VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VYSOKORYCHLOSTNÍ PAKETOVÉ DMA PŘENOSY Z FPGA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN KUBÁLEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN KOŘENEK, Ph.D.**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Kubálek Jan**
Obor: Informační technologie
Téma: **Vysokorychlostní paketové DMA přenosy z FPGA**
High-Speed Packet DMA Transfers from FPGA
Kategorie: Návrh číslicových systémů

Pokyny:

1. Seznamte se s technologiemi FPGA, PCI-Express a DPDK.
2. Navrhněte firmwarový modul pro přenos paketových dat z FPGA do paměti CPU. Při návrhu se zaměřte na dosažení co nejvyšší přenosové rychlosti.
3. Navržený firmwarový modul implementujte.
4. Ověřte funkční a výkonové vlastnosti modulu v simulacích a v čipu FPGA.
5. Zhodnoťte dosažené výsledky a diskutujte návrh modulu pro přenos dat v opačném směru.

Literatura:

- Podle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

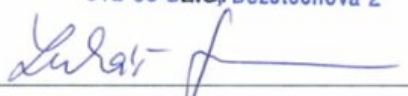
Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kořenek Jan, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

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



prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Pro zařízení, která na počítačových sítích zajišťují monitorování provozu a umožňují údržbu sítě, může být problémem nedostatečná rychlost přijímání dat pro analýzu. Aby dokázalo zařízení monitorovat síť s vysokou datovou propustností, musí být síťová karta zařízení schopna přijatá data rychle přenášet do paměti RAM. V rámci této práce byl proveden návrh, implementace a testování modulu pro FPGA čip na síťové kartě, který tyto přenosy řídí. Vytvořený modul podporuje přenášení paketů přes sběrnici PCI-Express na rychlosti 100 Gb/s a 200 Gb/s. Tyto přenosy jsou prováděny jako přenosy DMA v systému DPDK.

Abstract

Computer network devices that implement data-flow monitoring to allow network management require a high-speed receiving of a large amount of data for analysis. For a device to enable the monitoring of a network with high data traffic, its network interface card needs to be capable of transferring received data to a RAM at high speed. A new module for an FPGA chip on a network interface card, which can control these transfers, was designed, implemented and tested in the course of this thesis. The created module supports transfer of packets from the FPGA to the computer's memory via a PCI-Express bus at the speed of 100 Gb/s and 200 Gb/s. Packets are transferred by DMA in system DPDK.

Klíčová slova

FPGA, DMA, DPDK, PCI-Express

Keywords

FPGA, DMA, DPDK, PCI-Express

Citace

KUBÁLEK, Jan. *Vysokorychlostní paketové DMA přenosy z FPGA*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Kořenek, Ph.D.

Vysokorychlostní paketové DMA přenosy z FPGA

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Kořenka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Kubálek
5. května 2018

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce Ing. Janu Kořenkovi, Ph.D. za jeho pomoc a podporu při psaní této práce. Dále bych chtěl poděkovat kolegům z výzkumného týmu Liberouter, zejména pak Ing. Martinu Špinlerovi, Ing. Václavu Hummelovi a Ing. Vitoru Pušovi, Ph.D., za jejich odborné rady a pomoc při sestavování návrhu DMA modulu a za poskytnutí komponent, které jsem mohl použít jako jeho součást.

Obsah

Kapitola 1

Úvod

Při přenosu dat v počítačové síti dochází u jednotlivých komunikujících počítačů k přesouvání dat mezi síťovým rozhraním a pamětí RAM. V případě, že koncové zařízení není schopno dostatečně rychle ukládat data, která přicházejí ze sítě, je tím omezena výsledná rychlost příjmu. Pokud se jedná o server, jehož účelem je zpracovávat internetový provoz, znamená snížení propustnosti při příjmu dat zhoršení kvality služby, kterou poskytuje. Je-li požadovaná propustnost přenosů v řádu desítek až stovek gigabitů za sekundu, řešení ukládání dat do RAM se stává technicky náročnou záležitostí.

V dnešní době rozvoje Internetu se objem dat přenášených v síti neustále zvyšuje. Aby byla zajištěna stabilita a bezpečnost komunikace, nachází se v síti zařízení určená k monitorování a modifikaci síťových dat v reálném čase. Zapojení takového zařízení do Internetu by nemělo mít za následek zpomalení přenosů na síti. Z toho důvodu je v těchto zařízeních použita hardwarová akcelerace. Hardwarová akcelerace je postup implementace funkce zařízení, případně její části, ve specializovaném hardwaru — oproti běžnému přístupu, kdy je hardware obecný a funkce je implementovaná v softwaru. Specializované hardwarové zařízení dokáže funkci, pro kterou je určeno, vykonávat zpravidla rychleji a efektivněji než obecný hardware. Tento postup je v počítačových sítích široce používán a lze jej využít i k vysokorychlostnímu příjmu síťových dat do paměti počítače. Jednou z technologií používaných pro hardwarovou akceleraci je čip *FPGA* (*Field-Programmable Gate Array*) popsaný v sekci 2.1.

Data přicházejí do počítače přes síťové rozhraní, které je součástí síťové karty. Tato karta se chová jako periferní zařízení a data, která přijala ze sítě, musí přenést přímo do paměti, kde k nim bude moci přistupovat CPU. Přenosy mezi periferním zařízením a pamětí lze obecně implementovat různými způsoby. V této práci je to prováděno přenosy *DMA* (*Direct Memory Access*) (blíže viz sekce 2.2), které jsou pro tento účel vhodné.

Samotné DMA přenosy lze ještě dále impementovat specifickým způsobem především podle druhu periferního zařízení a přenášených dat. Systém *DPDK* (*Data Plane Development Kit*) (sekce 2.3) pro DMA je jedním z často používaných přístupů, které jsou v dnešní době rozvíjeny.

Pro zajištění dostatečné propustnosti z periferie do RAM je pro přenos dat nutno použít médium podporující přenosy s stejnou nebo vyšší propustností. Mezi komponentami v počítači jsou datové přenosy zprostředkovány datovými sběrnicemi. Konkrétně je v této práci uvažováno přenášení dat po sběrnici *PCI-Express* (*Peripheral Component Interconnect Express*) (sekce 2.4).

V souhrnu je mým cílem v této práci dosažení vysokorychlostních přenosů dat mezi FPGA čipem umístěným na síťové kartě a pamětí RAM přes sběrnici PCI-Express, a to

s využitím technologií DMA přenosů v systému DPDK. Za účelem dosažení tohoto cíle jsem navrhnul, implementoval a otestoval firmwarový DMA modul, který pro síťovou kartu zajišťuje tuto funkci.

Všechny výše zmíněné technologie jsou klíčové pro vytvoření návrhu, a proto jsem je v rámci přípravy nastudoval a popsal v kapitole 2. Na základě znalostí těchto technologií a dalších podmínek a omezení stanovených pro DMA modul jsem v kapitole 3 sestavil soupis požadavků pro správný návrh. Následně jsem v další kapitole 4 navrhnul DMA modul, který se snaží splnit tyto požadavky. Výsledky implementace tohoto modulu z pohledu pracovní frekvence a zdrojů využitých na FPGA čipu jsou shrnuty v kapitole 5 a v kapitole 6 je popsán postup při testování modulu v FPGA na síťové kartě. Kapitola 6 obsahuje také výsledky měření rychlosti datových přenosů provedených s využitím vytvořeného DMA modulu. V závěru práce v kapitole 7 jsou shrnuty výsledky práce a nastíněny možnosti dalšího vývoje v tomto směru.

Kapitola 2

Využité technologie

Abych byl schopen navrhnout funkční DMA modul, bylo zapotřebí seznámit se s existujícími technologiemi, které bude modul využívat. Každá z nich určuje část prostředků a omezení, které společně upřesňují funkci navrhovaného DMA modulu. Tyto technologie byly již vyjmenovány v úvodu a v této kapitole jsou popsány v jednotlivých sekcích: FPGA (sekce 2.1), DMA (sekce 2.2), DPDK (sekce 2.3) a PCI-Express (sekce 2.4). Tyto technologie jsou v praxi používány a informace o nich jsem získal z volně dostupných odborných zdrojů, které jsou uvedeny v úvodu příslušných sekcí.

2.1 FPGA

FPGA je počítačový čip, který je v našem případě umístěný na síťové kartě a který ovládá její funkce. Na tomto FPGA bude nahraná firmwarová archutektura, jejíž součástí bude DMA modul zajišťující úřenos dat do RAM. Proto je potřeba technologii FPGA dobře porozumět.

Charakteristickým znakem čipu FPGA je jeho programovatelnost. Do FPGA je možno nahrát firmware implementovaný v jazyce pro popis hardwaru a přeložený do formy *bitstreamu*, což je serializovaná forma konfiguračních dat pro danou verzi FPGA. Po spuštění FPGA s nahranou konfigurací začne čip vykonávat implementovanou funkcionalitu. Opakovaným nahráváním různých konfigurací tak lze jediný čip používat pro provádění různých činností.

Alternativní možností je *ASIC* (*Application-Specific Integrated Circuit*) [11]. Na rozdíl od FPGA je ASIC vyroben na míru konkrétnímu obvodovému návrhu. Vzniká tak čip schopný provádět stejnou funkci na vyšší frekvenci a s nižší spotřebou energie a prostoru. Protože výroba počítačového čipu je nákladná a časově náročná — zejména pokud se jedná o čip vyrobený na zakázku pro konkrétní aplikaci — není použití ASIC čipu vhodným postupem pro rychlý vývoj hardwaru. V našem případě budeme mít k dispozici síťovou kartu obsahující zapojený čip FPGA. Pro vyzkoušení implementace tak stačí z kódu vygenerovat bitstream, nakonfigurovat jím čip na kartě, spustit hardwarový test a zaznamenat výsledky.

Bližší popis struktury čipu FPGA se nachází v první podsekci 2.1.1. Detaily struktury a konkrétní hodnoty jsou uvedeny pro FPGA čipy rodiny *Virtex UltraScale+* od firmy *Xilinx*. V podsekci 2.1.2 jsou uvedeny body související se správným postupem při návrhu firmware pro FPGA.

2.1.1 Struktura FPGA

Pro správnou práci s FPGA je třeba znát vnitřní strukturu tohoto čipu. Tak lze zajistit vytvoření návrhu, který nejlépe využívá zdroje dostupné uvnitř programovatelného pole. Při neznalosti způsobu, jakým je na FPGA realizováno zapojení struktur popsaných v *HDL* (*Hardware Description Language*), může často i špatně napsaný řádek kódu znamenat obsazení většího množství zdrojů než by bylo potřeba při správném popisu. Ještě snadněji může nesprávně popsaný proces vytvořit při překladu logiku, která je příliš komplikovaná pro požadovanou frekvenci a návrh pak nesplňuje časování.

Hlavní část FPGA je tvořena soustavou základních komponent, jejichž kopie jsou rozloženy po celé ploše čipu. Například multiplexory, registry, look-up tabulky a bloky adresovatelné paměti. Tyto základní komponenty jsou propojeny hustou sítí vodičů. Právě spojováním nebo rozpojováním konkrétních dvojic vodičů pak dochází k propojování konkrétních bloků do složitějších logických struktur, a tak dochází k programování daného pole. Konkrétní základní jednotky, které se na čipu nacházejí, určuje výrobce a liší se v jednotlivých modelech FPGA. V této práci jsem vytvářel návrh určený pro model FPGA Virtex UltraScale+ VU7P a v této podsekci budu popisovat strukturu právě tohoto modelu. Pro detailnější popis struktury FPGA UltraScale+ lze nahlédnout do dokumentace [4, 5, 8], ze které jsem čerpal údaje pro tuto sekci.

Kromě samotného programovatelného pole obsahuje čip řadu bloků s pevně danou funkcí (*hard block*). Tyto bloky provádějí složitější funkce, které jsou v FPGA často využívány. Jejich implementace v programovatelném poli by ale nedosahovala dostatečné požadované rychlosti a nebo by zabírala příliš zdrojů oproti vytvořenému hard blocku. Příkladem takové funkce je přímá obsluha sběrnice PCI-Express nebo velkokapacitní paměť. Další důležitou částí čipu jsou generátory hodinového signálu a vodiče pro rozvod hodinového signálu po celé ploše pole. Pro hodinový signál jsou dedikovány speciální vodiče, protože je důležité, aby byl co nejpřesněji synchronizován ve všech místech čipu. V neposlední řadě je součástí čipu také napojení na vstupně výstupní piny, přes které komunikuje čip se svým okolím.

Hlavní programovatelné pole se na UltraScale+ skládá především z malých logických bloků zvaných *slice* a z blokových pamětí s náhodým přístupem (*BRAM*). BRAM je hard block sloužící jako adresovatelná paměť se dvěma zápisovými a dvěma čtecími rozhraními. V rámci jednoho rozhraní probíhá zápis i čtení synchronně, což u hardwarové komponenty znamená, že ke změně výstupů a k přijímání vstupů dochází pouze při nástupné hraně hodinového signálu. Každé rozhraní může při tom být řízeno vlastním hodinovým signálem. Jedna BRAM poskytuje paměťový prostor o velikosti 36 Kb. Samotná logika naprogramovaného firmwaru se provádí ve slicech. Každý slice obsahuje 8 šestivstupých look-up tabulek (*LUT*), které slouží k implementaci libovolných logických funkcí s šesti vstupy a jedním výstupem. Konkrétní logická funkce se do LUT nahrává při programování FPGA. Dále obsahuje slice 7 vestavěných multiplexorů 2:1 (*MUX*), 16 paměťových registrů a 1 jednotku *CARRY*, která implementuje rychlý přenos carry bitu při sčítání nebo odčítání dvou osmibitových hodnot.

Spojováním těchto komponent lze sestavovat jejich komplexnější verze. Například sčítačka šestnáctibitových hodnot je tvořena šestnácti LUT (které provádějí součet bez carry) a dvěma zřetězenými CARRY bloky. Využitím dvou šestivstupých LUT a jednoho MUX 2:1 může zase vzniknout logická funkce se 7 vstupy.

Kromě prostých logických sliců (označovaných také „sliceL“) obsahuje programovatelné pole také speciální paměťové slicy (nazývané „sliceM“). Paměťové slicy obsahují namísto obyčejných LUT, speciální paměťové LUT u kterých je možno měnit obsah a lze tak každou

využít jako asynchronní adresovatelnou paměť až pro 64 jednobitových položek. V případě potřeby lze paměťové slicy použít i pro provádění logických funkcí.

sliceL	98 520
sliceM	49 320
BRAM	1 440

Tabulka 2.1: Počty zdrojů pro FPGA Virtex UltraScale+ VU7P

Tabluka 2.1 ukazuje konkrétní hodnoty počtu logických sliců, paměťových sliců a BRAM pro FPGA čip Virtex UltraScale+ VU7P. Jak je vidět, tak zdaleka největší zastoupení mají základní logické bloky sliceL. Bloků sliceM, které mohou sloužit jako distribuovaná paměť, je na čipu dvakrát méně. Počet BRAM v celém poli je relativně omezený, a proto je třeba využívat při implementaci tento druh paměti opatrně a neplýtvat s ním.

Při vytváření návrhu jsem bral ohled na dostupnost jednotlivých zdrojů. Syntézní nástroje určené k překladu kódu popisu hardwaru umožňují zkontrolovat výsledný počet obsazených zdrojů u vygenerovaného bitstreamu. Po provedení implementace si tak návrhář může ověřit skutečnou velikost jednotlivých komponent architektury.

2.1.2 Návrh pro FPGA

Aby mohl být firmware správně navržen, je potřeba už při návrhu uzpůsobovat jeho podobu podle požadavků, které musí plnit po jeho nahrání do FPGA. Kromě již zmíněného využití dostupných zdrojů je další důležitou vlastností firmware splnění časování vyžadovaného zvolenou frekvencí hodinového signálu. Tyto požadavky vytvářejí hlavní rozdíl mezi návrhem software a návrhem firmware, který je patrný i při navrhování malých systémů provádějících jednoduchou funkci.

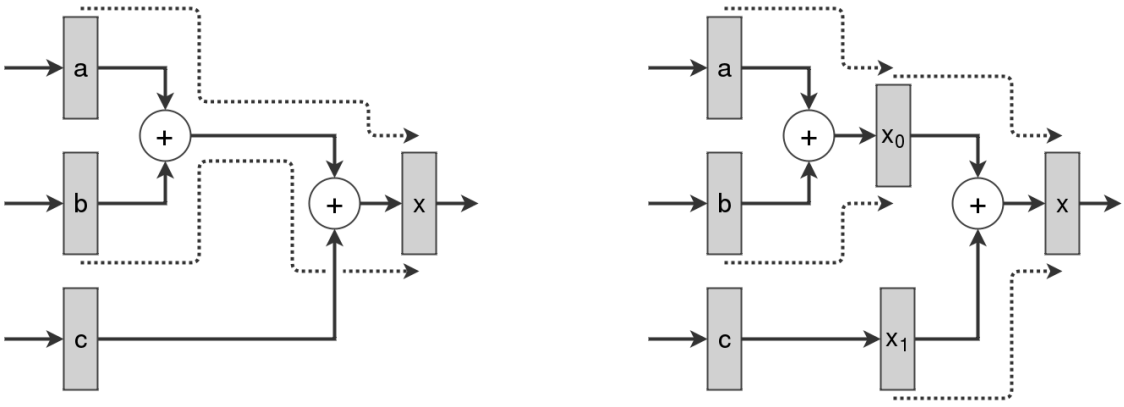
Při zvýšení pracovní frekvence modulu, který zpracovává data, dochází obecně ke zvýšení propustnosti. Zároveň se ale úměrně zkracuje perioda hodinového signálu (perioda je převrácenou hodnotou frekvence). Pokud se v návrhu vyskytuje složitý výpočet (logická funkce), který potřebuje být proveden během jednoho taktu, nesmí perioda hodin klesnout pod maximální dobu potřebnou pro tento výpočet. V opačném případě nelze zaručit korektní provedení výpočtu a obvod se bude z pohledu navrhovaného modulu chovat nedeterministicky (při bližším prozkoumání obvodu na nižší úrovni se chování už dá předpovídat, to by ale nemělo být součástí návrhu firmware).

Na začátku každého nového taktu jsou potřebné vypočtené hodnoty nahrány do registrů, kde jsou stabilně uloženy až do jejich přepsání (zpravidla na začátku příštího taktu). Aby mohl modul pracovat na vysoké frekvenci, musí být kombinační logika, která je prováděná v rámci jednoho taku hodin (v rozmezí jedné periody), co nejjednodušší.

Dalším faktorem, který ovlivňuje délku trvání operace v kombinačním logickém obvodu, je délka propojení mezi logickými prvky. Pokud bude ve výsledné architektuře jednoduchá funkce, která ale využívá sériové zapojení dvou sliců nacházejících se na opačných koncích FPGA, pak její provádění bude trvat déle, než provedení stejné funkce, která ale využívá dva sousedící slicy. K zapojení funkce pomocí vzdálených sliců může dojít, pokud větší počet operací pracuje s hodnotami ze stejného registru, nebo do stejného registru zapisují svůj výsledek (ne zároveň, ale každá ve speciálním případě). Některé operace zaberou logické zdroje v blízkosti sdíleného registru a ostatní pak musejí být připojeny prostřednictvím delších vodičů. K podobnému efektu dochází, pokud provádíme pouze jednu operaci,

ale nad daty o veliké bitové šířce. Rozložení prvků, které jsou pro jednotlivé funkce použity, je komplexní problém a při programování FPGA je řešen překladovým systémem pomocí heuristických iteračních algoritmů. Kvalita vytvořeného návrhu může ale výsledné rozložení také ovlivnit.

Aby bylo možno provádět složité operace nad daty při vysoké frekvenci, používá se při návrhu firmware metody zvané zřetězené zpracování (*pipelining*). Principem zřetězeného zpracování je rozdělení úlohy, kterou chceme nad daty provádět, na jednodušší sekvenční operace, z nichž každá je zakončena uložením hodnot do registrů, které jsou vstupem následující operace. Průchod vstupní hodnoty zřetězenou linkou o N stupních trvá N hodinových taktů. Protože je ale každý stupeň schopen přijmout a zpracovat jeden vstup každý takt a jednotlivé stupně mohou pracovat současně nad různými daty, dokáže zřetězená linka produkovat data stejně rychle, jako je přijímá. Zřetězená linka vytváří pouze latenci N taktů.



(a) Operace x = a+b+c bez zřetězení

(b) Operace x = a+b+c ve zřetězené lince

Obrázek 2.1: Demonstrace řešení operací pomocí zřetězeného zpracování

Na obrázku 2.1 je příklad použití zřetězeného zpracování pro řešení operace $x = a+b+c$, která do registru x vloží součet hodnot a, b a c. Operace součet je prováděna pomocí logických bloků $+$. Správný součet dvou vstupních hodnot se na výstupu bloku objeví až s určitým zpožděním, jehož maximum je Δ_+. Při zapisování vstupní hodnoty do registru vzniká zpoždění Δ_r, přičemž $\Delta_r < \Delta_+$. Zpoždění při přenosu hodnot po vodičích budeme zanedbávat.

Na obrázku 2.1a je zmíněná operace implementována v jedinem hodinovém taktu. Tečkované šipky označují „kritické cesty“, tedy zapojení, kde přenos z jednoho registru do druhého bude trvat nejdéle. V tomto případě se jedná o přenos z registrů a a b do registru x. Obě tyto cesty procházejí dvěma sčítacími bloky a pak zápisem do registru. Maximální doba trvání tohoto přenosu bude tedy $2 * \Delta_+ + \Delta_r$.

Abychom toto zpoždění snížili, můžeme obvod upravit přidáním jednoho stupně zřetězené linky, jako je ukázáno na obrázku 2.1b. Zde jsou doplněny registry x_0 a x_1, které po prvním taktu uchovají mezivýsledky operace. Ve druhém taktu jsou tyto hodnoty sečteny do výsledku v registru x, zatímco první stupeň zřetězené linky může řešit mezivýpočet pro nové hodnoty a, b a c. Kritické cesty jsou v tomto případě čtyři. Každá z nich obsahuje průchod sčítacím blokem a zápis do registru, což je přenos trvající maximálně $\Delta_+ + \Delta_r$. Zkrácením kritické cesty jsme umožnili snížení minimální periody hodinového signálu pro vykonávání této operace. Cenou za to je zvýšení latence modulu provádějícího tuto operaci z 1 na 2 hodinové takty a také zvýšení náročnosti na zdroje přidáním dvou registrů x_0 a x_1.

Existují případy, kdy některý stupeň zřetězené linky nedokáže vždy zaručit zpracování libovolného vstupu v jednom taktu. Pokud zpracování určitého vstupu nemůže být provedeno ihned, stupeň musí informovat předchozí stupeň, že není připraven na přepsání svých vstupních hodnot. K tomu se používá řízení pomocí „*handshaking protocol*“, který využívá signály „*source ready*“ a „*destination ready*“. Stupeň, který je připraven na nový vstup, informuje o tomto stavu předchozí stupeň nastavením signálu „destination ready“ na 1. Pokud má předchozí stupeň k dispozici validní data, zapíše je do registrů a nastaví na 1 signál „source ready“. Pokud mají oba tyto signály hodnotu 1, dochází k předávání validních dat mezi těmito dvěma stupni. Aby byla zachována maximální propustnost zřetězené linky jeden vstup za takt, musí pro každý stupeň existovat takový vstup, který může být zpracován v jednom taktu.

Dalším problémem při navrhování zřetězené linky je vytváření zpětných smyček. Uvažujme stupeň zřetězené linky, který si pro svou práci potřebuje uchovávat stav v registru. Na konci každého taktu je potřeba uložit nový stav, aby byl dostupný v příštím taktu. Pokud je následující stav závislý na stavu předchozím, vzniká zpětná smyčka. Když bude logika rozhodující o následujícím stavu pomocí stavu předchozího příliš složitá, nebude stupeň splňovat časování. Samotnou zpětnou smyčku nelze řešit zřetězeným zpracováním a jediným řešením je změna návrhu zřetězené linky.

2.2 DMA

DMA je technologie pro přenos dat mezi periferním zařízením počítače a jeho pamětí. Při DMA přenosech je periferní zařízení spojeno datovou sběrnicí přímo s pamětí RAM a provádí čtení a zapisování dat přímo bez aktivního zapojení CPU. Alternativou je přenášení veškerých dat přes procesor, který čte a zapisuje jak z periferního zařízení, tak z paměti. Při tom ale dochází k vytěžování procesoru, který aktivně řídí veškerý přenos. Pokud chceme do paměti přesouvat veliké množství dat vysokou rychlostí, jako je tomu při vysokorychlostním příjmu dat ze sítě, může být procesor značně zatížen, a proto je vhodné použít DMA přenosy.

Přímé přenosy dat mezi periferií a pamětí také vyžadují určité zapojení CPU. Hlavním důvodem je synchronizace čteného a zapisovaného paměťového prostoru. Periferní zařízení nemá samo povědomí o obsahu jednotlivých paměťových adres v RAM. Technologie DMA mu umožňuje číst a zapisovat data s libovolnou adresou, což může vést k bezpečnostnímu riziku a k nekonzistenci dat uložených v paměti. Informace o dostupném adresovém prostoru má ovladač periferního zařízení, který jakožto program běží na procesoru počítače. Ten proto musí komunikovat s periferním zařízením. Pokud chce ovladač poslat nějaká data z paměti do periferního zařízení, musí nejdříve předat informaci o adrese a velikosti těchto dat v paměti periferii. Ta následně přečte tato data pomocí DMA a poté oznámí ovladači, že data byla přečtena a mohou být uvolněna, či přepsána. V případě, že chce ovladač data přijímat, musí pro sebe alokovat prostor na tato data a opět poslat popsi tohoto prostoru periferii. Periferie zapíše na toto místo v paměti a opět oznámí ovladači, že zmíněný prostor nyní obsahuje validní data. Adresa a velikost prostoru v paměti jsou společně popsány v takzvaném „*deskriptoru*“.

2.3 DPDK

Jeden z používaných přístupů při řešení DMA přenosů je systém DPDK. Cílem systému je umožnit vysokorychlostní přenosy tím, že jednotlivé pakety jsou do paměti ukládány odděleně, a tím se softwaru ulehčuje jejich zpracování. Systém DPDK definuje způsob řízení přístupu do paměti pomocí deskriptorů, z nichž každý popisuje spojitý prostor v paměti. Součástí DPDK je také sada volně dostupných softwarových knihoven. Funkce z těchto knihoven může použít uživatelská aplikace, která chce přenášet data pomocí DPDK. Knihovny komunikují s ovladačem periferního zařízení, a tak umožňují aplikaci ovládat DMA přenosy. Aktuální dokumentaci systému DPDK lze nalézt na webových stránkách [3].

2.3.1 Komunikace software a hardware

Pro přenos paketů z FPGA do RAM musí uživatel alokovat v paměti jeden nebo více spojitých prostorů. Ukazatele na tyto prostory jsou předány ovladači, který podle nich vytvoří deskriptory. Samotné deskriptory uloží do paměti do kruhového bufferu. Adresu a velikost tohoto bufferu sdělí ovladač periferním zařízení, které čte obsah bufferu pomocí DMA přenosů. Při přenosech síťových dat v systému DPDK platí pravidlo, že deskriptory musí být hardwarem zaplňovány v pořadí a že prostor popasný jedním deskriptorem nesmí obsahovat data z více než jednoho paketu. To usnadňuje práci softwaru při čtení jednotlivých přijatých paketů. Hlavička paketu, která obsahuje informaci o jeho velikosti, je uložena hned na začátku deskriptoru. Podle velikosti paketu program zjistí, v kolika následujících deskriptorech je paket obsažen.

Používání jednotlivých deskriptorů se synchronizuje pomocí ukazatelů (softwarového a hardwarového), které jsou přístupné jak ovladači, tak FPGA. Ukazatele označují pozice v deskriptorovém bufferu (ukazatel s hodnotou N ukazuje na Ntý deskriptor). Na začátku jsou oba ukazatele inicializovány na nulu a buffer je prázdný, neboť do něj ovladač ještě nevložil žádný deskriptor. Když ovladač vytvoří nové deskriptory a uloží je do bufferu, posune SW ukazatel na adresu posledního použitelného deskriptoru plus jedna. Jakmile periferní zařízení zapíše data do prostoru popsaného deskriptorem, nastaví hodnotu HW ukazatele na první nevyužitý deskriptor v bufferu. Aby se minimalizovala komunikace mezi FPGA a CPU, nepřenáší se informace o hodnotě ukazatelů při každé inkrementaci, ale hromadně, v určitých časových intervalech. K aktualizaci HW ukazatele může dojít jen po odeslání celého paketu, takže deskriptor ležící na adrese o jedna menší než aktualizovaný HW ukazatel bude vždy obsahovat konec paketu.

(a) Plný buffer

(b) Prázdný buffer

Obrázek 2.2: Ukázka práce s deskriptorovým bufferem pomocí ukazatelů při přenosu z periferie do paměti

Obrázek 2.2 ukazuje ve dvou fázích práci s ukazateli na kruhovém deskriptorovém bufferu o velikosti osm položek. V první fázi 2.2a se buffer nachází poté, co do něj ovladač nahrál všechny deskriptory, které mohl. Buffer je tedy plný. Plný buffer je indikován tím, že SW ukazatel je o jedna nižší než HW ukazatel. Kdyby ovladač nahrál deskriptor i na pozici 2,

posunul by se SW ukazatel na pozici shodnou s HW ukazatelem, což by ale indikovalo, že je buffer prázdný (jako je tomu při inicializaci, kdy jsou oba ukazatele nulové). V následující fázi 2.2b periferní zařízení využije v pořadí deskriptory 3 až 7 a pak 0 až 1. Tím se HW ukazatel posune na hodnotu SW ukazatele. Jako další krok by ovladač přečetl data zapsaná do paměti a posunul SW ukazatel na hodnotu 1, čímž by naplnil buffer a všechny deskriptry kromě pozice 1 by mohly být opět použity.

2.4 PCI-Express

PCI-Express (dále jen PCIe) je sériová datová sběrnice použitá v této práci pro propojení mezi FPGA, RAM a CPU. Na této sběrnici se data mohou přenášet na paralelních linkách s obousměrným provozem (*full duplex*). Na jednotlivých linkách jsou data přenášena sériově. Samotná sběrnice vytváří dedikované *point-to-point* spojení pouze mezi dvěma zařízeními. Propojení více komponent mezi sebou (jako v tomto případě) je realizováno pomocí propojovacích uzlů, které obsahují více než dva PCIe porty. Komunikace přes sběrnici PCIe není pro připojené zařízení triviální záležitost. Technologie obsahuje řadu charakteristik a omezení, která mohou ovlivnit datovou propustnost. Abych mohl vytvořit DMA modul, který bude správně řídit rychlé přenosy přes PCIe, nastudoval jsem vlastnosti této technologie a popsal je v této sekci. Pro detailnější popis komunikace přes PCIe lze nahlédnout do [13]. Síťová karta, na které se nachází FPGA, používaná v této práci je připojena přes dvě rozhraní PCIe Gen 3 x16, což znamená generace 3 s 16 paralelními linkami.

2.4.1 Přenosy dat

Pro generaci 3 je uváděna propustnost jedné linky 8 Gigabitů za sekundu, což pro 16 linek znamená 128 Gb/s. To je ale pouze přenos čistých binárních dat. Skutečná propustnost je omezena celou řadou faktorů, které souvisejí s režií jednotlivých přenosů. Výslednou propustnost lze přesně analyticky určit pouze velmi těžko. Data jsou po sběrnici posílána v takzvaných „paketech“ (nejedná se o dříve zmiňované síťové pakety). Standatd sběrnice PCIe Gen3 definuje rozdělení funkcí zajišťujících přenos na tři vrstvy: fyzickou, linkovou a transportní. Každá z těchto vrstev obsahuje hlavičky a režijní pakety, které jsou také přenášeny po sběrnici a které snižují efektivitu přenosu po lince.

Fyzická vrstva zajišťuje bezpečný a spolehlivý přenos jednotlivých paketů po vodičích v lince. Aby se minimalizoval vliv vnějšího elektromagnetického rušení, které může změnit hodnoty bitů přenášených po lince, jsou data před přenosem kódována do podoby, která snižuje pravděpodobnost chyby v přenosu. Toto kódování ale mírně zvyšuje počet skutečně přenesených bitů. Na sběrnici PCIe Gen 3 se konkrétně jedná o kódování 128/130, které každý úsek dat o délce 128 bitů kóduje na data o délce 130 bitů.

Vrstva linková zajišťuje spolehlivý přenos dat mezi dvěma zařízeními přímo spojením point-to-point. Aby bylo spolehlivé spojení zajištěno, používá protokol na této vsrtvě posílání paketů *ACK* a *NACK* (*acknowledged* a *not acknowledged*), kterými přijímací strana potvrdí, respektive vyvrátí úspěšný příjem dat. V případě neúspěšného přenosu je stejný paket odeslán znovu. Dalším prvkem komunikace, který je na linkové vrstvě řešen, je „*Flow Control*“ neboli „řízení toku“. V případech, kdy odesílací strana na lince posílá data příliš vysokou rychlostí a přijímací strana je nedokáže zpracovávat, může Flow Control zamezit zahazování paketů a jejich opětovnému posílání. Aby se tomu předešlo, informují se opakovaně obě strany navzájem o tom, kolik volného prostoru mají ve svém přijímacím bufferu. Pokud chce zařízení odeslat paket, čeká, až bude protější strana schopna celý tento paket

uložit do svého bufferu. Posílání ACK, NACK a informací souvisejících s Flow Control ovšem také zvyšuje skutečný provoz na sběrnici a snižuje propustnost.

Transportní vrstva se zabývá přenosem dat z jednoho koncového zařízení do druhého, přičemž paket může být obecně posílán po cestě přes několik zařízení. Hlavní neefektivita, která na této vrstvě vzniká, je hlavička obsahující informace, jako například ID odesílatele a příjemce (unikátní v rámci dané propojovací sítě PCIe) a v případě přístupu do paměti také paměťovou adresu. Dalším zdrojem snížení efektivity přenosu je případ požadavku na čtení dat z paměti. Pokud zřízení odesílá požadavek na zapsání dat, vloží přímo do tohoto požadavku i zapisovaná data. V případě čtení musí ale zařízení odeslat do RAM požadavek s adresou a velikostí čtených dat a RAM následně odpoví zasláním příslušných dat. Samotný čtecí požadavek přitom nenese žádná validní data, a je proto považován za režijní komunikaci, která snižuje výslednou propustnos sběrnice.

Aby mohlo zařízení používat připojení přes PCIe, musí implementovat všechny tři tyto vrstvy. Součástí připojení je také počáteční konfigurace, kdy zařízení musí sbernici sdělit, jakým způsobem ji chce používat. Díky tomu je umožněno například připojení zařízení schopného komunikovat pouze přes 8 linek na sběrnici s 16 linkami. Existují ale další parametry, které jsou konfigurovatelné a které následně definují omezení pro přenos dat po sběrnici. Mezi ně patří *Maximum Payload Size* (MPS), která udává maximální velikost dat přenášených v jednom paketu, a *Maximum Read Request Size* (MRRS) udávající maximální velikost dat, která lze číst z RAM jedním čtecím požadavkem. Při čtení a zápisu do paměti RAM je také nutno dodržet omezení, že žádný čtecí ani zápisový požadavek se nesmí týkat dat, která zasahují do dvou a více stránek paměti. V takovém případě musí zařízení požadavek rozdělit tak, aby jednotlivé části pracovaly pouze na jedné stránce.

2.4.2 Používání sběrnice z FPGA

Na čipu FPGA Virtex UltraScale+ VU7P se nachází hard bloky „*GTH Transceiver*“ (přijímač/vysílač) [6] umožňující komunikaci přes jednu linku PCIe a hard bloky „*PCI Express IP*“ (označuje se jako PCIe Endpoint) [7], které slučují komunikaci z jednoho či více transceiverů do jednoho PCIe spojení mezi FPGA, RAM a CPU. Hard bloky zajišťují počáteční fázi konfigurace při připojení a jejich zapojení definuje způsob, jakým FPGA komunikuje s PCIe. Jak už bylo řečeno, je FPGA na cílové krtě připojeno na dvě rozhraní PCIe Gen 3 x16. Jeden endpoint může pro generaci 3 slouči až 16 linek, v rámci kterých dokáže rovnoměrně posílat data. Podmínkou je, že těchto 16 linek musí pocházet ze stejného rozhraní.

Při DMA přenosech rozděluje endpoint data na dva *streamy*, které jsou přístupné jako dvě oddělená rozhraní, podle toho, zda se jedná o komunikaci iniciovanou CPU (nazývaného v tomto případě jako „*completer*“), nebo iniciovanou FPGA (označováno „*requester*“). První ze jmenovaných streamů se týká především konfigurace FPGA z ovladače a komunikace v rámci DPDK. Druhý stream zahrnuje čtení a zápis FPGA do RAM. Tato rozhraní nejsou ve formátu PCIe, ale jde o rozhraní AXI (*Advanced eXtensible Interface*), jehož standard je z části definován firmou Xilinx [1, 2]. Při práci jsem měl zároveň k dispozici moduly pro FPGA, které rozhraní z completer streamu převádí na rozhraní MI (*Memory Interface*), jež je společné pro oba endpointy, a rozhraní DMA, jedno pro každý endpoint. Tato dvě rozhraní jsou blíže popsána v kapitole 3 v podsekcích 3.1.3 a 3.1.2

Kapitola 3

Požadavky pro návrh

DMA modul v FPGA je jako firmwarová komponenta definován svými vstupy, svými výstupy a svojí funkcí. Aby mohl být správně navržen, musí splňovat požadavky plynoucí z předchozí kapitoly o technologiích a některé další, které vyplývají z práce síťové karty. Tato kapitola se zabývá vysvětlením všech důležitých podmínek, které bude muset návrh modulu splňovat a které ještě nebyly zmíněny. Zároveň se tato kapitola od předchozí liší tím, že nepopisuje obecně známé technologie. Zde se již pohybujeme v prostředí, které je specifické pro cílovou síťovou kartu a architekturu nacházející se na FPGA, jejíž součástí je navrhovaný modul.

V sekci 3.1 je popsáno rozhraní sběrnic DMA a MI, přes které modul komunikuje s PCIe, a dále rozhraní, kterým do modulu přicházejí data určená k uložení do RAM. Upřesnění již zmíněných funkcí DMA modulu a dodatečné vedlejší funkce, které dosud nebyly uvedeny, jsou podrobně rozepsány v sekci 3.2. V poslední sekci 3.3 je pak soupis a shrnutí nejdůležitějších požadavků obsažených v celé kapitole.

3.1 Rozhraní DMA modulu

Na čipu FPGA bude navrhovaný DMA modul součástí většího celku zajišťujícího dohromady základní funkce síťové karty. Aby mohl být do této architektury zapojen nový modul vykonávající přenosy DMA do paměti počítače, musí tento modul přesně implementovat všechna vstupní i výstupní rozhraní, která jsou pro něj v architektuře připravena. Jednotlivá rozhraní jsou na hardwarové úrovni popsána jako sady vstupních a výstupních signálů. Součástí každého rozhraní je také protokol, kterým modul na tomto rozhraní komunikuje. DMA modul bude připojen k okolí přes tři oddělená rozhraní:

1. vstupní datové rozhraní, kterým do modulu přicházejí síťová data
2. výstupní datové DMA rozhraní pro komunikaci s RAM
3. konfigurační MI rozhraní pro komunikaci s CPU

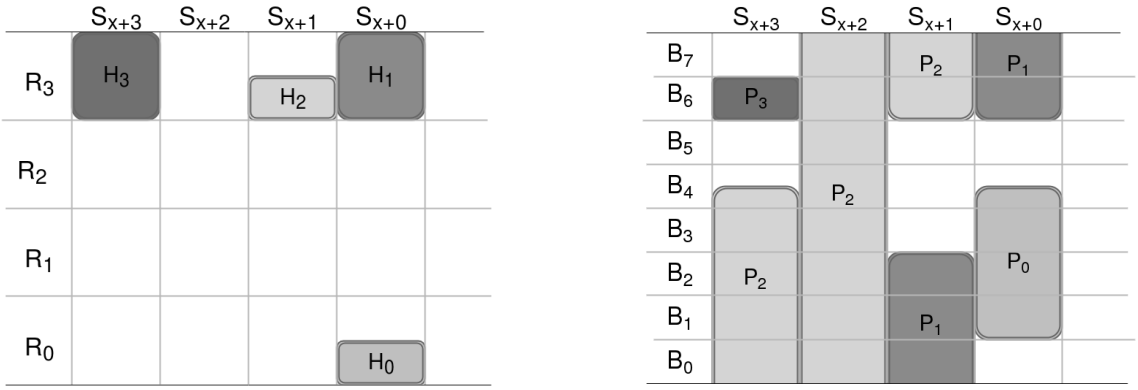
Vstupní datové rozhraní je popsáno v podsekci 3.1.1. Výstupní datové rozhraní DMA a konfigurační rozhraní MI již byly zmíněny v části zabývající se sběrnicí PCIe. Jedná se o převod dvou oddělených AXI streamů pomocí již existujících modulů AXI-to-DMA a AXI-to-MI, které samy dokáží zajistit maximální propustnos při komunikaci přes PCIe. Popis těchto rozhraní se nachází v podsekcích 3.1.2 a 3.1.3.

3.1.1 Vstupní datové rozhraní

Data, která do DMA modulu vstupují, přicházejí do karty přes síťové rozhraní. Nejedná se ale o čistá síťová data. V architektuře implementované v FPGA síťové karty se nacházejí komponenty, které s daty pracují a upravují je ještě předtím, než jsou poslána do DMA modulu. Výsledkem je, že na vstupu do modulu se nachází „datový buffer“ obsahující data jednotlivých paketů ze sítě a „hlavičkový buffer“, který obsahuje hlavičky s dodatečnými informacemi k paketům. DMA modul musí hlavičku každého paketu vložit na začátek před jeho data. Dalším vstupem je FIFO fronta obsahující informace o poloze jednotlivých paketů a jejich hlaviček ve zmíněných bufferech.

Jak datový, tak hlavičkový buffer je rozdělen na „slova“. Jedno slovo v datovém bufferu představuje množství dat, která dorazí ze sítě v jednom taktu hodinového signálu. Data uvnitř slova jsou adresována po blocích o velikosti 8 bajtů a ne po jednotlivých bajtech, což vede na zjednodušení logiky, která s daty pracuje. V datovém bufferu jsou pakety uloženy tak, že začátek paketu je zarovnán na blok, ale jeho konec zarovnaný být nemusí. V DMA modulu se pracuje s daty po celých blocích, takže délka paketu se zaokrouhluje nahoru na 8 bajtů. Paket může zasahovat do několika slov a mezi jednotlivými pakety mohou být mezery s nevalidními daty.

Hlavičkový buffer je rozdělen na stejný počet slov jako datový buffer. Slova v něm jsou také rozdělena na bloky o velikosti 8 bajtů, ale tyto bloky jsou sloučeny do pevného počtu regionů. Počet regionů je dán maximálním počtem paketů, které mohou přijít v jediném taktu (určeno specifikací síťového rozhraní) a platí, že na jeden region v hlavičkovém bufferu připadá celočíselný počet bloků v datovém bufferu. Pokud se v některém bloku v datovém bufferu připadajícím na region nachází začátek paketu, bude na začátku tohoto regionu uložena jeho hlavička. Velikost regionu je rovna maximální velikosti hlavičky a velikost hlavičky je vždy zarovnána na 8 bajtů.



(a) Hlavičkový buffer

(b) Datový buffer

Obrázek 3.1: Demonstrace dat ve vstupních bufferech

Obrázky 3.1a a 3.1b ukazují příklad obsahu hlavičkového a datového bufferu mezi slovy S_{x+0} až S_{x+3} obsahujícími čtyři pakety P_0 až P_3 a k nim patřící hlavičky H_0 až H_3. Na obrázku je každé slovo rozděleno na osmibajtové bloky B_0 až B_7, z čehož vyplývá, že v jednom taktu může naráz přijít 64 B dat. Hlavičkový buffer je v uvedeném příkladu rozdělen na čtyři regiony R_0 až R_3, takže na jeden region připadají dva datové bloky.

Paket P_0 začíná v bloku B_1, který je přiřazen k regionu R_0, a proto je v tomto regionu uložena jeho hlavička H_0. Velikost tohoto paketu je čtyři bloky, přičemž část dat v posledním bloku není validní. Ta ale mohou být do RAM přenášena také, neboť ovladač dokáže zjistit přesnou délku paketu v RAM z jeho hlavičky. Hlavičku lze z hlavičkového bufferu přečíst přesně, bez nevalidních dat na konci. Pakety P_1 a P_2 přesahují svou velikostí přes hranice slov, ale jejich příslušné hlavičky H_1 a H_2 leží pouze ve slově a v regionu, kde začínají pakety. Paket P_3 je příkladem paketu, který je menší než jeho hlavička H_3 v hlavičkovém bufferu. V některých případech může v modulu docházet i k posílání samotných hlaviček, ke kterým nejsou přiřazena žádná data (adresa dat je definována, ale délka dat je 0).

název	I/O	šířka
Read Addr	O	(počet datových bloků + počet hlavičkových bloků) * log_2(počet slov)
Read Enable	O	počet datových bloků + počet hlavičkových bloků
Read Data	I	(počet datových bloků + počet hlavičkových bloků) * velikost bloku
Read Pointer	O	log_2(počet slov)

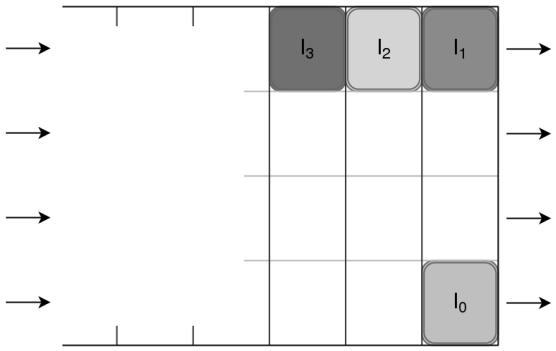
Tabulka 3.1: Rozhraní pro čtení ze vstupních bufferů

Rozhraní, kterým čte DMA modul data ze vstupních bufferů je popsáno v tabulce 3.1. Tabulka obsahuje názvy jednotlivých signálů, informaci, jestli se z pohledu DMA modulu jedná o vstupní (I) nebo výstupní (O) signál, a šířku daného signálu v bitech. V některých případech, kdy je šířka signálu vyjádřena pomocí logaritmu při zálkadu 2 z nějaké hodnoty, platí, že tato hodnota musí být nezápornou mocninou 2. Tuto podobu mají v této práci i ostatní tabulky popisující signály jednotlivých rozhraní.

Signál „Read Addr“ v tabulce 3.1 umožňuje DMA modulu nastavit pro každý blok v datovém a v hlavičkovém bufferu adresu slova, ze kterého bude číst. Pro přečtení hodnoty musí nastavit bit na pozici daného bloku v signálu „Read Enable“ na 1. Po přečtení dat tímto způsobem jsou přečtená data v následujícím taktu obsažena v části signálu „Read Data“ příslušící danému bloku. Když strana zapisující do bufferů zaplní všechna slova, musí mít možnost se dozvědět, která slova už byla DMA modulem celá úspěšně přečtena, aby mohla následně zapisovat do bufferu opět od adresy 0. K tomu slouží signál „Read Pointer“, který je DMA modulem nastaven na adresu posledního zcela přečteného slova plus jedna. Tento princip funguje stejně jako čtecí ukazatel v kruhovém bufferu.

Další částí vstupního datového rozhraní je „instrukční FIFO fronta“, která obsahuje informace o umístění dat a hlaviček v bufferech, a tudíž s nimi umožňuje pracovat. Každá položka ve frontě může popisovat jeden platný paket ve vstupních bufferech. Do instrukční fronty je v jednom taktu vloženo a je z ní čteno tolik položek, kolik je regionů v hlavičkovém bufferu — všechny popisují pakety začínající v jednom společném slově. Platné jsou ty položky, jejichž odpovídající region obsahuje hlavičku. Pokud slovo bufferu neobsahuje žádný začátek paketu — jako například slovo S_{x+2} na obrázku 3.1b — není do fronty vloženo nic. Kromě těchto položek čte DMA modul z fronty také adresu slov v datovém a v hlavičkovém bufferu, v nichž se nacházejí pakety popisované přečtenými instrukcemi.

Obsah instrukční fronty popisující data v bufferech uvedených na obrázcích 3.1 je demonstrován na obrázku 3.2. Jednotlivé sloupce obsahují platné instrukce I_0 až I_3 a také instrukce neplatné, znázorněné prázdnými poli. Sloupec, který se právě nachází na začátku



Obrázek 3.2: Demonstrace dat v instrukční FIFO frontě

fronty, obsahuje instrukce popisující pakety P_0 a P_1. Další sloupec obsahuje pouze instrukci paketu P_2 a třetí sloupec instrukci paketu P_3. O slově S_{x+2} není ve frontě žádný záznam, protože neobsahuje začátek žádného paketu.

název	I/O	šířka
Source Ready	I	1
Destination Read	O	1
Source Row	I	počet regionů * log_2(počet datových bloků v regionu)
Data Length	I	počet regionů * log_2(maximální velikost paketu) + 1
Header Length	I	počet regionů * log_2(maximální velikost hlavičky) + 1
DMA Channel	I	počet regionů * počet kanálů
Discard	I	počet regionů
Valid	I	počet regionů
Source Column	I	log_2(počet slov)

Tabulka 3.2: Rozhraní pro čtení z instrukční fronty

Tabulka 3.2 popisuje signály v rozhraní pro čtení instrukcí z instrukční fronty. „Source Ready“ a „Destination Ready“ jsou součástí handshaking protokolu. Pro každou instrukci vstupují do DMA modulu signály „Source Row“, „Data Length“, „Header Length“, „DMA Channel“, „Discard“ a „Valid“. Ty obsahují v tomto pořadí počáteční blok paketu, délku dat, délku hlavičky, DMA kanál, informaci, zda je paket určen pro zahození, a informaci, zda instrukce popisuje validní paket. Význam informace o DMA kanálu je vysvětlen později v podsekci 3.2.1. Signál „Source Column“ informuje o adrese slova, kde jsou popsané pakety obsaženy. Pokud ve slově začíná méně než maximální možný počet paketů, je signál „Valid“ nastaven u některých instrukcí na 0.

3.1.2 Sběrnice DMA

Sběrnice DMA je rozhraní umožňující DMA modulu vysílat čtecí a zápisové transakce na sběrnici PCIe a přijímat odpovědi na čtecí transakce v podobě přečtených dat. Jak už bylo řečeno v podsekci 2.4.2, vzniká DMA rozhraní transormací requester AXI streamu v modulu AXI-to-DMA. Tento modul byl již navržen a využívá se v existujících architekturách v FPGA na síťové kartě. Rozhraní DMA, které poskytuje modul, je z hlediska

firmwarového modulu snazší na ovládání než rozhraní AXI. Modul AXI-to-DMA přitom zajišťuje maximální efektivitu využití sběrnice PCIe.

Protože sběrnice DMA propojuje přímo pouze dva moduly, jedná se o sběrnici point-to-point. Aby mohl DMA modul generovat a přijímat transakce více podjednotkami, musí obsahovat rovněž propojovací uzly, které obsahují více než dvě DMA rozhraní a umožňují větvení sběrnice do spojovací sítě podobně, jako je to u sběrnice PCIe.

název	I/O	šířka
RX Source Ready	I	1
RX Destination Ready	O	1
RX Start of Packet (SOP)	I	1
RX End of Packet (EOP)	I	1
RX Data	I	512
RX Header	I	32
TX Source Ready	O	1
TX Destination Ready	I	1
TX Start of Packet (SOP)	O	1
TX End of Packet (EOP)	O	1
TX Data	O	512
TX Header	O	96

Tabulka 3.3: Rozhraní sběrnice DMA

Rozhraní popsané v tabulce 3.3 je rozděleno na signály pro příjem transakcí (RX) a odesílání transakcí (TX). Tyto dvě skupiny se liší pouze obrácením vlastností I/O a také šířkou signálu „Header“.

Signály „Source Ready“ a „Destination Ready“ řídí komunikaci pomocí handshaking protokolu. Jde o jediný mechanismus, který je zapotřebí pro řízení komunikace mezi dvěma moduly přímo spojenými sběrnicí DMA. Na rozdíl od sběrnice PCIe je totiž spojení provedeno mezi moduly nacházejícími se na stejném čipu, čímž odpadá celá řada problémů (například problém s elektromagnetickým rušením na dlouhém vodiči), kvůli kterým by bylo potřeba implementovat složitější způsob řízení.

Jednobitové signály „SOP“ a „EOP“ indikují, že data a hlavička přenášené v daném taktu obsahují začátek respektive konec paketu. Pokud jsou oba nastaveny při přenosu na 1, jedná se o paket, který je celý přenášen v jednom taktu. „Data“ je signál o šířce 512 bitů, který obsahuje data nebo část dat transakce. V případě, že se jedná o transakci, která nepřenáší data (například požadavek na čtení), je stav tohoto signálu přijímací stranou ignorován. „Header“ obsahuje hlavičku popisující transakci. Hlavička pro směr z DMA modulu (TX) obsahuje jiné informace než hlavička pro směr do DMA modulu (RX).

Odchozí hlavička má v sobě tyto informace:

- **typ** — čtení z paměti nebo zápis do paměti
- **délka** — počet čtyřbajtových slov, která chceme danou transakcí přečíst nebo zapsat
- **PCIe endpoint** — endpoint, po kterém má být transakce přenášena
- **tag** — slouží k identifikaci jednotlivých odpovědí na čtecí požadavky (odpověď na čtecí požadavek má vždy stejný tag, jako příslušný požadavek)

- **unit ID** — unikátní identifikátor podjednotky v DMA modulu, která transakci vytvořila (slouží ke směrování v propojovací síti DMA sběrnice)
- **adresa** — cílová adresa v RAM

Příchozí hlavička pak obsahuje tyto informace:

- **délka** — délka dané příchozí transakce ve čtyřbajtových slovech
- **flag „dokončeno“** — označuje poslední tranakci z odpovědi na jeden čtecí požadavek
- **tag** — slouží k identifikaci jednotlivých odpovědí na čtecí požadavky (odpověď na čtecí požadavek má vždy stejný tag, jako příslušný požadavek)
- **unit ID** — unikátní identifikátor podjednotky v DMA modulu, pro kterou je transakce určena

Pokud chce jednotka v DMA modulu provést čtení z RAM, nastaví signály v hlavičce výstupního DMA rozhraní na typ čtení, délku čtených dat, požadovaný PCIe endpoint, požadovaný tag, své unit ID a adresu v paměti, ze které chce číst. Signály SOP a EOP nastaví oba na 1 (jedná se o transakci v jednom taktu). Pomocí handshaking protokolu následně odešle transakci na sousední modul. Přečtená data mohou přijít v několika transakcích, přičemž poslední z nich má nastavený flag „dokončeno“ na 1. Všechny tyto datové transakce mají nastavené unit ID jednotky, která poslala čtecí požadavek, a tag je stejný jako u čtecího požadavku. Odpověď na čtecí požadavek přichází obvykle stovky až tisíce taktů po odeslání požadavku a jednotka si musí po celou tuto dobu pamatovat hodnotu tagu, která náleží danému požadavku, a nesmí ji použít pro jiný čtecí požadavek. Pokud jednotka pošle více čtecích požadavků, nemusejí odpovědi na ně přicházet ve stejném pořadí, v jakém byly vystaveny požadavky.

V případě zápisu nastaví jednotka hlavičku na zápis, SOP nastaví na 1 a do signálu Data vloží až prvních 512 b dat. Pokud jsou data větší než 512 b, odesílá jednotka transakci po částech ve více taktech, přičemž první část má SOP roven 1 a poslední část má na 1 nastaven EOP.

3.1.3 Sběrnice MI

Sběrnice MI vytváří rozhraní, kterým DMA modul komunikuje přes completer AXI stream s ovladačem běžícím na CPU počítače. Podobně jako je DMA sběrnice vytvářena jednotkou AXI-to-DMA, je MI rozhraní vytvářeno jednotkou AXI-to-MI. Opět přitom platí, že jednotka zajišťuje efektivní využití PCIe sběrnice.

MI rozhraní se od rozhraní DMA v mnohém liší.1 V první řadě je MI rozhraní společné pro oba PCIe endpointy a DMA modul nemůže určit, který endpoint bude použit. Dalším důležitým rozdílem je, že komunikace přes MI rozhraní už neprobíhá pomocí transakcí, ale je převáděna na přenosy prováděné v jednom taktu. Veškerá komunikace na této sběrnici se skládá z požadavků softwaru na čtení nebo zápis kontrolních a stavových registrů v DMA modulu. Do kontrolních registrů je softwarem zapisováno a modulem je z nich čteno. Do stavových registrů naopak modul zapisuje hodnoty, které software přes MI rozhraní čte. Z pohledu ovladače jsou adresy těchto registrů mapovány do adresového prostoru RAM a s jejich obsahem lze manipulovat instrukcemi pro přístup do paměti.

Každý z registrů přístupných přes MI sběrnici má velikost čtyři bajty. V této velikosti je do registrů zapisováno a z nich čteno v rámci jediného taktu. Pokud ovladač zapíše

nebo přečte z adresového prostoru registrů hodnotu větší než čtyři bajty, rozdělí modul AXI-to-MI tuto operaci na několik nezávislých zápisů nebo čtení.

název	I/O	šířka
Read Enable	I	1
Write Enable	I	1
Addr	I	32
Addr Ready	O	1
Byte Enable	I	4
Read Data	O	32
Data Ready	O	1
Write Data	I	32

Tabulka 3.4: Rozhraní sběrnice MI

Tabulka 3.4 popisuje signály obsažené v rohraní sběrnice MI. „Read Enable“ a „Write Enable“ jsou nastaveny na 1 v případě, že software chce číst respektive zapisovat v tomto taktu registry. Signál „Addr“ obsahuje adresu konkrétního registru, který je čten nebo zapisován. „Addr Ready“ nastaví modul na 1, pokud je připraven provést určenou operaci na dané adrese. Signál „Byte Enable“ je čtyřbitový signál určující, se kterými bajty ve čtyřbajtovém registru se má pracovat. Tak je umožněno ovladači manipulovat s jednotlivými částmi registrů po bajtech. V případě čtení z registru vystaví DMA modul hodnotu registru na signál „Read Data“ a současně nastaví „Data Ready“ na 1. Ze signálu „Write Data“ modul čte data v případě zápisu do registru.

Registrová sada DMA modulu obsahuje 16 registrů. Tabulka 3.5 zobrazuje jejich adresy v rámci registrové sady, jejich názvy, informaci, zda je daný registr určen pro čtení softwarem (R) nebo pro zápis (W), a stručný popis každého registru. Adresy registrů jsou uvedeny v hexadecimální soustavě. Protože některé z registrů souvisí s dosud nezmíněnými funkčními požadavky DMA modulu, jsou tyto registry blíže popsány v pozdějších sekcích této práce. Tato sada registrů byla pro návrh DMA modulu předem definována, neboť je přítomna v jeho starší verzi, se kterou musí být navrhovaný DMA modul kompatibilní.

Registr Descriptor Size ležící na adrese 0x7 je pouze dvoubajtový. Pro korektní zápis do tohoto registru nemusí být dolní dva bity signálu Byte enable nastaveny na 1. Pro všechny ostatní registry platí, že při zápisu nebo čtení musí mít Byte enable binární hodnotu 1111. V případě registrů Control a Status jsou jediné validní hodnoty 0 a 1. DMA modul totiž neumožňuje žádné jiné řízení než „zastavit“ (0) a „spustit“ (1) a nabývá pouze stavů „zastaven“ (0) a „spuštěn“ (1). Každá dvojice registrů označených jako „Low“ a „High“ společně tvoří jeden registr s osmibajtovou hodnotou.

Registry Cntr Recieved a Cntr Discarded jsou čítače, do kterých ukládá DMA modul počet příjatých respektive zahozených paketů. K zahození paketu může dojít v jednom ze tří případů. Buď je paket určen k zahození už při vstupu do DMA modulu, nebo přijde v době, kdy je DMA modul zastaven, nebo proto, že modul nemá momentálně staženy deskriptory, které by stačily k uložení paketu do paměti. Zápisem libovolné hodnoty do některého z těchto registrů dojde k vynulování čítače.

adresa	název (čtení/zápis)	popis
0x00	Control (W)	Slouží k řízení DMA modulu ovladačem.
0x04	Status (R)	Informuje o stavu DMA modulu.
0x08	SW Pointer (W)	Softwarový ukazatel pro DPDK.
0x0C	HW Pointer (R)	Hardwarový ukazatel pro DPDK.
0x10	Pointer Mask (W)	Maska určující velikost deskriptorového bufferu pro DPDK.
0x14	-	Rezervováno.
0x18	Timeout (W)	Minimální počet taktů mezi dvěma změnami hardwarového ukazatele.
0x1C	Descriptor Size (W)	Velikost prostoru popisovaného deskriptory.
0x20	Desciptor Addr Low (W)	Adresa deskriptrového bufferu pro DPDK (dolní polovina).
0x24	Descriptor Addr High (W)	Adresa deskriptrového bufferu pro DPDK (horní polovina).
0x28	Update Addr Low (W)	Adresa pro ukázání změn v hardwarovém ukazateli (dolní polovina).
0x2C	Update Addr High (W)	Adresa pro ukázání změn v hardwarovém ukazateli (horní polovina).
0x30	Cntr Recieved Low (R)	Čítač přijatých paketů (dolní polovina).
0x34	Cntr Recieved High (R)	Čítač přijatých paketů (horní polovina).
0x38	Cntr Discarded Low (R)	Čítač zahozených paketů (dolní polovina).
0x3C	Cntr Discarded High (R)	Čítač zahozených paketů (horní polovina).

Tabulka 3.5: Adresový prostor DMA modulu

3.2 Funkční požadavky

Kromě správného napojení na vnější rozhraní a jejich korektního používání musí návrh DMA modulu splňovat ještě další požadavky spojené s funkcí síťové karty. Některé z nich — jako například práce s deskriptory popisovaná v podsekci 3.2.2 nebo efektivita využití zdrojů na FPGA zmíněna v podsekci 3.2.3 — vyplývají z rozboru použitých tecgnologií uvedeného v kapitole 2. Jiné nemusejí mít s použitými technologiemi nic společného, ale přesto jsou pro DMA modul důležité a musí na ně být brán ohled. Sem patří především práce s DMA kanály popsaná v podsekci 3.2.1.

3.2.1 DMA kanály

Data, která přicházejí na vstup DMA modulu a která jsou určena k uložení do paměti, s sebou nesou informaci o přiděleném kanálu. Tato informace je k jednotlivým síťovým paketům přidána během jejich předchozího zpracování v FPGA čipu a je obsažena v instrukcích čtených z Instrukční fronty popisované již dříve v podsekci 3.1.1. Účelem rozdělování paketů do kanálů je možnost klasifikace paketů v hardwaru a pozdější paralelní zpracování jednotlivých kanálů na oddělených jádrech CPU. Aby mohly progamy pracovat s pakety na jednotlivých kanálech nezávisle, musí být nezávisle řízeny jejich DMA přenosy do paměti. Nastavení firmware při překladu udává maximální počet podporovaných kanálů (je roven mocnině 2). Při spuštění může uživatel kartu nastavit na klasifikaci paketů

do několika z těchto kanálů a každému kanálu zpravidla přiřadí jedno jádro CPU. Ovladač společně s DMA modulem pak dodávají pakety z těchto kanálů do paměti podle deskriptorů spravovaných procesem běžícím na příslušném jádře.

Pro DMA modul z toho plyne požadavek na samostatnou konfiguraci jednotlivých kanálů (jedna sada MI registrů pro každý kanál) a na udržování deskriptorů z jednotlivých kanálů odděleně. Paket na určitém kanálu může být uložen pouze do paměti popsané deskriptory na stejném kanálu a deskritpory na tomto kanálu musejí být využívány postupně. Každý kanál je spouštěn a zastavován zvlášť. Každý má svůj vlastní deskriptorový buffer, ze kterého DMA modul stahuje deskriptory, a proto má i vlastní softwarový a hardwarový ukazatel.

Pro návrh DMA modulu je žádoucí, aby podporoval co nejvyšší počet kanálů (32, 64, 128 a více). Zároveň ale hrozí, že v návrhu se budou vyskytovat komponenty a logické prvky, které budou kopírovány pro každý kanál (například MI registry), a to povede k růstu zdrojů využívaných na FPGA úměrně s růstem počtu kanálů. Některé logické obvody se mohou navíc s vyším počtem kanálů stát složitějšími (například výběr jednoho deskriptoru podle kanálu pomocí multiplexoru), což bude mít za následek vznik delších kritických cest a zhoršení časování. Na tyto problémy musí být brán ohled při vytváření návrhu DMA modulu.

3.2.2 Práce s deskriptory

V sytému DPDK slouží deskriptory k popisu spojitého prostoru v paměti, který může být použit k uložení jednoho paketu nebo jeho části. Každý deskriptor je tvořen 64 b dlouhou adresou do RAM a 16 b douhým vyjádřením popisované délky v bajtech. Pro zkrácení budou tyto dvě hodnoty označovány jako adresa deskriptoru a délka deskriptoru (nebo velikost deskriptoru). Adresa i délka deskriptoru musí být zarovnány na 8 bajtů, takže dolní tři bity jsou vždy rovny 0. Maximální velikost deskriptoru je rovna velikosti stránky v paměti. V prostředí, kde budeme DMA přenosy provádět, je velikost stránky 4096 B. Hodnota velikosti deskriptoru je stejná pro všechny deskriptory na jednom kanálu od okamžiku spuštění kanálu až do jeho zastavení. Proto je tato hodnota zapisována ovladačem do MI registru Descriptor Size před spuštěním daného kanálu.

Dalším prvkem DPDK přenosů, který je konfigurován před startem kanálu, je adresa a velikost deskriptorového bufferu. Adresa je zapisována do registru Descriptor Addr. Velikost bufferu je vyjádřena v počtu deskriptorů, které jsou v něm uloženy, a to pomocí masky v registru Pointer Mask. Hodnota tohoto registru nabývá od nejnižší pozice tolik jedniček, na kolika bitech může být vyjádřen ukazatel do deskriptorového bufferu. V případě deskriptorového bufferu o velikosti N deskriptorů, kde N je mocnina 2, je hodnota masky rovna $log_2(N)-1$. Přestože registr Pointer Mask má 32 bitů, je maximální velikost bufferu, kterou DMA modul podporuje, 2^{16}. To vede ke zjednodušení logiky pracující s ukazateli do bufferu, neboť si tak modul vystačí pouze s ukazateli vyjádřenými na 16 bitech.

Hodnota deskriptoru je v deskriptorovém bufferu vyjádřena na 64 bitech. Protože v dřívější verzi DMA modulu byla délka deskriptoru uložena na horních 16 bitech této hodnoty a nový návrh musí být s touto verzí kompatibilní, je adresa deskriptoru vyjádřena pouze na zbylých 48 bitech. Horních 16 bitů adresy každého deskriptoru musí být shodných s horními 16 bity deskriptorového bufferu, ve kterém je uložen. Tak se DMA modul může tuto část adresy dozvědět z registru Descriptor Addr. Horních 16 bitů hodnoty deskriptoru staženého z paměti je v novém DMA modulu ignorováno, protože délka deskriptoru je uložena v registru Descriptor Size.

O aktuální hodnotě SW ukazatele informuje ovladač zápisem této hodnoty do registru SW Pointer. Modul naopak předává aktuální hodnotu HW ukazatele ovladači tak, že ji v určitých intervalech zapisuje do paměti na adresu danou registrem Update Addr. Důvodem, proč k tomu nemůže být využit registr HW Pointer, je nejistota zpoždění při přenosech přes PCIe. Poté, co DMA modul vyšle přes DMA rozhraní transakci na zápis paketu do deskriptoru, trvá nějakou dobu, než jsou data skutečně zapsána. DMA modul nemůže sám určit, jak dlouho bude tento přenos trvat, a proto nedokáže ve správnou chvíli nastavit správnou hodnotu MI registru pro ukazatel. Ani kdyby DMA modul poslal hodnotu HW ukazatele po sběrnici PCIe až po transakci s daty, nebylo by zajištěno, že transakce s daty bude skutečně doručena do RAM dříve než ukazatel. Aby dokázal určit chvíli, kdy je paket bezpečně uložen do paměti, využívá vlastnosti pro zachování konzistence dat při přenosech přes PCIe. Tato vlastnost zaručuje, že pokud je poslána do RAM čtecí transakce, jsou nejprve odeslány všechny zápisové transakce, které byly na daný endpoint vystaveny před ní. Jakmile tedy chce modul informovat ovladač o posunutí HW ukazatele, odešle čtecí transakci na každý z připojených PCIe endpointů. Když se vrátí přečtená data, ignoruje je a vyšle transakci pro zápis validní hodnoty HW ukazatele na adresu uloženou v registru Update Addr. Důvod, proč se v registrové sadě přesto vyskytuje registr HW Pointer, je vysvětlen v pozdější části této podsekce.

Hodnota HW ukazatele smí být nastavena pouze po přenosu celého paketu. Aby nedocházelo k přílišnému zatěžování sběrnice čtením a zápisem upadate adresy, je toto prováděno pouze v určitých intervalech. Minimální počet taktů mezi aktualizací HW ukazetele na daném kanálu je nastaven v registru Timeout.

Po nastavení adresy a velikosti bufferu, velikosti deskriptorů, update adresy, SW ukazatele (na nulu) a intervalu pro update (Timeout) může ovladač spustit DMA kanál zapsáním hodnoty 1 do registru Control. DMA modul nastaví svůj stav pro tento kanál na spuštěný změnou hodnoty registru Status na 1 a začne stahovat deskriptory z deskriptorového bufferu. V okamžiku, kdy má dostatek stažených deskriptorů, začne DMA modul přijímat pakety přicházející na daném kanálu a odesílat je do paměti. Posouváním HW ukazatele a jeho ukládáním na aktualizační adresu informuje software o jejich úspěšném zápisu. Software opětovně tyto deskriptory uvolní posunutím SW ukazatele v MI registru.

Když chce software zastavit běh DMA kanálu, zapíše do registru Control hodnotu 0. DMA modul nejdříve přestane stahovat deskriptory pro tento kanál a již stažené deskriptory si označí jako nevalidní. Dále musí modul počkat, až všechny pakety, které již byly přijaty a byly jim přiděleny deskriptory, budou zpracovány. Pak odešle HW ukazatel na aktualizační adresu, nastaví hodnotu registru HW Pointer na aktuání HW ukazatel a hodnotu registru Status na 0. Když software indikuje, že Status je roven 0, začne porovnávat hodnotu v registru SW Pointer s hodnotou ukazatele na aktualizační adrese. Jak už bylo řečeno, může přenos posledního paketu a následný přenos ukazatele do paměti nějakou dobu trvat. Jakmile je hodnota HW ukazatele v paměti rovna hodnotě v registru (která je považována za referenční), může si být software jist, že DMA modul se již nepokusí zapsat do žádného deskriptoru na daném kanálu, a může tedy uvolnit celý buffer.

3.2.3 Efektivní využití zdrojů a rozšiřitelnost

DMA modul musí být schopen zpracovávat veliké množství dat v co nejkradším čase. Jeho propustnost je primárně dána dvěma parametry: šířkou vstupních dat čtených v jednom taktu a frekvencí čtení dat ze vstupních bufferů. Výsledná propustnost se spočítá jako počet datových bloků v jednom slově $*$ velikost bloku $*$ čtecí frekvence. Pokud by vstup ob-

sahoval 8 bloků velikých 8 bajtů a ty bychom četli s frekvencí 200 MHz, měli bychom se dostat na propustnost 102,4 Gb/s. Ve skutečnosti bychom se ale kvůli dalším omezením k této propustnosti jen přiblížili.

Prvním omezením ovlivňujícím propustnost je rozložení dat ve vstupním datovém bufferu. Protože v datovém bufferu jsou ukládána data přicházející ze síťového rozhraní, mají jednotlivé pakety mezi sebou mezery, čím se snižuje množství čtených validních dat. Aby bylo slovo v datovém bufferu zaplněno na sto procent validními daty, musí obsahovat pouze jeden paket nebo jeho část. Se snižující se délkou příchozích paketů klesá poměr validních dat přečtených z bufferu k počtu bloků, protože v bufferu se vyskytuje více mezer.

Dalším významným omezením je možnost odeslat na sběrnici PCIe zápisovou transakci pouze na část paměti popsanou jediným deskriptorem. Sběrnice PCIe dosahuje nejvyšší efektivity využití při zápisu do paměti pouze pokud všechny zápisové transakce mají velikost MPS. Efektivitu využití lze vypočítat jako podíl průměrné velikosti zápisové transakce a MPS. Pokud bude například Maximum Payload Size (MPS) pro připojenou sběrnici PCIe 1024 b a velikost deskriptorů bude 1280 b, musí být zápis do prostoru každého deskriptoru proveden minimálně dvěma transakcemi o délkách 1024 b a 256 b ($1024 + 256 = 1280$). V tomto případě bude využití sběrnice při zápisu maximálně $\frac{640}{1024}$, což vychází 0,625. Z tohoto pohledu je ideální, pokud je velikost deskriptorů celočíselným násobkem MPS. Podobným způsobem hraje svoji roli také velikost přenášeného síťového paketu. Zápis do paměti musí být navíc rozdělen na dvě transakce, pokud zapisujeme do oblasti na hranici dvou stránek v paměti.

I přesto jsou ale frekvence a vstupní datová šířka nejdůležitějšími parametry pro ovlivnění propustnosti DMA modulu. Z hlediska vstupní datové šířky bude pro návrh modulu platit požadavek, že vstupní datová šířka musí být nastavitelná parametricky a modul by měl být schopen pracovat s šířkami 512 b, 1024 b a 2048 b. Pro dosažení vysoké frekvence musí být návrh vytvořen tak, aby se minimalizovala práce s přenášenými daty. Protože data jsou pro architekturu FPGA čipu objemná, je každé jejich přenesení a uložení náročné na zdroje. Rozhodovací logika určující rozdělení dat do zápisových transakcí a zajišťující odeslání těchto transakcí na PCIe by měla pracovat výhradně s informacemi dostupnými v instrukční frontě.

Výsledný návrh by měl být flexibilní a nastavitelný z hlediska počtu DMA kanálů, počtu PCIe endpointů a vstupní datové šířky.

3.3 Shrnutí požadavků

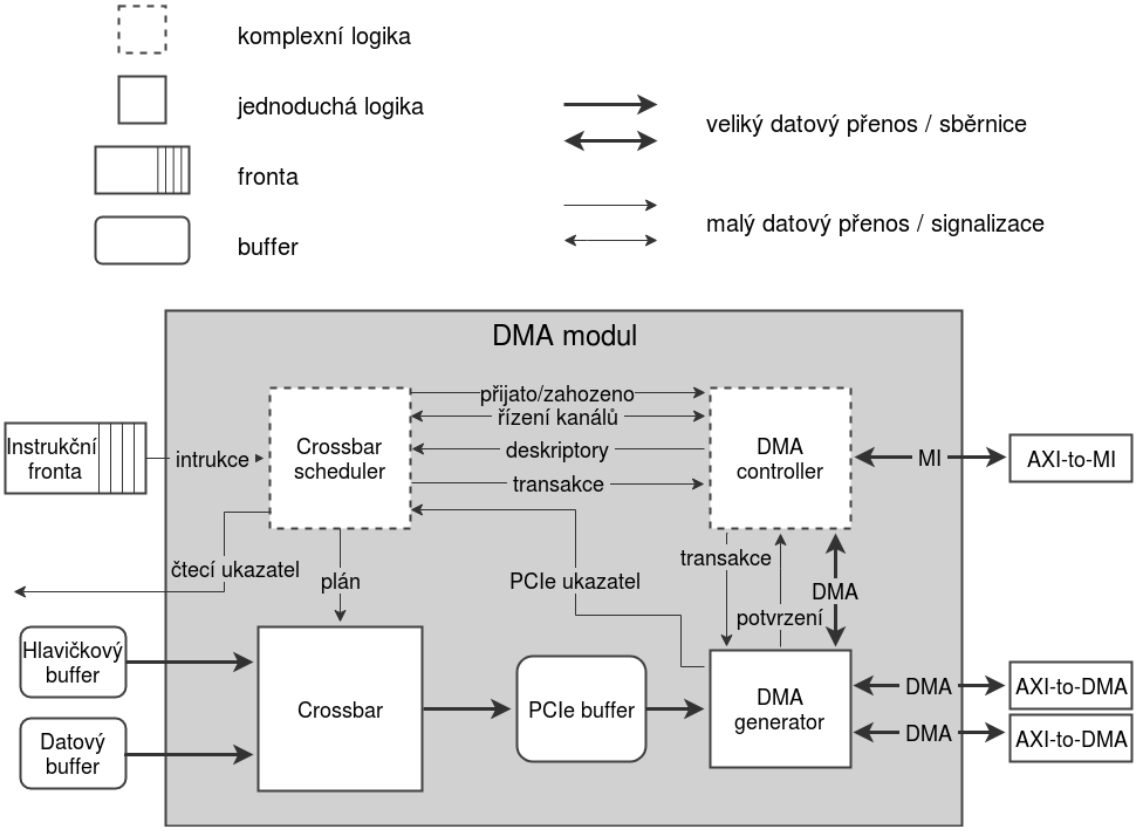
Všechny výše zmíněné požadavky by měl návrh DMA modulu splňovat, aby mohl být nasazen do FPGA čipu na síťové kartě a použit pro přenos dat ze sítě do RAM v systému DPDK. Zde jsou tyto požadavky shrnuty, aby bylo v pozdější části práce usnadněna kontrola jejich splnění:

1. kompatibilita se všemi rozhraními (sekce 3.1)
2. kompatibilita přenosu deskriptorů v DPDK (podsekce 3.2.2)
3. rozdělení datových přenosů na vysoký počet kanálů (podsekce 3.2.1)
4. schopnost generovat transakce na PCIe (s ohledem na omezení PCIe) s co nejvyšší propustností (podsekce 3.2.3)
5. snadná rozšiřitelnost a flexibilita (podsekce 3.2.3)

Kapitola 4

Návrh DMA modulu

Nyní, když byly definovány všechny důležité požadavky, které musí modul pro DMA přenosy splňovat, lze přejít k samotnému návrhu. DMA modul pro DPDK nebyl jediný, který byl v době mojí práce vytvářen. Paralelně s ním vznikal i návrh DMA modulu jiného typu. Sám jsem autorem návrhu komponent zabývajících se fungováním modulu pro systém DPDK. Některé komponenty, které byly navrženy v rámci návrhu druhého typu DMA modulu, jsou ale použity i v mém návrhu, neboť s ním jsou kompatibilní. Tyto komponenty jsou popsány v sekci 4.1. V sekcích 4.2 a 4.3 jsou popsány mnou navržené části. Nakonec v sekci 4.4 je vyhodnocena míra splnění daných požadavků.



Obrázek 4.1: Architektura DMA modulu

Obrázek 4.1 obsahuje blokový diagram celkového návrhu DMA modulu s přiloženou legendou. Šedý blok představuje samotný DMA modul. Vlevo se nacházejí komponenty Instrukční fronta, Hlavičkový buffer a Datový buffer, ze kterých modul přijímá datový vstup. Řídicí a výstupní datové rozhraní jsou vpravo a vedou do jednotek AXI-to-MI a AXI-to-DMA. Tučně vyznačené šipky bez popisků označují široké datové sběrnice. Sběrnice DMA a MI jsou zobrazeny obousměrnými šipkami s popiskem. Tenké šipky s popisky představují ostatní komunikaci mezi komponentami. Komponenty Crossbar scheduler a DMA controller, které jsou ohraničeny čárkovaně, obsahují nejsložitější logiku sloužící k řízení datové cesty. Zároveň se jedná o komponenty, jejichž obsah jsem navrhoval.

DMA modul z obrázku 4.1 funguje následujícím způsobem. Pokud se v Instrukční frontě nacházejí intrukce popisující data ve vstupních bufferech, jsou přečteny jednotkou Crossbar scheduler. Tato jednotka rozhodne, zda mohou být data odeslána do RAM a jak budou uložena v jednotlivých PCIe transakcích. Podle toho vytvoří plán přenosu, který předá jednotce Crossbar. Ta přenese data ze vstupních bufferů do PCIe bufferu, přičemž v PCIe bufferu jsou data uložena v takové podobě, jak budou posílána na sběrnici PCIe. Informaci o vytvořených transakcích pošle Crossbar scheduler do jednotky DMA controller a ta následně do DMA generator (rozhraní „transakce“). Jednotka DMA generátor podle této informace přečte data z PCIe bufferu a vytvoří zápisové transakce pro sběrnici DMA, které odešle. Na obrázku 4.1 se nachází rozhraní pro dvě jednoty AXI-to-DMA, což odpovídá situaci pro přenos dat přes dva PCIe endpointy. DMA controller obsahuje sadu registrů pro MI rozhraní a je také zodpovědný za stahování deskriptorů a aktualizaci HW ukazatele v paměti. Z toho důvodu je rovněž připojen na sběrnici DMA.

4.1 Popis předpřipravených komponent

V této sekci budou popsány komponenty DMA modulu, které jsou součástí jiné paralelně vznikající verze DMA modulu a které jsem měl k dispozici. Protože jsou tyto komponenty použitelné pro oba typy DMA modulu, mohl jsem je použít ve svém návrhu, aniž bych je musel znovu navrhovat. Do této skupiny patří komponenty Crossbar, PCIe buffer a DMA generator. Jak je patrné na obrázku 4.1, jsou tyto komponenty součástí datové cesty pro přenos velkého množství dat určených k odeslání na PCIe. Z toho důvodu byly navrženy tak, aby neprováděly nad daty žádné logické operace a pouze je přesouvaly na základě instrukcí z řídicích komponent Crossbar scheduler a DMA controller.

4.1.1 Crossbar

Účelem jednotky Crossbar je přenos dat z Hlavičkového bufferu a Datového bufferu do PCIe bufferu. Čtení dat ze vstupních bufferů je přitom prováděno přes čtecí rozhraní uvedené v podsekci 3.1.1. Tato jednotka pracuje s daty po blocích o velikosti 8 bajtů. Jejím úkolem není rozhodnout, odkud a kam budou data přenášena. Crossbar dostane od řídicí jednotky Crossbar scheduler plán přenosu v podobě zdrojových adres ve vstupních bufferech a cílových adres v PCIe bufferu. Crossbar pak provede čtení ze vstupních bufferů, přeskládá přečtená data pomocí multiplexorů a zapíše je do PCIe bufferu.

4.1.2 PCIe buffer

Podobně jako Datový buffer je i PCIe buffer rozdělen na slova, která jsou rozdělena na bloky. V PCIe bufferu jsou navíc slova sdružena do skupin, kde každá skupina představuje prostor

o velikosti MPS. Jedna skupina obsahuje data určená pro jednu PCIe zápisovou transakci. To umožňuje číst jednotlivé transakce z bufferu a přímo je odesílat na rozhraní sběrnice DMA.

4.1.3 DMA generator

Jednotka DMA generator čte data pro jednotlivé transakce z PCIe bufferu a odesílá transakce na sběrnici DMA. Protože prostor pro transakci nemusí být plně obsazen daty a buffer neobsahuje informaci o jejich skutečné délce, je tato informace do DMA generatoru poslána z Crossbar scheduleru pomocí rozhraní „transakce“. V DMA controlleru je navíc přidána informace o PCIe endpointu, na který má být transakce odeslána. DMA controller potřebuje následně potvrzení, že transakce byla kompletně odeslána na výstup (odesílání může trvat několik taktů). Teprve po přijetí potvrzení může DMA controller přejít k odeslání aktualizovaného HW ukazatele. DMA generator přitom zajišťuje, že potvrzená transakce bude odeslána před začátkem aktualizace.

4.2 Crossbar scheduler

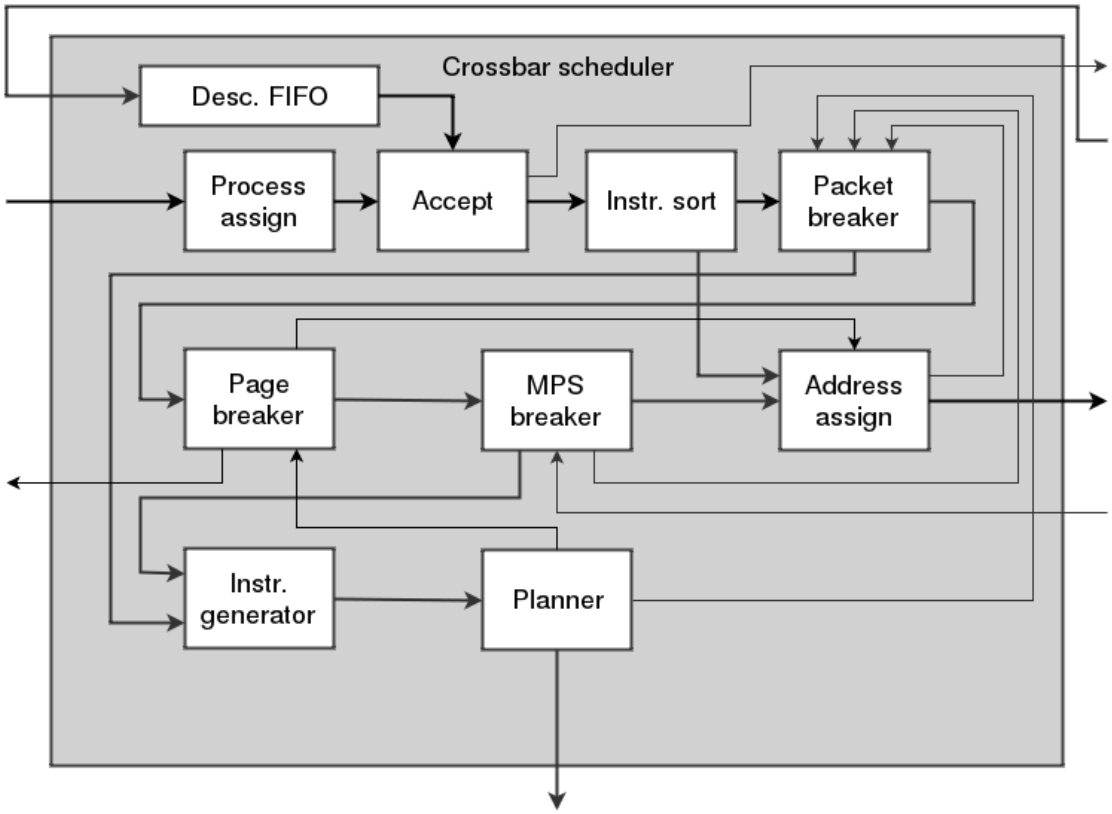
Crossbar scheduler je hlavní jednotka určující přiřazení jednotlivých síťových paketů a jejich hlaviček do transakcí pro jejich zápis do paměti. Z Instrukční fronty přečte jednotka sadu instrukcí popisujících posílané pakety a z DMA controlleru obdrží deskriptory pro uložení těchto paketů. Jejím výstupem je především plán pro přenos dat pomocí jednotky Crossbar a popis vytvořených transakcí odeslaný do DMA controlleru.

Na obrázku 4.2 se nachází blokové schéma tohoto modulu. Zobrazená vstupní a výstupní rozhraní jsou shodná s rozhraním ve schématu DMA modulu 4.1. Z důvodu zjednodušení je z nákresu odebráno zapojení pro řízení kanálů. Toto rozhraní je popsáno v podsekci 4.2.11. Celý modul lze popsat jako zřetězenou linku, která zpracovává vstupní instrukce postupně v jednotkách Process assign, Accept, Instruction sort, Packet breaker, Page breaker a MPS breaker. V jednotce MPS breaker se tato linka rozděluje na část vytvářející plán (Intruction generator a Planner) a část produkující popis transakcí (Address assign). Rozhraní mezi jednotlivými komponentami je popsáno v podsekcích zabývajících se těmito komponentami.

4.2.1 Descriptor FIFO

Modul Descriptor FIFO představuje FIFO frontu pro deskriptory. Do této jednotky vstupují deskriptory z DMA controlleru a jsou zde uloženy, dokud nejsou odebrány jednotkou Accept, která je přiřadí paketům. Jednotka neustále informuje DMA controller, zda má dostatek prostoru pro uložení jedné celé trasakce deskriptorů (DMA controller čte deskriptory z paměti RAM po blocích o velikosti 64 deskriptorů). Deskriptory jsou ukládány odděleně pro každý kanál tak, aby byly pro každý kanál využívány v pořadí. Z toho důvodu se náročnost této jednotky zvyšuje s narůstajícím počtem kanálů. Z každého deskriptoru je zde uložena adresa o šířce 45 b. Víme totiž, že z celkové adresy o 64 b je horních 16 b společných pro všechny deskriptory na jednom kanálu a dolní 3 b mají hodnotu 0 z důvodu zarovnání adresy. Výsledný počet bitů adresy ukládané ve frontě je proto $64 - 16 - 3 = 45$.

Aby Descriptor FIFO umožnilo uložení dostatečného množství deskriptorů a zároveň okamžité přečtení hodnoty deskriptoru, je rozděleno na dvě části: buffer a cache (vyrovnávací paměť). Po příchodu nové sady deskriptorů (deskriptory přicházejí ve skupinách po 8, protože tolik jich přijde v jednom taktu po DMA sbernici) jsou deskriptory uloženy



Obrázek 4.2: Architektura Crossbar scheduleru

do bufferu implementovaného pomocí BRAM bloků. Tento buffer umožňuje uložit veliké množství dat, ale nelze z něj přečíst hodnotu během jednoho taktu. Z bufferu jsou proto deskriptory automaticky přenášeny do paměti cache implementované v paměťových LUT, které mají menší objem, ale mají téměř okamžitou odezvu při čtení. Teprve odtud mohou být přečteny jednotkou Accept. Jednotka Accept určí, z jakého kanálu chce data přečíst, a Descriptor FIFO podle toho nastaví čtení konkrétní položky z konkrétní LUT. Složitost tohoto procesu je rovněž přímo závislá na počtu DMA kanálů.

Kromě jednotlivých deskriptorů informuje Descriptor FIFO jednotku Accept také o celkovém objemu dat, který je popsaný dohromady všemi deskriptory na daném kanálu uloženými v bufferu nebo v cache. Podle této informace se může Accept rozhodnout, zda je vůbec DMA modul schopen určitý paket odeslat do paměti.

4.2.2 Accept

Jednotka Accept má za úkol rozhodnout, zda je možno daný paket přijmout, a přiřazuje přijatým paketům deskriptory z Descriptor FIFO. Odmítnuté pakety jsou označeny pro zahození. Instrukce, které jsou již předem označeny k zahození, jsou pouze přeposlány dál. Accept je rozdělen na několik vzájemě nezávislých paralelně pracujících jednotek Accept process. Při datových šířkách vstupních dat 1024 a 2048 b je z instrukční fronty přečtena více než jedna instrukce v jednom taktu. Pokud jsou tyto instrukce validní a každá popisuje paket na jiném kanálu, musí být Accept schopen je zpracovávat naráz. Proto má

každý Accept process na starost práci s jednou instrukcí. Každý má také nezávislé rozhraní pro čtení deskriptorů z jednotky Descriptor FIFO. Platí, že každý proces musí být schopen číst deskriptor z kteréhokoli kanálu, ale nikdy nečtou dva ze stejného. Z toho důvodu obsahuje Descriptor FIFO jednu kopii cache pro každý proces, čímž dochází ke zvětšování zdrojů pro větší datové šířky.

V jednom taktu dokáže Accept process vygenerovat instrukci popisující část paketu určenou pro jeden deskriptor nebo celý paket, který je určen k zahození. Pokud je pro paket dostatek volných deskriptorů, ale ty se nenacházejí v cache, musí Accept process počkat a nemůže v tomto taktu generovat výstup. Velikost jednotlivých deskriptorů získá Accept process čtením příslušných MI registrů v DMA controlleru (toto rozhraní zde není zachyceno, neboť se jedná pouze o čtení registrů). Nový vstup je do jednotky Accept přijat, až když všechny procesy dokončí práci na paketech, na kterých pracují teď.

Výstupní instrukce obsahuje kromě informací z Instrukční fronty také adresu přiřazeného deskriptoru. Délka a adresa v Datovém bufferu je upravena tak, aby odpovídala dané části paketu. K instrukcím je přidán flag „konec paketu“ označující instrukci s koncem paketu.

Jednotka Accept také generuje výstup informující DMA controller o přijatých a zahozených paketech. Tento výstup se pro každý kanál skládá z flagu „přijato“ a flagu „zahozeno“. Když některý nastaví na 1, odešle tím do DMA controlleru zprávu, že došlo k přijetí nebo zahození jednoho paketu na daném kanálu. DMA controller na základě této informace inkrementuje MI registry čítající přijaté a zahozené pakety (Cntr Recieved a Cntr Discarded, tabulka 3.5, podsekce 3.1.3).

4.2.3 Process assign

Pokud je z Instrukční fronty přečteno najednou více instrukcí na stejném kanálu, nesmí být odeslány na nezávislé procesy v jednotce Accept. Nebylo by pak možno zajistit, že deskriptory na tomto kanálu budou využity v pořadí. V takovém případě zajistí jednotka Process assign, aby byly do Accept odeslány pouze instrukce na různých kanálech. První validní instrukce na opakujícím se kanálu a všechny po ní musejí být v jednotce Process assign pozdrženy, dokud Accept nezpracuje instrukce předchozí.

4.2.4 Instruction sort

Protože procesy v jednotce Accept pracují nezávisle, může se stát, že proces pracující na pozdějším paketu vygeneruje instrukci dříve než proces pracující na dřívějším paketu (paketu nacházejícím se na nižší adrese v Datovém bufferu). Pro Crossbar scheduler je důležité, aby pakety byly zpracovány postupně nebo naráz. Jednotka Instruction sort zajišťuje, aby výstupy z jednotky Accept, které jsou generovány nezávisle, byly odebírány postupně.

Představme si následující situaci: Accept process Pr_0 pracuje na instrukci pro paket, který bude uložen do dvou deskriptorů, a Accept process Pr_1 současně pracuje na instrukci pro paket, který se vejde do jednoho deskriptoru. V prvním taktu vygeneruje každý z nich instrukci pro jeden deskriptor. Protože ale instrukce od procesu Pr_0 nepopisuje konec paketu, může být přečtena pouze ona. V následujícím taktu nevygeneruje Pr_0 nic, protože čeká, až bude načten deskriptor do cache. Proto nemůže být v tomto taktu přečtena žádná instrukce. Ve třetím taktu vytvoří konečně Pr_0 instrukci pro deskriptor s koncem paketu. V tomto taktu může být přečtena i instrukce od Pr_1.

Pro N vstupních instrukcí je Instruction sort implementován jako dvě registrová pole, každé o velikosti $2 * N$. Do jednoho jsou ukládány adresy deskriptorů z instrukcí a do druhého zbytek instrukcí. Čtení z těchto polí probíhá současně a to pomocí čtecího ukazatele

do těchto polí. Na výstupu je N instrukcí nacházejících se v poli tak, že čtecí ukazatel ukazuje na první z nich. Čtení je řízeno z jednotky Packet breaker, do které jsou odesílány instrukce. Adresy deskriptorů nejsou odesílány s instrukcemi po zřetězené lince, ale jsou posílány do jednotky Address assign, kde jsou uloženy do FIFO fronty. To proto, že ve zřetězené lince je není potřeba a ukládání 45 bitů dlouhé adresy pro každou instrukci v každém stupni linky by bylo náročné na zdroje. Z adresy je ale v instrukci ponechána část, určující adresu v rámci jedné stránky v paměti (9 bitů), aby mohlo být později detekováno přetečení stránky.

Pokud nejsou z této jednotky odebírány výstupy, může se v registrových polích uložit až $2 * N$ instrukcí. Mezi validními instrukcemi přitom nejsou mezery, a proto může výstup obsahovat instrukce pro více deskriptorů pro stejný paket (na rozdíl od výstupu z jednotky Accept).

4.2.5 Packet breaker

Crossbar dokáže zpracovávat v jednom taktu pouze jedno slovo ze vstupních bufferů. Ve stejné rychlosti jsou také data do bufferů ukládána, což znamená, že není nutno tuto rychlost zvyšovat. Proto se nachází v Crossbar scheduleru jednotka Packet breaker, která vstupní instrukce rozděluje na instrukce popisující pouze části nacházející se v jediném slově. Všechny instrukce na jejím výstupu popisují pakety v jednom slově. Pokud instrukce pokračuje do následujícího slova, je její zbytek uložen v Packet breakeru do dalšího taktu. Na výstupu jednotky se pro N vstupních instrukcí nachází až $N+1$ validních instrukcí (vstupní instrukce + zbytek instrukce z předchozího slova). Z Instruction sort je vyčítáno, až když je odesílána poslední část poslední instrukce z momentálního výstupu Instruction sort.

V instrukcích, které jsou posílány do Page breakeru, je upravena délka tak, že je vytvořen součet délky hlavičky a dat pro část uvnitř jednoho slova. Přidává se informace o tom, zda instrukce popisuje začátek deskriptoru a zda popisuje konec deskriptoru. Pokud je instrukce rozdělena na více částí, pak všechny části obsahují stejnou zkrácenou adresu deskriptoru. Z instrukcí je zde také odebrána informace o adrese v datovém bufferu. Tato informace a informace o oddělených délkách hlavičky a dat je odesílána do jednotky Instruction generator v podobě „subinstrukcí“.

Až do tohoto bodu platilo, že data byla mezi jednotkami v Crossbar scheduleru předávána pomocí mechanismu handshaking protocol. Výstup z Packet breakeru a všechna následující rozhraní už takto řízena nejsou. Data jsou jednoduše vložena na výstup a v příštím taktu jsou přepsána novými daty. V libovolném taktu musí být u takového rozhraní následující jednotka schopna přijmout nový vstup. ednotlivé instrukce mohou být označeny jako validní nebo nevalidní. Tímto způsobem se snižuje složitost jednotek předávajících si data a nevznikají kritické cesty vlivem složitého generování signálu destination ready.

Pro řízení vstupu dat do této části zřetězené linky se používají tři signály vstupující do Packet breakeru.

1. Address assign almost full (AddAFull)
2. Planner almost full (PlaAFull)
3. PCIe buffer almost full (PciAFull)

Tyto signály jsou generovány jednotkami Address assign, Planner a MPS breaker. Pokud má kterýkoli z těchto signálů hodnotu 1, nastaví Packet breaker svůj destination ready na 0 a sám pozastaví produkci validních instrukcí a subinstrukcí.

Jednotka Address assign obsahuje výstupní FIFO frontu pro odesílání transakcí do DMA controlleru. Pokud DMA controller nemůže číst transakce dostatečně rychle (odesílání transakcí na sběrnici DMA trvá příliš dlouho), může dojít k naplnění této fronty. Nesmí ale dojít k tomu, aby do Address assign přišly transakce, které nebude možno uložit do fronty (transakce je zde označení pro výstup z jednotky MPS breaker). Proto musí Address assign detekovat, kdy se fronta blíží zaplnění, a v tu chvíli nastaví signál AddAFull na 1. Jakmile je fronta uvolněna, je signál opět nastaven na 0. Při detekci blížícího se zaplnění musí Address assign dbát na množství validních transakcí, které k němu mohou přijít ještě poté, co nastavil AddAFull. Jedná se o transakce vytvořené z instrukcí momentálně zpracovávaných v Page breakeru a v MPS breakeru. Rezerva pro almost full se nastavuje jako konstanta a předpokládá nejhorší možný případ, kdy všechny stupně zřetězené linky mezi Packet breakerem a Address assignem jsou plné validních instrukcí.

Podobným způsobem funguje také signál PlaAFull, který je generovaný jednotkou Planner. I Planner obsahuje FIFO frontu, která se může blížit naplnění, což musí Planner detekovat a zastavit linku pomocí PlaAFull.

Signál PciAFull je generován jednotkou MPS breaker na základě vstupního signálu PCIe ukazatel z jednotky DMA generator. Nastavením PciAFull na 1 oznamuje, že hrozí zaplnění PCIe bufferu, a proto je třeba zastavit generování plánu pro Crossbar.

4.2.6 Page breaker

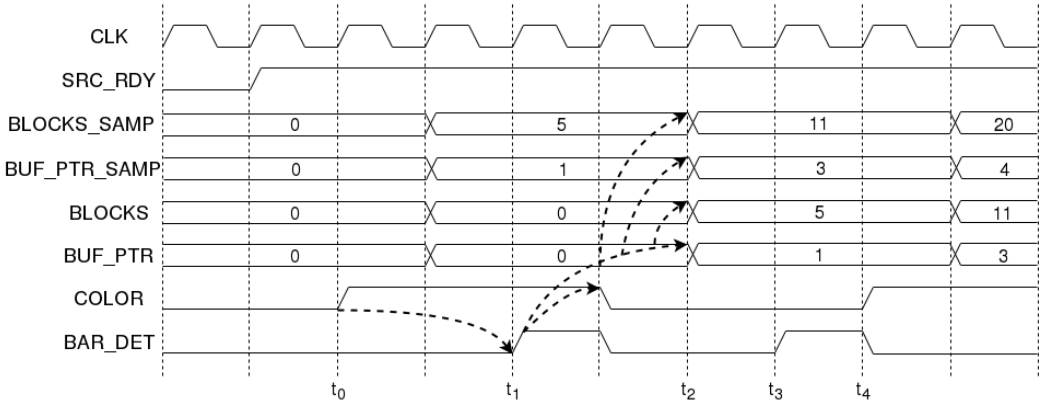
Úkolem Page breakeru je rozdělit instrukce, které by byly zapsány na hranici dvou stránek v paměti, na dvě instrukce. Protože vstupem je $N+1$ instrukcí, z nichž každá může představovat část popsanou jiným deskriptorem, může každá zapisovat na hranici mezi stránkami. Aby se zachovala struktura zřetězené linky bez handshaking protokolu, je na výstupu z Page breakeru $(N+1)*2$ instrukcí.

Jednotka detekuje přetečení hranice stránky podle zkrácené adresy deskriptoru a délky instrukce. Pokud některá příchozí instrukce nepopisuje začátek deskriptoru, je její informace o zkrácené adrese v paměti neplatná (nevyjadřuje skutečnou adresu, kde budou popisovaná data uložena). To řeší Page breaker ukládáním posunuté adresy z předchozích instrukcí. Pokud je u validní příchozí instrukce nastaven flag „konec deskriptoru“ na 0, je k její zkrácené adrese přičtena délka a výsledná adresa je uložena do registru jako adresa pokračování. Mezi instrukcemi přicházejícími v jednom taktu se může vyskytovat pouze jedna taková instrukce, protože v rámci jednoho slova může jen jedna instrukce pokračovat do dalšího slova. Zároveň platí, že příští sada instrukcí bude na prvním místě obsahovat pokračování této instrukce (flag „začátek deskriptoru“ zde bude nastaven na 0). U tohoto pokračování se použije uložená zkrácená adresa místo adresy v instrukci.

Informace o zkrácené adrese deskriptoru a flag „začátek deskriptoru“ už nejsou v dalších částech linky zapotřebí, a proto nejsou obsaženy v instrukcích na výstupu z Page breakeru. Naopak je přidán flag „hranice stránky“, který označuje instrukci končící na konci stránky.

Dalším úkolem Page breakeru je počítat sumu zpracovaných osmibajtových bloků a posouvat čtecí ukazatel pro vstupní buffery DMA modulu. Množství zpracovaných bloků dat se počítá pouze pro pakety, které nejsou zahozeny, a je odesíláno do jednotky Address assign. Čtecí ukazatel se posouvá i pro zahazované pakety a je výstupním signálem Crossbar scheduleru (a také celého DMA modulu; viz signál Read Pointer, tabulka 3.1, podsekce 3.1.1). Toto je poslední místo, kde jsou instrukce o zahazovaných paketech zapotřebí, a proto už nejsou na výstup z Page breakeru posílány.

Signály čítače bloků a čtecího ukazatele smějí být pro konkrétní sadu instrukcí vystaveny na výstup, až když byly tyto instrukce předány jako plán pro Crossbar a když máme jistotu, že data popsaná těmito instrukcemi byla úspěšně přenesena ze vstupních bufferů do PCIe bufferu. Aby toto mohlo být zaručeno, komunikuje Page breaker s jednotkou Planner pomocí „obarvování“ instrukcí a detekce bariéry. Pokaždé, když Page breaker odesílá sadu instrukcí, nese výstup také informaci o „barvě“. Barva může mít hodnotu 0 nebo 1 (jde o jednobitový signál). Tato informace putuje s instrukcemi až do jednotky Planner, která, když detekuje změnu barvy (takzvanou „bariéru“) na vstupních instrukcích, ohlásí tuto skutečnost Page breakeru signálem „barrier detected“.



Obrázek 4.3: Časový diagram pro změnu barvy a detekci bariéry

Princip systému barev a detekce bariér je demonstrován na časovém diagramu 4.3. Hodnoty signálů v tomto diagramu jsou synchronizovány nástupnou hranou hodinového signálu CLK. Signál SRC_RDY je abstraktní signál. Hodnota 1 u tohoto signálu znamená, že Page breaker má na výstupu validní instrukce. Výstupní signál Page breakeru pro čítač přenesených bloků je vyjádřen signálem BLOCKS a čtecí ukazatel zase signálem BUF_PTR. Signál COLOR udává pro výstupní instrukce v daném taktu jejich barvu. Signálem BAR_DET oznamuje Planner, že detekoval změnu barvy. BLOCKS_SAMP a BUF_PTR_SAMP jsou vnitřní registry Page breakeru, které ukládají navzorkované hodnoty čítače bloků a čtecího ukazatele předtím, než jsou propagovány na výstup. Hodnoty vzorků i výstupních signálů jsou na začátku inicializovány na 0.

V čase t_0 odesílá Page breaker první validní výstup s barvou 0. Pro následující výstup hned změní barvu na 1, aby zajistil, že v Planneru bude detekována změna. Aby měl Page breaker čas inkrementovat hodnoty čítače bloků a čtecího ukazatele i pro instrukce z času t_0, provede propagaci předchozích vzorků (s hodnotami 0 a 0) na výstup a vytvoření nových vzorků (5 a 1) až o takt později. V čase t_1 detekuje Planner instrukce s barvou 1 a nastaví signál BAR_DET, což pro Page breaker znamená, že může opět začít používat barvu 0. V čase t_2 proto Page breaker opět produkuje instrukce s barvou 0. Pak znovu provede propagaci navzorkovaných hodnot a nové vzorkování. Hodnoty 5 a 1 signálů BLOCKS a BUF_PTRjsou nyní platné pro okamžik po odeslání výstupu v čase t_0. V čase t_3 je detekována změna barvy z 1 na 0. Další navzorkované hodnoty (11 a 3) jsou propagovány na výstup a jsou vytvořeny nové vzorky (20 a 4) platné pro čas t_4.

Ve skutečnosti Planner produkuje signál BAR_DET až s určitým konstantním zpožděním, aby se ujistil, že všechny instrukce s barvou před detekcí změny byly odeslány na Crossbar a provedeny.

4.2.7 MPS breaker

Jednotka MPS breaker dostává na vstup instrukce, přiřazuje jim adresy konkrétních transakcí v PCIe bufferu a posílá je do jednotky Instruction generator. Dokončené transakce odesílá do jednotky Address assign.

Protože Page breaker má na výstupu pouze instrukce, které nebyly určeny k zahození, musí být ke každé instrukci přiřazena adresa transakce v PCIe bufferu, kam budou přenesena popisovaná data ze vstupních bufferů. O použitelných adresách v PCIe bufferu se dozvídá pomocí signálu PCIe ukazatel, který do Crossbar scheduleru vstupuje z DMA generatoru a který ukazuje na poslední uvolněné pole pro transakci v PCIe bufferu. Na počátku tento signál ukazuje na poslední pole pro transakci v PCIe bufferu. Adresy transakcí jsou v MPS breakeru využívány postupně od 0. Pokud se hodnota použité adresy přibílží hodnotě PCIe ukazatele, je to znamení, že hrozí zaplnění PCIe bufferu a MPS breaker nastaví signál PciAFull na 1, jak již bylo řečeno v podsekci 4.2.5.

Nově příchozí instrukce nemusí nutně znamenat vytvoření nové transakce. Nová transakce je do jednotky Address assign odeslána pouze pokud pro instrukci platí některá z následujících podmínek označujících konec transakce:

1. instrukce obsahuje konec deskriptoru (příznak konec deskriptoru)
2. instrukce končí na hranici stránky (příznak hranice stránky)
3. po přidání instrukce přesahuje transakce hodnotu MPS

Pokud pro instrukci neplatí žádná z těchto podmínek, je pro ni vyhrazena adresa v PCIe bufferu a vygenerována instrukce do Address assign, ale rozpracovaná transakce je místo odeslání do Address assign uložena do registru, aby byla informace o ní uchována do příchodu další části. V jednom taktu (v rámci jednoho slova) může přijít pouze jedna taková instrukce. Zároveň platí, že první instrukce, která přijde v dalším taktu (pokud není linka pozastavena v Packet breakeru), bude obsahovat pokračování. Pro toto pokračování se použije adresa rozpracované transakce a přičte se její velikost.

Vytvářená transakce, která překračuje MPS, musí být ukončena a vygenerována na výstup. Pokud zároveň obsahuje příslušná instrukce příznak konec deskriptoru nebo hranice stránky, jsou vygenerovány dvě transakce. V opačném případě je zbytek uložen jako rozpracovaná transakce.

Pokud je hodnota MPS větší než vstupní datová šířka, pak platí, že jediná instrukce, která může způsobit přetečení MPS, je ta, která navazuje na rozpracovanou transakci. V opačném případě může přetéci kterákoliv instrukce, a to dokonce vícekrát. Protože se stále pohybujeme ve zřetězené lince bez handshaking protokolu, musí být počet výstupů připraven na nejvyšší možný počet transakcí.

Instrukce, které jsou posílány do Instruction generatoru, jsou generovány při každé přijaté vstupní instrukci. Tyto instrukce obsahují pouze délku, valid bit, adresu transakce v PCIe bufferu a offset. Offset slouží jako informace o odsazení instrukce, která není na začátku transakce (navazuje na rozpracovanou transakci).

Transakce posílaná do Address assign obsahuje adresu transakce, délku, valid bit, DMA kanál, flag „konec deskriptoru“ a flag „konec paketu“. Tyto flagy slouží později v DMA controlleru k aktualizaci HW ukazatele.

4.2.8 Address assign

Do jednotky Address assign vstupují z jednotky Instrukction sort deskriptory a z MPS breakeru transakce. Na jeden validní deskriptor připadá jedna nebo více validních transakcí. Platí přitom, že transakce přicházejí ve stejném pořadí jako příslušné deskriptory. Úkolem Address assign je zjistit na základě délek deskriptorů a transakcí, které transakce patří ke kterým deskriptorům a přiřadit jednotlivým transakcím adresu v RAM získanou z deskriptoru.

Tato jednotka obsahuje 3 FIFO fronty:

1. fronta pro deskriptory
2. fronta pro transakce
3. výstupní fronta

Protože deskriptory přicházejí do jednotky dříve než transakce, musejí být uchovávány ve frontě. Z této fronty je přední deskriptor odebrán, až když jsou mu přiřazeny všechny příslušné transakce. Rozhraní DMA sběrnice, po kterém budou posílány transakce, dokáže na jednom endpointu zpracovat nejvýše jednu transakci za takt. Address assign proto musí v jednom taktu generovat jednu transakci pro každý endpoint. Protože ale MPS breaker dokáže vytvářet v jednom taktu více transakcí, musí být tyto transakce uloženy do fronty a teprve odtud mohou být přiřazeny k deskriptorům. Po přiřazení adresy v RAM k transakci je transakce přesunuta do třetí fronty. Tato fronta je výstupní frontou Crossbar scheduleru a z ní jsou transakce odebírány DMA controllerem.

Velikost výstupní fronty je nastavena na maximální počet transakcí, které lze uložit do PCIe bufferu. Díky tomu se tato fronta nemůže přeplnit, neboť dříve dojde k nastavení signálu PciAFull v jednotce MPS breaker. Velikost fronty pro deskriptory je nastavena tak, aby dokázala obsáhnout maximální počet deskriptorů, které popisují instrukce od začátku Packet breakeru až po transakce v MPS breakeru. Tento počet je roven počtu stupňů linky od začátku Packet breakeru do konce MPS breakeru násobený počtem instrukcí na vstupu Packet breakeru. Proto se ani tato fronta nemůže nikdy přeplnit.

V případě fronty pro transakce ale může dojít k tomu, že položky budou odebírány pomaleji než jsou přidávány a to po libovolně dlouhou dobu. U této fronty musí být proto nastaven limit pro blížící se zaplnění. V případě hrozícího zaplnění této fronty se nastaví výstupní signál AddAFull na 1, což způsobí zastavení linky v Packet breakeru.

Do jednotky Address assign vstupuje signál čítače bloků z Page breakeru. Jak bylo řečeno v podsekci 4.2.6, slouží tento signál k tomu, aby byly z Crossbar scheduleru odesílány pouze ty transakce, které se již určitě nacházejí v PCIe bufferu. Pro Address assign to znamená, že u čtecího rozhraní z výstupní fronty se nachází čítač odeslaných bloků zvyšující se s každou odeslanou transakcí. Pokud platí, že po odeslání následující transakce, by tento čítač převýšil hodnotu čítače z Page breakeru, pak je výstup zablokován a čeká se, až se zvýší hodnota čítače posílaná z Page breakeru.

4.2.9 Instruction generator

Jednotka Instruction generator přijímá z Packet breakeru subinstrukce obsahující adresy ve vstupních bufferech a z MPS breakeru instrukce obsahující adresy v PCIe bufferu. Jejím úkolem je spojit je a vytvořit sadu instrukcí vhodných pro Crossbar, které odešle do jednotky Planner. Na jednu subinstrukci může připadnout více instrukcí, neboť instrukce byly v Page breakeru děleny podle hranic stránek v paměti.

Na výstupu Instruction generatoru se nachází jedna instrukce pro každý jeden vstupní datový blok. Každá z nich obsahuje adresu zdrojového slova ve vstupních bufferech, adresu cílového slova a bloku v PCIe bufferu a barvu, která je obsažena i ve vstupních instrukcích.

4.2.10 Planner

Úkolem jednotky Planner je přijmout instrukce z Instrukction generatoru a přeskládat je tak, aby bylo jejich provádění v Crossbaru co nejrychlejší. Protože se tato jednotka nacházela i v Cossbar scheduleru určeném pro druhý navrhovaný typ DMA modulu, použil jsem ji bez nutnosti jejího opětovného návrhu.

Kromě rozhraní předávájícího plán instrukcí do Crossbaru vystupuje z Planneru také signál detekce bariéry reagující na změnu barvy instrukcí a signál PlaAFull, který informuje, že se blíží zaplnění fronty pro instrukce, jež se v Planneru nachází. Tyto signály byly již popsány v podsekcích 4.2.6 a 4.2.5.

4.2.11 Řízení kanálů

název	I/O	šířka
ChStop	I	počet kanálů
ChStart	I	počet kanálů
ChStopAck	O	1

Tabulka 4.1: Rozhraní Crossbar scheduleru pro řízení kanálů

Rozhraní pro řízení kanálů mezi Crossbar schedulerem a DMA controllerem se skládá ze signálů ChStop a ChStart vstupujících do Crossbar scheduleru a vystupujícího signálu ChStopAck. Jak signál ChStop, tak ChStart obsahují jeden bit pro každý kanál. Pokud je bit daného kanálu u ChStart nastaven na 1, znamená to požadavek DMA controlleru na spuštění kanálu. Nastavení bitu v signálu ChStop znamená požadavek k zastavení kanálu. Musí platit, že v libovolném taktu může být v obou těchto signálech dohromady jen jeden bit nastaven na 1. Jednobitovým signálem ChStopAck dává Crossbar scheduler najevo, že dokončil zastavování kanálu.

Pokud DMA controller žádá o spuštění kanálu, může to být provedeno okamžitě. Je to provedeno tak, že si Crossbar scheduler v interním registru označí tento kanál za spuštěný. V případě zastavování kanálu je situace složitější. Po přijmutí výzvy k zastavení kanálu musí Crossbar scheduler přestat přijímat pakety na daném kanálu. Instrukce, které již byly na kanálu přijaty, byly jim přiřazeny deskriptory a nacházejí se ve zřetězené lince, musejí být ale zpracovány a odeslány, aby mohl být správně posunut HW ukazatel v DMA controlleru. Také instrukce z tohoto kanálu, které se nacházejí v Accept jednotce a jsou jim přiřazovány deskriptory, musejí být zpracovány. Jakmile jsou všechny zpracovávané transakce na za-

stavovaném kanálu bezpečně odeslány z Crossbar scheduleru, potvrdí jednotka zastavení signálem ChStopAck. Teprve poté může DMA controller požádat o zastavení jiného kanálu.

Postup při zastavování kanálu v Crossbar scheduleru je následující:

1. Signál na zastavení kanálu je poslán do Process assign. Pokud Process assign právě posílá instrukce na tomto kanálu do Accept jednotky, musí počkat, až jsou tyto instrukce odeslány. Potom si sám označí kanál jako zastavený a odteď všechny instrukce na tomto kanálu označuje k zahození. Tím zaručí, že u následujících instrukcí na daném kanálu se nebude v Accept jednotce uvažovat jejich přijetí. Process assign potvrdí zastavení kanálu ve své části linky.

2. Signál na zastavení je následně odeslán do jednotek Accept a Descriptor FIFO. Protože máme nyní zaručeno, že Accept nebude požadovat žádné deskriptory na zastavovaném kanálu, jsou v jednotce Descriptor FIFO vynulovány čítače pro obsažené deskriptory na tomto kanálu, a tím se deskriptory odstraní. Descriptor FIFO neuchovává informaci o zastavení kanálů a neobsahuje mechanizmus pro odmítání deskriptorů, které přijdou na zastaveném kanálu. Z toho důvodu musí DMA controller zaručit, že už když odesílá požadavek na zastavení, nebude na zastavovaném kanálu odesílat další deskriptory. Po vynulování čítačů signalizuje Descriptor FIFO potvrzení zastavení kanálu v jeho části. Z jednotky Accept je signál pro zastavení kanálu propagován do Instruction sort až po odeslání předchozích instrukcí. Tento signál je vázán na sadu instrukcí, které Accept přijal společně se signálem a nesmí dojít k tomu, že tento signál předběhne ve zřetězené lince tuto sadu instrukcí.

3. Z jednotky Instruction sort je signál odesílán do Address assign společně s deskriptorem. Je tam ale odeslán i v případě instrukce pro zahazovaný paket, kdy se žádný deskriptor do Address assign nevkládá.

4. V Address assign je simulován průchod signálu frontou pro deskriptory a výstupní frontou tak, aby nedošlo k předběhnutí instrukcí předcházejících signálu. Průchod frontou je simulován uložením signálu do registru společně s hodnotou momentální velikosti fronty. Tato velikost je dekrementována s každým čtením položky z fronty. Jakmile je hodnota 0, je signál přečten z registru. Tento systém lze použít, protože víme, že v Crossbar scheduleru se může naráz nacházet pouze jeden požadavek na zastavení. Po průchodu signálu jednotkou Address assign signalizuje jednotka potvrzení zastavení ve své části.

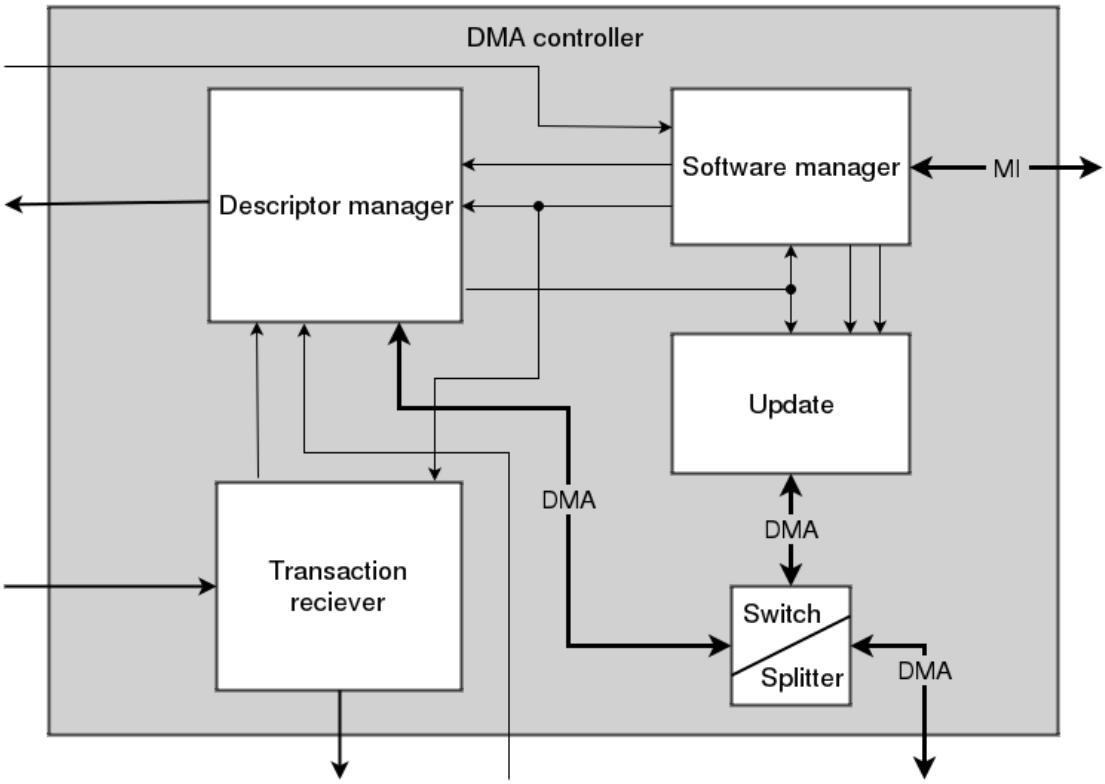
5. Jakmile získá Crossbar scheduler potvrzení o zastavení od jednotek Descriptor FIFO i Address assign, nastaví signál ChStopAck na 1 a potvrdí zastavení kanálu.

Pro zastavování kanálu v Crossbar scheduleru je důležité, že signál o zastavení musí projít zřatezenou linkou, i když zrovna neobsahuje zádné validní instrukce (nepřichází žádný vsup z instrukční fronty).

4.3 DMA controller

Jednotka DMA controller zajišťuje řízení DMA modulu, stahování deskriptorů z paměti počítače a aktualizaci HW ukazatele v paměti. Do jednotky vstupují rozhraní sběrnic MI a DMA. S jednotkou Crossbar scheduler komunikuje odesíláním deskriptorů a přijímáním transakcí. Na základě přijatých transakcí aktualizuje DMA modul HW ukazatel a následně

transakce odesílá do DMA generatoru. Od DMA generatoru se pak zpětně dozvídá o dokončení odesílání jednotlivých transakcí pomocí potvrzovacího rozhraní.



Obrázek 4.4: Architektura DMA controlleru

Návrh architektury tohoto modulu je zobrazen na obrázku 4.4. Jako v případě schématu jednotky Crossbar scheduler platí, že zapojení je ekvivalentní se zapojením na schématu DMA modulu na obrázku 4.1 vyjma rozhraní pro řízení kanálů, které bylo odebráno pro zjednodušení. Jednotka se skládá z komponent Descriptor manager, Software manager, Transaction receiver, Update a Switch/Splitter. Všechy signály propojující tyto jednotky jsou popsány v podsekcích zabývajících se jednotlivými jednotkami.

Software manager se stará o komunikaci DMA modulu přes rozhraní MI, uchovává veškeré MI registry a poskytuje přístup k nim ostatním jednotkám. Descriptor manager spravuje stahování deskriptorů z paměti přes sběrnici DMA a jejich odesílání do Crossbar scheduleru. Také je v něm uchován HW ukazatel, který je odesílán při aktualizaci do paměti. Jednotka Update přijímá od Descriptor manageru požadavky na aktualizaci HW ukazatele a provádí je prostřednictvím rozhraní DMA. Transaction receiver získává transakce z Crossbar scheduleru a posílá je do DMA controlleru.

Jednotka Switch/Splitter je propojovací uzel sběrnice DMA, který umožňuje připojení jednotek Descriptor manager a Update na jedno rozhraní sběrnice. Tato jednotka je běžná v komponentách používajících propojení přes DMA sběrnici a její architektura není součástí návrhu DMA modulu.

4.3.1 Transaction receiver

Jednotka Transaction receiver přijímá transakce z Crossbar scheduleru, připojuje k nim horní část adresy do RAM a odesílá je do DMA generatoru. Vrchních 16 bitů adresy deskriptoru zjistí čtením MI registru z jednotky Software manager. Jednotka také předává informace o tom, zda posílaná transakce obsahuje konec deskriptoru a konec paketu, do jednotky Descriptor manager. Tyto informace jsou zapotřebí, když se v Descriptor manageru inkrementuje HW ukazatel a odesílá se aktualizace. Na výstupu z Transaction receiveru směrem do Descriptor manageru jsou tyto informace ukládány do fronty, odkud jsou čteny, teprve když Descriptor manager získá od DMA generatoru potvrzení o odeslání příslušných transakcí.

4.3.2 Descriptor manager

Hlavní funkce Descriptor manageru je stahování deskriptorů z deskriptorového bufferu v paměti. Descriptor manager také uchovává informaci o HW ukazateli a o interním HW ukazateli. HW ukazatel je inkrementován, když dojde k odeslání transakce obsahující flag „konec deskriptoru," a je odesílán do jednotkek Update a Software manager, když je odeslána transakce s koncem paketu (podle DPDK může být HW ukazatel aktualizován pouze po přenesení celého paketu). Interní HW ukazatel je inkrementován, když Descriptor manager zažádá o čtení deskriptorů, a obsahuje adresu deskriptoru v deskriptorovém bufferu, který bude Descriptor manager číst jako další.

Porovnáním interního HW ukazatele a SW ukazatele v MI registru dokáže jednotka zjistit množství volných deskriptorů uložených v deskriptorovém bufferu. Jednotka může naráz zažádat o čtení nejvýše 64 deskriptorů, neboť je omezena maximální velikostí čtecího požadavku na sběrnici PCIe. Pokud je v bufferu na spuštěném kanálu alespoň 64 volných deskriptorů a zároveň je v Descriptor FIFO volné místo pro jednu transakci (tato informace je propagována signálem z Descriptor FIFO), vytvoří Descriptor manager čtecí transakci na DMA sběrnici pro čtení těchto deskriptorů a zvýší interní ukazatel o 64. Přečtených 64 deskriptorů přichází po DMA sběrnici ve skupinách po 8 v jednom taktu (velikost deskriptoru je 64 bitů a DMA rozhraní má datovou šířku $8 * 64 = 512\,\mathrm{b}$). Tyto přijaté deskriptory jsou ihned vloženy do Descriptor FIFO. Teprve když je všech 64 deskriptorů přijato, zkontroluje opět Descriptor manager, zda je místo v Descriptor FIFO a volné deskriptory ke stažení, a odešle další čtecí požadavek.

Aby mohl Descriptor manager přijímat deskriptory z více kanálů naráz a nemusel čekat na přijetí 64 deskriptorů z jednoho kanálu, než bude číst z jiného kanálu, používá označení transakcí pomocí tagů nacházejících se v hlavičce DMA transakce. Jednotka odešle čtecí požadavky postupně na všechny spuštěné kanály, každý s jiným tagem. U příchozí odpovědi pak identifikuje kanál, ke kterému deskriptory patří, podle tagu transakce, který musí být stejný jako u odpovídajícího požadavku.

Další funkcí Descriptor manageru je inkrementace HW ukazatele pro odeslané datové transakce. Aby to mohl dělat, potřebuje Descriptor manager informaci od DMA generatoru, že byla transakce odeslána z DMA modulu na DMA sběrnici, a také informace o kanálu transakce, zda se jednalo o transakci s koncem deskriptoru a zda s koncem paketu. Informace o transakci přečte Descritpor manager z rozhraní Transaction receiveru ve chvíli, kdy od DMA generatoru dostane potvrzení o odeslání. HW ukazatel se inkrementuje pouze pro transakce s koncem deskriptoru. K odeslání požadavku na aktualizaci HW ukazatele do jednotky Update a Software manager dochází jen u transakcí s koncem paketu. Požadavek o aktualizaci se skládá z čísla kanálu, hodnoty ukazatele a valid bitu.

4.3.3 Software manager

Jednotka Software manager obsahuje sadu MI registrů popsaných v tabulce 3.5 a umožňuje ovladači přístup k nim pomocí rozhraní MI.

Pomocí signálů z jednotky Accept inkrementuje Software manager registry Cntr Recieved a Cntr Discarded. Jednotkám Descriptor manager a Transaction receiver poskytuje rozhraní pro čtení hodnot registrů Descriptor Addr a jednotce Descriptor manager také ke čtení registrů Pointer Mask a SW Pointer. Jednotkám Accept a Descriptor FIFO v Crossbar scheduleru umožňuje Software manager číst registr Descriptor Size (Accept porovnává velikosti deskriptorů s velikostmi paketů a Descriptor FIFO počítá celkovou velikost stažených deskriptorů). Upadte čte ze Software manageru hodnoty registrů Timeout a Update Addr.

Když Descriptor manager vytvoří požadavek na aktualizaci, je v Software manageru aktualizována hodnota registru HW pointer. Protože registrem Control řídí ovladač zastavování a spouštění DMA modulu, je spouštění i zastavování v rámci modulu inicializováno v Software manageru a po provedení některé z těchto akcí je zde nastavena hodnota registru Status.

Protože DMA modul musí obsahovat jednu sadu registrů pro každý kanál, bude složitost Software manageru vzrůstat s počtem kanálů.

4.3.4 Update

Update zajišťuje aktualizaci HW ukazatele uloženého v paměti na adrese Update Addr. Obsahuje svůj vlastní registr s HW ukazatelem, který nastavuje při přijetí požadavku na aktualizaci od Descriptor manageru. Dále obsahuje vnitřní čítač, který se inkrementuje s každým taktem. Když čítač dosáhne hodnoty Timeout pro daný kanál, je provedena aktualizace HW ukazatele podle hodnoty v registru. To se děje postupem popsaným dříve v podsekci 3.2.2.

Tato jednotka byla používána i v dřívějších verzích DMA modulu, a proto nebyla navrhována v rámci této práce.

4.3.5 Řízení kanálů

Při spouštění kanálu zapisuje ovladač přes MI rozhraní do registru Control hodnotu 1. Tento zápis vygeneruje v DMA controlleru signál pro spuštění kanálu. Stejně jako u Crossbar scheduleru se spuštění kanálu provádí nastavením interních registrů jednotlivých jednotek informujících o stavu kanálů. Tento signál je následně propagován do Crossbar scheduleru. Descriptor manager vynuluje HW registr a interní HW registr a začne stahovat deskriptory pro tento kanál a hodnota registru Status tohoto kanálu se změní na 1.

Stejně jako v jednotce Crossbar scheduler je i zde postup při zastavování složitější:

1. Při zastavování kanálu je přes MI rozhraní zapsána do registru Control hodnota 0. Software manager vygeneruje signál pro zastavení pro jednotky Descriptor manager a Crossbar schduler.

2. Descriptor manager přestane stahovat deskriptory na tomto kanálu a v případě, že do něj přijdou DMA transakce obsahující deskriptory, o jejichž čtení žádál už dříve, ignoruje je a neposílá je do jednotky Descriptor FIFO.

3. Když se z jednotky Crossbar scheduler vrátí potvrzení o zastavení kanálu, dostane se do Transaction receiveru a je vloženo do výstupní fronty společně s informacemi

o odesílané transakci. Odtud je přečteno Descriptor managerem, když DMA generator potvrdí odeslání této transakce. Transaction receiver musí být schopen předat potvrzení o zastavení Descriptor manageru i v případě, že v DMA modulu nejsou zrovna žádná data a z Crossbar scheduleru nepřicházejí instrukce. To lze docílit pomocí jednobitového signálu „Force read“. Pokud je nastaven na 1, Descriptor manager přečte potvrzení o zastavení kanálu a nečeká na potvrzení od DMA generatoru.

4. Descriptor manager odešle požadavek na aktualizaci HW ukazatele a také pošle požadavek na zastavení kanálu do jednotky Update.

5. Když Update provede poslední aktualizaci HW ukazatele, posílá potvrzení o zastavení kanálu do Software manageru.

6. Software manager nastaví hodnotu registru Status na 0, čímž dává najevo úspěšné zastavení kanálu v DMA modulu.

Aby mechanizmus spuštění a zastavování kanálů fungoval, smí ovladač v jednu chvíli zastavovat nebo spouštět pouze jeden kanál. Pokud není u všech kanálů hodnota registru Status rovna hodnotě registru Control, nesmí měnit žádný z registrů Control.

4.4 Shrnutí návrhu

Na konci předchozí kapitoly v sekci 3.3 bylo uvedeno shrnutí důležitých požadavků, které by měl návrh DMA modulu splňovat. Podle těchto požadavků můžeme nyní můžeme zhodnotit správnost a kvalitu návrhu.

Kompatibilita DMA modulu s rozhraními je zajištěna jednotkami, které jsou k jednotlivým rozhraním připojeny, a návrh modulu byl vytvořen tak, aby bral ohled na jejich podobu. S tím souvisí postup při odesílání a přijímání transakcí na sběrnici DMA, přístup k registrům pomocí rozhraní MI a schopnost přijímat data a instrukce.

Schopnost DMA modulu pracovat s deskriptory a s ukazateli do deskriptorového bufferu je zajišťována především jednotkami Descriptor manager a Update. Navržený DMA modul je schopen přenášet data do paměti v souladu se systémem DPDK.

Zvyšování počtu kanálů u DMA modulu povede ke zvýšení složitosti a prodlužování kritických cest na několika místech v návrhu. Tato místa jsou však v návrhu explicitně zmíněna, což umožní jejich snazší optimalizaci v případě, že to bude zapotřebí.

Pro efektivní využití sběrnice PCIe byl Crossbar scheduler navržen tak, aby přesouval posílaná data do PCIe bufferu přinejmenším tak rychle, jak přicházejí do DMA modulu. V případě vyšší rychlosti dat přicházejících do modulu lze reagovat zvýšením frekvence, na níž Crossbar scheduler pracuje.

Aby mohl být návrh považován za flexibilní, jsou v popisu jednotlivých částí zmíněny ty proměnné, které mohou strukturu těchto částí měnit. Nejčastěji se může jednat o změnu počtu kanálů (má vliv především na Software manager, Descriptor manager a Descriptor FIFO) a změnu vstupní datové šířky (ovlivňuje počet vstupních instrukcí v Crossbar scheduleru). Rozšiřitelnost ze strany vstupní datové šířky je umožněna minimalizací práce se damotnými daty. Data, která mají být poslána do RAM, se v DMA modulu nacházejí pouze v Crossbar scheduleru, v PCIe bufferu a DMA generatoru.

Na základě tohoto shrnutí lze říci, že všechny důležité požadavky stanovené pro návrh DMA modulu byly splněny.

Kapitola 5

Implementace

Pro imlementaci komponenty DMA modulu byl zvolen jazyk VHDL. Hlavním důvodem byla kompatibilita s existujícími komponentami architektury, do které bude DMA modul zapojen. Jazyk VHDL [9] je standardizovaný jazyk pro popis hardwaru a je podporován všemi nástroji potřebnými k vývoji pro FPGA.

Pro dobrou přehlednost a udržovatelnost kódu byl DMA modul rozdělen na komponenty popsané ve zvláštních souborech a i ty byly rozděleny na menší komponenty. Rozdělení na komponenty odpovídá obrázkům 4.1, 4.2 a 4.4 z kapitoly o návrhu.

Modul byl implementován pro vstupní datové šířky 512 b a 1024 b. Pokud by dokázal při datové šířce 512 b pracovat na frekvenci 200 MHz, dokázal by teoreticky zpracovat 102,4 Gb vstupních dat za sekundu ($200\,Hz * 10^6 * 512\,b = 102,4 * 10^9\,b/s$).

V sekci 5.1 je popsáno řešení implementace pro specifické části, jejichž složitost škáluje podle počtu kanálů. Sekce 5.2 popisuje naměřené výsledky zabraných zdrojů a časování pro implementovanou architekturu. V poslední sekci 5.3 je shrnut postup při verifikaci implementovaného modulu v simulaci.

Součástí implementace bylo i zapojení komponenty do existující architektury určené pro síťovou kartu. DMA modul v architektuře karty se skládá ze dvou částí. Přijímání dat ze sítě zajišťuje strana označovaná jako „RX“ a data z počítače do sítě odesílá strana „TX“. Protože náš DMA modul přenáší přijatá data z FPGA do RAM v počítači, je zapojen jako RX DMA modul.

5.1 Řešení kritických částí

Speciální pozornost jsem věnoval kritickým částem, jejichž složitost se zvyšuje s rostoucím počtem DMA kanálů. Jak bylo řečeno v kapitole 4, těmito kritickými částmi jsou především Software manager v komponentě DMA controller a Descriptor FIFO v komponentě Crossbar scheduler.

V Software manageru se nachází sada MI registrů pro každý kanál. Některé z registrů nepotřebují být implementovány jako 32bitové, protože do nich nikdy není nahrána 32bitová hodnota. Například registru Control stačí pouze 1 bit. Všechny registry v jedné sadě dohromady proto zabírají celkem 418 b. Pro většinu registrů v sadě platí, že je do nich zapisováno v jeden moment pouze na jednom kanálu (přes MI rozhraní může ovladač přistoupit naráz pouze k jednomu registru) a to stejné platí i pro jejich čtení. Proto mohou být tyto registry implementovány s využitím paměťových LUT (LUT uvnitř sliceM použitá jako distribuovaná paměť). Jedna paměťová LUT obsáhne jeden bit registru pro 64 kanálů

současně a umožní přístup k nim přes nezávislé čtecí a zápisové rozhraní. Některé registry v sadě však nemohou být implementovány tímto způsobem, neboť z nich čte více komponent DMA modulu a zároveň může probíhat čtení z různých kanálů. Tyto registry musí být implementovány jako registrová pole. Jedná se o registry: Control (1 b), Status (1 b), Pointer Mask (32 b) a Descriptor Size (16 b).

V jednotce Descriptor FIFO způsobuje vysoký počet kanálů nárůst zabraných zdrojů a značné prodloužení kritických cest v rámci logiky pro čtení deskriptorů. Descriptor FIFO se skládá z bufferu, který je implementován pomocí BRAM, a cache implementované pomocí sady paměťových LUT. Další součástí jednotky jsou čítače, které uchovávají informaci o zaplnění bufferu a cache a o čtecích a zápisových ukazatelých do nich pro každý kanál. Také je zde čítač sumy prostorů, které deskriptory uložené v této jednotce na každém kanálu popisují. Při přijímání nových deskriptorů, při čtení deskriptorů z jednotky a při přenosu deskriptorů z bufferu do cache musí být tyto hodnoty aktualizovány.

Aby nemusela být logika provádějící aktualizaci kopírována pro každý kanál, byla pro implementaci těchto čítačů použita speciální komponenta realizující multiportovou paměť. Pomocí paměťových LUT a registrových polí tato komponenta simuluje paměť s libovolným počtem zápisových a čtecích portů za cenu latence jednoho taktu při čtení. Protože počet různých kanálů, na kterých je naráz nutno s jednotlivými čítači pracovat, je podstatně nižší než celkový počet kanálů, sníží se dramaticky množství aktualizační logiky, a vylepší se časování. U čítače sumy prostorů, čtecího ukazatele z cache a zaplnění cache je počet portů závislý na počtu vstupních instrukcí Crossbar scheduleru (určuje počet Accept procesů, které čtou z jednotky Descriptor FIFO). Například u čítače zápisového ukazatele do bufferu jsou ale pouze 2 čtecí a 2 zápisová rozhraní, protože v jednom taktu se s touto hodnotou pracuje nejvýše ve 2 různých DMA kanálech (inkrementace hodnoty při zápisu nových deskriptorů a resetování hodnoty při zastavení kanálu).

Tímto způsobem implementace se minimalizuje vliv zvýšení počtu kanálů na složitost architektury DMA modulu, což umožní provozovat modul na vysoké frekvenci.

5.2 Spotřeba zdrojů a časování

Pro zjištění zdrojů, které DMA modul na FPGA čipu Virtex UltraScale+ zabírá, a maximální bezpečné pracovní frekvence jsem použil nástroj Vivado 2017.4 od firmy Xilinx. Tento nástroj umožňuje přeložit VHDL kód a syntetizovat z něj obvodové zapojení specifické pro dané FPGA. U syntetizovaného modulu lze následně zjistit kolik komponent nacházejících se na FPGA bylo pro modul použito a zda je splněno časování na jednotlivých kombinačních logických cestách. Je nutno poznamenat, že výsledky analýzy časování nejsou u samotné syntézy úplně přesné, neboť časování je závislé na rozložení komponent na FPGA, což není součástí procesu syntézy.

Tabulky 5.1 a 5.2 zobrazují počet LUT, paměťových LUT (LUTmem), registrů (REG), BRAM a dosaženou frekvenci DMA modulu na FPGA Virtex UltraScale+ VU7P pro různý počet DMA kanálů. První z nich obsahuje informace o DMA modulu v nastavení s datovou šířkou 512 b a druhá s datovou šířkou 1024 b. U každé hodnoty počtu spotřebovaných zdrojů je uvedeno také procentuální využití dostupných zdrojů na čipu.

Tabulky ukazují nárůst všech spotřebovaných zdrojů s vyšším počtem kanálů. Přesto ale DMA modul spotřebuje nejvýše asi 11 % dostupných LUT na FPGA, a to v nastavení s datovou šířkou 1024 b a 256 DMA kanály. Spotřeba paměťových LUT a registrů je rovněž relativně nízká. Míra využití paměťových bloků BRAM je vysoká, ale tyto paměti jsou v DMA modulu zapotřebí. Nejvíce se jich nachází v jednotce Descriptor FIFO, kde jsou

DMA kanály	LUT ([%])	LUTmem ([%])	REG ([%])	BRAM ([%])	Frekvence [MHz]
8	10 524 (1)	3 569 (1)	12 105 (1)	57,5 (4)	329
16	12 029 (2)	3 783 (1)	12 906 (1)	73,5 (5)	329
32	15 188 (2)	4 209 (1)	14 789 (1)	105,5 (7)	336
64	20 644 (3)	5 451 (1)	18 526 (1)	169,5 (12)	329
128	33 463 (4)	8 537 (2)	26 032 (2)	297,5 (21)	333
256	59 930 (8)	14 811 (4)	41 981 (3)	553,5 (38)	326

Tabulka 5.1: Výsledky syntézy RX DMA modulu na FPGA Virtex UltraScale+ VU7P pro datovou šířku 512 b

DMA kanály	LUT ([%])	LUTmem ([%])	REG ([%])	BRAM ([%])	Frekvence [MHz]
8	30 388 (4)	5 012 (1)	22 626 (1)	97 (7)	278
16	32 024 (4)	5 446 (1)	23 472 (1)	113 (8)	304
32	35 700 (5)	6 280 (2)	25 800 (2)	145 (10)	287
64	42 147 (5)	8 490 (2)	30 418 (2)	209 (15)	278
128	57 226 (7)	13 792 (3)	39 642 (3)	337 (23)	281
256	87 710 (11)	24 418 (6)	59 033 (4)	593 (41)	282

Tabulka 5.2: Výsledky syntézy RX DMA modulu na FPGA Virtex UltraScale+ VU7P pro datovou šířku 1024 b

do nich ukládány deskriptory. Je třeba si uvědomit, že u kompletní architektury určené pro síťovou kartu můžou ostatní komponenty obsadit i více než polovinu celého čipu.

Výsledky syntézy ukazují, že modul dokáže pracovat na frekvenci 200 MHz na libovolném ze zvolených nastavení. Ve většině z nich dosáhne i frekvence 300 MHz. Není ale jisté, jestli těchto frekvencí skutečně dosáhne, až bude společně s celou architekturou umístěn na FPGA čip.

V tabulce 5.3 jsou poměry zdrojů jednotlivých komponent vyjádřeny porovnáním počtu obsazených LUT pro 8 a pro 256 DMA kanálů. Hodnoty počtu LUT jsou v tomto případě součtem počtu logických a paměťových LUT. Poměry počtů využitých registrů jsou uvedeny v tabulce 5.4. Komponenty jsou v tabulkách rozděleny na ty, které jsou obsaženy v Crossbar scheduleru, a ty, které jsou součástí DMA controlleru (Transaction receiver, Descriptor manager a Software manager). Hodnoty využití BRAM zde nejsou zobrazeny v žádné tabulce. Jediná z uvedených jednotek DMA modulu, která obsahuje hard bloky BRAM, je Descriptor FIFO. Při změně počtu kanálů se počet BRAM v této komponentě zvyšuje přímo úměrně, protože buffer v jednotce Descriptor FIFO obsahuje pro každý kanál konstantní počet BRAM.

Největší poměr zvětšení v počtu LUT (23krát) byl zaznamenán u jednotky Transaction receiver. Pro 256 DMA kanálů tvoří většinu této jednotky jediný multiplexor, který vybírá horních 16 bitů adresy z jednoho z MI registrů (Descriptor Addr High) podle kanálu a přiřazuje je do transakcí posílaných na PCIe. Pokud ale porovnáme absolutní počet LUT v této jednotce s ostatními jednotkami, vidíme, že její velikost je i pro 256 DMA kanálů zanedbatelná. Ani z hlediska časování nevznikaly v této komponentě žádné kritické cesty

komponenta	LUT (8)	LUT (256)	poměr zvětšení
Process assign	34	462	13,6
Accept	227	1872	8,2
Descriptor FIFO	2552	36254	14,2
Instruction sort	135	140	1
Packet beaker	142	148	1
Page breaker	43	40	0,9
MPS breaker	219	234	1,1
Address assign	621	680	1,1
Trans. receiver	48	1104	23
Descriptor manager	1033	3665	3,5
Software manager	1083	16316	15

Tabulka 5.3: Porovnání počtu LUT obsazených komponentami RX DMA modulu pro 8 a 256 DMA kanálů

komponenta	REG (8)	REG (256)	poměr zvětšení
Process assign	47	310	6,5
Accept	300	400	1,3
Descriptor FIFO	2425	15401	6,4
Instruction sort	317	337	1,1
Packet beaker	192	217	1,1
Page breaker	136	148	1,1
MPS breaker	172	192	1,1
Address assign	677	727	1,3
Trans. receiver	91	96	1,1
Descriptor manager	299	1941	6,5
Software manager	394	10859	27,6

Tabulka 5.4: Porovnání počtu registrů obsazených komponentami RX DMA modulu pro 8 a 256 DMA kanálů

a proto pro počet kanálů 256 a méně není třeba tuto komponentu nijak optimalizovat. Podobně tomu je i u komponent Process assign, Accept a Descriptor manager.

Tabulky 5.3 a 5.4 ukazují, že zdaleka největší vliv na spotřebu zdrojů při vzrůstu počtu kanálů mají jednotky Descriptor FIFO a Software manager, což potvrzuje původní předpoklad, že se jedná o kritické části. I navzdory optimalizacím provedeným v těchto jednotkách způsobilo zvýšení počtu kanálů z 8 na 256 navýšení množství potřebných LUT v nich přibližně 15krát. U jednotky Software manager vzrostl počet obsazených registrů dokonce více než 27krát. Přesto ale díky optimalizacím došlo ke značnému zlepšení těchto poměrů a rovněž ke zvýšení maximální pracovní frekvence DMA modulu. V dřívější fázi implementace, kdy byly MI registry v Software manageru a čítače v Descriptor FIFO implementovány jako registrová pole, byly tyto komponenty i několikanásobně větší a způsobovaly potíže s časováním.

5.3 Verifikace

Pro ověření funkčnosti implementace jsem vytvořil verifikační prostředí testující schopnost DMA modulu přenášet data v simulaci. Jako simulační nástroj byl použit program ModelSim a verifikaci jsem napsal v jazyce SystemVerilog [10], což je objektový jazyk určený k verifikaci hardwaru.

Simulace chování DMA modulu spočívá v jeho zapojení do komponenty „testbench“ a následném generování validních hodnot na vstupní signály. Pro verifikaci funkčnosti byl simulován příchozí datový provoz, řízení modulu přes MI sběrnici a přístup do virtuální paměti přes sběrnici DMA. Porovnáním dat odeslaných modulem do virtuání paměti s daty poslanými na vstup byla ověřena správnost chování. Verifikační program testoval modul na opakované spouštění a zastavování kanálů a na nejrůznější nastavení.

parametr	možné hodnoty
DMA kanály	2, 4, 8, 16, 32
datová šířka [b]	512, 1024
PCIe endpointy	1, 2
velikost paketů (min) [B]	64, 128, 256, 512, 1024
velikost paketů (max) [B]	128, 256, 512, 1024, 2048
velikost deskriptorů (min) [B]	64, 128, 256, 512
velikost deskriptorů (max) [B]	256, 512, 1024, 2048, 4096

Tabulka 5.5: Varianty nastavení verifikačních běhů DMA modulu

V tabulce 5.5 je obsažen výčet parametrů, které byly u verifikovaného DMA modulu nastavovány, a hodnot, které jim byly přidělovány. Velikost generovaných paketů a deskriptorů byla náhodně vybírána z rozsahu minimální až maximální nastavené velikosti. Důvodem, proč některé možné hodnoty nebyly vyzkoušeny (například počet DMA kanálů 64 a více), bylo snížení počtu možných kombinací nastavení. Při verifikaci nebyly z časových důvodů vyzkoušeny všechny možné varianty (dohromady 3610 možných variant). Místo toho bylo spuštěno několik desítek verifikačních běhů, z nichž v každém byly jednotlivé parametry nastaveny náhodně.

Kromě mnou vytvořené verifikace RX DMA modulu byla komponenta verifikovaná také po zapojení do obecného DMA modulu pomocí již existující verifikace této jadnotky. Tato verifikace byla již dříve používána pro testování jiných verzí DMA modulu, a mohla tak ověřit správnost zapojení nového RX DMA modulu a správnost mojí verifikace.

Po opravení řady chyb v implementaci bylo dosaženo správného fungování nového modulu ve verifikačním prostředí. Dalším krokem při ověření funkčnosti bylo testování v FPGA na síťové kartě, které je popsáno v následující kapitole.

Kapitola 6

Testování v FPGA

Aby byla zaručena funkčnost DMA modulu pro síťovou kartu, bylo u implementovaného a verifikovaného modulu provedeno testování v FPGA na síťové kartě zapojené do testovacího serverového stroje. K testování byla použita karta NFB-200G2QL. Na této kartě se nachází FPGA čip Virtex UltraScale+, který je cílovou platformou tohoto DMA modulu. Karta je připojena k počítači přes dvě PCIe Gen3 rozhraní, každé se 16 linkami, a obsahuje také dvě ethernetová rozhraní s rychostí 100 Gb/s. Díky tomu je karta schopna přenášet data ze sítě do RAM počítače a naopak až rychlostí 200 Gb/s. Pro dosažení této rychlosti s DMA modulem podporujícím rychlost 100 Gb/s je nutné do architektury v FPGA zapojit dva paralelně pracující DMA moduly, přičemž každý je připojen k jednomu PCIe Endpointu patřícímu k jednomu PCIe rozhraní. Při takovém zapojení má každý DMA modul svou vlastní sadu DMA kanálů a z pohledu softwaru je tak počet kanálů dvojnásobný oproti počtu nastaveném v jednom DMA modulu. Při testování modulu jsem používal pouze jeden DMA modul (teoretická maximální rychlost 100 Gb/s).

Aby při testování nedocházelo k problémům s časováním, byla architektura vytvářena v konfiguraci s 16 DMA kanály a s datovou šířkou 512 b. Pro ověření funkčnosti RX DMA modulu byl na jeho vstup posílán datový provoz a modul byl ovládán DPDK aplikací komunikující s ovladačem síťové karty. Pro vytváření datového provozu bylo použito několik metod:

1. generátor před RX
2. generátor za TX
3. generování nástrojem Spirent

V prvním případě byla architektura vytvořena s komponentou generátoru provozu připojenou na datový vstup RX DMA modulu. Druhý způsob spočíval v zapojení generátoru provozu na výstup z TX DMA modulu. Data z tohoto generátoru byla odesílána na ethernetové rozhraní a kabelem přivedena zpět na vstup do karty (*loopback*). Data následně procházela architekturou na čipu až k RX DMA modulu, který data odesílal do paměti. Při třetím způsobu byly datové pakety generovány pomocí speciálního nástroje pro generování síťového provozu Spirent určeného k testování síťových zařízení. Spirent generoval data na lince připojené na vstup do karty, podobně jako v případě 2.

Každý z těchto generátorů umožnil generovat síťové pakety o konkrétní délce. Architektura v FPGA umožňovala ze softwaru nastavit počet DMA kanálů, na které bude provoz distribuován.

Statistiky o průběhu přenosu dat bylo možno získat z několika různých míst datové cesty:

1. statistická jednotka v DMA modulu
2. MI registry RX DMA modulu
3. statistiky IBUFu
4. statistiky DPDK aplikace

Součástí DMA modulu je statistická jednotka, která zaznamenává počet přijatých a zahozených paketů v registrech, které jsou u připojené karty dostupné softwaru. Podobně lze přistoupit i k MI registrové sadě DMA modulu, kde můžeme sledovat změny HW a SW ukazatele při přenosech DPDK a čítače zahozených a přijatých paketů v DMA modulu. Komponenta IBUF se nachází za vstupním ethernetovým rozhraním karty. V případě, že RX DMA modul nestíhá přijímat data, naplní se v IBUFu buffer a jednotka začne zahazovat pakety. Čítače přijatých a zahozených paketů IBUFu lze rovněž číst ze softwaru. Poslední možností jsou statistiky přijímaných paketů DPDK aplikace, které udávají počet paketů úspěšně přijatých do paměti RAM.

Cílem testování je ověřit funkčnost a změřit propustnost RX DMA modulu na provozu obsahujícím pakety délek 64 B až 1516 B. Funkční testování je popsáno v sekci 6.1 a výsledky měření výkonnosti jsou uvedeny v sekci 6.2.

6.1 Funkční testování

Při funkčním testování nového RX DMA modulu v FPGA bylo cílem ověřit správné chování síťové karty při přijímání dat z ethernetového rozhraní. Správné fungování modulu lze poznat podle následujících znaků:

- Počet přijatých paketů v IBUFu roste.
- Počet zahozených paketů v IBUFu neroste, nebo roste podstatně pomaleji než počet přijatých paketů.
- Je-li DMA kanál zastavený:
  - Neroste počet přijatých paketů v MI registrech ani ve statistické jednotce.
  - Roste počet zahozených paketů v MI registrech a ve statistické jednotce.
- Je-li DMA kanál spuštěný:
  - Roste počet přijatých paketů v MI registrech a ve statistické jednotce.
  - Počet zahozených paketů v MI registrech a ve statistické jednotce neroste, případně roste podstatně pomaleji než počet přijatých paketů.
  - Roste počet přijatých paketů v DPDK aplikaci.

Toto chování bylo testováno pro délky paketů od 64 B do 1516 B pro generátor v RX a generátor v TX. U generování provozu nástrojem Spirent byly vyzkoušeny pouze některé vybrané délky.

Nejpravděpodobnějším projevem chyby při tomto testování je stav, kdy RX DMA modul přestane přijímat data a v IBUFu se následně začnou zahazovat všechny pakety. Takový stav může nastat, pokud v některé části zřetězené linky modulu dojde k trvalému nastavení signálu „destination ready“ na hodnotu 0. Příčinou může být trvalé naplnění některé z FIFO front nebo některého z bufferů. Tato chyba může být také způsobena nesprávným chováním jednotky Descriptor FIFO, kdy jednotka oznámí Accept processu, že má k dispozici dostatek deskriptorů na přijetí paketů, ale ve skutečnosti tyto deskriptory dostupné nejsou.

Další chybou, která může nastat, je nadměrné zahazování paketů v DMA modulu na kanálu, který je spuštěný. To nastane, pokud DMA modul není schopen dostatečně rychle číst volné deskriptory z paměti. Ve chvíli, kdy nemá DMA modul v Descriptor FIFO žádné použitelné deskriptory, pakety se zahazují. Takovou chybu lze vyřešit zvětšením objemu Descriptor FIFO, zvětšením descriptorového bufferu v paměti (pokud je příliš malý) nebo snížením hodnoty Timeout pro aktualizaci HW ukazatele (je-li příliš vysoká).

Po dosažení funkčnosti RX DMA modulu pro všechny testované paketové délky bylo možno přistoupit k měření výkonnosti modulu při přenosu paketů.

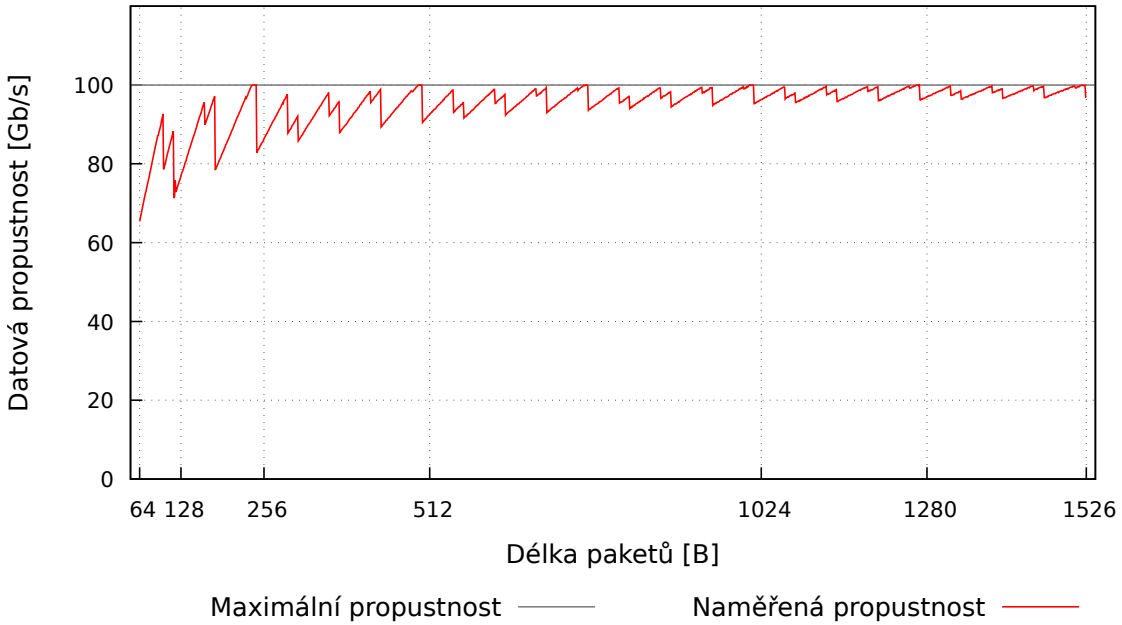
6.2 Výkonnostní testování

Hlavní metrikou pro RX DMA modul je kromě počtu podporovaných DMA kanálů také rychlost, s jakou dokáže řídit přenášení dat z FPGA do RAM. Tato rychlost je měřena v jednotkách gigabit za sekundu (Gb/s) a vyjadřuje počet bitů dat paketů přicházejících ze sítě úspěšně přenesených do RAM za jednu sekundu. V případě, že je DMA kanál v modulu zastaven, lze měřit rychlost, s jakou DMA modul přijímá data ze svého vstupu a následně je zahazuje (pokud nedokáže modul zahazovat data dosti rychle, jedná se také o neefektivitu).

Při přenosu dat v systému DPDK komunikuje DMA modul se softwarovou DPDK aplikací, která mu poskytuje deskriptory. Z toho důvodu může dojít ke zpomalení přenosu, pokud tato aplikace nepracuje dostatečně rychle. Rychlost zpracování dat aplikací závisí na složitosti operací, které provádí nad přijímanými pakety. Protože ale v softwaru může být každý DMA kanál zpracováván jiným jádrem CPU, lze nedostatečnou rychlost aplikace kompenzovat zvýšením počtu kanálů, na které je datový provoz distribuován. V našem případě byla použita jednoduchá aplikace, která po přijetí dat do RAM pouze přeskočila data nově přijatých paketů až k poslednímu nově použitému deskriptoru, přičemž četla velikost jednotlivých paketů a následně aktualizovala SW ukazatel. Ke snížení propustnosti vlivem softwaru proto docházelo pouze při použití méně než 4 DMA kanálů. (Při spouštění přenosu nemusí být počet použitých DMA kanálů mocninou 2, pouze nesmí být větší než počet kanálů podporovaný DMA modulem v daném nastavení.) Výkonnost DMA modulu byla měřena při spuštění přenosů na 16 DMA kanálech.

Jak bylo řečeno na začátku této kapitoly, teoretická rychlost příjmu dat na vstupu do DMA modulu je 102,4 Gb/s (při datové šířce 512 b a frekvenci 200 MHz). V těchto datech jsou započteny i hlavičky síťových paketů, protože i ty jsou přenášeny z FPGA do RAM. Této rychlosti lze ale dosáhnout pouze při generování provozu generátorem na vstupu do RX, který generuje na vstup datová slova plně obsazená pakety. Při příjmu dat z ethernetového rozhraní je maximální propustnost omezena rychlostí Ethernetu na 100 Gb/s. Pokud při provozu přicházejícím z Ethernetu nejsou v IBUFu ani v DMA modulu zahazovány žádné pakety, pak lze tvrdit, že DMA modul přenáší data do RAM rychlostí 100 Gb/s. Výslednou propustnost v konkrétním časovém intervalu lze vypočítat jako $100 * (P_{gen} - P_{IBUFdis} - P_{DMAdis})/P_{gen}$, kde P_{gen} je počet vygenerovaných paketů, $P_{IBUFdis}$ je počet paketů zahozených v IBUFu a P_{DMAdis} je počet paketů zahozených

v DMA modulu. Pokud bychom počítali rychlost přijímání paketů na vstupu do DMA modulu (pakety zahozené v DMA modulu jsou započteny jako přijaté do DMA modulu), vypočítala by se propustnost jako $100 * (P_{gen} - P_{IBUFdis})/P_{gen}$.



Obrázek 6.1: Graf naměřené datové propustnosti na paketových délkách 64 B až 1526 B.

Graf 6.1 ukazuje výsledek měření propustnosti při příjmu paketů s konkrétními délkami. Vyzkoušeny byly všechny délky v rozmezí od 64 B do 1526 B (včetně 64 a 1526). Pro generování paketů byl v tomto případě použit gerátor umístěný za TX. Tento graf zobrazuje propustnost v gigabitech za sekundu v porovnání s maximální propustností 100 Gb/s v celém rozmezí délek 64 B až 1526 B.

V grafu lze sledovat periodické propady na násobcích 64 B, z nichž první je na délce 117 B. Tyto propady jsou zapříčiněny způsobem, jakým jsou datové transakce odesílány na DMA sběrnici. V jednom taktu hodinového signálu je odeslána část jedné zápisové transakce o velikosti maximálně 64 B. Pokud není velikost transakce násobkem 64 B, pak při odesílání poslední části dochází k neefektivnímu přenosu. U každého paketu, který je přijat do karty, jsou nejprve odebrány 4 bajty CRC a následně přidána hlavička o velikosti 16 bajtů. Když tedy paket vstoupí do DMA modulu, je o 12 bajtů větší. Pokud se podíváme na délku 117 B, můžeme zjistit, že takový paket bude na DMA sběrnici odeslán ve třech částech ($117 + 12 = 129 = 64 + 64 + 1$). Při přenosu těchto paketů je tedy v každém třetím taktu odeslán pouze jeden bajt dat, čímž se teoretická maximální propustnost snižuje z 102,4 Gb/s na pouhých 68,8 Gb/s. Tuto neefektivitu by bylo možno vyřešit, pokud by DMA modul nepoužíval DMA sběrnici. Pokud by posílal transakce na sběrnici PCIe tak, aby v případě končící transakce poslal zároveň začátek příští transakce, pak by pokaždé naplno využil 64 B své výstupní datové šířky. Menší propady v průběhu grafu jsou způsobeny přenosem dat přes více sběrnic a rozhraní, které k datům přidávají ruzně veliké hlavičky a běží na různých frekvencích a datových šířkách.

V případech, kdy je délka paketů přicházejících do DMA modulu již téměř rovna násobku 64 B, pak podle grafu dosahuje DMA modul plné propustnosti 100 Gb/s. V těchto

případech se nezahazují žádné z příchozích paketů a všechny jsou úspěšně přijaty aplikací v softwaru.

Z výsledků měření propustnosti lze tedy vyvodit, že DMA modul je podle předpokladů schopen dosáhnou předpokládané datové propustnosti 100 Gb/s pro některé z testovaných délek. Na většině délek je ale jeho propustnost omezována neefektivním využitím rozhraní pro přenos dat na sběrnici PCIe. Toto omezení plánuji odstranit nahrazením používané DMA sběrnice za jiný druh rozhraní, při kterém nebude DMA modul omezen na vyslání části pouze jedné zápisové transakce. Tato úprava bude rovněž vyžadovat změnu práce s komponentou PCIe buffer (zobrazen na diagramu návrhu modulu 4.1), neboť v současnosti jsou data do bufferu ukládána separátně pro každou transakci.

Kapitola 7

Závěr

Cílem práce bylo navržení, implementace a otestování firmwarového modulu pro FPGA, který provádí vysokorychlostní DMA přenosy z FPGA do paměti počítače přes sběrnici PCI-Express Gen3 v systému DPDK.

Po nastudování technologií jsem sestavil seznam základních požadavků pro správně navržený DMA modul, podle kterých bylo později možno zhodnotit kvalitu tohoto návrhu. Samotný návrh modulu sestával z obecného návrhu celého modulu, z návrhu jednotlivých částí a z identifikace a optimalizace kritických částí souvisejících především s podporou vysokého počtu DMA kanálů. Následně proběhla implementace modulu, po jejímž dokončení mohly být zjištěny zdroje obsazené DMA modulem po syntéze pro cílové FPGA. Výsledky syntézy ukázaly, že i pro 256 DMA kanálů byl samotný modul schopen pracovat na požadované frekvenci 200 MHz a využití zdrojů se pohybovalo kolem 10 % s vyjímkou BRAM pamětí, jejichž využití vzrostlo přibližně na 40 %. Po celkové verifikaci a otestování funkčnosti na skutečném zařízení, byl modul podroben testu propustnosti při příjmu paketů o délkách 64 B až 1526 B. Měření ověřilo schopnost DMA modulu přijímat pakety učitých délek na plné propustnosti při konfiguraci na 100 Gb/s. Protože ale byla propustnost mírně omezována u většiny paketových délek, byly navrženy možné úpravy pro její zlepšení.

Výsledkem práce je DMA modul vyhovující požadovaným parametrům přenosu s rychlostí 100 Gb/s. Při použití dvou paralelních modulů v jedné architektuře lze ve stejné konfiguraci přenášet data na rychlosti 200 Gb/s. Po provedení několika chystaných úprav za účelem zvýšení efektivity a zlepšení spolehlivosti bude dalším krokem ve vývoji optimalizace pro dosažení propustnosti 400 Gb/s. Této propustnosti bude moci DMA modul dosáhnout zapojením dvou modulů, kde každý bude pracovat s dvojnásobnou datovou šířkou. Na zvětšení datové šířky je modul již připraven, ale bude třeba zajistit jeho schopnost zpracovávat tato data na požadované frekvenci. V současné době ještě také není dostupná síťová karta s FPGA, která by takovouto propustnost umožňovala.

Na základě této práce byl vytvořen článek, který byl prezentován na Studentské Konferenci Inovací, Technologií a Vědy v IT s názvem Excel@FIT 2018. Na této konferenci získala ocenění odborného odborným panelem za efektivní řešení komplexní problematiky datových přenosů [12].

Literatura

[1] AXI Reference Guide.
URL https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

[2] AXI Reference Guide.
URL https://www.xilinx.com/support/documentation/ip_documentation/axi_interconnect/v2_1/pg059-axi-interconnect.pdf

[3] Data Plane Development Kit Documentation.
URL https://dpdk.org/doc

[4] UltraScale Architecture Configurable Logic Block.
URL https://www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf

[5] UltraScale Architecture Memory Resources.
URL https://www.xilinx.com/support/documentation/user_guides/ug573-ultrascale-memory-resources.pdf

[6] UltraScale Devices Gen3 Integrated Block for PCI Express v4.4.
URL https://www.xilinx.com/support/documentation/user_guides/ug576-ultrascale-gth-transceivers.pdf

[7] UltraScale+ Devices Integrated Block for PCI Express v1.3.
URL https://www.xilinx.com/support/documentation/ip_documentation/pcie4_uscale_plus/v1_3/pg213-pcie4-ultrascale-plus.pdf

[8] UltraScale Plus FPGA Product Tables and Product Selection Guide.
URL https://www.xilinx.com/support/documentation/selection-guides/ultrascale-plus-fpga-product-selection-guide.pdf

[9] IEEE Standard VHDL Language Reference Manual. *IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002)*, Jan 2009: s. c1–626, doi:10.1109/IEEESTD.2009.4772740.

[10] IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language. *IEEE Std 1800-2012 (Revision of IEEE Std 1800-2009)*, Feb 2013: s. 1–1315, doi:10.1109/IEEESTD.2013.6469140.

[11] Einspruch, N.: *Application Specific Integrated Circuit (ASIC) Technology*. VLSI Electronics, Elsevier Science, 2012, ISBN 978-03-231-5323-2.

[12] Kubálek, J.: High-speed DMA packet transfer in system DPDK.
URL http://excel.fit.vutbr.cz/submissions/2018/012/12.pdf

[13] Lawley, J.: White Paper: UltraScale and Virtex-7 FPGAs, Understanding Performance of PCI Express Systems. Technická Zpráva WP350 (v1.2), October 2014.