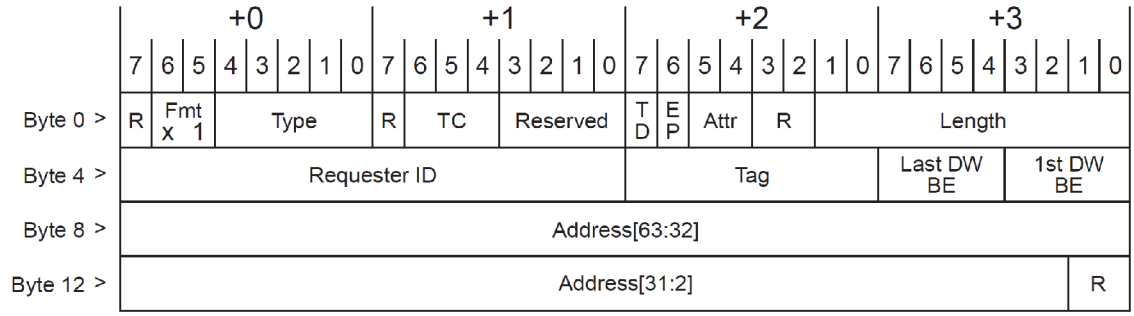
K V/V adresovému prostoru se přistupuje zvláštními instrukcemi procesoru a tento přístup je dnes využíván zřídka. Nehodí se pro vysokorychlostní přenosy, protože je možné v jednom paketu zaslat pouze 32 bitů dat a všechny operace jsou blokující, tzn. i pro zápis je nutné čekat na potvrzení o úspěšném ukončení transakce. Oproti tomu jsou operace v rámci paměťově mapovaného prostoru jednodušší (nevyžadují například potvrzení zápisu) a vytváří pro procesor abstrakci, že jsou řídící registry vstupně/výstupních zařízení součástí hlavní paměti. Díky tomu je také možné využít pro přenosy dat mezi zařízeními a hlavní pamětí řadiče DMA a tím dosáhnout řádově vyšší propustnosti datových přenosů.

Paměťové transakce s PCI Express

Jak již bylo popsáno dříve, komunikace po sběrnici PCI Express je založena na zasílání paketů. Pro obsáhlost zde bude prezentována pouze výměna dat v rámci paměťově mapovaného prostoru.

Paměťové operace používají hlavičku paketu z obrázku 2.5. Za hlavičkou mohou být podle typu operace připojena data. Typ operace (čtení/zápis) se nastavuje parametry *Fmt* a *Type*. Pole *Length* definuje, kolik dvouslov(32bitů) operace zahrnuje. Pole *Requester ID* je jednoznačná identifikace odesílatele, *tag* identifikuje číslo operace, pokud jich je momentálně rozpracovaných více. *Last BE(byte enable) a First BE* pole slouží příjemci zprávy a označují platnost dat v prvním a posledním dvojslově datové části paketu. Ostatní parametry slouží k nastavení priorit paketů a dalším doplňkovým službám.
Na obrázku 2.5 je hlavička pro 64-bitový adresový prostor, ve 32-bitovém se pouze vynechají bajty 8-11.

Pro odpověď na žádosti čtení se používá odlišná hlavička, která obsahuje identifikaci příjemce a zejména údaje o datové části paketu, jako je délka v bajtech a adresa prvního bajtu. Adresa je zde zejména proto, že žádostí o čtení je možné zažádat o větší část paměti (typicky 4kB), než je maximální možná délka paketu (typicky 128B až 1kB, podle konkrétní



Obrázek 2.5: PCI Express: Hlavička paměťových operací pro 64-bitový adresový prostor. Převzato z [9].

implementace). Data jsou pak odeslána ve více paketech a tato adresa napomáhá správné orientaci v příchozích datech. Více informací je k nalezení ve specifikaci PCI Express 1.1[9].

2.3 Rozhraní Ethernet

Ethernet je v současnosti jedna z nejrozšířenějších technologií pro budování lokálních sítí LAN. Byl standardizován institutem IEEE jako norma IEEE 802.3. Bylo definováno několik dalších standardů lišících se hlavně přenosovými rychlostmi. V současnosti jsou nejvíce rozšířené standardy pro rychlosti 10/100Mb a 1Gb. Na páteřních spojích se častěji setkáváme s rychlostmi 10,40 a 100Gb. Jako přenosové médium se pro rychlosti do 10Gb používá několik párů kroucené dvojlinky z mědi Pro vyšší rychlosti se již ve většině případů používají optické kabely.

Komunikace probíhá po blocích dat zvaných rámce o maximální délce 1518 bajtů. Formát rámce je jednoduchý, obsahuje na začátku pouze cílovou a zdrojovou adresu zařízení, informaci o obsahu rámce a na konci je připojena část s kontrolním součtem pro detekci chyb. MAC adresy(adresy zařízení) v Ethernetu mají 48bitů. Zbylé místo je vyhrazeno protokolům vyšších vrstev. Standard Ethernetu definuje mnoho různých druhů přenosového média, od použití s koaxiálními kabely, přes kroucené páry vodičů až po optické rozvody.

Samotný protokol Ethernet se nevyužívá pro komunikaci, slouží jen jako přenosový protokol pro vyšší protokoly, jako je IP, TCP a UDP protokol. Protokol Ethernet zajišťuje přenos dat pouze mezi dvěma přímo spojenými uzly nebo více uzly spojenými rozbočovačem (anglicky *Hub*). Pro zasílání paketů v rámci počítačové sítě spojené přepínači (anglicky *Switch*) byl navržen protokol IP. Protokoly TCP a UDP staví na protokolu IP. Zatímco TCP zajišťuje spolehlivou komunikaci a ošetřuje chyby přenosů, protokol UDP ztrátu a poškození datagramů neošetřuje. UDP se nezdržuje přeposíláním a potvrzováním přijatých dat, hodí se tak lépe pro vysokorychlostní přenosy, ve kterých nám nevadí občasná ztráta informace, jako je video streaming nebo přenos hlasu.

2.4 Příklady zpracování obrazu

V této podkapitole je prezentováno několik příkladů zpracování obrazu, které lze implementovat na čip FPGA a mohly by tak využít potenciál vysokorychlostních datových linek.

Digitální zpracování obrazu je vědecko-technická disciplína, která se zabývá zpracováním digitálních obrazových dat. Zdroje obrazu mohou být různé: videosekvence z kamery, snímky z fotoaparátu, data z ultrazvuku či jiných lékařských zobrazovacích technik a další. Zpracování obrazu je podoborem digitálního zpracování signálu, protože se zabývá zpracováním 2D digitálních signálů.

Mezi úlohy zpracování obrazu patří například:

- Vylepšení obrazu (odstranění šumu, zaostření/rozmazání, ...)
- Segmentace obrazu
- Detekce hran, objektů, geometrických primitiv
- Korespondence obrazů
- a další...

Detekce hran

Detekce hran je základním nástrojem pro zpracování obrazu, kde využití nachází zejména v oblasti počítačové vidění. Hrana je místo v obraze, kde se prudce mění jeho jas, tyto místa nesou často nejvíce obrazových informací. Na jejich základě je možné rozpoznat hranice různých objektů, což může být výhodné pro rozpoznání obsahu snímku. Detekce hran se tak dá využít například pro rekonstrukci 3D scény, vyhledávání korespondujících bodů obrazu nebo i sledování pohybu objektu.

Pokud hranu definujeme jako velkou změnu jasové funkce, bude v místě hrany velká hodnota derivace jasové funkce. Maximální hodnota derivace bude ve směru kolmo na hranu. Kvůli jednoduššímu výpočtu se ale hrany detekují jen ve dvou, resp. ve čtyř směrech. Velká skupina metod na detekci hran aproximuje tuto derivaci pomocí konvoluce s vhodným jádrem. Konvolucí se rozumí postupný průchod obrázku konvolučním jádrem a přepočítání každého vstupního pixelu na základě koeficientů jádra a pixelů obrazu, které překrývá aktuální pozice masky.

Mezi nejrozšířenější operátory pro měření 2D gradientu obrazu patří Robinsonův, Kirschův, Prewittové a Sobelův operátor. Tyto operátory mají obvykle rozměry konvolučních jáder 3x3 pixely, ale jsou definovány i větší rozměry, které vedou na přesnější detekci. Všechny operátory jsou založeny veskrze na stejném principu, liší se pouze koeficienty masky. Na obrázku 2.6 nalezneme konvoluční jádra Sobelova operátoru pro detekci různých směrů hran. Hranu detekuje vždy ta maska, která má nuly souběžně s hranou v obraze. Tento operátor dává větší váhu středu (vyšší koeficienty), mělo by proto docházet k lepší lokalizaci hran než například u operátoru Prewittové, kde jsou všechny koeficienty sice stejně rozložené, ale se stejnými hodnotami. Ukázka detekce hran Sobelovým operátorem se nachází na obrázku 2.8.

Při aplikaci Sobelova operátoru se nejčastěji setkáváme s použitím masek pouze pro horizontální a vertikální hrany (Obrázek 2.6a,b). Mezivýsledky po konvoluci s každou z masek jsou použity pro výpočet absolutní velikosti gradientu v každém bodě podle vzorce na obrázku 2.7a. Výpočet lze také aproximovat podle vzorce na obrázku 2.7b.

a) $\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$ b) $\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$ c) $\begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$ d) $\begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$

Obrázek 2.6: Sobelův hranový detektor: a) Konvoluční maska pro detekci vertikálních hran; b) Horizontálních hran; c),d) Úhlopříčných hran.

$$|G| = \sqrt{G_x^2 + G_y^2} \quad |G| = |G_x| + |G_y|$$

Obrázek 2.7: Sobelův hranový detektor. vlevo: Výpočet velikosti gradientu z mezivýsledků konvoluce; vpravo: Aproximace výpočtu velikosti gradientu.

Detekce objektů

Detekce objektů je technika počítačového vidění, jejímž úkolem je nacházet požadované objekty v obrazu nebo videosekvencích. Člověk dokáže rozpoznat velké množství objektů na snímku s minimálním úsilím a to i přesto, že se předměty mohou lišit v různých aspektech, například úhlu pohledu, velikosti, otočení nebo dokonce jsou-li překryté nějakým jiným objektem. Rozpoznávání objektů v takové míře je stále velkou výzvou pro počítačové systémy.

Rozpoznávání objektů ve videosekvencích je jednou z úloh, která je potencionálně dobře implementovatelná v čipu FPGA. Algoritmus detekce objektů je výpočetně velmi náročný, ale zároveň je dobře paralelizovatelný. Jedna z možných implementací detektoru je publikována ve článku *Hardware Detection of Scalable Objects Based on Adaboost Classifier*[7].

Appearance-based metody

Appearance-based metoda detekce objektů(metoda založená na podobnosti) provedla revoluci mezi tradičními postupy počítačového vidění. Místo toho, aby metoda vycházela z 3D

Obrázek 2.8: Sobelův hranový detektor. Vlevo se nachází původní obrázek, vpravo je výsledek po aplikaci Sobelova operátoru na detekci horizontálních a vertikálních hran. Převzato z [12].

modelu prostředí a z porovnávání s předem definovanými 3D modely předmětů, *Appearance-based* metody pracují pouze s 2D informacemi.

Metoda je založená na skenování obrazu klasifikátorem. Klasifikátor pracuje nad malým výřezem snímku(dále jen oknem) a má za úkol ohodnotit, zda se ve výřezu objekt nachází či nikoli. Tímto způsobem se musí v obraze ohodnotit postupně všechny možné pozice okna, až poté je klasifikace hotova. Tím, že prozkoumáváme všechny pozice docílíme invariance klasifikátoru vůči posunutí hledaného objektu. Invariance vůči natočení objektu se může řešit buď otočením klasifikovaného výřezu obrazu nebo použitím klasifikátoru, který je sám o sobě invariantním vůči otočení. Detekce objektů různých velikostí se dá řešit například zmenšením obrazu nebo zvětšením okna klasifikátoru.

Problémem metody je, že je nutné klasifikovat obrovské množství pozic v obraze, řádově milióny. Proto je metoda relativně hodně náročná na výpočetní výkon a tím pádem pomalá. Dalším neduhem je, že pracujeme pouze se 2D obrazem, ale samotné objekty existují ve 3D prostoru. Detekovaný objekt tak může být na obrázku různě pootočen, což ztěžuje detekci.

Klasifikátory Klasifikator je jednoduchý algoritmus, který dokáže pro nějaký výřez obrazu rozhodnout, zda se v něm nachází hledaný objekt. Klasifikátory mivají obvykle fixní velikost (např. 24x24 pixelů) a procházení obrazu takovým klasifikátorem dokáže detekovat objekty právě této velikosti. Pro detekci větších či menších objektů je nutné obraz, ve kterém hledáme, zvětšit či zmenšit.

Tímto postupem získáme pro každé místo obrazu informaci, zda se na něm hledaný objekt nachází či nikoli. Pokud obraz prohledáváme i v jiných rozlišeních, dostaneme i informaci o velikosti objektu. Protože ale prohledáváme klasifikátorem všechny možné pozice, je velmi pravděpodobné, že v těsném okolí jedné detekce bude i druhá, stejného předmětu. Je proto nutné provést jednoduchý clustering (shlukování), který potlačí tyto vícenásobné detekce.

Algoritmus Adaboost Adaboost je metoda, která umožňuje spojit několik slabých klasifikátorů a tím vytvořit jeden silný klasifikátor. Slabé klasifikátory mají většinou jeden příznak, např. treshold, histogram apod. Jedinou podmínkou pro jejich použití je, že úspěšnost jejich detekce musí být větší jak 0.5, tj. lepší jak náhodné. Silný klasifikátor je pak dán jejich lineární kombinací:

$$f(x) = \sum_{t \in T}^{n} (\alpha_t * h_t(x)) \tag{2.1}$$

α je vahou klasifikátoru, tj. procento, jakým se výsledný slabý klasifikátor podílí na výsledku detekce. Algoritmus Adaboost slouží právě k výpočtu těchto vah u jednotlivých klasifikátorů. Tento postup se nazývá trénováním. K trénování je zapotřebí velká sada trénovacích a testovacích dat, obrázků, na kterých je hledaný objekt i obrázků, na kterých se nenachází.

Za zmínku stojí varianta algoritmu od autorů Viola a Jones[11]. Algoritmus Adaboost používá vážení trénovací sady váhami D_t, které jsou ze začátku nastaveny rovnoměrně. V každé iteraci algoritmu se pak provádí následující:

- výběr slabého klasifikátor s nejmenší chybou klasifikace vzorů při váhách D_t
- ověření, že chyba klasifikátoru nepřekročila hodnotu 0.5

- výpočet koeficientu (váhy) slabého klasifikátoru podle jeho chyby
- aktualizace jednotlivých vah D_t trénovací sady

Celý postup je opakován do doby, dokud nedosáhneme například akceptovatelné chyby silného klasifikátoru.

Více informací o detekci objektů a konkrétní implementaci na čip FPGA je publikováno v článku *Hardware Detection of Scalable Objects Based on Adaboost Classifier*[7].

Kapitola 3

Specifikace zadání

V této kapitole se nachází specifikace jednotlivých bodů zadání a jsou zde stanoveny cíle, kterých má být v rámci této práce dosaženo. Součástí kapitoly je také popis hardwarových prostředků, které splňují požadavky zadání a budou použity pro řešení implementační části práce.

3.1 Cíle projektu

Práce se zaměřuje na vytvoření vysokorychlostních komunikačních rozhraní PCI Express a Ethernet na čipu FPGA. Hlavním cílem jejich využití je podpora projektů, které se zaměřují na zpracování obrazu, ale nejen jich. Základním blokem každého projektu je tak uživatelský obvod, který může využívat těchto rozhraní pro přenos dat do/z čipu FPGA. Protože mohou být vytvořená rozhraní znovupoužívána v dalších aplikacích, je třeba je navrhnout tak, aby vyhovovala různým druhům požadavků připojených obvodů. Samozřejmostí je jednoduché rozhraní, které obsahuje veškerou řídící logiku, kterou tak není třeba implementovat a testovat vývojářem.

Pro úlohy zpracování obrazu se velice hodí mít k dispozici zdroj snímků, jakým je například kamera. Jedním z úkolů je tak připojení odpovídající kamery k FPGA a vytvoření řídícího obvodu, který tato data vhodným způsobem předává uživatelským obvodům.

Mnohé algoritmy zpracování obrazu obsahují operace, které lze jen těžce či vůbec implementovat obvodově na čipu FPGA, například rekurzivní algoritmy, výstavba/průchod stromovou strukturou dat a další. Tyto operace ale není těžké provádět na CPU počítače. K realizovaným rozhraním je tak třeba vytvořit softwarové API, které zpřístupní výměnu dat mezi čipem FPGA a hlavní pamětí počítače. Samozřejmostí je vytvoření ovladače pro operační systém Linux.

Funkčnost obou rozhraní je dobré otestovat na konkrétním příkladu, například s obvodem pro detekci hran pomocí Sobelova operátoru. Jedna z možných implementací detektoru hran je publikována v článku *Image Edge Detection Based on FPGA*[6]. Zdařilou demonstrací funkčnosti celého řešení může být připojení engine pro detekci objektů, který je publikován v rámci článku *Hardware Detection of Scalable Objects Based on Adaboost Classifier*[7].

Postup vypracování

Nejsložitější částí projektu bude realizace PCI Express zařízení, proto bude vhodné začít právě tímto rozhraním. Navíc může dobře posloužit pro testování dalších částí projektu. Prvním milníkem bude vytvoření a nasimulování části PCI Express zařízení, která implementuje vnitřní registry. Tato verze by měla být odladěna přímo na hardwaru. Současně vznikne ovladač zařízení, se kterým se provede test funkčnosti. Posléze po řádném otestování lze zapracovat pokročilejší řízení, jako jsou obvody pro DMA přenos dat. Pro ověření jejich funkčnosti se bude hodit paměťový blok tvořený obvody BlockRAM, případně větší externí pamětí DDR.

Teprve po zhotovení a testování celého PCI Express zařízení bude vhodné začít s pracemi na dalších obvodech. Prostřednictvím tohoto zařízení bude možné jednoduše testovat ostatní obvody realizované v FPGA. Dalším krokem by mělo být připojení kamery a přenos streamovaných obrazových dat dat do počítače, kde budou zobrazována. Teprve poté je dobré začít s implementací a připojením některého z jednodušších obvodů pro zpracování obrazu z kamery.

3.2 Hardwarové platformy

Podkapitola obsahuje stručný popis hardwarových komponent vhodných pro řešení problémů popsaných ve specifikaci zadání.

Kit Xilinx SP605

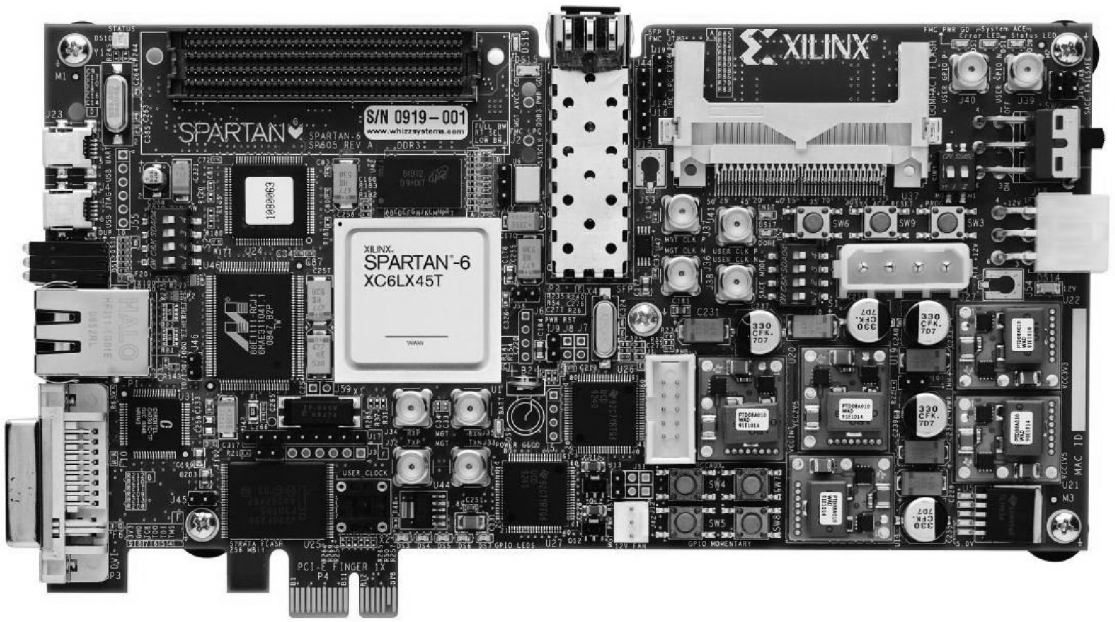
Pro implementaci projektu byl vybrán kit od společnosti Xilinx[13], model SP605[4](viz Obrázek 3.1). Kit obsahuje FPGA dostatečně velké i pro implementaci složitějších algoritmů zpracování obrazu, jako je například detektor objektů prezentovaný v článku *Hardware Detection of Scalable Objects Based on Adaboost Classifier*[7]. Zároveň nabízí pestrou paletu komunikačních rozhraní, jako je PCI Express, Ethernet a další.

Na kitu je k dispozici:

- FPGA XC6SLX45T firmy Xilinx
- 128MB DDR3 paměti
- PCI Express x1 konektor
- zásuvka Ethernet a čip pro realizaci 10/100/1000 fyzické vrstvy Ethernetu
- FMC LPC konektor, USB-to-UART převodník, SMA konektory a další.

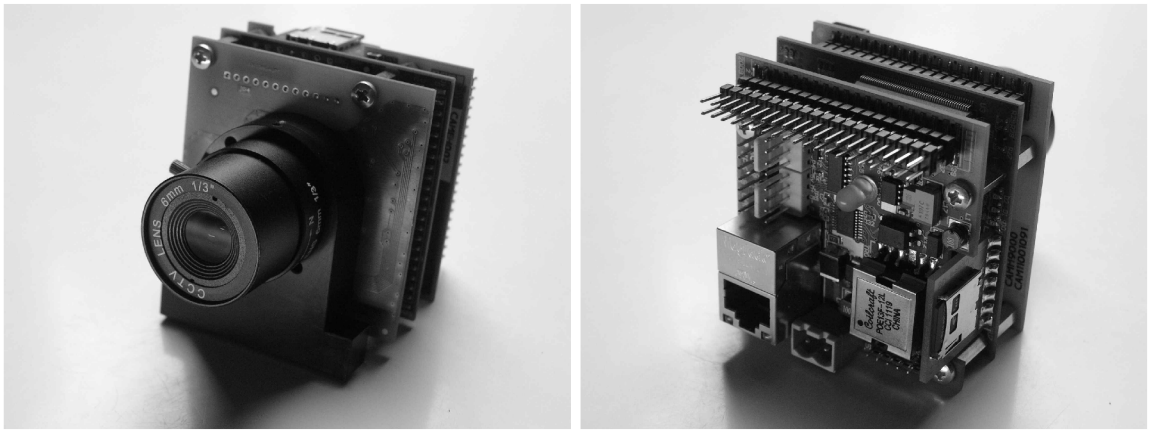
Kamera Unicam M621

Pro detekci objektů potřebujeme zajistit zdroj obrazu, ideálně videosekvencí. Pro tento účel bude výhodné použít kameru Unicam M621 od firmy Camea spol. s r.o.(Obrázek 3.2). Kamera obsahuje samostatnou řídící jednotku, která se stará o zpracování obrazu ze snímače. Obrazový výstup o rozlišení 752x478 pixelů je streamově přeposílán na Ethernetové rozhraní, kde je celý obraz rozdělen do několika paketů konstantní velikosti. Řízení kamery probíhá zasíláním krátkých Ethernetových rámců. Kamera vysílá pakety s obrazovými daty ihned po připojení napájení a navíc v pevném formátu, což bude s výhodou využito při



Obrázek 3.1: Xilinx SP605 evaluation board.

návrhu síťového rozhraní v FPGA, kdy dojde ke zjednodušení výsledné implementace a zároveň k úspoře zdrojů dostupných na čipu FPGA.



Obrázek 3.2: Kamera Unicam M621.

Výstupní rozhraní kamery

Kamera Unicam M621 komunikuje skrze rozhraní Ethernet. Výstupní pakety se dělí na dvě skupiny, obrazová data a obecné pakety. Mezi obecné pakety se řadí veškerá komunikace spojená s identifikací v síti a konfigurací kamery(ARP, ICMP, TCP protokol a další). Obrazová data jsou zasílána prostřednictvím protokolu UDP v paketech pevné délky 1024 bajtů(viz Tabulka 3.1). Odesílání paketů s obrazovým obsahem začíná ihned po startu kamery a děje se tak automaticky, bez předchozí konfigurace. Protokol kamery je navržen pro streamové odesílání obrazových dat, každý paket proto obsahuje informace o přenášených

datech. Příjemce nepotvrzuje přijetí paketu.

L2 vrstva - 14B	Dest. MAC addr. Src. MAC addr. Type	6B 6B 2B
IP vrstva - 28B	IP header UDP header	20B 8B
Obsah	Data	966B
Datová patička	Reserved Image counter Block counter Control word	4B 4B 4B 4B

Tabulka 3.1: Kamera M621: Formát datového paketu.

Obrazová data kamery jsou rozdělena po snímcích(viz Tabulka 3.2). S každým novým snímkem se inkrementuje čítač *Image counter*. Při restartu zařízení nebo při zastavení a spuštění grabbování dat je *Image counter* inicializován na nulu. Každý snímek je rozdělen na jednotlivé pakety. S každým paketem se inkrementuje čítač *Block counter*, toho lze využít například pro detekci ztráty paketu. S novým snímkem začíná čítač *Block counter* znovu od nuly.

Rozlišení kamery je 752*478 obrazových bodů v 8-bitových odstínech šedi a snímač kamery dodává 60 snímků/s. Celková velikost jednoho snímku v obraze tak není při současném rozlišení dělitelná číslem 996 beze zbytku. Poslední paket proto obsahuje ze začátku platná data, zbývající obsah je nedefinován. Po obrazových datech následuje vždy jeden epilogový paket, kde jsou uloženy doplňkové informace k pořízenému snímku(parametry obrazových dat, stav systému, verze firmware, časové razítko, ...). Formát a velikost epilogového paketu jsou jinak shodné s obrazovým. Epilogový paket je označen příznakem v *Control word* - v LSB je nastaven nejvyšší bit na log. 0. Obrazové pakety mají na této pozici log. 1.

Popis	Image counter	Block counter	Control word
1. datový paket - celý platný	N	0	0x00000080
2. datový paket - 534 B platných	N	1	0x00000080
Epilogový paket	N	2	0x00000000
1. datový paket dalšího snímku	N+1	0	0x00000080
...	...	...	...

Tabulka 3.2: Kamera M621: Příklad přijetí obrázku o velikosti 1500B.

Kapitola 4

Návrh komunikačních obvodů

V této kapitole se nachází rozbor požadavků na implementaci rozhraní PCI Express[9] a Ethernet[8] a způsoby jejich možného splnění. Je zde prezentován podrobnější návrh koncového zařízení PCI Express a výběr vhodného modelu výměny dat s hlavní pamětí počítače. Nechybí ani zvážení různých způsobů komunikace zařízení s operačním systémem počítače a nad ním běžící obslužnou aplikací. Návrh Ethernetového řadiče pro připojení kamery obsahuje úvahy nad možným zjednodušením architektury a implementace síťových protokolů pro konkrétní použití.

4.1 Návrh obvodu PCI Express

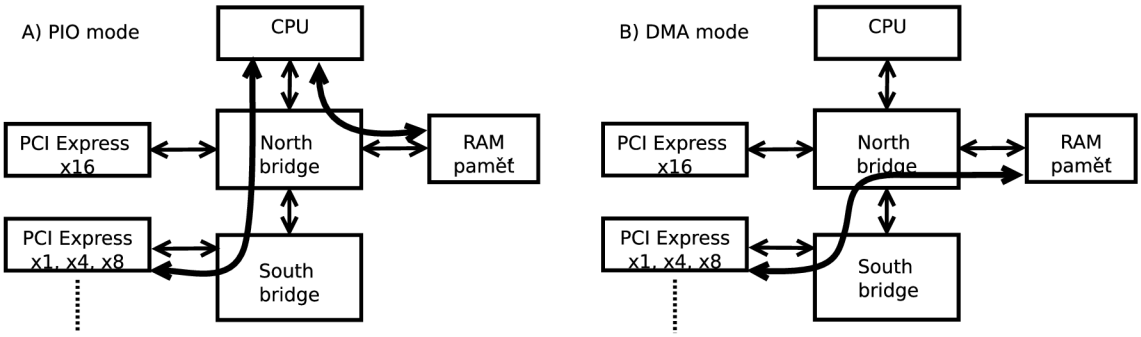
Kartu Xilinx SP605 je třeba připojit vhodným rozhraním s počítačem. Na něm může běžet podle potřeby software, který bude dále zpracovávat data přijátá z karty. Zde může docházet k relativně vysokým požadavkům na rychlost komunikace, kdy bude přenášena například celá videosekvence z připojené kamery Unicam M621. Pro tento účel je dobré využít připojení ke sběrnici PCI Express[9].

Požadavky na propustnost

Na kitu je možnost připojení jedné linky PCI Express v1.1 s teoretickou propustností 250MB v každém směru. Dostupná šířka pásma bohatě stačí na přenos nekomprimovaného videa v 8-bitových odstínech šedi o rozlišení 752x478 a 60-ti snímcích za vteřinu. Tyto vlastnosti má právě kamera Unicam M621 a pro přenos takového videa je třeba propustnost cca 22MB/s. Linka by tak s rezervou zvládla přenést video až o rozlišení Full-HD 1920x1080 se 60-ti snímky/s nebo například více samostatných videosekvencí právě na rozlišení 752x478. Je zde tak potencionální prostor pro budoucí rozšíření.

Realizace datových přenosů

Aby bylo možné s kartou nějakým způsobem komunikovat a vyměňovat data, musí být realizováno vhodné komunikační rozhraní na straně karty. Procesor pak musí mít možnost z karty číst a zapisovat data pomocí svých instrukcí. Každý řadič periferií či rozšiřující karta proto standardně obsahuje sadu řídících registrů. Procesor pak zápisem do těchto registrů řící jeho činnost a čtením kontroluje stav zařízení. Dříve se tímto způsobem přenášela i data, jedná se o tzv. PIO mód(obrázek 4.1a), což je zkratka pro *Processor Input Output*.



Obrázek 4.1: a) Programable Input Output - PIO mode, b) First-party DMA mode.

Nevýhodou tohoto přístupu je velmi malá rychlost přenosu dat, protože data lze číst pouze po blocích velikosti jednoho registru, nejčastěji 32 bitů. Při takové komunikaci vzniká velká režie pro přenos dat, protože s každými čtyřmi bajty dat je například na sběrnici PCI Express nutné odeslat hlavičku paketu čítající 12-16 bajtů. Není možné ani čtení ani zápis zřetězit, další operace může být zahájena až po dokončení předchozí. Tento přístup je pro požadovaný vysokorychlostní přenos nepoužitelný nejen pro rychlost přenosu, ale i pro příliš velké vytížení procesoru, který musí neustále provádět operace čtení/zápisu pro každých 32 bitů dat.

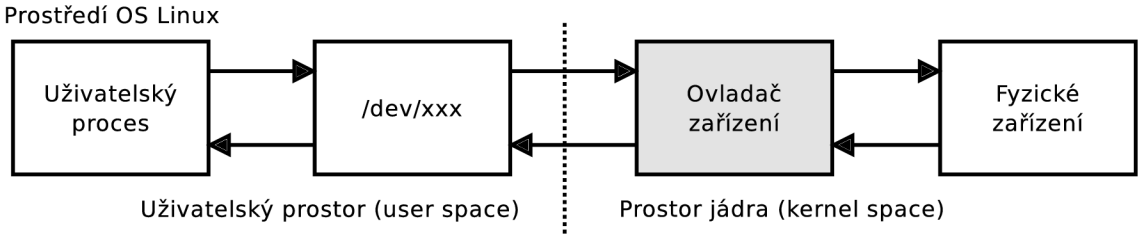
Správným řešením z pohledu rychlosti přenosu dat je využití DMA řadiče. Výhoda spočívá v tom, že procesor může nastavit jednou DMA řadič a přenos dat proběhne na pozadí bez asistence procesoru. Data jsou také přenášena po větších blocích, čímž se skryje režie přenosu.

DMA řadič

DMA řadič lze implementovat i v rámci koncového zařízení PCI Express, kdy se jedná o tzv. *first-party DMA* přenos(Obrázek 4.1b). Sběrnice má totiž podporu tzv. *bus mastering* režimu, kdy má koncové zařízení možnost řídit sběrnici. Kit tak může generovat například žádosti čtení a zápisu dat do hlavní paměti počítače zcela samostatně, což ještě více snižuje režii řízení procesorem a zvyšuje maximální možnou rychlost přenosu. Tento princip přenosu dat je s výhodou použit v implementaci.

Při použití DMA přenosů je nutné ošetřit ještě jednu podstatnou věc. Po nastavení hardwaru a spuštění DMA už nemá procesor nad řízením přenosu dat kontrolu a neví, zda již byl přenos dokončen nebo zda ještě pokračuje. Jednou z možností je cyklicky testovat dokončení přenosu například čtením z kontrolního registru DMA řadiče. To je ale značně neefektivní způsob, který plýtvá procesorovým časem a zahlcuje sběrnici. Proto je koncové zařízení rozšířeno o podporu hardwarového přerušení procesoru. Jedná se o metodu pro asynchronní obsluhu událostí. Jakmile je vyvoláno přerušení, je spuštěna obslužná rutina jádra operačního systému, která následně spustí obslužnou rutinu ovladače.

Přerušení je navíc možné využít k vyvolání obslužné rutiny ovladače ze strany zařízení. Například pokud výpočetní obvod dokončí operaci výpočtu, vyvolá přerušení a ovladač se postará o ošetření této situace.



Obrázek 4.2: Ovladač zařízení a jeho umístění v rámci operačního systému linux.

4.2 Ovladač pro operační systém

Ovladač zařízení(*device driver*) je software, který umožňuje operačnímu systému pracovat s hardwarem. Některé obecné ovladače jsou součástí operačního systému, jiné se dodávají spolu se zařízením, například grafickou kartou. Ovladač zajišťuje řízení hardware a zároveň komunikuje se zbytkem operačního systému pomocí obecnějšího rozhraní, které tvoří abstrakci zařízení. Tím pádem je možné komunikovat s odlišnými zařízeními skrze stejné rozhraní. Je již věcí ovladače, jak příkazy interpretuje. Návrh ovladače pak výrazně ovlivňuje možnosti hardwaru, protože některé funkce například není schopen ve stávající verzi podporovat. Stejně tak toto platí i pro výslednou rychlost komunikace s periferií, kdy může být ovladač navržen neefektivně a není schopen plně využít možnosti hardwaru.

Při návrhu ovladače pro systém linux bylo čerpáno z knihy *Linux Device Drivers*[5].

Virtualizace paměti

Protože pracujeme s operačním systémem, který vytváří nad hardwarem počítače abstrakci a virtualizuje operační paměť, není tak snadné implementovat přímé přenosy dat z/do kitu, je třeba využívat funkcí ovladače zařízení. Hardware samozřejmě pracuje s adresami v rámci fyzického adresového prostoru, ale aplikace běžící pod operačním systémem musí pracovat ve svém vlastním virtuálním adresovém prostoru. Každá aplikace má k dispozici svůj virtuální adresový prostor, ve kterém je spuštěna a je zcela odstíněna od práce s adresami fyzickými, které se od nich liší. Operační systém se pak stará o přidělování fyzického prostoru aplikacím a jeho mapování do virtuálního prostoru. Pro získání fyzické adresy je proto potřeba využít služeb operačního systému, který provede potřebný překlad virtuální adresy. Tento proces se nejčastěji odehrává v ovladači zařízení.

Jak již bylo zmíněno výše, zařízení připojené přes sběrnici PCI Express obsahuje několik registrů, přes které probíhá komunikace se zařízením. Aby bylo možné k těmto registrům přistupovat, musí být začleněny do fyzického adresového prostoru. Tomuto procesu se říká mapování, při něm se registrům zařízení vyhradí místo v adresovém prostoru. Přístup na tuto adresu probíhá z pohledu procesoru stejně jako do hlavní paměti, jen se místo operace s pamětí provede operace s registry namapovaného zařízení.

Správa přerušení

Navržené PCI Express zařízení pracuje v režimu *Bus Master* proto, aby bylo možné implementovat DMA přenosy mezi zařízením a hlavní pamětí počítače. Tím, že je řízení přenosu dat předáno zařízení, nemůže procesor zjistit, zda je DMA přenos dat již dokončen. K tomuto účelu je vhodné využívat podsystém přerušení. Zařízení pak po dokončení DMA

operace vyvolá přerušení procesoru, který následně provede jeho obsluhu. Přerušení využijeme i v případě, kdy je třeba ze strany zařízení upozornit ovladač, například na dokončení nějaké uživatelské operace.

Při startu počítače jsou všem připojeným zařízením přidělena čísla přerušení. Ovladač pak pouze zaregistruje u operačního systému příjem tohoto přerušení a přiřadí ho k volání obslužné funkce. V rámci ní proběhne přečtení stavového registru zařízení a vyvolání příslušných akcí.

Paměť zařízení

Pro navrhované PCI Express zařízení je vhodné implementovat paměťový blok, který bude sloužit jako mezipaměť pro přijatá a odeslaná data. Ostatní obvody pak mohou k této paměti přistupovat. Využití najde například jako framebuffer snímků kamery, které sem během zpracovávání postupně kopíruje uživatelský obvod. Ten poté může dostupnými prostředky, například přerušením, upozornit ovladač, že jsou k dispozici data pro přenesení.

4.3 Řadič Ethernet

Pro řešení projektu je k dispozici upravená kamera Unicam M621 od firmy Camea. Kamera se připojuje přes rozhraní Ethernet, na kterém běží jednoduchý komunikační protokol postavený nad UDP protokolem. Komunikace je velice zjednodušená, kamera posílá nepřetržitě na výstup UDP datagramy se snímaným obrazem. Pakety jsou konstantní délky a obsahují přesně 966 pixelů obrazu. Následují počítadla snímků a paketů v rámci snímků(viz Kapitola 3.2). Kamera začíná odesílat data obrazu okamžitě po startu a nevyžaduje žádné potvrzovací či konfigurační pakety ze strany připojeného zařízení. Nicméně i tak je možné zasílat kameře jednoduché povely spojené zejména s kvalitou obrazového výstupu.

Připojení kamery

Důležitý je výběr z možností připojení kamery do designu. Protože je kamera připojena přes Ethernetové rozhraní a implementuje standardní komunikační protokol UDP pro přenos obrazu, je vcelku nasnadě připojit kameru k síťové kartě počítače. Výhody tohoto řešení jsou zřejmé, ovladače síťové karty jsou k dispozici a každý operační systém podporuje všechny hlavní síťové protokoly, včetně kamerou podporovaného UDP. Obslužná aplikace pro čtení dat z kamery, která by běžela na počítači, je v tomto případě jednoduchá a snadno realizovatelná. Výhodou pak může být ještě fakt, že kopie snímků by již byly uchovány v paměti počítače a z karty by se přenášely zpět například jen výsledky zpracování.

Toto řešení má ale jednu zřejmou nevýhodu. Jak již bylo zmíněno v předchozí kapitole, v kameře není implementována podpora kódování videa. To znamená relativně velké přenosy dat mezi síťovou kartou počítače a vývojovým kitem, zvlášť při vysokém rozlišení obrazu nebo vyšším FPS. Pro případ kamery M621 s černobílým obrazem o rozměrech 752x478 a 60-ti snímky/s je zapotřebí přenést 22MB dat za vteřinu. Kit je vybaven připojením ke sběrnici PCI Express a jeho teoretická přenosová rychlost je výrazně větší, což pro přenos videa stačí. Nepřijatelné ovšem může být zatížení jak procesoru, tak hlavní paměti obslužného počítače spojené se zpracováváním příchozích paketů.

Existuje zde i jiná varianta řešení problému. Vývojový kit SP605[4] obsahuje zásuvku a čip Marvell Alaska 88E1111 pro realizaci fyzické vrstvy Ethernetu. Je tak možné zapojit

kameru přímo ke kartě, což řeší problém předchozího návrhu. Další problém ale nastává v implementaci transportního protokolu UDP a síťového IP potokolu, což není v FPGA triviální problém. Velkou výhodou při implementaci je fakt, že kamera po spuštění odesílá obraz automaticky bez ohledu na to, zda někdo naslouchá. Zároveň nevyžaduje žádné potvrzování přijetí dat. Pokud je ale kamera připojena přímo ke kitu, je zde možnost navrhnout řadič v FPGA tak, aby jen naslouchal kamerou streamovaná data. Odpadá tak implementace síťových protokolů, což značně zjednodušuje řadič. Tento návrh se zatím jeví jako nejvhodnější pro implementaci.

4.4 Vyrovnávací paměť snímků

Protože jsou přenosy dat z kamery velice rychlé a po poměrně velkých blocích(celé řádky), je vhodné implementovat nějaký druh vyrovnávací paměti příchozích snímků. Může totiž docházet k situacím, kdy je obvod pro zpracování obrazu dočasně plně vytížen a hrozí, že dojde ke ztrátě nově příchozích dat. Nejjednoduším řešením je použití pamětí *BlockRAM*, které se nacházejí na čipu FPGA. Tyto paměti jsou dvouportové, je z nich možné v jednom hodinovém taktu zároveň číst i zapisovat a tím pádem se skvěle hodí pro implementaci vyrovnávacího bufferu. Bohužel velikost jednoho paměťového bloku není nijak závratná, pouhých 18kb. To při šířce řádku například 752 pixelů(v 8-bitových odstínech šedi) stačí pouze na 3 řádky.

Paměťové bloky BRAM se dají velice dobře spojovat do větších celků, problémem je ale jejich dostupný počet na čipu. Implementace některých obvodů pro zpracování obrazu jich může zabrat poměrně velké množství. Například detekční engine prezentovaný v článku *Hardware Detection of Scalable Objects Based on Adaboost Classifier*[7] potřebuje k činnosti celkem 96 ze 116 BRAM bloků dostupných na FPGA kitu SP605. Několik bloků potřebuje také PCI Express zařízení, proto je jich k dispozici velice omezené množství. Stěží je tak možné vytvořit vyrovnávací paměť na více než pár desítek řádků. Pokud by byly požadavky na paměť přece jen příliš velké, je zde ještě možnost využít 128MB DDR3 pamětí dostupných na vývojovém kitu SP605[4].

Kapitola 5

Realizace komunikačních rozhraní

Kapitola se zaměřuje na popis implementace obvodů navržených v kapitole 4. Nachází se zde popis architektur obou komunikačních rozhraní a konkrétních technik použitých při jejich realizaci. Samozřejmostí je podrobný popis výstupních rozhraní, která budou dostupná pro připojení obvodů pro zpracování dat. Nechybí souhrn klíčových vlastností obvodů, jako je test maximální přenosové rychlosti, požadavky na zdroje, taktovací frekvence a další. Závěr kapitoly je věnován představení hardwarové aplikaci pro detekci hran v obraze s použitím zde prezentovaných rozhraní.

5.1 PCI Express zařízení

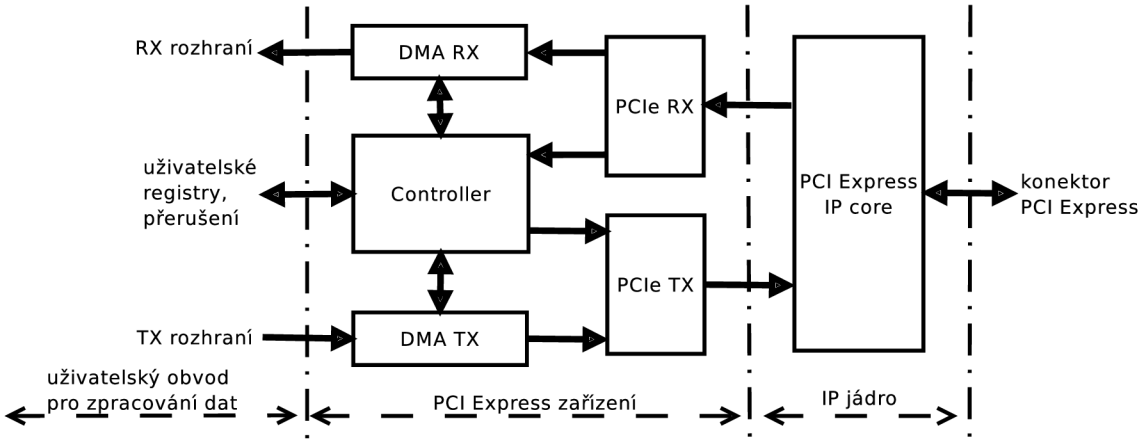
Podkapitola se zaměřuje na samotnou implementaci PCI Express zařízení. Jsou zde popsány a vysvětleny všechny klíčové vlastnosti zařízení a použité techniky návrhu a realizace.

Architektura zařízení

Obvod zařízení se skládá z několika dílčích částí(viz Obrázek 5.6). Základním stavebním kamenem je IP jádro PCI Express, kde je jeho velká část realizována jako tzv. *hard* jádro, tedy vyrobená přímo na čipu. IP jádro zajišťuje funkcionalitu nižších vrstev protokolu PCI Express, jeho výstupem jsou až pakety nejvyšší, transportní vrstvy[9]. Jádro je poskytováno pod licencí společnosti Xilinx[13], která je výrobcem čipu FPGA. Ostatní řídící logika byla doimplementována.

K IP jádru jsou připojeny dva obvody, *PCIe RX* a *TX*, jež mají za úkol dekódovat informace z hlaviček paketů či naopak hlavičky vytvořit z dostupných dat. Údaje z hlaviček a datová část jsou pak zvlášť předávány řídícím obvodům. Tvoří tak mezivrstvu mezi IP jádrem a ostatními obvody, která upravuje a do značné míry zjednodušuje stávající rozhraní IP jádra.

Nejsložitější částí obvodu je *Controller*. Má za úkol zpracovávat údaje hlaviček paketů, udržovat jejich kontext a na jejich základě vykonávat příslušné akce a řídit obvody pro zpracování DMA přenosu. Blok obsahuje sadu řídících registrů, které jsou namapované do paměťového prostoru počítače. *Controller* dále řídí přenosy DMA a k tomu přitom využívá pomocných bloků *DMA RX a TX*.



Obrázek 5.1: PCI Express zařízení: Znázornění architektury obvodu.

IP jádro PCI Express

Základem celého zařízení je jádro *Integrated Endpoint Block v2.3 for PCI Express*, které je kompatibilní s architekturou Spartan-6. Dostupná je jedna linka PCI Express kompatibilní se specifikací *PCI Express Base Specification v1.1*. Dle specifikace je teoretická propustnost tohoto rozhraní 250MB/s. Jádro implementuje standardizované *AXI4-Stream* rozhraní pro uživatelské obvody. Rozhraní je navrženo pro maximální propustnost, je proto plně duplexní. Veškerá odesílaná a přijímaná data jsou jádrem interně bufferována, což rozhraní rozšiřuje o jednoduchou kontrolu toku dat. Zároveň ale existuje možnost zakázat bufferování příchozích paketů a přeposílat je přímo uživatelskému obvodu tak, aby byla eliminována část latence vzniklé ukládáním paketů v bufferech. Jádro podporuje maximální velikost příchozích i odchozích paketů až 512B. IP jádro realizuje nižší vrstvy protokolu PCI Express, jeho výstupem jsou pakety transportní vrstvy[9].

PCIe RX a TX obvody

Vstupem a výstupem IP jádra jsou kompletní pakety transportní vrstvy protokolu PCI Express[9]. Tyto dva obvody zajišťují dekódování nebo naopak skládání informací do hlaviček paketů a oddělují tak řídící informace od samotných dat. Hlavní část obvodu *PCIe RX* tvoří stavový automat. Jakmile ho IP jádro informuje o příchodu nového paketu, začne se čtením hlavičky. Data postupně dekóduje a vystavuje na samostatné výstupy. Poté, co je hlavička dekódována, informuje o této skutečnosti signálem *"ready"* navazující obvody. Těm jsou prezentovány dekódované informace a následně předáno řízení čtení zbývající části paketu, kde se nachází již samotná data. Jedná se o rozdělení složitosti řízení do více vrstev, kdy se navazující obvody nemusí zabývat formátem nebo dekódováním paketu.

Obdobně je implementován i obvod *PCIe TX*, který naopak na žádost navazujících obvodů hlavičky sestavuje. Řídící obvody zapíší na porty informace pro sestavení hlavičky a činnost obvodu zahájí vystavením signálu *"enable"*. V tom okamžiku se spustí automat, který z dostupných parametrů sestaví hlavičku a odešle ji IP jádru. Jakmile je hlavička hotova, obvod požádá řídící obvody o datovou část paketu.

Controller

Hlavním řídícím prvkem celého zařízení je blok *Controller*. Jak již název napovídá, tento obvod řídí a synchronizuje činnost ostatních obvodů.

Typické periferní zařízení obsahuje registry, jejichž obsah lze měnit vykonáním některých instrukcí procesoru. Implementace registrů se nachází právě v tomto obvodu. Aby k nim bylo možné přistupovat, *Controller* implementuje jednoduchou logiku pro příjem a odesílání paketů, která je třeba pro provádění čtení, zápisu a generování odpovědí na čtení registru. Další funkcionalitou obvodu je řízení DMA přenosů. Obvod je proto přímo napojen na bloky *DMA RX a DMA TX*, jejichž činnost řídí na základě obsahu registrů zařízení. Součástí řízení DMA je i mechanismus vyvolání přerušení v hostitelském počítači. Tento obvod je dále popsán v kapitole 5.1.

Obvody DMA RX a TX

Tyto bloky mají za úkol řídit veškerou agendu spojenou s DMA přenosy. Jejich činnost řídí *Controller*. Ten na základě hodnot ve svých registrech, do kterých zapisuje nastavení hostitelský počítač, inicializuje činnost DMA obvodů.

DMA RX obvod nejen že řídí DMA přenos dat směrem do zařízení, ale také poskytuje rozhraní uživatelským obvodům odebírajícím přenesená data. Toto rozhraní obsahuje kromě datových a synchronizačních vodičů také adresu přenesených dat. Nejedná se o adresu v rámci paměťového prostoru počítače, ale o adresu v rámci vlastního paměťového prostoru zařízení, kterou je s každým přenosem možné libovolně měnit zápisem do registrů zařízení. Tato vlastnost není klíčová například pro správnou funkci přenosu streamovaných dat, kde se adresa nemusí uplatnit(a nastavení registrů lze ignorovat), ale významně rozšiřuje a zjednodušuje možnosti zařízení pro připojení s jinými obvody, například pamětí DDR či BlockRAM. Dále je možné tuto adresu využít pro adresaci dat jednomu z více uživatelských zařízení připojených současně na datové rozhraní(tzv. *Chip select*).

Obdobně pracuje také obvod *DMA TX*, který je určen pro odesílání dat do hlavní paměti počítače. Obvod musí kontrolovat maximální velikost paketu, kterou je možno odeslat a nedopustí její překročení tím, že rozdělí stream dat do více paketů. Obvod vyžaduje od uživatelských obvodů data z adresy, která je nastavena v registrech zařízení. Adresa může být uživatelskými obvody ignorována, ale platí, že tak může výrazně zjednodušit připojení například paměťových modulů, protože není třeba implementovat vlastní generátor adres.

Řadič zařízení - Controller

Controller je hlavní částí celého zařízení, řídí a koordinuje činnost ostatních obvodů. Činnost obvodu by se dala rozdělit do následujících kategorií.

Zpracování read/write instrukcí

Instrukce čtení a zápisu do registru zařízení provádí v naprosté většině případů procesor při vykonávání kódu ovladače zařízení. Pro úplnost existuje i možnost čtení/zápisu od jiného zařízení na sběrnici, což ale není případ tohoto zařízení. Interní registry zařízení jsou mapovány do paměťového prostoru počítače, takže čtení a zápis dat do registrů probíhá pro procesor zcela transparentně s transakcemi z/do hlavní paměti(viz Podkapitola 2.2).

Příchozí transakce jsou zpracovávány stavovými automaty. Jakmile obvod *PCIe RX* upozorní na nově příchozí paket, který spadá do paměťového prostoru registrů, je spuštěna obsluha příslušné operace. Výhodou paměťově mapovaných registrů je fakt, že pro zápisové instrukce není třeba odesílat potvrzení o úspěšném zápisu, jako je to v případě IO instrukcí procesoru, které používají vlastní adresový prostor. Dochází tak z vyšší propustnosti zápisových operací, kterých je v současné implementaci většina. Operace čtení registru samozřejmě odesílají odpověď vždy, proto musí mít *Controller* sdílený přístup ke sběrnici obvodu *PCIe TX*.

Registry zařízení jsou implementovány jako jednoportová distribuovaná RAM paměť s asynchronním čtením, je tak řešen sdílený přístup různých obvodů ke čtení registrové sady. Pokud jde o nastavování příznaků do stavového registru, například indikace dokončení DMA přenosu, tak je tento případ ošetřen speciálním stavem obvodu pro odesílání dat, který na základě shody adresy čtení a adresy stavového registru odešle obsah stavového registru a ne hodnotu uloženou v registrové sadě.

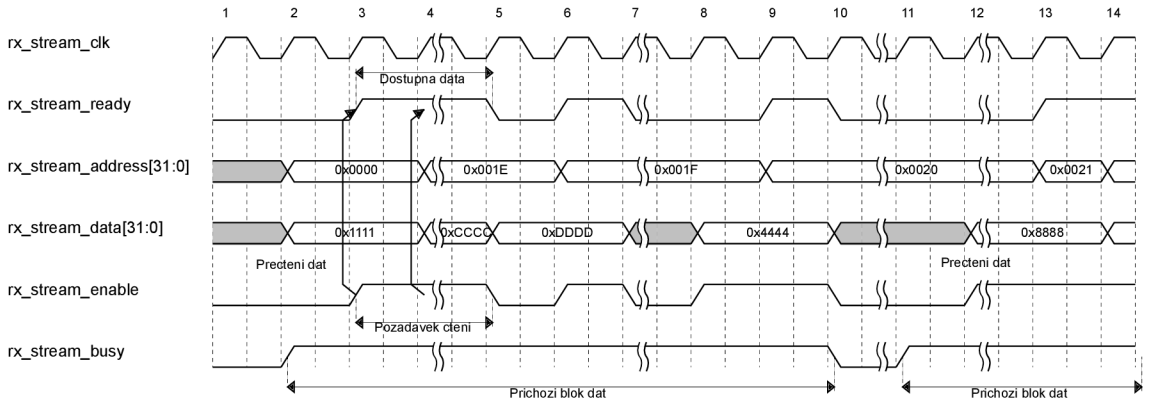
V zařízení může nastat situace, kdy dorazí další instrukce pro práci s registry dříve, než je předcházející zpracována. Může se tak stát například u instrukce čtení, kdy odeslání odpovědi může trvat delší dobu kvůli sdílení výstupní sběrnice s dalšími obvody. IP jádro je na tento případ vybaveno signálem blokování, který v interní vstupní frontě pozdrží pakety se žádostmi čtení/zápisu po dobu vystavení tohoto signálu v log. 1. Zároveň mohou být tyto pakety předběhnuty jinými, určenými například DMA obvodům. Nedochází tak ke zbytečným prodlevám při přenosu dat.

Řízení DMA přenosů

DMA bloky pracují téměř autonomně. Úkolem *Controlleru* je správně inicializovat jejich parametry a následně je spustit. Parametry DMA přenosu nastavuje ovladač zařízení zápisem do příslušných registrů. Pro DMA přenos je třeba do registrů zapsat fyzickou adresu dat v hlavní paměti. Samozřejmostí je podpora 64-bitového adresového prostoru. Další, ale volitelnou položkou je adresa v rámci adresového prostoru karty, na kterou se budou data přenášet. Toto nastavení má smysl pouze pro případ, že uživatelský obvod bude využívat adresových výstupů PCI Express zařízení. Posledním parametrem je délka přenášeného bloku dat.

Existují dva limity velikosti paměti, kterou je možné na jednu transakci přenést. Jedním je celková velikost transakce a je závislá na ovladači zařízení. Konkrétně jde o velikosti bufferů, které mu pro DMA přenosy dovolí jádro operačního systému alokovat. Přenosy se proto mohou odehrávat pouze do velikostí těchto bloků, pro větší bloky paměti je třeba spuštění DMA opakovat. O řešení tohoto problému se stará ovladač a tak je tento limit pro aplikační software neviditelný. Tento limit platí pro oba směry DMA přenosu.

Druhý limit platí pouze pro přenos dat z hlavní paměti do zařízení. Je jím maximální velikost bloku paměti, o kterou si může při čtení z hlavní paměti PCI Express zařízení zažádat v jedné transakci. Ta je definována parametrem *MAX_READ_REQUEST_SIZE*, který nastavuje operační systém počítače v době enumerace PCI Express sběrnice. Limit je zaveden proto, aby si zařízení nemohla naalokovat příliš velkou část přenosového pásma hlavní paměti a nedošlo tak k výraznému omezování ostatních zařízení. Problém platí pouze pro přenos dat směrem z hlavní paměti, je spjat s instrukcí čtení paměti. Protože je ale podle standardu možné mít současně až 32 rozpracovaných transakcí na jedno PCI Ex-



Obrázek 5.2: Příjem dat z DMA rozhraní PCI Express zařízení.

press zařízení, tak Controller spravuje několik(parametrem ve zdrojovém kódu lze nastavit) rozpracovaných transakcí tak, aby došlo k překrytí latence mezi jednotlivými žádnostmi o paměť a tím zvýšení maximální přenosové rychlosti DMA přenosu.

Správa přerušení

S implementací DMA přenosů úzce souvisí i možnost vyvolání přerušení procesoru. Jedná se o efektivní způsob, jak ovladači zařízení oznámit, že byl DMA přenos dokončen. IP jádro již implementuje řídící signály pro zasílání přerušení, stačí proto pouze vystavit několik signálů, aby se tak stalo. S dokončením DMA operací jsou spojeny příznaky, které se mapují do stavového registru. Po vyvolání přerušení ovladač provádí čtení tohoto registru a zjistí tak, které operace byly dokončeny. Následným zápisem zpět do stavového registru potvrzuje, že bere na vědomí konec těchto operací.

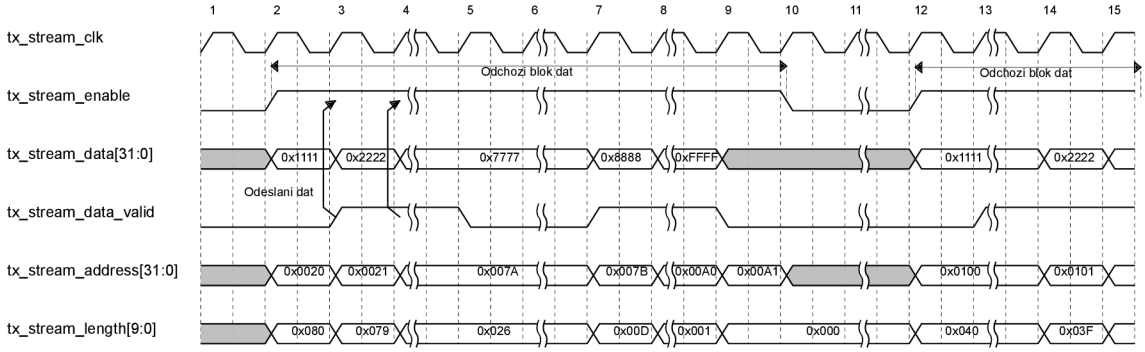
Controller mapuje celkem tři interní přerušení(v rámci *Controlleru*) na jedno systémové. Dvě interní přerušení vyvolávají RX a TX kanály DMA a třetí může přijít z uživatelského obvodu. Nejnižší tři bity stavového registru reprezentují jednotlivá přerušení, kdy log.1 znamená vystavení přerušení. Potvrzení přijetí jednotlivých přerušení proběhne zapsáním slova do stavového registru, které má logické jedničky na místech těch přerušení, která ovladač potvrzuje.

Uživatelské rozhraní

Rozhraní zařízení je navrženo s ohledem na co nejjednoduší napojení dalších obvodů. Navíc není řešeno jednoúčelově, ale naopak se snaží o univerzální řešení, které je použitelné pro širší škálu použití. Připojené obvody jsou navíc zcela odstíněné od skutečnosti, že pracují se sběrnicí PCI Express. Všechny signály uvedené v této části podkapitoly jsou aktivní v logické 1.

DMA rozhraní

Rozhraní pro čtení je výstupem obvodu *DMA RX*. Tímto rozhraním prochází pouze a jen data přenesená pomocí DMA přenosu, ostatní data, například řídící informace nebo další transakce na sběrnici PCI Express jsou vyfiltrovány. Rozhraní má charakterem blízko k frontě FIFO. K dispozici je 32-bitová datová sběrnice *rx_stream_data*, platnost dat na ní vystavených je potvrzována signálem *rx_stream_ready*. Uživatelský obvod signalizuje



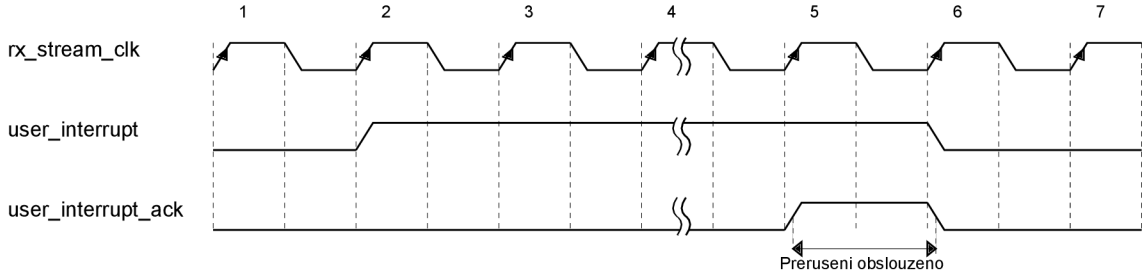
Obrázek 5.3: Odesílání dat přes DMA rozhraní PCI Express zařízení.

čtení vystavením signálu *rx_stream_enable*. Současným vystavením signálu *rx_stream_ready* a *rx_stream_enable* při náběžné hraně hodin *rx_stream_clk* dojde k úspěšnému přečtení dat ze sběrnice. Připojený obvod navíc může pomocí přerušovaného vystavování signálu *rx_stream_enable* libovolně řídit rychlost toku dat. Časový průběh signálů rozhraní je znázorněn na obrázku 5.2.

Rozšířením RX rozhraní je 32-bitová adresová sběrnice *rx_stream_address*, která datům přiřazuje adresy v rámci vnitřního adresového prostoru zařízení(viz. Podkapitola 5.1).Využití najde například při připojení paměťových modulů ke sběrnici. Dalším signálem je *rx_stream_busy*. Komunikace po sběrnici PCI Express probíhá po paketech, mezi kterými se může vyskytovat různě velká časová prodleva. Za tímto účelem rozhraní obsahuje právě signál *rx_stream_busy*, který je vystaven vždy po dobu přenosu jednoho bloku dat. Tato signalizace se dá s výhodou využít při připojení PCI Express zařízení k obvodům, která pracují s pakety či bloky dat, jako například obvod pro zápis do DDR pamětí a podobně. V případě DDR pamětí tak může po skončení přenosu bloku dat dojít k zahájení zápisu do paměti, který je relativně zdlouhavý a překrýt tak tuto režii s čekáním na další blok dat z PCIe linky.

Vstup obvodu *DMA TX* tvoří rozhraní pro odesílání dat. TX rozhraní tvoří opět 32-bitová datová sběrnice *tx_stream_data*. Přenos dat je řízen *DMA TX* obvodem a signál *tx_stream_enable* signalizuje operaci čtení. Vystavením tohoto signálu do log. 1 je uživatelský obvod vyzván k vystavení dat na datovou sběrnici. Ten poté signálem *tx_stream_data_valid* potvrzuje, že se na sběrnici *tx_stream_data* zapsal platná data. Současným vystavením signálů *tx_stream_enable* a *tx_stream_data_valid* jsou data ze sběrnice úspěšně přenesena. Pozdržením signálu *tx_stream_data_valid* je možné řídit rychlost toku dat obdobně, jako je tomu v případě RX rozhraní.

Rozhraní je rozšířeno také o 32-bitovou adresovou sběrnici *tx_stream_address*, která datům přiřazuje adresy v rámci vnitřního adresového prostoru zařízení(viz. Podkapitola 5.1). Adresa najde využití například při připojení paměťových bloků k PCI Express zařízení. Vylepšením rozhraní je specifická sběrnice *tx_stream_length*, která udává délku právě čteného bloku dat. Velikosti bloků jsou odvozovány z maximálních velikostí paketů, které smí PCI Express zařízení odesílat po sběrnici (maximální velikost paketů je nastavována systémem). Toto vylepšení může najít uplatnění v lepší práci s obvody podporujícími blokové přenosy dat, jako jsou například DDR paměti. Časový průběh signálů rozhraní je znázorněn na obrázku 5.3.



Obrázek 5.4: Signalizace hardwarového přerušení pomocí zařízení PCI Express.

Registry a přerušení

Rozhraní zařízení také nabízí možnost implementovat až 992 uživatelských registrů o šířce 32 bitů. Jejich obsah je plně v režii uživatelských obvodů a může je využít například pro komunikaci s aplikací běžící na počítači. Rozhraní obsahuje adresovou sběrnici *user_reg_address*, na kterou je vždy vystavena adresa registru, se kterým se pracuje. Dále je k dispozici datová sběrnice *user_reg_data_in*, na kterou uživatelský obvod vystavuje obsah čteného registru. Čtení registrů probíhá asynchronně, na této sběrnici by měly být vždy vystavena data korespondující k adrese vystavené na adresové sběrnici *user_reg_address*. Na datovou sběrnici *user_reg_data_out* jsou PCI Express zařízením vystavována data pro zápis do uživatelských registrů. Zápisová operace je synchronní s vystavením signálu *user_reg_write_enable* do logické 1.

Uživatelský obvod má také možnost vyvolat přerušení a tak asynchronně zaslat zprávu aplikaci. Žádost o přerušení je signalizována vystavením signálu *user_interrupt*. Ten musí být držen v logické 1 do doby, dokud neproběhne potvrzení přechodem signálu *user_interrupt_ack*. V tomto okamžiku bylo již přerušení správně obslouženo ovladačem zařízení. Bezprostředně poté musí uživatelský obvod svou žádost ukončit přechodem signálu *user_interrupt* do logické 0. Průběhy obou signálů jsou znázorněny na obrázku 5.4.

Možnosti připojení

Připojení uživatelských obvodů v FPGA je velice jednoduché. Je třeba pouze základní řídící logiky pro čtení/zápis dat. Realizace přenosu dat z a do FPGA je jinak zcela v režii ovladače a PCI Express zařízení. Aplikace běžící na hostitelském počítači tak řídí skrze volání ovladače, která data budou odkud kam přenesena.

Rozhraní zařízení je navrženo nejen pro streamové přenosy dat, ale díky pomocným signálům a mechanismu generování adres se výborně hodí pro přímé připojení komponent BlockRAM. Dále je možné blok připojit i k DDR pamětím (na kitu SP605 je 128MB DDR3 paměti), ale je třeba implementovat alespoň základní řízení a správu blokového čtení/zápisu, které je nutné pro práci tímto druhem periferie. Potřebné obvody byly implementovány v rámci této práce a lze je tak v případě potřeby použít.

Ovladač zařízení

Nedílnou součástí jakékoli periferie počítače, tedy i PCI Express zařízení je ovladač zařízení. Ovladač vytváří abstrakci nad cílovým zařízením - řeší všechny implementační detaily komunikace s kartou a navenek poskytuje jednotné rozhraní pro uživatelské aplikace.

Knihovna sp605.h

Pro práci s ovladačem byla vytvořena knihovna *sp605.h*, která má za úkol co nejvíce zjednodušit programování aplikace. Na začátku práce s knihovnou je nutné zavolat funkci *initDriver*, která připraví ovladač a inicializuje knihovnu. Před ukončením aplikace musí být volána funkce *closeDriver*, která korentním způsobem ukončí práci s ovladačem.

Základní podporu tvoří funkce pro manipulaci s registry zařízení. Pro čtení a zápis do registru slouží funkce *readReg* a *writeReg*. Parametry funkcí jsou číslo registru a 32-bitová hodnota v případě zápisu. Při čtení registru je parametrem ukazatel, na který bude po skončení operace zapsán 32-bitový obsah čteného registru.

Pro práci s DMA přenosy jsou k dispozici funkce *dmaRead* a *dmaWrite*. Názvosloví je z pohledu aplikace, kdy *dmaRead* provádí čtení dat ze zařízení v FPGA do RAM paměti počítače. Obě funkce potřebují stejné tři parametry, ukazatel na pole s daty(nebo pro uložení dat), délku datového přenosu v bajtech a počáteční adresu přenosu v rámci adresového prostoru FPGA zařízení. Volání těchto funkcí je blokující a po návratu z volání funkce *dmaRead* jsou v na místě předaného ukazatele platná data. Totéž platí i pro *dmaWrite*, kdy návrat z volání funkce znamená úspěšné zapsání všech dat do zařízení na čipu FPGA.

Průběh DMA přenosu

Tento odstavec si klade za cíl vysvětlit průběh přenosu dat od volání knihovní funkce pro zahájení DMA přenosu po příjem dat na straně uživatelského obvodu v FPGA.

Uvažujme aplikaci na PC, která chce odeslat data přes sběrnici PCI Express do čipu FPGA, kde se nachází (v této práci prezentované) PCI Express zařízení spolu s uživatelským obvodem na zpracování dat. Aplikace si připraví data do souvislého bloku paměti (např. pole) a funkci *dmaWrite* předá ukazatel na tento blok společně s velikostí dat v bajtech. Součástí volání funkce je i adresa v rámci adresového prostoru zařízení v FPGA.

Knihovna pomocí volání funkcí *ioctl* sdělí ovladači parametry přenosu a posléze ho dalším voláním této funkce zahájí. Ovladač provede všechny operace nutné pro přenos, jako zkopírování přenášených dat do interních bufferů ovladače, překlad virtuální adresy tohoto bufferu na fyzickou a následný přenos parametrů DMA do registrů zařízení.

Zápisem do kontrolního registru spustí ovladač činnost PCI Express zařízení. To začne postupně generovat žádosti o čtení hlavní paměti z adresy, kterou mu předal ovladač. Jak data postupně přicházejí do zařízení, prochází *DMA RX* obvodem na výstupní rozhraní, kde jsou prezentována uživatelskému obvodu. Zde se uplatní interní adresa předaná při volání knihovní funkce obslužnou aplikací. Tato adresa je na výstupním rozhraní postupně inkrementována s příjatými daty. Adresové vodiče se dají velmi dobře využít pokud je připojeným obvodem paměťový blok, například BlockRAM.

Pokud je přenášená velikost dat příliš velká a ovladač není schopen kvůli velikosti interních bufferů přenést data v rámci jednoho DMA přenosu, knihovna tento případ ošetří a inicializuje potřebný počet DMA přenosů na přenesení celého objemu dat.

Obdobný scénář se odehrává pro případ přenosu dat přes PCI Express zařízení do RAM paměti počítače. Uživatelská aplikace volá funkci *dmaRead*, které jako parametry předá prázdný buffer pro uložení dat a velikost dat ke čtení. Může také předat adresu v rámci paměťového prostoru zařízení, ze které se bude provádět čtení. Tyto údaje jsou ovladačem zapsány do registrů zařízení. Po následném spuštění DMA operace nastaví *controller* blok

DMA TX a ten začne samostatně pracovat. Obvod DMA vystavuje požadavky na čtení dat z uživatelských obvodů (zde se může opět uplatnit zadaná interní adresa) a ty pak, rozdělené na pakety maximální dovolené velikosti, odesílá po sběrnici PCI Express do paměti RAM.

Přenosové operace jsou dokončeny vždy před návratem z volání funkcí *dmaRead* nebo *dmaWrite*.

Vlastnosti implementace

Zařízení je úspěšně implementováno a otestováno na vývojovém kitu Xilinx SP605[4] s rozhraním PCI Express x1 verze 1.1[9]. Na FPGA rodiny Spartan-6 zabírá obvod celkem 714 bloků Slice(pro FPGA čip na kitu to představuje 10%). Kmitočet obvodu zařízení je omezen na hodnotu 125MHz, která je daná tím, že rozhraní IP jádra pracuje právě na této frekvenci. Jádro si pro realizaci interních bufferů vyžádá 2 - 18 komponent BlockRAM(2 - 21% dostupných bloků), jejich počet je ale nastavitelný a závisí na něm výsledná maximální propustnost rozhraní.

Teoretická propustnost rozhraní PCI Express 1.1[9] je 250MB/s, přičemž po započtení režie přenosů(hlavičky paketů) se rychlost redukuje na maximálně 200MB/s. Prezentované zařízení dokáže podle měření přenášet streamy dat rychlostí až 182MB/s při čtení paměti RAM a 177MB/s při zápisu do paměti RAM(viz. Tabulka 5.1). Tyto údaje jsou ale závislé na několika faktorech. Mezi ně patří například použitý hardware (zejména chipset a jeho vlastnosti), dále operační systém a ovladač zařízení, jehož implementace může mít na rychlost zásadní vliv. Mezi neméně důležité parametry patří konstanty *MAX_READ_REQUEST_SIZE* a *MAX_PAYLOAD_SIZE*, které nastavuje systém během enumerace zařízení na sběrnici. Parametr *MAX_PAYLOAD_SIZE* definuje, jakou maximální délku může mít datová část paketu. Běžná velikost je 128B, ale některé systémy podporují velikosti větší.

MAX_READ_REQUEST_SIZE je parametr, který udává o jakou velikost paměti může zařízení zažádat v jedné transakci čtení z hlavní paměti (operace *"memory read"*). Tento parametr zabraňuje tomu, aby si některé zařízení alokovalo příliš velkou šířku přenosového pásma hlavní paměti a tím pádem omezovala ostatní zařízení.

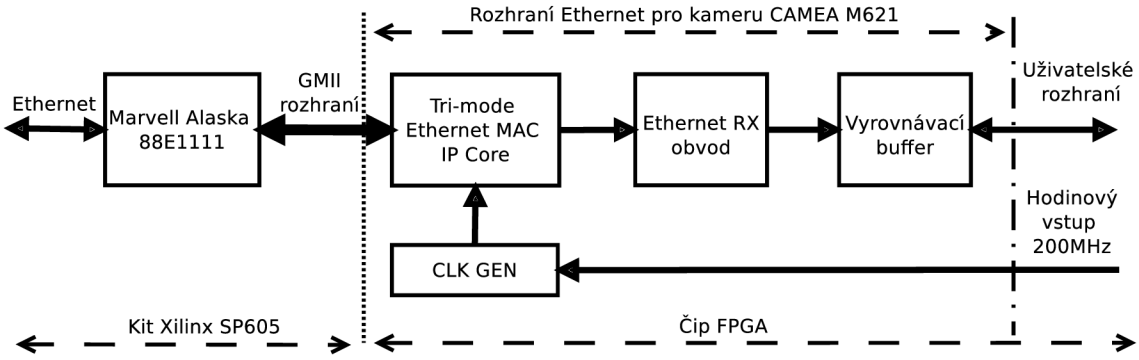
Maximální rychlost datového přenosu přes PCI Express zařízení může významně ovlivňovat také propustnost připojených uživatelských obvodů, které mají možnost řídit rychlost datového toku.

př. rychlost [MB/s]	4kB	8kB	16kB	32kB	64kB	128kB	256kB	512kB	1MB
RAM ⇒ FPGA	70.3	107.9	143.6	158.1	169.8	173.3	176.0	176.5	177.1
FPGA ⇒ RAM	83.6	118.8	151.7	166.3	179.1	182.3	182.5	182.7	182.7

Tabulka 5.1: Tabulka dosažených přenosových rychlostí pro různě velké bloky dat.

5.2 Rozhraní kamery

Tato podkapitola pojednává o architektuře a specifických vlastnostech Ethernetového rozhraní kamery Unicam M621. Obsahuje také popis výstupního rozhraní obvodu, kterým jsou uživatelským obvodům předávána obrazová data z kamery.



Obrázek 5.5: Znázornění architektury Ethernetového zařízení.

Architektura rozhraní

Pro realizaci fyzické vrstvy Ethernetu je na kitu Xilinx SP605[4] k dispozici čip Marvell Alaska 88E1111[1]. Čip podporuje Ethernetovou komunikaci o rychlosti 10/100/1000 Mb/s a je s FPGA propojen rozhraním GMII. Celé zařízení se skládá z IP jádra *Ethernet MAC* a obvodů *Ethernet RX*, *Image Buffer* a *Clock generator*(viz Obrázek 5.5). Vzhledem k povaze zařízení a výstupního rozhraní kamery není nutné implementovat odchozí (TX) část Ethernetového protokolu.

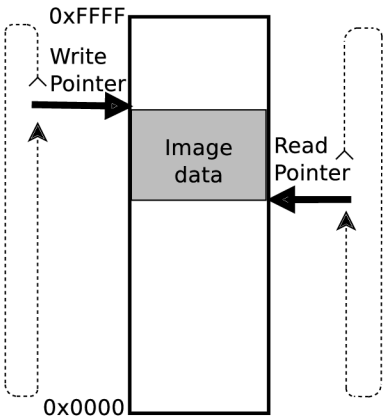
IP jádro Ethernet MAC

Pro realizaci Ethernetového rozhraní v FPGA bylo zvoleno IP jádro *LogiCORE IP Tri-Mode Ethernet MAC v4.5*[2]. Jádro spravuje veškerou komunikaci s čipem Marvel Alaska přes rozhraní GMII, čímž značně zjednodušuje návrh celého zařízení. Dále implementuje kontrolní mechanismy, které s MAC(*Medium Access Control*[8]) vrstvou úzce souvisí. Jedná se například o detekci a ošetření kolizí v případě poloduplexní komunikace nebo o kontrolu CRC součtu v Ethernetovém rámci. IP jádro také dovoluje implementovat jednoduchý filtr MAC adres. Samozřejmostí je pak podpora 10/100/1000 Mbit Ethernetu.

IP jádro je navrženo pro co možná nejmenší latence při zpracování dat. To se promítá do uživatelského rozhraní, které nepodporuje žádnou možnost řízení toku dat. IP jádro neobsahuje žádné vyrovnávací paměti, veškeré příchozí rámce jsou tak přímo posílány na výstupní rozhraní a tak si musí připojený obvod buď implementovat vlastní buffer nebo jinak zajistit, aby byl vždy připraven přijímat data a nedocházelo k jejim ztrátám. Totéž platí i pro odchozí rámce. Jakmile je inicializováno odesílání rámce, tak v každém taktu hodin musí mít obvod přichystána další data na vstupním rozhraní IP jádra.

Generátor hodinového signálu

IP jádro vyžaduje pro gigabitový přenos dat referenční hodinový signál o frekvenci 125MHz. Na kitu SP605 jsou ale k dispozici pouze hodinové vstupy na 200MHz. Pro převod frekvence byl proto v FPGA použit obvod fázového závěsu PLL(komponenta *PLL_BASE*), který podle nastavených parametrů generuje z referenčního kmitočtu požadovaný výstupní kmitočet.



Obrázek 5.6: Rozhraní kamery: Znázornění implementace kruhové vyrovnávací paměti.

Obvod Ethernet RX

Tento obvod, jak již název napovídá, spravuje veškerou příchozí komunikaci. Jedná se o stavový automat, který implementuje základní třídění přijatých rámců. Během kontroly probíhá ověřování některých konstant, jako jsou například kódy vyšších protokolů nebo čísla portů (u protokolů TCP a UDP). Obvod je navržen pro snadné doplnění další funkcionality, pro použití s kamerou je ale upraven tak, aby přijímal pouze Ethernetové rámce zasílané kamerou a ostatní provoz ignoroval.

Výstupní rozhraní tohoto obvodu je připojeno přímo k vyrovnávací paměti snímků. Obvod *Ethernet RX* je navržen tak, aby odfiltroval veškeré nedůležité informace, jako jsou hlavičky protokolů a předával obvodu vyrovnávací paměti pouze datovou část paketu protokolu UDP.

Vyrovnávací paměť snímků

Vstupem bloku vyrovnávací paměti je výstup obvodu *Ethernet RX*, který filtruje data hlaviček síťových protokolů a na výstup přeposílá pouze samotnou datovou část. Ta neobsahuje pouze obrazová data, ale také řídící informace, jako je číslo bloku a snímku.

Hlavní částí celého řídícího obvodu bufferu je stavový automat, který dokáže extrahovat samotná obrazová data od řícících informací. Jeho konstrukci zjednodušuje fakt, že veškeré pakety posílané Ethernetovou kamerou mají konstantní velikost a formát. Lze tak snadno detekovat konec obrazových dat a posléze začít s vyčítáním řícících informací. Podle dokumentace ke kameře Unicam M621(viz podkapitola 3.2) je délka paketu vždy přesně 1024B, z toho 42B patří hlavičkám síťových protokolů, dalších 966B tvoří obrazová data a čtyři části, každá po 4B, obsahují čítač paketů, snímků a stavové informace.

Samotná vyrovnávací paměť je tvořena jedním blokem paměti BlockRAM. Ta je využívána jako kruhový buffer(viz Obrázek 5.6), tzn. příchozí data se postupně zapisují do paměti a neošetřuje se přetečení adresového čítače, které po dosažení konce paměťového rozsahu vrátí zapisovací ukazatel opět na začátek paměti. Stejný princip je používán i pro čtecí ukazatel. V případě, že se oba ukazatele sobě rovnají, fronta je prázdná. Pokud tomu tak není, je možné z paměti číst.

Velikost fronty

Buffer je koncipován pouze jako vyrovnávací a existuje zde možnost přetečení a ztráty dat v případě, že se nebudou z paměti číst data dostatečně rychle a zapisovací ukazatel předběhne čtecí o celou velikost bufferu. V tom případě budou oba ukazatele stejné, buffer se bude jevit jako prázdný a dojde ke ztrátě dat o velikosti kapacity bufferu. Vzhledem k tomu, že je přenos obrazu z kamery streamovaný a neexistuje žádná možnost pozastavení streamu, je tato možnost přípustná. Omezit případný vznik tohoto jevu je možné zapojením více bloků BlockRAM a tím pádem zvýšením kapacity kruhového bufferu. Toto řešení ale stejně nemůže garantovat, že k přetečení a ztrátě dat nedojde. Může pouze zabránit ztrátě dat při dočasném zpoždění vyčítání dat.

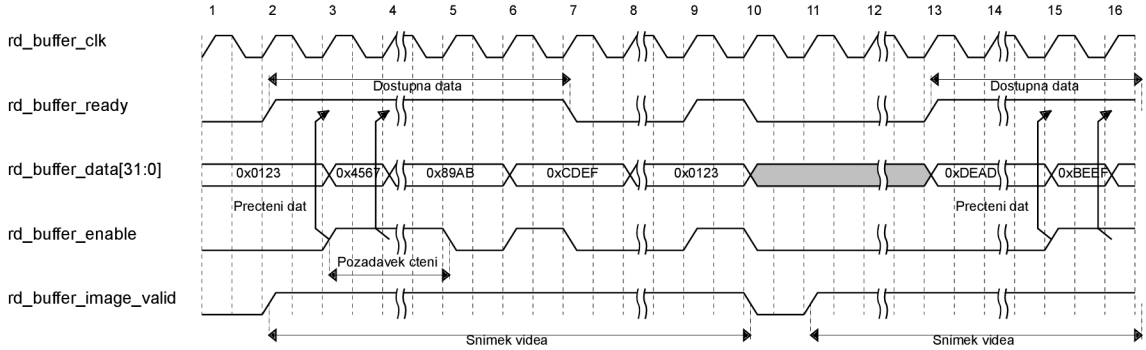
Vyrovnávací paměť je vystavěna nad komponentou BlockRAM. Tento paměťový blok je dvouportový, je tak možné z něj číst a zapisovat data v jednom okamžiku. Každý port paměti navíc může pracovat na jiném kmitočtu. Tato vlastnost se velice hodí pro kruhový buffer, kdy se nemusí operace čtení a zápisu dělit o jeden port paměti a obě operace mohou probíhat zaráz. Na druhou stranu mohou vznikat konflikty způsobené právě odlišnými kmitočty obou portů. To je ošetřeno tím, že logika spojená se správou fronty je čistě kombinační a není tak závislá ani na jednom z kmitočtů portů, což by způsobovalo značné problémy a hazardní stavy. Mohlo by například dojít k situaci, kdy by proběhla čtení dvou slov z paměti, přičemž bylo ve skutečnosti uloženo jen jedno, ale řídící logika synchronizovaná hodinami by nedokázala detekovat stav "prázdná fronta"před druhým čtením a tím mu v operaci zabránit. V současné implementaci je tak použita pouze kombinační logika pro detekci stavů vyrovnávacího bufferu.

Řízení fronty

Obrazová data jsou odesílána kamerou po blocích o velikosti 966B. Protože ale počet pixelů v jednom obraze není dělitelný 966 beze zbytku a poslední paket tak obsahuje kromě zbytku obrazových dat i výplň do délky 966 bajtů, implementuje správa vyrovnávací paměti i čítač přijatých pixelů spolu s komparátorem. Na vstupy komparátoru je tak připojen jak čítač přijatých pixelů, tak konstanta, která definuje počet pixelů snímku při daném rozlišení obrazu kamery, v tomto případě je konstanta nastavena na počet pixelů rozlišení 752x478. V případě připojení kamery s jiným rozlišením, ale stejným protokolem stačí pro správnou funkci vyrovnávací paměti pouze zaměnit tuto konstantu. Komparátor je pak použit pro povolování zápisu do bufferu. Toto řešení se jeví jako nejlepší vzhledem k tomu, že informace o odeslaných datech jsou až na konci kamerou zasílaného rámce a není tak snadné detekovat jiným způsobem konec obrazové části.

Synchronizace snímků

Vyrovnávací paměť je vybavena také signalizací konce snímku. Za tímto účelem je v rámci bufferu implementován registr, do kterého se v případě konce snímku uloží adresa, na které je uložen poslední pixel obrazu. Prostá komparace tohoto registru vůči čtecímu ukazateli nefunguje z důvodu, že buffer má mnohonásobně menší kapacitu než je třeba k uložení celého snímku, docházelo by tak k oznámení vždy, když se budou ukazatele rovnat a to je mnohokrát za snímek. Proto je vždy společně s koncem snímku vystaven signál



Obrázek 5.7: Čtení obrazových dat z rozhraní Ethernetového obvodu.

img_end, který potvrdí platnost tohoto ukazatele. Jakmile pak čtecí strana narazí na konec snímku, vystaví signál *end_of_image_ack*, kterým tuto skutečnost potvrdí. Z důvodu, že obě strany, jak čtecí tak zapisovací, mohou pracovat na odlišných kmitočtech, signál o potvrzení *end_of_image_ack* je držen v log. 1 do doby, dokud signál konce snímku *img_end* nepřejde zpět do log.0, tedy dokud není zapisovací strana obeznámena s dosažením konce snímku. Je tak implementován jednoduchý handshake protokol, který se vypořádá s rozdílnými hodinami jednotlivých částí.

Uživatelské rozhraní

Rozhraní obvodu se skládá ze dvou částí, uživatelského a obecného rozhraní. Obecné rozhraní tvoří signály, který se mapují přímo na výstupy čipu FPGA. Pro jednoduchost jsou již tyto vývody v rámci stávající implementace mapovány na IOB komponenty. Dále má obvod vstup hodinového signálu a resetu. Vstupní hodiny musí být taktovány přesně na hodnotu 200MHz, protože z nich se fázovým závěsem generují hodiny pro gigabitový ethernet o frekvenci 125MHz.

Uživatelské rozhraní je navrženo pro streamový přenos obrazu. Tomu odpovídá i složení výstupních signálů. Data jsou přenášena po 32-bitové sběrnici, kde první pixel v pořadí je uložen na nejvyšším bajtu, tedy na pinech 31-24. Použitá kamera je monochromatická, pixel je tedy reprezentován 8-bitovou hodnotou šedi.

Hodinový signál pro čtení dat přivádí na vstup rozhraní uživatelský obvod. Frekvence může být díky vhodnému návrhu vyrovnávacího bufferu libovolná, musí být ale brán ohled na výslednou rychlost čtení dat, jinak by mohlo docházet k případům, kdy obvod nebude dostatečně rychle odebírat data a bude docházet k přetečení bufferu.

Vystavením signálu *rd_buffer_ready* do log. 1 dává obvod vyrovnávací paměti najevo, že na sběrnici se nachází platná data. Uživatelský obvod pak signálem *rd_buffer_enable* potvrzuje jejich přečtení. K tomu dojde při vystavení obou signálů s náběžnou hranou hodin.

Rozhraní je dále vybaveno výstupem pro signalizaci začátku a konce snímku. Po dobu vystavení signálu *rd_buffer_image_valid* v log.1 patří všechny čtené pixely do téhož snímku. Přechod tohoto signálu do log.0 značí konec aktuálního snímku. Naopak přechod zpět do log.1 označuje začátek nového snímku. Po resetu obvodu je zajištěno, že na výstup bude přenášen až první kompletní snímek. Průběh řídících signálů při přenosu obrazových dat je znázorněn na obrázku 5.7.

Vlastnosti implementace

Zařízení je úspěšně implementováno a otestováno na vývojovém kitu Xilinx SP605[4]. Na FPGA dostupném na kitu zabírá obvod celkem 141 bloků *Slice* (cca 2% všech *Slice*). Realizace vyrovnávací paměti obrazu si vyžádá jeden blok paměti BlockRAM(<1% dostupných bloků). Kmitočet čtecího rozhraní ethernetového zařízení není omezen fixní hodnotou, připojený obvod může poskytnout libovolný kmitočet hodinového signálu.

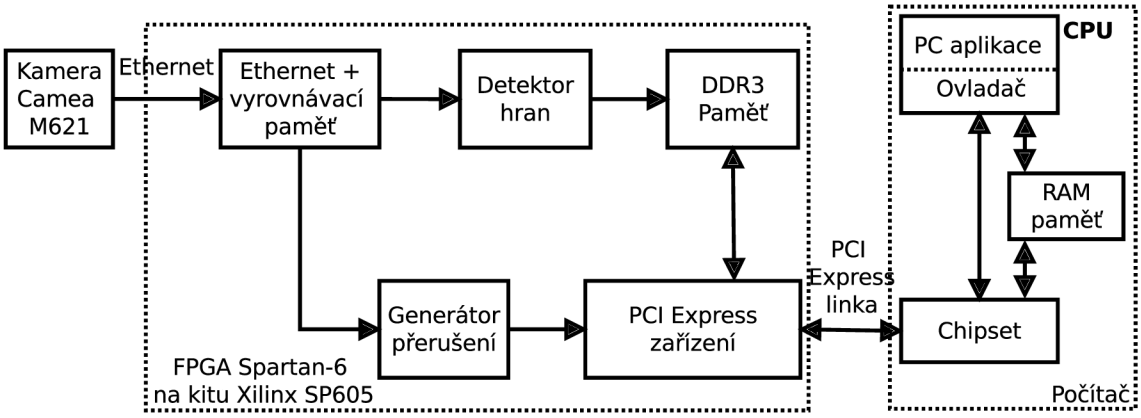
Implementovaný obvod realizuje síť Ethernet pouze o rychlosti 1Gb/s, protože nižší rychlost nepřipadá pro připojení kamery v úvahu. Rychlost, s jakou se plní fronta obrazových dat je přímo závislá na samotné kameře Unicam M621. Uživatelský obvod musí být navržen tak, aby stíhal streamově zpracovávat data z kamery. Současná kamera má rozlišení 752x478 při 60FPS, což odpovídá přenosu cca 22MB/s. Vzhledem k tomu, že je použit pouze vyrovnávací buffer s omezenou kapacitou tří řádků obrazu, je dobré zpracovat řádek dříve, než dorazí další. V průměru tato doba činí 34us na jeden řádek, což je pro taktovací frekvenci 200MHz celých 6800 taktů hodinového signálu.

5.3 Příklad použití

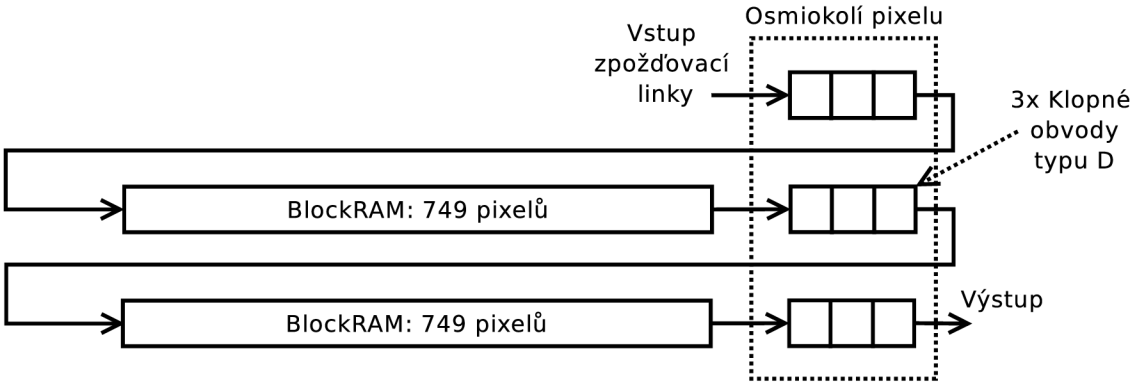
Kapitola se zaměří na ukázku implementace algoritmu pro zpracování obrazu s využitím rozhraní, která byla prezentována a implementována v rámci této práce. Jako ukázkový příklad streamového zpracování byl implementován výpočet Sobelova operátoru pro detekci hran ve videosekvenci.

Detekce hran pomocí Sobelova operátoru

Sobelův operátor je jedním z často používaných operátorů pro výpočet 2D gradientu obrazu. Detekce je provedena několika konvolucemi snímku s maskami Sobelova operátoru. Každá z konvolucí podle použitého jádra detekuje hrany obrazu v jednom směru. Gradient je pro každý pixel obrazu vypočítán z jeho osmiokolí. Konvoluční masky tak mají velikost 3x3 pixelů a jejich koeficienty jsou uvedeny na obrázku 2.6.



Obrázek 5.8: Zpracování obrazu: Architektura obvodu na detekci hran.



Obrázek 5.9: Detekce hran: Zpožďovací linka pro realizaci výpočtů nad osmiokolím pixelu.

Schéma zapojení

V této úloze jsou využívána obě komunikační rozhraní vytvořená v rámci této práce. Zdrojem obrazu pro detekci hran je kamera Unicam M621, kde obraz z kamery zpřístupňuje Ethernetový obvod spolu s vyrovnávacím bufferem snímků (viz Kapitola 5.2). K jeho rozhraní je připojen detektor hran spolu s obvodem pro generování přerušení. Výsledek detekce je pak ukládán do paměti DDR3, odkud je cyklicky čten PCI Express zařízením(viz Kapitola 5.1) a přenášen do počítače. Přenos obrazu inicializuje aplikace běžící na PC, která přenos výsledků zahájí po přijetí přerušení. Blokové schéma zapojení je na obrázku 5.8.

Zpožďovací linka

Obrazovým vstupem detektoru hran je kamera Unicam M621 připojená přes Ethernetový obvod. Data obrazu jsou streamovaná a vyrovnávací pamět Ethernetového obvodu tyto data pouze bufferuje. V takovém případě se pro výpočet osmiokolí hodí zpožďovací linka, kterou prochází vstupní stream obrazových dat. Ve vhodném místě této linky jsou připojeny obvody, které vyhodnocují hodnotu gradientu.

Protože se výpočet sobelova operátoru odehrává nad okolím 3x3 pixelů, je třeba mít linku dostatečně dlouhou pro uložení dvou řádků obrazu a tří pixelů třetího řádku. Protože by bylo při délce řádku 752 pixelů a osmi bitech na pixel náročné a nehospodárné vytvořit na čipu tolik klopných obvodů typu D, byla paměť řádku vytvořena z komponent BlockRAM. Pro oba řádky byla použita jedna bloková paměť, která implementuje zpožďovací linku pro 749 pixelů. Zbylá část řádku vždy o délce tří pixelů je sestavena z klopných obvodů typu D proto, aby bylo možné zaráz číst jejich výstupy a provádět tak výpočty nad osmiokolím. Zpožďovací linka je znázorněna na obrázku 5.9.

O řízení činnosti zpožďovací linky se stará konečný automat. Ten řídí synchronizaci čtení pixelů pro zpožďovací linku a zápisu výsledků do DDR paměti. Na začátku činnosti musí automat naplnit zpožďovací linku prvními pixely. Poté je spuštěna synchronní část řízení, kdy se čtení nových dat podřizuje řízení zápisu, které se pro DDR paměti provádí po blocích. Pokud jsou na vstupu dostupné pixely obrazu a paměť DDR je připravena na zápis, provede se s každým hodinovým taktem posun pixelů ve zpožďovací lince. Tato část řízení probíhá až do doby, kdy jsou všechny pixely patřící jednomu snímku kamery přečteny. Poté se zpožďovací linka přepne do režimu vyprázdnění.

Obrázek 5.10: Detekce hran: Snímek obrazovky se spuštěnou aplikací.

Výpočet Sobelova operátoru probíhá v jedné několikastupňové výpočetní pipeline, kde řízení výpočtu probíhá současně s posunem dat na zpožďovací lince. Výpočetní blok proto nemá vliv na řízení toku pixelů. Jediná vlastnost, se kterou je třeba počítat, je zpoždění pipeline, o které se prodlužuje příchod prvního výsledku na výstup. Pro jednoduchost výpočtu byla použita verze s aproximací gradientu, protože se obejde bez výpočtu odmocniny a přesto poskytuje velmi dobré výsledky detekcí.

Obslužná aplikace

Přenos zpracovaného výsledku do operační paměti je synchronizován vystavením přerušení, které vyvolává generátor po přijetí každého snímku. Obslužná aplikace na PC nejprve detekuje příchod přerušení pomocí procedur ovladače. Následně provede nastavení PCI Express zařízení, které spustí DMA přenos dat z interní paměti DDR3 do hlavní paměti počítače. Nakonec provede aplikace vykreslení snímku s detekovanými hranami na obrazovku počítače. K vykreslení jsou použity procedury knihovny OpenCV. Tento postup je opakován pro každý nový snímek obrazu.

Na obrázku 5.10 je zachycen snímek obrazovky se spuštěným detektorem hran. Další výsledky ukázkové aplikace jsou prezentovány v příloze B.

Kapitola 6

Závěr

Cílem této diplomové práce byl návrh a realizace vysokorychlostních komunikačních rozhraní na čipu FPGA, která najdou využití v úlohách přenosu a zpracování rastrových dat. Všechny body stanovené v zadání byly v rámci této práce splněny. Prvním úkolem bylo nastudování dostupné literatury na téma datových přenosů. Nejdůležitější informace o komunikačních rozhraních, které byly důležité pro splnění zadání práce byly shrnuty v kapitole 2. Zpracování druhého a třetího bodu zadání se nachází v Kapitolách 3 a 4. Jsou zde uvedeny důvody pro implementaci PCI Express zařízení a Ethernetové kamery jako zdroje obrazových dat. Pro obě dílčí části byly vypracovány požadavky, které musí cílové zařízení splňovat a na jejichž základě byly vybrány z pohledu specifikace nejlepší alternativy.

Čtvrtý bod zadání, samotná implementace a demonstrace dosažených výsledků je představena v kapitole 5. Obě rozhraní jsou zde prezentována po implementační stránce. Protože se jedná o obvody, které budou nasazovány v různých projektech, je zde pečlivě popsáno jejich rozhraní, které bodou využívat různé obvody pro zpracování dat. Každému z nich je věnována podkapitola, která shrnuje dosažené vlastnosti, výkonnost a požadavky na zdroje čipu FPGA. V závěru kapitoly je představen experiment se zpracováním videosekvence pomocí Sobelova operátoru pro detekci hran, který využívá obě implementovaná komunikační rozhraní.

Vytvořené PCI Express zařízení nabízí, i přes dostupnost pouze jedné linky, relativně vysokou propustnost až 175MB/s oběma směry a přitom zabírá na čipu FPGA relativně málo zdrojů. Využití tak nalezne i v dalších projektech, kde je vyžadována vysoká přenosová rychlost mezi čipem FPGA a pamětí počítače. Nemusí se jednat pouze o realizace různých obrazových filtrů, jako v případě demonstračního příkladu, ale například o detektor objektů, jednoduchý raytracer a mnoho dalších. Zejména doplnění detektoru objektů prezentovaného v článku *Martin Musil, P. M.: Hardware Detection of Scalable Objects Based on Adaboost Classifier* je vhodným kandidátem pro pokračování této práce.

Rozšířit by se v případě potřeby mohlo i Ethernetové rozhraní pro připojení kamery Unicam M621. Současný design zvládne bez problémů příjem obrazu s vyšším rozlišením. Možné by ale bylo například rozšíření obvodu pro příjem obrazu z několika kamer současně. Toho by se dalo docílit pouze zkopírováním vyrovnávací paměti snímků a několika drobnými úpravami stávajícího obvodu. Zapojení více kamer by se dalo využít například pro zpracování 3D scény v projektech počítačového vidění.

Literatura

[1] Integrated 10/100/1000 Ultra Gigabit Ethernet Transceiver [online].
http://www.marvell.com//transceivers/assets/Marvell-Alaska-Ultra-88E1111-GbE.pdf.

[2] LogiCORE IP Tri-Mode Ethernet MAC v4.5 User Guide [online].
www.xilinx.com/support/documentation/ip_documentation/tri_mode_eth_mac_ug138.pdf.

[3] Spartan-6 FPGA CLB [online].
http://www.xilinx.com/support/documentation/user_guides/ug384.pdf.

[4] Xilinx SP605 Evaluation kit [online].
http://www.xilinx.com/products/boards-and-kits/EK-S6-SP605-G.htm.

[5] Corbet, J.; Rubini, A.; Kroah-Hartman, G.: *Linux device drivers*. O'Reilly Software Series, O'Reilly, 2005, ISBN 9780596005900.

[6] Guo, Z.; Xu, W.; Chai, Z.: Image Edge Detection Based on FPGA. *International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, ročník 0, 2010: s. 169–171,
doi:http://doi.ieeecomputersociety.org/10.1109/DCABES.2010.39.

[7] Martin Musil, P. M.: Hardware Detection of Scalable Objects Based on Adaboost Classifier. In *Proceedings of the 18th Conference STUDENT EEICT 2012*, Brno University of Technology, 2012.

[8] Miller, F.; Vandome, A.; McBrewster, J.: *Ethernet*. VDM Publishing House Ltd., 2009, ISBN 9786130089047.

[9] PCI-SIG: PCI Express Base Specification Revision 1.1 [online]. http://www.pcisig.com, 2005-05-28.

[10] Pedroni, V.: *Circuit design with VHDL*. MIT Press, 2004, ISBN 9780262162241.

[11] Viola, P.; Jones, M.: Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, ISSN 1063-6919.

[12] WWW stránky: css.engineering.uiowa.edu.
http://css.engineering.uiowa.edu/~dip/LECTURE/contents.html.

[13] WWW stránky: Xilinx Inc. http://www.xilinx.com/.

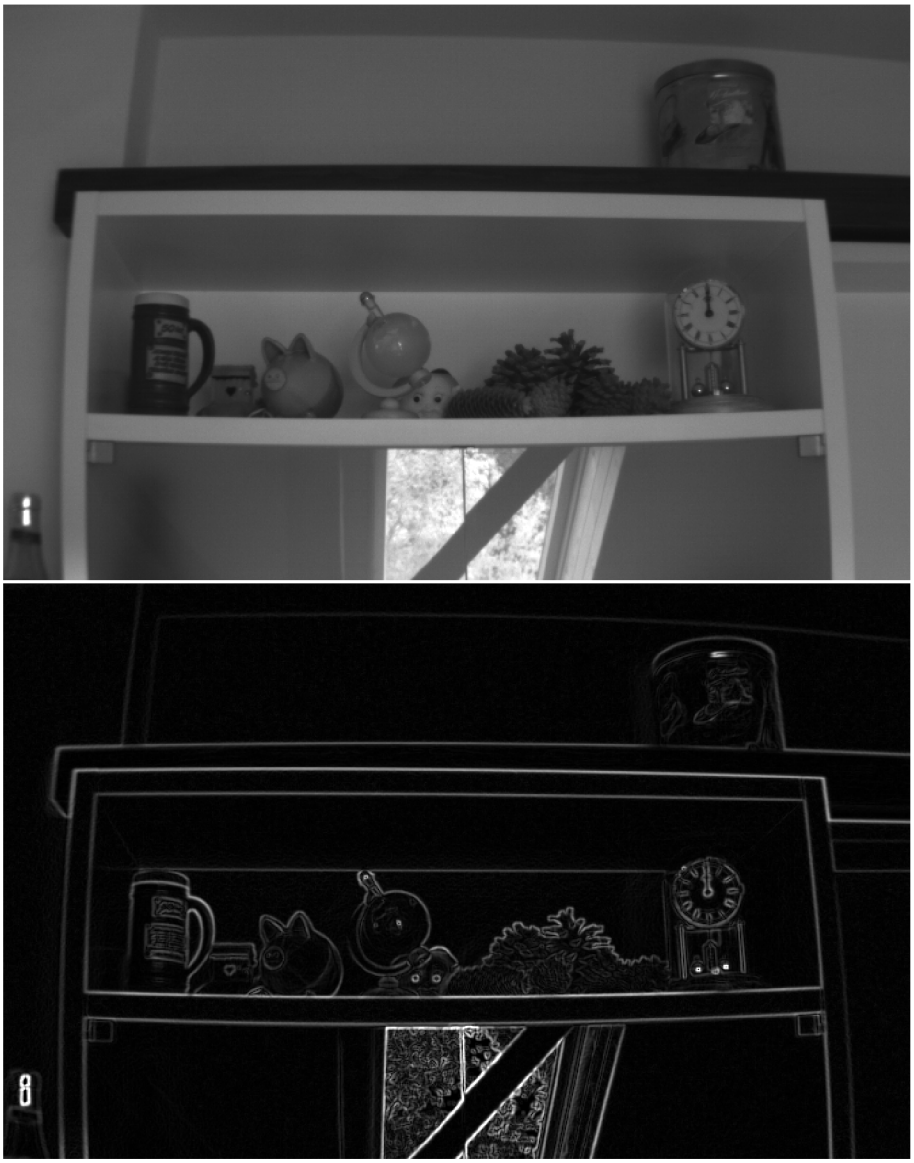
Příloha A

Obsah DVD

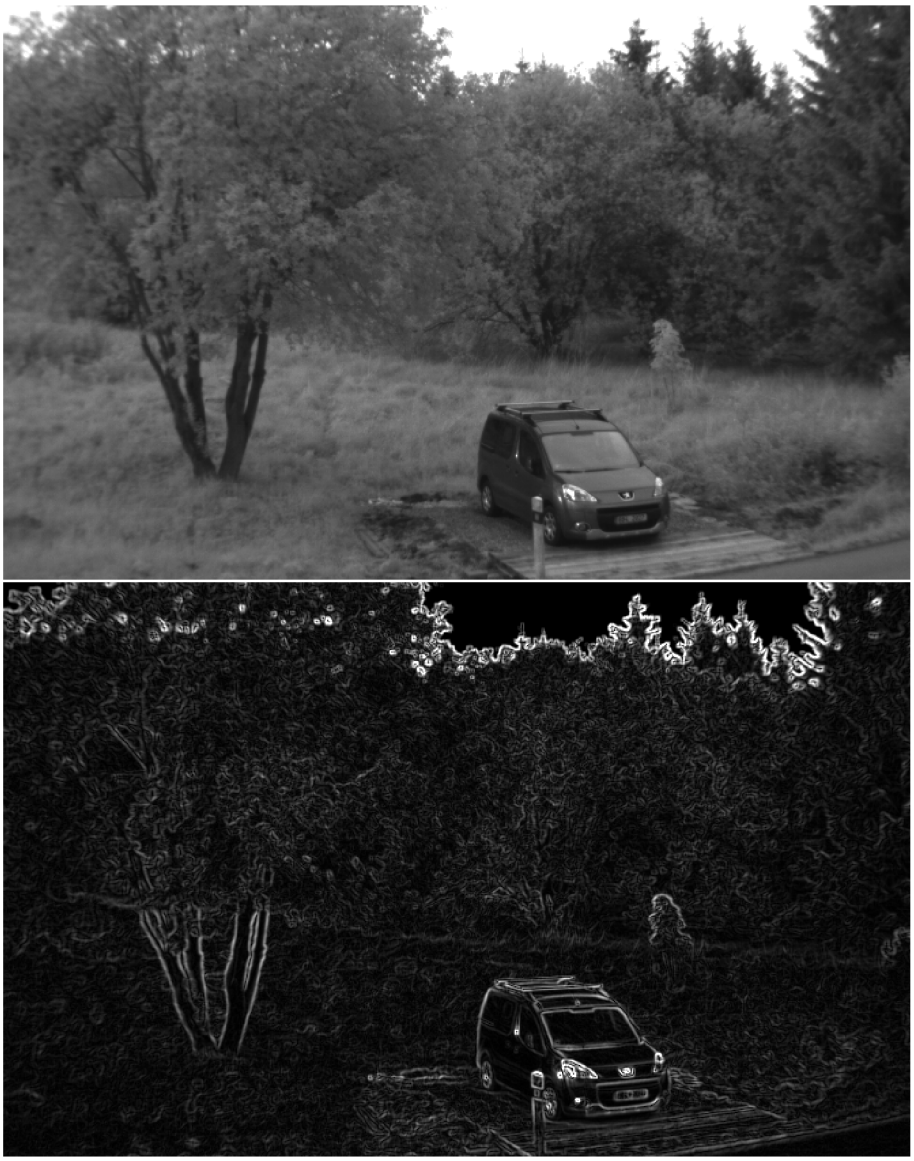
- Zdrojové kódy ve VHDL
- Ovladač PCI Express zařízení
- Ukázková aplikace
- Instalační příručka

Příloha B

Výsledky detekce hran



Obrázek B.1: Snímek interiéru před a po použití Sobelova operátoru.



Obrázek B.2: Snímek auta v krajině před a po použití Sobelova operátoru.