VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

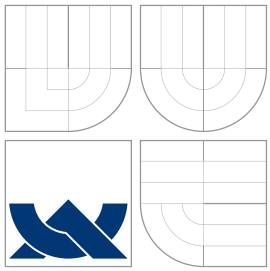
SYSTÉM INTERNÍCH SBĚRNIC PRO ČIPY S TECHNOLOGIÍ FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

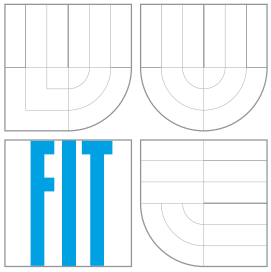
AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ MÁLEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SYSTÉM INTERNÍCH SBĚRNIC PRO ČIPY S TECHNOLOGIÍ FPGA

SYSTEM OF INTERNAL BUSES FOR CHIPS WITH FPGA TECHNOLOGY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. TOMÁŠ MÁLEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ MARTÍNEK
SUPERVISOR

BRNO 2008

Abstrakt

Tato práce se zabývá návrhem a implementací propojovacího systému interních sběrnic pro čipy s technologií FPGA. Systém zajišťuje jak komunikaci mezi interními komponentami v rámci čipu, tak jejich komunikaci s ostatními prvky počítače, které jsou namapovány do paměti hostitelského systému. Architektura vysokorychlostní plně duplexní paketově-orientované sběrnice vychází ze stromové topologie. Šířka datové sběrnice je genericky nastavitelná, individuálně pro každou její část. Díky tomu je možné vybudovat jednotný hierarchický systém sběrnic na různých rychlostech, který propojuje různě rychlé komponenty. Navržený propojovací systém byl implementován v jazyce VHDL a je nasazen v projektu Liberouter, který je součástí výzkumného záměru CESNETu Programovatelný hardware.

Klíčová slova

Sběrnice, propojovací systém, PCI, PCI-X, PCI Express, FPGA, VHDL

Abstract

This thesis deals with design and implementation of interconnection bus system for chips with FPGA technology. The system ensures both communication between internal components on a chip and their communication with other computer elements which are mapped to the host system memory. The buses are high-speed, full duplex and packet-oriented and their architecture is based on tree topology. The data width is configurable, individually for every bus part. Due to this feature, it is possible to build uniform hierarchical system of internal buses with different speed that interconnects differently fast components. Proposed interconnection system was implemented in VHDL language and it is utilized in the Liberouter project which is the part of CESNET research intention Programable Hardware.

Keywords

Bus, interconnection system, PCI, PCI-X, PCI Express, FPGA, VHDL

Citace

Tomáš Málek: Systém interních sběrnic pro čipy s technologií FPGA, diplomová práce, Brno, FIT VUT v Brně, 2008

Systém interních sběrnic pro čipy s technologií FPGA

Prohlášení

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

.....
Tomáš Málek
2. května 2008

Poděkování

Především bych rád poděkoval vedoucímu své diplomové práce panu Ing. Tomáši Martínkovi za odborné vedení a čas věnovaný konzultacím. Také bych chtěl poděkovat kolegům z projektu Liberouter za poskytnutí informací a zajištění technické podpory.

Obsah

Kapitola 1

Úvod

V minulém desetiletí nastal prudký rozvoj v oblasti informačních technologií. Výpočetní technika začala být stále častěji nasazována do nejrůznějších sfér lidské činnosti, aby přispívala k efektivnějšímu zpracování informací a pomáhala při řešení nejrůznějších problémů. Počítače byly chápány především jako obecný výpočetní prostředek.

S rozšiřováním počítačů však souvisí i postupné zvyšování výkonnostních požadavků, které jsou na ně kladeny. Řešení založená na platformě PC začínají v některých případech narážet na značná omezení způsobená především limitující propustností sběrnic a nedostatečným výkonem univerzálního procesoru. Týká se to například oblastí počítačových sítí, počítačové grafiky či vědeckých výpočtů. Zde se nabízí řešení přesunout implementaci takovýchto počítačových systémů, celých nebo v případě hardwarové akcelerace pouze jejich částí, na úroveň aplikačně specifických obvodů (ASIC) [14] nebo programovatelných hradlových polí (FPGA) [22]. Pro vybrané problémy pak obě technologie poskytují potencionálně vyšší výkonnost než obecné procesory.

Technologie ASIC (Application specific integrated circuit) představuje integrovaný obvod speciálně navržený pro konkrétní aplikaci a jeho struktura je napevno vytvořena již při výrobě. Řešení založená na této technologii umožňují dosahovat vysokou výkonnost, mají nižší spotřebu elektrické energie a při velkém počtu vyrobených kusů i nízkou cenu. Na druhou stranu je však pro výrobu takového obvodu nutné navrhnout a vytvořit tzv. masku, jejíž cena je poměrně vysoká. Každá změna návrhu tedy přináší další nezanedbatelné náklady.

Oproti tomu obsahuje FPGA (Field Programable gate array) pole programovatelných logických buněk a vestavěných bloků, jejichž konfiguraci a tím i funkci je možné měnit. Díky rekonfiguraci je tato technologie flexibilnější a nabízí jistý kompromis mezi výkonem obvodů ASIC a flexibilitou univerzálních procesorů. Uplatnění FPGA čipů narůstá zejména v oblasti komunikací a průmyslu a jejich podíl na trhu se neustále zvyšuje.

S rostoucí složitostí výpočetního systému na libovolné úrovni se zvyšuje i nutnost navrhnout takový systém dostatečně modulárně a pokud možno co nejobecněji, aby byl dobře spravovatelný a udržovatelný. Jednotlivé moduly si pak mezi sebou definovaným způsobem předávají data a musí tak být vzájemně propojeny. To platí i pro systémy, které jsou postavené na platformě FPGA. Komunikovat mezi sebou musí jednak komponenty, které jsou na čipu, a v případě, že je FPGA hradlové pole součástí zařízení připojeného například systémovou sběrnicí k PC, musí být zajištěna i komunikace s procesorem a ostatními prvky počítače.

Důležitou součástí každého systému implementovaného na čipu FPGA je propojovací sběrnicový systém. Vzhledem k tomu, že propustnost sběrnice bývá často omezujícím faktorem, je nutné, aby byla dostatečně rychlá. Zároveň je žádoucí, aby celý propojovací systém

byl kompaktní a nezabíral příliš místa na čipu na úkor vlastní implementované aplikace. Další požadavky kladené na sběrnicové systémy přicházejí ze strany připojovaných uživatelských komponent. Každá implementovaná aplikace se obecně může skládat z mnoha takovýchto komponent, a proto se jejich nároky na vlastnosti a možnosti propojení často výrazně liší.

V posledních dvaceti letech bylo vyvinuto několik sběrnicových systémů sloužících k propojení jednotlivých komponent v rámci čipu FPGA. Některé z nich vynikají svými výkonnostními parametry. Kvůli vysokým frekvencím a rychlostem však nezvládají pokrýt všechny funkční požadavky, zabírají příliš mnoho místa na čipu a jejich implementace je často velmi složitá. Naproti tomu existují i systémy, kterou jsou relativně jednoduché, kompaktní a nabízejí uživateli širokou škálu možností. Mezi jejich nedostatky však obvykle patří např. nízká propustnost.

Tato práce se zabývá návrhem a implementací propojovacího systému interních sběrnic pro čipy s technologií FPGA, jehož cílem je vyváženým způsobem pokrýt typické funkční a výkonnostní požadavky. Systém zajišťuje jak komunikaci mezi interními komponentami v rámci čipu, tak jejich komunikaci s ostatními prvky počítače, které jsou namapovány do paměti hostitelského systému. Architektura vysokorychlostní plně duplexní paketově-orientované sběrnice vychází ze stromové topologie. Šířka datové sběrnice je genericky nastavitelná, individuálně pro každou její část. Díky tomu je možné vybudovat jednotný hierarchický systém sběrnic na různých rychlostech, který propojuje různě rychlé komponenty na čipu. Propojovací systém je vyvíjen v rámci projektu Liberouter, který je součástí výzkumného záměru CESNETu Programovatelný hardware [16]. Celý systém je implementován v jazyce VHDL [10], konkrétním hardwarovým prostředkem pro realizaci návrhu jsou karty COMBO6X z rodiny karet COMBO [11] a ML555 [9].

Dokument je logicky členěn do několika částí. V úvodu je čtenář zasvěcen do základní problematiky, kterou se práce zabývá. Podrobnější teoretický rozbor propojovacích systémů, jejich principů a problémů, které jsou s nimi spojené, následuje v kapitole 2. Kapitola 3 popisuje existující sběrnicové systémy a je poukázáno na jejich nedostatky. Rozbor samotného navrženého systému interních sběrnic je zahrnut v kapitole 4. Detailním způsobem je popsána například jeho topologie, komunikační protokol nebo architektura jednotlivých komponent. Kapitola 5 shrnuje dosažené výsledky a v závěru jsou rekapitulovány důležité vlastnosti navrženého systému a jsou zmíněny další možnosti pokračování projektu.

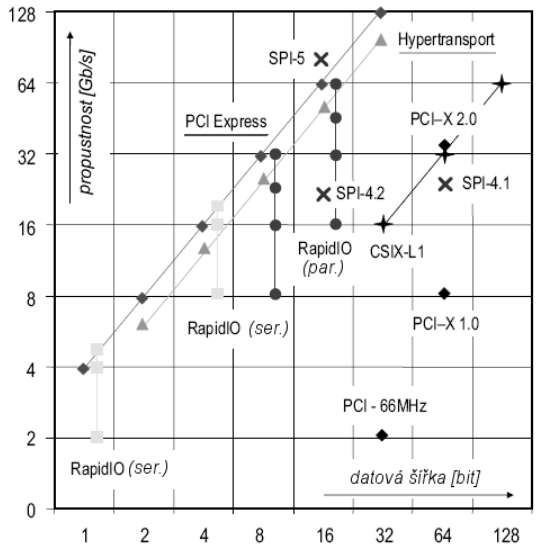
Kapitola 2

Propojovací systémy

Sběrnicí se v oblasti počítačových architektur rozumí systém, který slouží k přenosu dat mezi komponentami nebo mezi samotnými počítači. Logické propojení několika zařízení na stejné úrovni je realizováno pomocí jedné množiny vodičů a je vytvořen celistvý systém, v rámci kterého spolu mohou komunikovat libovolné dva jeho prvky.

Se sběrnicemi je možné se setkat na mnoha úrovních hierarchie počítačového systému. Opomeneme-li propojení samotných počítačů, můžeme na nejvyšší úrovni identifikovat tzv. systémovou sběrnici, která slouží k propojení hlavních komponent. Vedle ní existuje i řada dalších sběrnic, na které je systémová sběrnice konvertována a které slouží k připojení přídavných adaptérů, vstup-výstupních zařízení atd. Sběrnice jsou rovněž potřeba i na nižších úrovních v rámci jednotlivých adaptérů.

Vývoj sběrnic je vždy úzce spojen s vývojem komponent, které jsou pomocí nich propojeny. S rostoucí propustností pamětí a výkonností procesoru a dalších komponent souvisí i nutnost zlepšovat parametry sběrnic tak, aby se nestaly úzkým hrdlem celého systému. Obrázek 2.1 znázorňuje vývoj propustností vybraných typů sběrnic v závislosti na jejich datové šířce [2].



Obrázek 2.1: Propustnost různých typů sběrnic v závislosti na datové šířce

Z obrázku můžeme vysledovat, že nejvyšších propustností dosahují sériové sběrnice a to i při nižší datové šířce (oproti paralelním sběrnicím). To vystihuje i trend postupného přechodu od přenosů po paralelních vodičích (např. sběrnice ISA, EISA, PCI nebo PCI-X) k bitově orientovaným přenosům po škálovatelných sériových linkách (PCI Express, HyperTransport, RapidIO a další).

V rámci této práce se budeme zabývat pouze propojovacími systémy na úrovni jednotlivých adaptérů, které obsahují čip s programovatelným hradlovým polem (FPGA). Tato kapitola popisuje různé principy, algoritmy a schémata, které se v takových systémech uplatňují. Jejich výběr úzce souvisí s vlastnostmi a požadavky uživatelských komponent. Mezi ty nejdůležitější patří:

- propustnost a s ní související pracovní frekvence,
- šířka sběrnice,
- latence čtecích operací,
- možnost iniciovat datové přenosy nebo
- podpora specifických vlastností (např. zarovnání dat podle adresy, kontrola pořadí čtecích a zápisových operací atd).

Neméně podstatné jsou i vlastnosti a požadavky kladené na propojovací systémy jako na celek. Jedná se zejména o:

- množství spotřebovaných zdrojů,
- škálovatelnost (např. počet připojitelných komponent),
- platformovou nezávislost a přenositelnost,
- otevřenost k případným rozšířením nebo
- modularitu a udržovatelnost.

Existuje mnoho přístupů, jak dosáhnout požadovaných vlastností. K typickým otázkám, které je při návrhu propojovacího systému nutné řešit, patří výběr vhodné sběrnicové topologie, arbitračního schématu a komunikačního protokolu, podporované typy přenosů a transakcí, problematika související s pořadím čtecích a zápisových transakcí, řešení citlivosti na vzdálenost, různé chybové stavy a další.

2.1 Topologie sběrnicových systémů

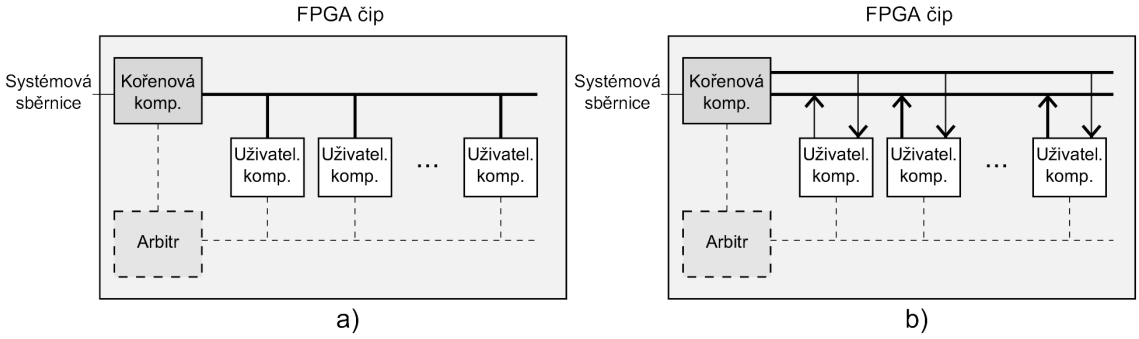
V kontextu propojovacích a sběrnicových systémů charakterizuje topologie (prostorové uspořádání) způsob, jakým jsou mezi sebou jednotlivé elementy propojeny. Je to vlastnost systému, která má významný vliv na to, jak bude systém výkonný, jaké bude mít parametry, podporované funkční rysy atd.

S topologií se setkáváme na všech úrovních propojovacích systémů, od počítačů až po elementární komponenty v rámci jednotlivých adaptérů. V praxi se můžeme setkat s mnoha obměnami architektur založených například na klasických sběrnicích, sítích či směrování a přepínání. Dále budou podrobněji rozebrány čtyři konkrétní sběrnicové architektury, které jsou díky svým vlastnostem použitelné pro cílovou technologii FPGA.

2.1.1 Sběrnicová architektura

Sběrnicová architektura představuje základní topologii, která určuje způsob propojení jednotlivých komponent. S rostoucími požadavky na propojovací systémy se klasická sběrnicová topologie postupně vyvíjela a různými transformacemi a modifikacemi upravovala do různých podob. Některé změny byly natolik významné, že vznikly nové typy topologií [13].

Na obrázku 2.2a je znázorněna základní architektura klasické sběrnicové topologie. Důležitou jednotkou je tzv. kořenová komponenta, která vytváří most mezi rozhraními systémové a interní sběrnice. Arbitrační komponenta (plánovač) slouží k řízení datových přenosů, avšak pouze v případě centralizovaného způsobu přidělování sběrnice. Proto je ohraničena přerušovanou čarou. Další schémata přidělování sběrnice jsou podrobněji popsána v kapitole 2.2.



Obrázek 2.2: Sběrnicová architektura:
(a) klasická varianta, (b) upravený model se dvěma duplexními směry

Jednou z možností, jak upravit základní model sběrnicové architektury, je vytvoření dvou duplexních směrů (viz obrázek 2.2b):

- směr z kořene do vnitřní části systému (tzv. downstream linka) a
- směr z vnitřní části systému ke kořenu (tzv. upstream linka).

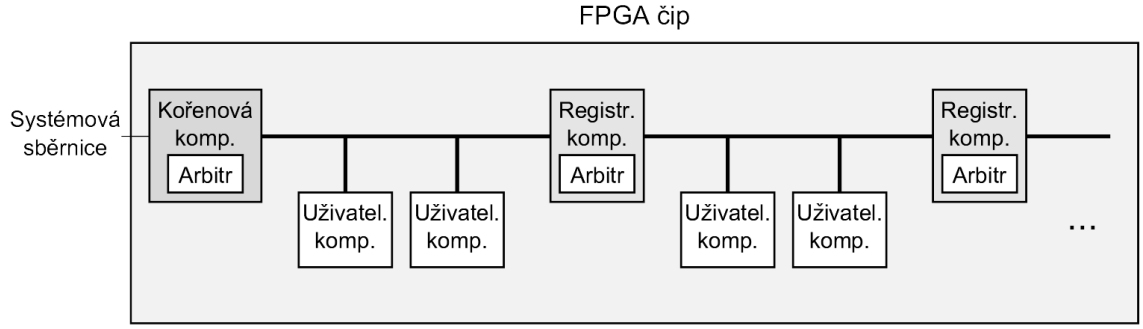
Po downstream lince přicházejí pouze požadavky ze strany kořenové komponenty. Tato linka je tak určena pouze pro shora iniciované čtecí a zápisové transakce a jednotlivé komponenty na ni nemohou vystavovat svá data.

Upstream linka naopak slouží uživatelským komponentám k vystavování dat, která jsou určená k zaslání mimo čip FPGA. Jedná se buď o odpovědi na čtecí transakce nebo o přenosy iniciované ze strany komponent (tzv. master přenosy). Přípojné místo na upstream linku je realizováno pomocí logického hradla OR. Tento druh spoje je velice jednoduchý, rychlý a vyvarovává se použití tří-stavových sběrnic. Provádění arbitrace při přístupu ke sběrnici je nutné pouze pro upstream linku, protože přenosy po downstream lince iniciuje pouze kořenová komponenta.

Doposud jsme neuvažovali variantu, kdy je možná i vzájemná komunikace jednotlivých interních komponent na čipu. Realizace takového případu klade na funkci arbitra další nároky a hraje jednu z důležitých rolí při výběru vhodného arbitračního schématu.

Dalším problémem, který je potřeba řešit v souvislosti s cílovou technologií FPGA, je tzv. citlivost na vzdálenost. Touto problematikou se podrobněji zabývá kapitola 2.6. V zásadě jde o to, že s rostoucí datovou šířkou sběrnice je obtížnější dosáhnout vysoké pracovní

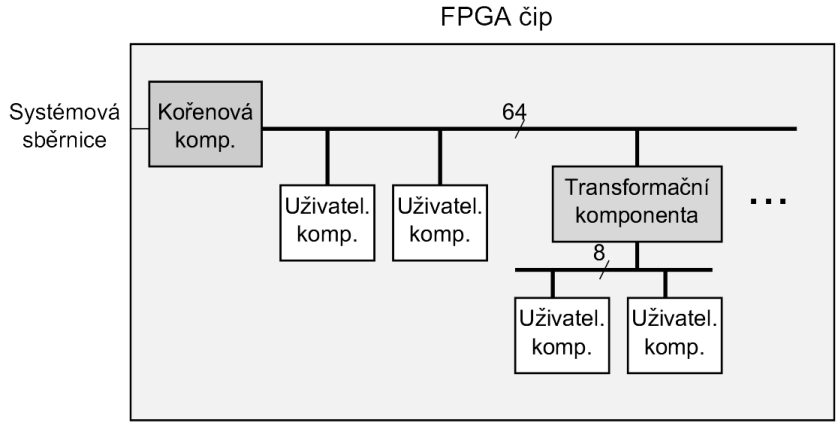
frekvence a tím i propustnosti. Možným řešením je zasadit do sběrnicové architektury speciální registrovací komponenty, které umožní rozdělení připojených komponent do několika stupňů. Tím se problémy spojené s délkou vodičů eliminují.



Obrázek 2.3: Registrovaná sběrnicová architektura

Tuto úpravu sběrnicové topologie znázorňuje obrázek 2.3. Pro každý stupeň zahrnující několik komponent musí být prováděna arbitrace. Příslušná logika je většinou zabudována v jednotlivých registrovacích komponentách a v případě prvního stupně přímo v kořenu. Takovéto řešení je na rozdíl od předchozích variant dobře škálovatelné a umožňuje připojení libovolného počtu komponent, aniž by byla omezena pracovní frekvence.

Speciálním případem uvedené úpravy je situace, kdy se v každém stupni sběrnice nachází pouze jedna uživatelská komponenta. V takovém případě odpadá arbitrace jednotlivých komponent a příslušná logika může být redukována. Navíc je možné, aby mohly být datové přenosy inicovány i ze strany uživatelských komponent, což je rovněž velmi důležitou vlastností propojovacího systému.



Obrázek 2.4: Sběrnicová architektura modifikovaná přidáním transformačních komponent

Další vlastností, se kterou se základní sběrnicová topologie není schopná efektivně vypořádat, jsou různé požadavky jednotlivých připojených komponent na propustnost. Např. řadič dynamické paměti obvykle vyžaduje vysokou propustnost pro časté čtecí a zápisové operace z nebo do externí paměti, zatímco jiné jednodušší komponenty potřebují jednou za čas vyčíst nebo zapsat hodnotu stavových a řídících registrů. Pokud by byly komponenty připojeny k jedné sběrnici, musely by respektovat datovou šířku příslušící nejvyšší poža-

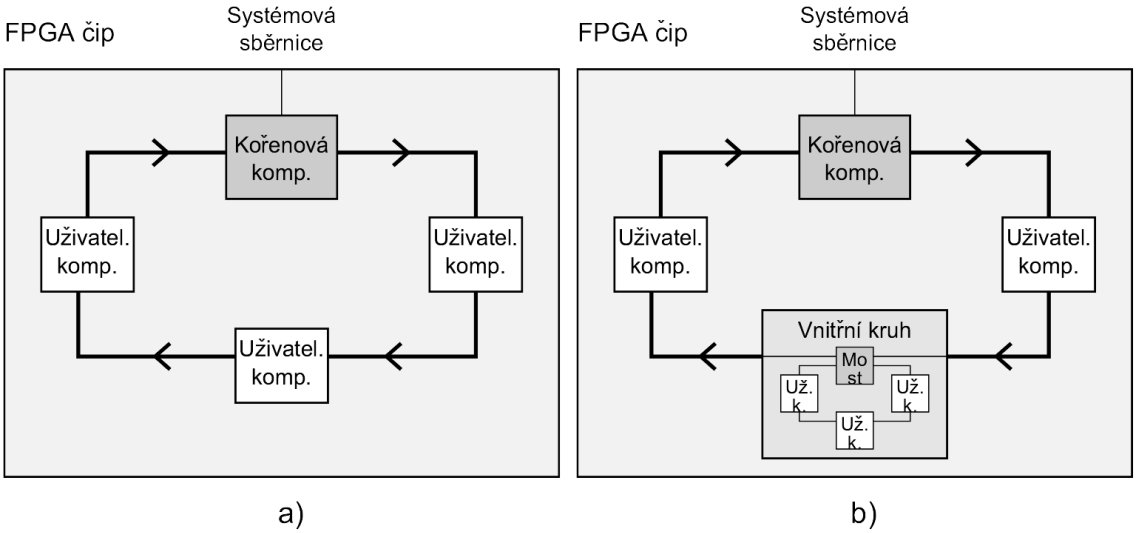
dované propustnosti. Vzhledem k tomu, že šířka sběrnice má vliv na složitost adresových dekodérů a další logiky, zabíraly by jednoduché komponenty neúměrné množství zdrojů.

Do propojovacího systému lze proto integrovat speciální transformační komponentu, která konvertuje šířku sběrnice. Díky tomu je možné vytvořit celou hierarchii sběrnic s různými šířkami, které by vyhovovaly všem připojeným komponentám. Příklad takto upraveného systému je znázorněn na obrázku 2.4.

Jak jsme si v této kapitole ukázali, existují různé způsoby, jakými lze klasickou sběrnicovou topologii upravit a zlepšit tak její výkonností a funkční parametry. Některé z nich vedou dokonce ke vznikům nových typů topologií. Příkladem může být použití transformační komponenty, které vede k vytvoření stromové hierarchie. Lze tak říci, že klasická sběrnicová architektura představuje základ pro většinu dalších typů topologií.

2.1.2 Kruhová architektura

Vzájemným propojením uživatelských komponent do kruhu vytvoříme tzv. kruhovou topologii. Základní schéma takové architektury je zobrazeno na obrázku 2.5a. Každá komponenta je jednosměrnou sběrnicí spojena se dvěma sousedy, s nimiž může komunikovat přímo. Komunikace s ostatními členy kruhu probíhá zprostředkovaně přes ostatní komponenty, kterými musí přenášená data projít.



Obrázek 2.5: Kruhová architektura: (a) základní schéma, (b) přidání vnitřního kruhu

Každá připojená komponenta má v rámci systému přiřazený jedinečný identifikátor. Data, která jsou vyslána do kruhu, obsahují mimo jiné i zdrojové a cílové identifikační pole. Při příchodu transakce do komponenty se pak na základě těchto polí rozhoduje, zda budou data přeposlána dál nebo budou vyňata z kruhu a zpracována cílovou komponentou.

Jednou z komponent zapojených do kruhu je také kořenová komponenta, která tvoří rozhraní mezi interním sběrnicovým systémem a sběrnicí hostitelského počítače. Tato komponenta data nezpracovává, ale automaticky je přeposílá buď směrem mimo čip FPGA nebo dovnitř do kruhu.

Uspořádání komponent kruhové topologie umožňuje vzájemnou komunikaci jednotlivých interních komponent. V případě, že je vyžadován pouze tento typ komunikace, může

být zapojení topologie realizováno i bez kořenového prvku [19]. Kruhová architektura umožňuje jak přenosy iniciované ze strany nadřazeného systému, tak master přenosy, kterými interní komponenty komunikují s prvky umístěnými mimo čip FPGA. Díky přepínací logice, která je součástí každého přípojného místa, není nutná přítomnost žádného centrálního arbitru. V některých kruhových topologiích se přidává i protiběžný okruh.

Nevýhodou této topologie je, že každá transakce zatěžuje zbytečně několik komponent, přes které musí projít, než dojde ke svému cíli. Krajním případem je situace, kdy data přichází z nadřazeného systému do kořenového prvku a směřují do komponenty, která je hned vedle vlevo od něj. Transakce pak prochází přes všechny zbylé interní komponenty a nadbytečně je zatěžuje. Řešením této situace je vytvoření dalšího vnitřního kruhu místo jedné z uživatelských komponent. Do něj se umístí jednotky, pro které není důležitá rychlá odezva a vysoká propustnost. V hlavním kruhu tak zůstane nezbytně nutný počet komponent, které vyžadují vysokou výkonnost.

Obrázek 2.5b znázorňuje začlenění vnitřního kruhu do základní kruhové topologie. Využitím speciálního mostu je možné vytvářet celé hierarchie kruhů. V případě, že most umí transformovat i šířku sběrnice, může být vytvořena hierarchie různě rychlých sběrnic.

Jednou z výhod kruhové topologie je také to, že se u ní neprojevuje citlivost na vzdálenost, protože sběrnice je registrována postupně díky každé uživatelské komponentě. Mezi nevýhody patří například obtížnost při detekci chyby ve sběrnicovém systému. To je dáno tím, že v případě zhroucení jednoho uzlu je zablokována celá sběrnice (u jiných topologií se může zablokovat pouze část sběrnice).

2.1.3 Stromová architektura

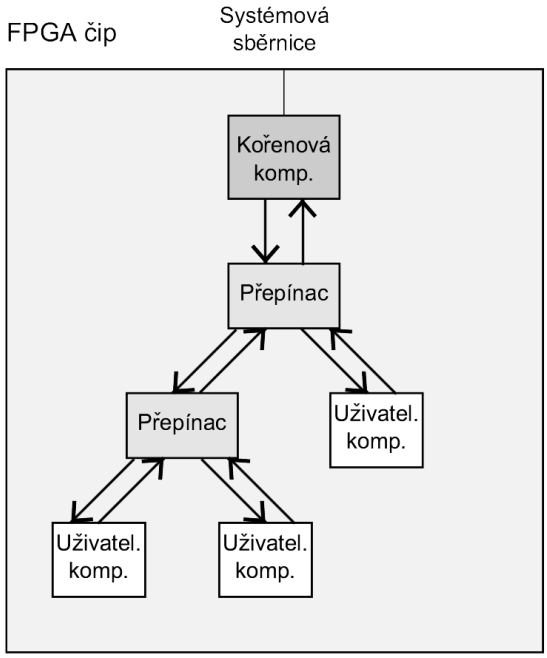
Stromová architektura je jedna z možných zapojení tzv. přepínané topologie, která je založena na propojení jednotlivých zařízení pomocí přepínačů. Podobnou topologii má i sběrnice PCI Express [15], která je jedním z potencionálních nadřazených sběrnicových systémů, ke kterým je čip FPGA připojen. Jak je patrné z obrázku 2.6, sběrnice je tvořena dvěma duplexními směry, tzv. downstream a upstream linkou. Všechny prvky jsou pak těmito přímými (point-to-point) linkami spojeny. V kontextu stromové struktury rozlišujeme v rámci topologie tři typy stromových uzlů:

- kořen,
- vnitřní uzly a
- koncové uzly (listy).

Kořenová komponenta převádí rozhraní sběrnice hostitelského počítače na rozhraní pro interní sběrnicový systém. Pokud jsou v systému zapojeny více než dvě komponenty (listy), potom jsou odděleny pomocí 3-portového přepínače (vnitřní uzel).

Úlohou přepínače je směrovat tok dat mezi jeho třemi porty. K tomu není potřeba žádného centrální arbitra. Směrování nahrazuje proces arbitrace u klasických sběrnic a je možné je provést pouze na základě cílové adresy. Zapojení přepínače umožňuje realizovat komunikaci mezi libovolnými dvěma uzly v systému, přičemž lokální komunikace neovlivní propustnost v jiných částech sběrnice.

Další možnou funkcí přepínače je transformace datové šířky sběrnice. V případě, že tuto funkcionalitu podporuje, může libovolný přepínač představovat kořen nového podstromu s rozdílnou datovou šířkou. Stejně jako u klasické sběrnicové a kruhové topologie je



Obrázek 2.6: Stromová architektura

pak možné vybudovat stromovou hierarchii různě rychlý sběrnic. V neposlední řadě tvoří přepínač prvek, který registruje interní sběrnici a snižuje tak citlivost na vzdálenost.

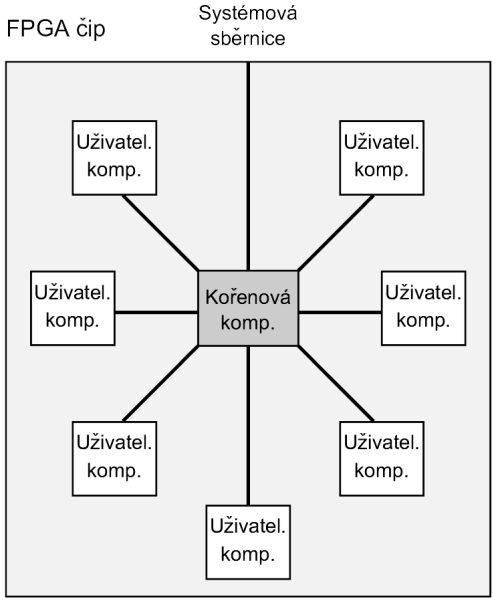
Podporovány jsou jak přenosy z nadřazeného systému do FPGA, tak přenosy iniciované ze strany uživatelských komponent. V případě, že není nutné spojení se systémovou sběrnicí, je v kořenu stromu umístěný místo kořenové komponenty přepínač.

2.1.4 Hvězdicová architektura

Zobecněním stromové topologie je hvězdicová architektura. Příklad zapojení komponent podle tohoto schématu je znázorněn na obrázku 2.7. Centrálním prvkem této architektury je tzv. hub nebo také přepínač. Jeho úkolem je zajišťovat propojení a komunikaci mezi všemi připojenými uživatelskými komponentami. Jedná se o obecnou variantu přepínače, který známe ze stromové architektury (stromový přepínač odpovídá dvouportovému hvězdicovému přepínači). Hvězdicový přepínač má jeden tzv. upstream port a, na rozdíl od stromového, obecně libovolný počet downstream portů. Upstream port slouží k připojení systémové sběrnice, zatímco downstream porty poskytují rozhraní pro jednotlivé uživatelské komponenty.

Princip směrování je podobný jako u stromové topologie a je založený na porovnávání cílové adresy nebo speciálně definované identifikace. Jestliže spolu komunikují dva uzly, je mezi nimi dočasně vytvořeno virtuální spojení, které není po dobu transakce přerušeno. V jeden okamžik může být vytvořeno $n/2$ virtuálních spojení, kde n je celkový počet portů centrálního prvku (např. v systému se 6 komponentami spolu mohou souběžně komunikovat až 3 dvojice).

Stejně jako předchozí typy topologií může být i hvězdicové schéma modifikováno různými způsoby tak, aby bylo dosaženo lepších funkčních a výkonnostních parametrů. Připojením dalšího centrálního prvku místo jedné z uživatelských komponent je možné vytvořit další



Obrázek 2.7: Hvězdicová architektura

hvězdici na nižší úrovni. Pokud bude tento přepínač schopný také transformovat šířku sběrnice, může být opět vytvořena hierarchie sběrnic (hvězdic) s různými rychlostmi. Další modifikací může být například využití plně duplexních spojovacích linek.

Podporovány jsou jak přenosy iniciované ze strany systémové sběrnice, tak ze strany uživatelských komponent. Problém citlivosti na vzdálenost je v případě dlouhých vodičů mezi centrálním prvkem a uživatelskou komponentou nutné řešit vložením speciální registrovací komponenty jako například u klasické sběrnicové topologie.

2.2 Principy přidělování sběrnice

Před vlastním datovým přenosem musí proběhnout proces, jehož výsledkem je rozhodnutí o tom, která komponenta žádající o přístup ke sběrnici dostane povolení komunikovat, respektive která transakce z jednoho ze vstupních rozhraní projde na výstup.

V zásadě existují dva základní principy přidělování (arbitrace) sběrnice [12]: centralizovaný a distribuovaný. V prvním případě existuje v propojovacím systému centrální prvek (arbitr), který přijímá požadavky od všech žadatelů. Na základě prioritního systému rozhoduje o tom, které komponentě bude sběrnice přidělena.

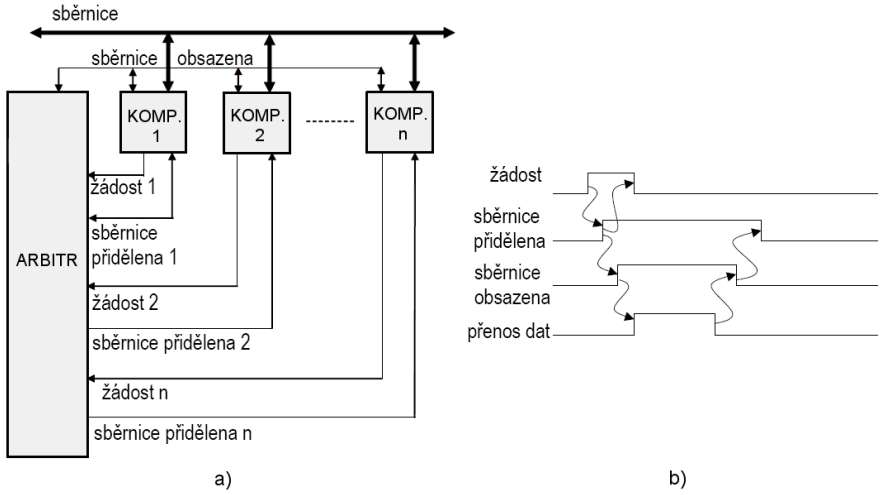
U distribuovaného přidělování sběrnice centrální prvek neexistuje. Arbitrační rozhodnutí mohou před vlastním přenosem provést jednotlivé komponenty mezi sebou. Jestliže transakce putuje do cílového místa přes více uzlů, je možné provádět rozhodování postupně v průběhu celé trasy putující transakce. Takové arbitrační schéma se nazývá směrování nebo také přepínání.

2.2.1 Centralizovaná schémata

Prvním schématem je tzv. centrálně řízené přidělování sběrnice podle důležitosti požadavku. Připojené komponenty mohou mít buď fixní prioritu stanovenou na základě důležitosti jimi

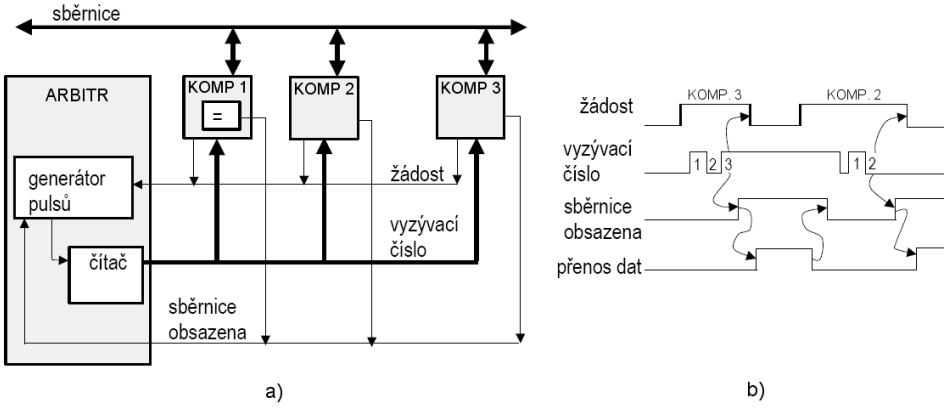
generovaných požadavků nebo prioritu proměnnou. V tom případě určuje prioritu zvolená strategie.

Nejčastěji je využívána tzv. strategie cyklické priority, pro niž se ustálilo anglické označení Round-robin. Tento přístup vybírá jednotlivé žadatele postupně v cyklickém pořadí. Jestliže je komponenta na řadě a nemá připravená data, automaticky je vybrán další žadatel v pořadí, který má data k dispozici.



Obrázek 2.8: Centrálně řízené přidělování sběrnice podle důležitosti požadavku: (a) bloková struktura, (b) komunikační protokol

Při přidělování sběrnice podle důležitosti požadavku generují jednotlivé komponenty signál s *žádostí* o přístup. Centrální arbitr na základě zvolené strategie vyhodnotí žadatele s nejvyšší prioritou a odpoví mu signálem *sběrnice přidělena*. Vybraná komponenta pak prostřednictvím signálu *sběrnice obsazena* informuje ostatní komponenty, že přebírá kontrolu nad sběrnicí a zahajuje přenos dat. Popsanou situaci demonstrují obrázky 2.8a a 2.8b.

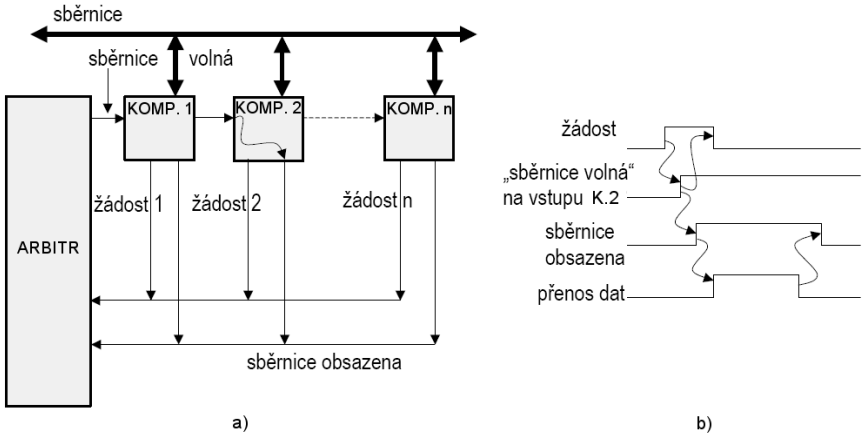


Obrázek 2.9: Centrálně řízené přidělování sběrnice na výzvu: (a) bloková struktura, (b) komunikační protokol

Dalším centralizovaným schématem je přidělování sběrnice na výzvu. Tento princip je nedemokratický, protože je ve hře prioritní systém. Priorita je fixní a určuje ji tzv. vyzývací číslo, které má každá komponenta předem dané.

Systém centrálně řízené sběrnice na výzvu je zobrazen na obrázcích 2.9a a 2.9b. Jednotlivé komponenty generují *žádosti* o přístup do společného vodiče, který je monitorován arbitrem. Ten v případě aktivní žádosti začne vysílat vyzývací číslo. Komponenta, která žádá o přidělení sběrnice rozpozná své číslo a generuje signál *sběrnice obsazena*. Pak následuje vlastní přenos dat.

Třetí demonstrované schéma představuje centrálně řízená postupná obsluha sběrnice a je zobrazeno na obrázcích 2.10a a 2.10b. Jednotlivé komponenty vysílají signály *žádosti* do společného vodiče. Arbitr jej monitoruje a při aktivní žádosti odpoví vysláním signálu *sběrnice volná*. Ten je postupně distribuován skrze jednotlivé komponenty, v kterých je vyhodnocován. V případě, že dorazí k aktivnímu žadateli, je jeho odeslání dalším komponentám zablokováno. Následně je nastaven signál *sběrnice obsazena* a začíná přenos dat. Prioritní systém je uplatněn pořadím zařízení na vodiči *sběrnice volná*. Zařízení, která jsou blíže arbitra, mají vyšší prioritu.



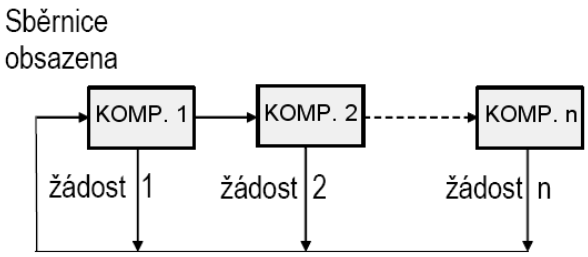
Obrázek 2.10: Centrálně řízená postupná obsluha sběrnice:
(a) bloková struktura, (b) komunikační protokol

2.2.2 Distribuovaná schémata

Arbitrační rozhodování u distribuovaných schémat neprovádí jeden prvek, ale podílejí se na něm všechny komponenty, které jsou danou sběrnicí propojeny či přes které transakce prochází.

Na obrázku 2.11 je znázorněn jednoduchý případ, kdy jsou komponenty propojeny pouze pomocí jedné datové sběrnice. Transakce tak nemusí procházet skrze jiné, např. přepínací komponenty, a komunikace může probíhat přímo mezi zdrojem a cílem přenosu. Obrázek zahrnuje také řídící vodiče, pomocí kterých je možné rozhodnout, kterému prvku bude sběrnice přidělena.

Komponenta mající připravená data generuje *žádost* do společného vodiče, což následně vyvolá nastavení signálu *sběrnice obsazena*. Ten je postupně vyhodnocován jednotlivými komponentami. Jakmile se signál dostane na vstup žadatele, který žádost vygeneroval, je jeho přenos do následujícího zařízení zablokován a může začít vlastní přenos dat.



Obrázek 2.11: Distribuované schéma arbitrace

U složitějších topologií (než je klasická sběrnicová architektura) vyžaduje hierarchické uspořádání komponent existenci distribuovaného směrovacího schématu arbitrace. Podle něj transakce postupně putují jednotlivými segmenty různě široké víceúrovňové sběrnice a prochází přes speciální přepínací a transformační komponenty. Každý takový uzel zná adresové rozsahy svých portů a na základě této informace a cílové adresy provádí směrování. S tímto schématem se můžeme setkat např. u stromové či hvězdicové architektury.

2.3 Komunikační protokol

Při výběru typu komunikačního protokolu je třeba přihlédnout k několika aspektům. Mezi ně patří např. režie protokolu, využití kapacity přenosových linek, latence čtecích operací, minimální a maximální dosažitelná propustnost nebo složitost a potažmo potřebné množství zdrojů pro zpracování vybraného protokolu. Důležitým faktorem pro rozhodování je také typ rozhraní, pro které je komunikační protokol vybírán.

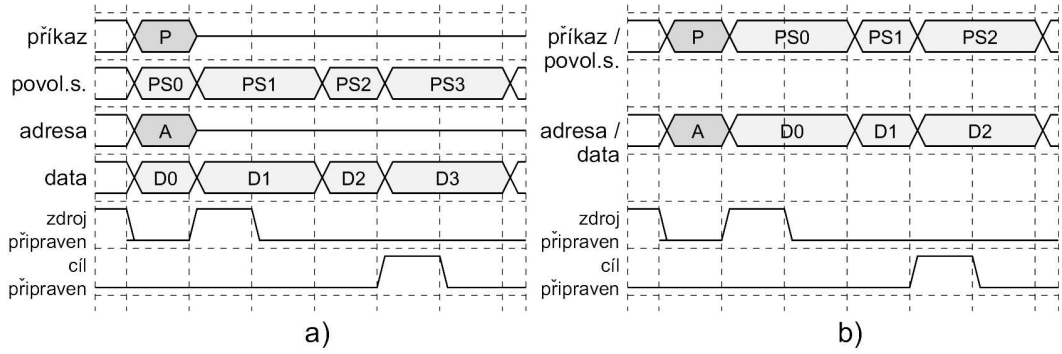
2.3.1 Typy protokolu

První variantou je jednoduchá paralelní sběrnice [13], u níž mají všechny potřebné řídící (příkaz, adresa, povolovací signály atd.) a datové signály vyhrazené vlastní vodiče. Jak ukazuje obrázek 2.12a, pro každé přenášené datové slovo jsou zvlášť nastavovány všechny řídící signály. Díky tomu je sice režie protokolu nulová a kapacita přenosové linky může být teoreticky využita maximálně, avšak nastávají jiné problémy. Ty souvisejí s tzv. citlivostí na vzdálenost, množstvím využitých zdrojů a omezenými prostředky pro mapování potřebné logiky do hradel a vodičů programovatelného hradlového pole FPGA. Tato problematika je podrobněji popsána v kapitole 2.6.

Další možností je využití tzv. multiplexované sběrnice [17], jejíž protokol je znázorněn na obrázku 2.12b. Ten je podobný předchozí paralelní variantě, avšak díky multiplexování vodičů dochází k výrazné redukci jejich počtu a tím i částečné eliminaci problémů paralelní sběrnice. Režií protokolu je první takt každé transakce, v kterém je přenášena adresa a příkaz.

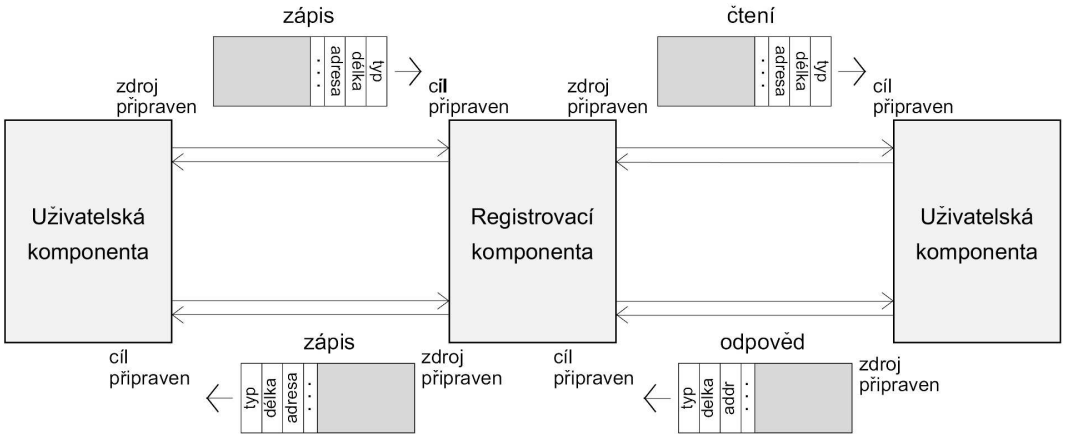
Zobecněním předchozího případu je tzv. paketově-orientovaný protokol [25]. Toto schéma slouží k přenosu datových jednotek (paketů), které se skládají z hlavičky a volitelné datové přílohy. Hlavička nese řídící informace (typ transakce, délka, adresa atd.), které jsou nutné ke zpracování přenášených dat.

Přenos dat je podobně jako u předchozích typů protokolu založený na dvou signálech poskytovaných přijímací (*cíl připraven*) a vysílací (*zdroj připraven*) stranou. Pouze v případě,



Obrázek 2.12: Komunikační protokol: (a) paralelní schéma, (b) multiplexované schéma

že jsou oba signály aktivní, dochází k přenosu. Každý paket se obvykle skládá z více slov, a proto bývá jeho začátek a konec označen speciálními značkami (signály). Popsané schéma demonstruje obrázek 2.13, kde spolu dva prvky komunikují skrze registrovací komponentu.



Obrázek 2.13: Paketově-orientované schéma komunikace

Režii paketového protokolu představuje hlavička paketu a výrazněji se projevuje pouze u krátkých transakcí, jejichž délka dat mnohonásobně nepřesahuje délku hlavičky. Při objemných přenosech se však využívá transakcí maximální povolené délky, u kterých je režie zanedbatelná a je tak možné dosáhnout vysoké propustnosti. Vzhledem k tomu, že jednotlivým řídícím informacím nepřísluší vlastní vodiče, je tento typ protokolu vhodný pro propojení komponent na dlouhé vzdálenosti (komponenty mohou být rozmístěny po celé ploše čipu FPGA). Na druhou stranu však vyžaduje zpracování takového protokolu větší množství spotřebované logiky.

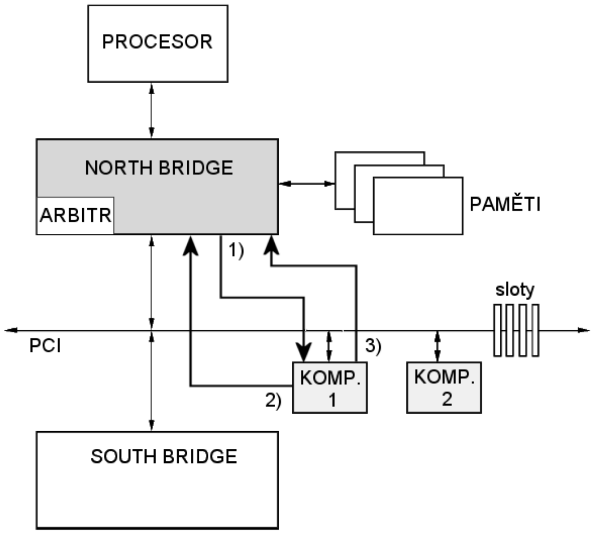
2.3.2 Protokoly čtecí operace

Doba provedení čtecí operace má velký vliv na využitelnou kapacitu přenosové linky, která představuje jedno z důležitých kritérií při výběru komunikačního protokolu. Každé čtení má dvě oddělené fáze, které nutně nemusí následovat těsně po sobě. První z nich je doručení čtecího požadavku ze zdrojové komponenty do prvku obsahujícího požadovaná data.

Druhou fází je pak příjem přečtených dat a jejich transport ke komponentě, která čtení vyvolala. Mezi těmito fázemi může být obecně libovolně dlouhá prodleva, kterou potřebuje komponenta na to, aby připravila požadovaná data. To se označuje jako latence čtecí operace a je to parametr specifický pro každou připojenou komponentu.

S cílem zachovat maximální využití kapacity přenosové linky i při velkých čtecích latencích bylo navrženo několik přístupů. Ty se mmj. postupně uplatňovaly i v jednotlivých generacích sběrnice PCI [17], na jejímž příkladě si navržené principy demonstrujeme.

U samotné sběrnice PCI se využívá tzv. Retry nebo Disconnect protokol (obrázek 2.14). Čtecí operace u Retry protokolu vypadá následovně: (1) zdrojová komponenta inicializuje čtecí operaci směrem k cílovému zařízení. (2) to rozpozná svoji adresu a potvrdí příjem operace. Jestliže bude cílová komponenta schopná připravit data rychle, může na sběrnici vložit až 16 čekacích stavů (taktů), po kterých začne vysílat odpověď. (3) v případě, že čtecí latence je příliš velká, odloží cíl operaci příkazem Retry (zkusit znovu). Zdrojová komponenta pak zkusí inicializovat čtecí operaci později. Celý cyklus se opakuje, dokud nejsou data přenesena. Tento přístup je v případě vícenásobného odkládání transakce neefektivní, protože dochází k plýtvání kapacity sběrnice.

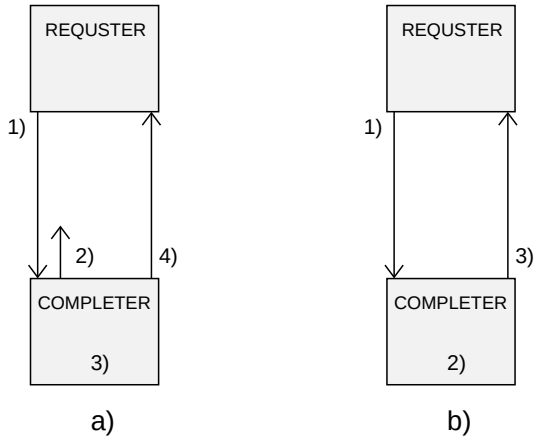


Obrázek 2.14: Architektura systému se sběrnicí PCI a Retry/Disconnect protokolem

Retry protokol je doplněn tzv. Disconnect protokolem. Díky němu může v bodě (2) cílová komponenta začít přenášet data, i když předem neví, jak bude přenos dlouhý. Jestliže je v průběhu přenosu zjištěno, že další data již nejsou k dispozici, dojde v bodě (3) k odložení operace příkazem Disconnect (odpojit). Zdrojová komponenta pak zkusí znovu založit operaci později a to od pozice, kde došlo k rozpojení transakce. Celý cyklus se opakuje, dokud nejsou přenesena všechna požadovaná data.

U sběrnice PCI-X [17] se objevil tzv. model rozdělených transakcí (obrázek 2.15), který byl později vylepšen u zatím poslední sběrnice z rodiny PCI, sběrnice PCI Express [15]. Tento model je cílený zejména pro plně duplexní přenosové linky, které mají pro každý směr vyhrazený jeden nezávislý komunikační kanál. U každé operace se rozlišuje tzv. žadatel (requester), který iniciuje čtecí operaci, a poskytovatel (completer), který dodává požadovaná data.

Princip čtení u sběrnice PCI-X je následující: (1) žadatel iniciuje čtecí operaci. (2) cílová komponenta detekuje svoji adresu a pokud zjistí, že není schopna poskytnout data okamžitě, rozpojuje transakci. (3) pak začne příprava dat do vnitřní vyrovnávací paměti komponenty. (4) v okamžiku, kdy jsou všechna data k dispozici (u PCI-X zná poskytovatel na rozdíl od PCI délku dat), je zahájena jedna nebo více zápisových operací, ve kterých jsou přečtená data poslána žadateli. Nedochází tak k opakovanému vyzývání ze strany zdrojové komponenty jako v případě PCI a pro načtení dat jsou potřeba maximálně dvě operace.



Obrázek 2.15: Model rozdělených transakcí: (a) PCI-X, (b) PCI Express

U sběrnice PCI Express má každý spoj dvě nezávislé přenosové linky a podle toho byl model rozdělených transakcí dále upraven. Každá čtecí operace se skládá ze 3 částí: (1) nejprve je vyslána čtecí transakce, která končí v okamžiku převzetí cílovou komponentou. (2) pak je prováděna příprava dat, zatímco obě linky jsou volné a může na nich probíhat komunikace. (3) až jsou data k dispozici, je odeslána odpověď, která je s původní čtecí transakcí spjata jednoznačným identifikátorem.

2.3.3 Rozhraní v propojovacích systémech

V propojovacích systémech se setkáváme s dvěma základními typy rozhraní: rozhraní interních sběrnic a uživatelská rozhraní. Interní sběrnice na nejvyšší úrovni by měla být co nejvýkonnější, aby pokryla i vysoké požadavky na propustnost. S rostoucí velikostí čipu je rovněž důležité, aby se u sběrnice neprojevovala tzv. citlivost na vzdálenost. To je problém především širokých rozhraní, která se skládají z velkého množství vodičů (např. jednotlivým řídícím informacím jsou přiděleny zvláštní signály).

Ze současných řešení je nejvhodnější využití paketově-orientovaného komunikačního protokolu. Protože součástí každé aplikace jsou i pomalé komponenty, je interní sběrnice konvertována buď na nižší šířky nebo na jiné typy pomalejších sběrnic. Ta pak podle požadavků (např. na rozmístění komponent, propustnost, latenci atd.) mohou mít rovněž paketová nebo např. multiplexovaná rozhraní.

Uživatelská rozhraní slouží k připojení uživatelských komponent k propojovacímu systému. Tyto komponenty obsahují různé paměti a registry, ke kterým je třeba přistupovat. Proto se většinou jeví jako vhodné použití jednoduché paralelní sběrnice, která je jednoduše napojitelná na adresové dekodéry a paměti.

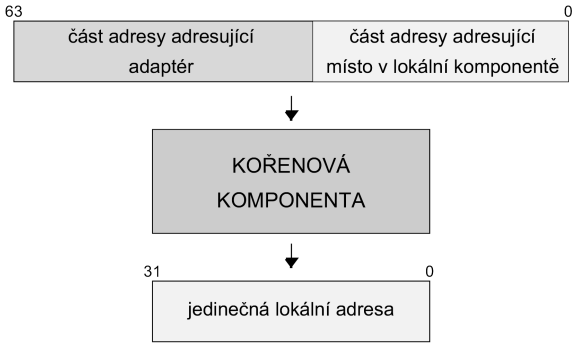
Jako uživatelská rozhraní chápeme především zápisové a čtecí rozhraní, ale také např. rozhraní pro iniciování přenosů a další. Nejsložitější z nich je čtecí rozhraní, v souvislosti s kterým existuje problém se čtecí latencí, který byl podrobně popsán v kapitole 2.3.2. V zásadě se můžeme setkat se dvěma typy čtecích rozhraní: V prvním případě je čtecí požadavek (identifikovaný adresou a délkou) uživatelské komponentě jednorázově předán. To vyžaduje více logiky ke zpracování na straně cílové komponenty, avšak interní sběrnice není zbytečně blokována. Je to vhodné pro komponenty vyžadující velkou propustnost nebo nacházející se na externím čipu. Pro vyčítání stavových informací z jednoduchých komponent se zase hodí druhá varianta čtecího rozhraní, kdy je čtecí požadavek předáván uživatelské komponentě postupně a při každém dalším slově se adresa inkrementuje.

Důležitou vlastností každého rozhraní, která ovlivňuje množství využitých zdrojů, propustnost a frekvenci, je datová šířka. Záleží nejen na aplikaci, které bude propojovací systém součástí, ale i na konkrétních komponentách v rámci aplikace, jak rychlou sběrnici o jaké datové šířce je vhodné použít. Jednotlivé komponenty mají zpravidla rozdílné požadavky na propustnost, a proto je důležité vybudovat v rámci propojovacího systému jakousi hierarchii různě rychlých sběrnic. Vyčítání stavových informací z běžných komponent je typickým příkladem situace, kdy postačuje pomalá komunikace. Naproti tomu například připojení řadiče dynamické paměti vyžaduje rychlý přísun dat a dostatečnou propustnost. V případě fixní datové šířky je do propojovacího systému nutné začlenit více typů sběrnic s různými rozhraními. Jestliže je datová šířka rozhraní flexibilní, je možné vybudovat celý systém na základě jedné sběrnice.

2.4 Způsoby komunikace a typy přenosů

Než-li se budeme zabývat jednotlivými způsoby komunikace a typy přenosů, uvedeme si, s jakými typy adresových prostorů se můžeme setkat. V kontextu adaptéru (potencionálně osazeného čipem FPGA) a hostitelského systému hovoříme o globálním a lokálním adresovém prostoru.

Globální adresový prostor celého systému zahrnuje jak fyzickou paměť počítače, tak i paměť namapovanou ze všech adaptérů. Podle typu systému má tento prostor velikost 4 GB (32 bitů) nebo 16 EB (64 bitů). Při návrhu propojovacího systému je pro obecnost nutné uvažovat větší variantu. Transakce, které přistupují ke globálnímu prostoru tak budou používat globální adresy o velikosti 64 bitů, s čímž je třeba počítat (například při rezervování potřebného místa v hlavičce paketu).

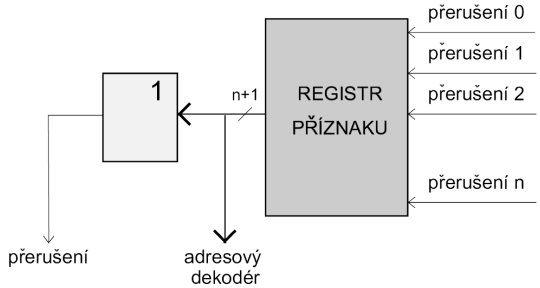


Obrázek 2.16: Překlad mezi globální a lokální adresou

Lokální adresový prostor je platný pouze v rámci adaptéru, v kterém mají jednotlivé komponenty definovanou jedinečnou lokální adresu. Její šířka musí být odpovídající potřebné velikosti paměti interních komponent. Překlad mezi globálními a lokálními adresami bývá většinou zajištěn kořenovou komponentou interního propojovacího systému (viz obrázek 2.16). O přenosech iniciovaných ze strany prvku, který se nachází mimo adaptér, a směřujících do lokálního adresového prostoru hovoříme jako o tzv. slave přenosech. Pokud je operace zahajována ze strany komponenty uvnitř FPGA a je adresovaná do globálního prostoru (obecně jakékoliv jiné komponentě), mluvíme o tzv. master přenosu.

Mezi základní způsoby komunikace počítače a adaptéru patří programové (softwarové) řízení, přerušení a blokové přenosy. Programem řízená čtecí nebo zápisová operace probíhá mezi procesorem (iniciátor přenosu) a registry nebo pamětí na adaptéru. Přenos je řízen instrukcemi procesoru a jsou přenášena samostatná slova. Vzhledem k tomu, že je v rámci jedné instrukce přenášen pouze malý blok dat, je na procesor kladena velká zátěž. Navíc se výrazně projevuje režie komunikačního protokolu (hlavička paketu u paketové komunikace, přenos adresy u multiplexované sběrnice). Aktivním čekáním procesoru (tzv. pooling) je také možné provádět zpracování asynchronní události. Toto řešení je realizováno opakovaným testováním hodnoty registru v adaptéru, což opět zatěžuje procesor a zbytečně blokuje sběrnici. Programové řízení přenosu je proto vhodné pouze pro jednoduché přenosy jako je např. odstartování periferní operace či nastavení řídících informací.

Dalším způsobem komunikace je přerušení, které je vhodné zejména pro signalizaci asynchronní události ze strany adaptéru. Při přerušení není procesor zatížen komunikací a na vzniklou událost je možné reagovat okamžitě. Tímto způsobem je možné odstranit režii, kterou sebou nese programové zpracování asynchronní události. V případě, že se v interním systému adaptéru může vyskytnout více různých událostí, je nutné použít registr příznaků. Tento případ je demonstrován na obrázku 2.17. Po vyvolání přerušení je nutné přečíst obsah registru a zjistit tak, která událost přerušení způsobila. Priority je možné řešit na programové úrovni.



Obrázek 2.17: Demonstrace více zdrojů přerušení na jednom adaptéru

Důležitým způsobem komunikace jsou také blokové přenosy, které umožňují přímý přístup do paměti počítače (anglicky Direct Memory Access, DMA), aniž by byl zatížen procesor. Dlouhé přenosy dat snižují komunikační režii a umožňují dosáhnout plnou propustnost sběrnice. Nevýhodou však může být blokování sběrnice pro ostatní transakce. Přenosy jsou realizovány mezi adaptérem a pamětí počítače nebo pouze v rámci adaptéru. V obou případech není procesor zatížen a může se věnovat jiným výpočetním úlohám.

Ukončení přenosů je signalizováno pomocí přerušení. Zde je potřeba dále řešit, kdy přesně má být přerušení generováno. Obecně lze totiž říci, že odeslání dat z konkrétní komponenty a okamžik skutečného zápisu dat do cílového místa jsou dvě rozdílné události

s různým časem výskytu. Je žádoucí, aby byl o ukončení přenosu procesor informován ve chvíli, kdy je jisté, že všechna data byla úspěšně zapsána. K tomu, aby se zdroj dozvěděl o ukončení zápisu a mohl generovat přerušení, lze využít např. zápisového potvrzovacího paketu. Tento způsob však vede na vytvoření nového typu transakce a dalšímu zesložitění logiky. Jednodušším řešením je vyčítání řídící informace z pomocného registru. Zmíněná problematika je poměrně rozsáhlá a v kontextu této práce se jí nebudeme podrobněji zabývat.

2.5 Pořadí čtecích a zápisových operací

Iniciování čtecích a zápisových operací je ze strany procesoru prováděno sekvenčně a po celou dobu putování těchto transakcí systémem musí být v rámci jednotlivých adres zachováno jejich pořadí. Díky tomu se zabraňuje nežádoucím situacím, kdy by mohla být data z dané adresy přečtena příliš brzy (čtecí transakce by předběhla zápisovou transakci a přečetla by se nesprávná data) nebo příliš pozdě (zápisová transakce by předběhla čtecí transakci a původní správná data by se nestačila přečíst).

Zachování pořadí v rámci jednotlivých adres lze pomocí technických prostředků realizovat několika způsoby. Ty se liší především množstvím přídavné logiky nutné k zajištění potřebných porovnání a mírou omezení výkonnosti.

Nejjednodušším přístupem, který však nejvíce omezuje propustnost dané části sběrnice, je model striktního dodržení pořadí čtecích a zápisových transakcí s čekáním na dokončení čtecí operace. Tento způsob nepovoluje žádné změny v pořadí transakcí bez ohledu na cílovou adresu. Následující operace nemůže začít dříve, než skončí operace předchozí.

Vypuštěním nutnosti čekání na dokončení čtecí operace získáme uvolněnější variantu předchozího modelu. Vynechání čekání lze však provést pouze tehdy, pokud zápisová transakce nezasahuje do prostoru, který je právě čten. U tohoto přístupu je již využití sběrnice efektivnější, ovšem za cenu přídavné logiky. Ta je nezbytná k provádění kontrol tzv. kolizí typu WaR (anglicky Write after Read), kdy zápis bezprostředně následuje za čtením ze stejné adresy.

Třetí možností je adaptivní přeuspořádávání čtecích a zápisových operací, při kterém se jednotlivé transakce mohou vzájemně předbíhat. Čtecí operace může předejít zápisovou, pokud nečte data z prostoru, do kterého zápisová operace zapisuje. A podobně tomu je i v situaci, kdy zápisová operace předchází čtecí. Tento přístup efektivním způsobem využívá sběrnici, avšak vyžaduje poměrně komplexní testy při přeuspořádávání. Ty vedou na nezanedbatelné zesložitění logiky.

Zmíněné postupy je možné aplikovat na různých úrovních hierarchie propojovacího systému. Čím vyšší je úroveň, na které probíhají kontroly pořadí transakcí, tím větší je část adresového prostoru, u jehož komponent je následkem kontrol omezena propustnost. Proto je vhodné uplatňovat tyto postupy až na konkrétní přípojná místa těch komponent, které zachování pořadí transakcí skutečně potřebují.

Zachování pořadí v rámci jednotlivých adres lze kontrolovat i pomocí programového vybavení počítače. Tento způsob využívá speciálních paměťových bariér, které buď představují samostatné instrukce, nebo jsou součástí speciálních čtecích a zápisových operací. Paměťová bariéra zablokuje provádění zápisových transakcí na určité adresy, dokud nedojde k dokončení všech čtecích operací z tohoto prostoru. Tento přístup znamená pro propojovací systém vytvoření nového speciálního typu transakce, který reprezentuje paměťovou bariéru. Dále je nutné přidání jednoduché logiky, která po detekci bariéry pozastaví provádění zápisových operací.

2.6 Citlivost na vzdálenost

Problém tzv. citlivosti na vzdálenost je specifický zejména pro technologii hradlových programovatelných polí. Na rozdíl od technologie ASIC disponuje čip FPGA omezeným množstvím dostupných zdrojů. Zvláště v případě dlouhých a širokých paralelních sběrnic je obtížné namapovat potřebnou logiku do dostupných hradel a vodičů a současně splnit požadovaná časová omezení. Problém se navíc s rostoucím zaplněním čipu zvětšuje.

Řešením je tzv. registrování obvodu (pipelining). Základní princip je založený na rozdělení většího úseku logiky na menší části, mezi něž se vloží jednotlivé registry. U dlouhých sběrnic navíc hraje roli zpoždění samotných vodičů.

Registrování datových cest sebou kromě latence nenese žádný další vedlejší efekt. Problém nastává u kontrolních signálů, které řídí přenos dat. Vzhledem k tomu, že připravenost k přenosu musí být signalizována jak z vysílací, tak z přijímací strany, vyskytuje se v obvodu zpětná vazba. V takovém případě nelze využít obyčejných registrů, ale je nutné implementovat speciální registrovací komponentu. Ta provede zpoždění dat o jeden takt korektním způsobem a umožní eliminovat problém citlivosti na vzdálenost. Její architektura může být založena např. na principu fronty, která má dvě položky. Ve standardním případě (zdroj i cíl jsou současně připraveni k přenosu) je zaplněna pouze jedna položka. Jestliže vysílací strana iniciuje přenos a příjemce není připraven, zaplní se i druhá položka a čeká se, dokud přijímací strana frontu neuvolní.

2.7 Chybové stavy

Při komunikaci po sběrnici může docházet k různým chybovým stavům. Obecně rozlišujeme dva základní typy chyb [15]: opravitelné a neopravitelné. Opravitelné chyby nevedou k žádným fatální funkčním chybám a způsobují pouhou degradaci výkonu. Z takových chyb je systém schopný se zotavit sám bez jakékoliv ztráty informace a zásah procesoru není nutný. Příkladem může být situace, kdy příjemce zjistí chybu v příchozí transakci. Takový problém lze vyřešit opětovným zasláním chybného paketu. Neopravitelné chyby vedou k funkčním chybám a je těžké navrhnout obecný platformově nezávislý postup, jak se v takových situacích zachovat. Tyto chyby lze dále dělit podle toho, zda způsobují nestabilní a nespolehlivé chování celých komponent nebo pouze jednotlivých transakcí. V prvním případě je nutné postižené komponenty resetovat, zatímco ve druhém je možné dát šanci programovému vybavení pokusit se o zotavení.

Vzniklé chyby je možné detekovat několika způsoby. Jednoduchým nativním způsobem je kontrola řídících informací použitého protokolu. Takovýmto způsobem je možné detekovat např. situaci, kdy hodnota délky v hlavičce paketu neodpovídá skutečné délce dat. Další možností je použití různých zabezpečovacích kódů, které mohou být pouze detekční nebo současně i opravné. Nejjednodušším příkladem je zabezpečení sudou nebo lichou paritou. Poslední možností detekce chyb, která bude zmíněna, je zabezpečení bloku dat kontrolním součtem, který je připojen za přenášená data. Cílová komponenta pak nad daty provede stejný součet a obě hodnoty porovná.

Jestliže je detekována chyba, je dále vhodné ji nějakým způsobem signalizovat. Je možné identifikovat tři základní mechanismy signalizace: bitové pole v hlavičce paketu indikující stav transakce, bit v hlavičce paketu indikující chybu a speciální chybové zprávy (pakety). Bitové pole v hlavičkách paketů se vyskytuje např. u dokončovacích transakcí, které nesou odpověď na čtecí požadavek. Toto pole pak může informovat žadatele buď o úspěšně dokončené operaci nebo o typu chyby, ke které došlo. To může být například situace, kdy byla

žádána data z neexistující adresy. Jednodušším mechanismem pro indikaci stavu/chyby v paketu je jeden bit v hlavičce, který říká, zda je paket poškozený či nikoliv. Posledním, nejobecnějším mechanismem je zasílání speciálních paketů, které informují o vzniklých chybách. Tento postup vyžaduje přídavnou podporu všech zúčastněných zařízení a z pohledu složitosti a množství zabraných zdrojů je poměrně náročný.

Typickým chybovým stavem je tzv. uváznutí, které bude podrobněji popsáno v kapitole 2.7.1. Obecně je problematika chyb u sběrnicových systémů velmi rozsáhlá a složitá. Podrobnější charakteristika chybových stavů, jejich vzniku, detekce, signalizace a opravy je však nad rámec tohoto textu a nebude dále popisována.

2.7.1 Problém uváznutí

Uváznutí (anglicky deadlock) představuje obecný problém, který se může vyskytovat od nejnižších hardwarových úrovní až po softwarová uváznutí, v rámci jednoho počítače nebo mezi různými počítači v síti atd. Tyto problémy poprvé komplexně popsal již v roce 1971 E.G. Coffman [3].

V kontextu interních sběrnic a propojovacích systémů se jedná o situace, kdy komunikující komponenta uvázne v některém ze svých vnitřních stavů. K tomu může dojít v případě, kdy skupina agentů (paketů) nemůže postupovat, protože čekají jeden na druhého, až uvolní společné prostředky (např. vyrovnávací paměti nebo virtuální kanály). Tvoří-li čekající agenti smyčku, dochází k uváznutí.

Takovýmto problémům se lze zpravidla vyhnout použitím mechanismu vypršení časového limitu nebo pomocí pravidel pro uspořádání transakcí. Vypršení časového limitu je nejčastěji aplikováno v souvislosti s čtecími transakcemi, ke kterým nepřijde odpověď (ani ve formě prázdné odpovědi, která by pouze indikovala výskyt chyby). V takovém případě je pak např. možné čtecí požadavek opakovat. U některých existujících sběrnicových systémů je tento mechanismus implementován i pro zajištění toho, aby pakety nečekali v přepínači déle než je stanovený limit [20].

Pravidla pro uspořádání transakcí definují, jakým způsobem mohou být pakety ve frontě přeuspořádány a tedy kdy je možné, aby jeden paket mohl předběhnout jiný. Například, jestliže jsou paměti pro uložení rozpracovaných čtecích transakcí nějaké komponenty plné, pak je díky tomuto mechanismu možné, aby zápisové transakce předbíhaly čtecí a nebyly zbytečně blokovány.

Kapitola 3

Dostupná řešení

Za posledních dvacet let bylo vyvinuto mnoho sběrnicových systémů sloužících k propojení jednotlivých prvků počítače na různých úrovních. Pro připojení periférií k základní desce se používají především sběrnice z rodiny PCI (PCI, PCI-X, PCI Express), které poskytují velmi komplexní komunikační protokol. Každá obvodová deska se dále skládá z jedno či více čipů, které spolu potřebují komunikovat. K tomu je určena např. sběrnice RapidIO [19].

Nás budou nejvíce zajímat propojovací systémy, které slouží k propojení jednotlivých komponent v rámci každého čipu. Některé z nich byly vyvinuty komerčními společnostmi (např. IBM Core Connect nebo AMBA), jiné vznikly jako výsledek tzv. open source projektů (např. WishBone). Z každé z těchto dvou kategorií si popíšeme jednoho zástupce, který se stal standardem a je široce používán při vývoji složitějších systémů v rámci jednoho čipu.

Core Connect [4] je komplexní propojovací systém vyvinutý firmou IBM, který používá více jak 1500 technologických společností po celém světě. Systém se skládá ze třech základních typů sběrnic: PLB (Processor Local Bus), OPB (On-chip Peripheral Bus) a DCR (Device Control Register bus). Všechny typy sběrnic jsou synchronní a obsahují jak adresové, tak datové vodiče, čímž se zvyšuje jejich citlivost na vzdálenost. K dispozici jsou také speciální transformační komponenty, které umožňují napojení na sběrnici AMBA [1].

Vysokorychlostní PLB sběrnice [7] slouží k připojení rychlých komponent, které vyžadují vysokou propustnost. Datovou šířku lze vybrat z několika možností (32, 64 nebo 128 bitů), které mají vliv na celkovou výkonnost. Sběrnice podporuje model rozdělených transakcí, možnost iniciování přenosů ze strany uživatelských komponent, dále nechybí podpora blokových přenosů či uzamykání a registrování sběrnice. Sběrnice OPB [5] slouží k připojení pomalejších komponent, které by jinak zbytečně omezovaly propustnost PLB. Její šířka je nastavitelná na hodnoty 8, 16 nebo 32 bitů. Komunikační protokol nepodporuje model rozdělených transakcí. Místo něj je využíván tzv. Retry protokol, který pro čtení z komponent s velkou latencí není příliš efektivní. Posledním typem je sběrnice DCR [6], která slouží pro přenos řídících dat a komunikaci se stavovými a kontrolními registry. Celý Core Connect systém je tedy založen na třech různých komunikačních protokolech a je složen z velkého množství různých řídících a transformačních komponent. To vede k horší udržovatelnosti takového systému.

WishBone [13] představuje jeden z nejznámějších propojovacích systémů, který vznikl jako výsledek projektu s otevřenými zdrojovými kódy (tzv. open source projekt), a je široce používán v oblasti integrovaných obvodů. Sběrnice podporuje šířky 8, 16, 32 a 64 bitů a je založena na velmi jednoduchém komunikačním protokolu. Podporovány jsou také různé druhy sběrnicových topologií, mezi které patří např. klasická sběrnicová topologie, přímé propojení (tzv. point-to-point) nebo propojení pomocí křížového přepínače. Wishbone zahr-

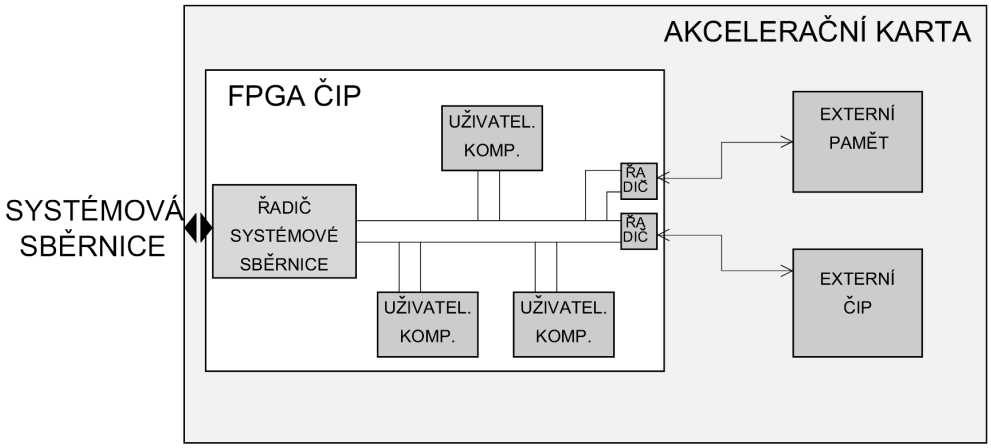
nuje jak adresové, tak datové vodiče, čímž se zvyšuje citlivost sběrnice na vzdálenost. Velkou nevýhodou je i absence podpory modelu rozdělených transakcí. Problém komponent s dlouhou čtecí latencí je řešen pouze neefektivním Retry protokolem. Existují některé techniky, které blokování sběrnice při čtení z pomalejších komponent eliminují. Příkladem může být transformace čtecích požadavků na zápisové transakce do zvláštního adresového prostoru komponenty. Tato technika však pro propojovací systémy není nativní a komplikuje práci návrháře.

Cílem této práce je navrhnout a implementovat sběrnicový systém, který slouží pro propojení jednotlivých komponent v rámci jednoho čipu. Systém bude zajišťovat i napojení na systémovou sběrnici počítače, což umožní komunikaci interních komponent s ostatními prvky hostitelského systému. Navržená plně duplexní paketově-orientovaná sběrnice bude mít genericky nastavitelnou šířku. Díky tomu bude možné vybudovat jednotný hierarchický systém různě rychlých sběrnic pro připojení komponent s různými výkonnostními a funkčními požadavky. Zohledněna bude podpora přenosů inicovaných ze strany uživatelských komponent, realizace modelu rozdělených transakcí a další požadavky. I přes bohatou podporu různých vlastností a vysokou propustnost bude systém díky jednomu typu sběrnice (s různou datovou šířkou) velmi dobře spravovatelný a bude složen z malého počtu komponent. Důležitou vlastností bude také oddělení platformově závislých a nezávislých částí. Díky tomu bude možné jak připojení různých moderních systémových sběrnic, tak přenesení celého interního systému na jinou technologii.

Kapitola 4

Navržené řešení

Cílem této práce je především navrhnout a implementovat propojovací systém interních sběrnic pro čipy s technologií FPGA. Těmito čipy jsou osazovány např. akcelerační karty a adaptéry jako je ML555 nebo COMBO6X. Tuto situaci zachycuje obrázek 4.1. Akcelerační karta je připojena k hostitelskému počítači prostřednictvím systémové sběrnice, která zajišťuje vzájemnou komunikaci procesoru s ostatními prvky systému. Uvnitř čipu FPGA je připojen speciální řadič (ve formě komerčního modulu nebo vestavěné jednotky), který řídí komplexní komunikační protokol systémové sběrnice a poskytuje jednodušší rozhraní pro připojení komponent akcelerované aplikace. Toto rozhraní pak představuje přípojné místo pro navrhovaný propojovací systém.



Obrázek 4.1: Začlenění navrhovaného propojovacího systému v hierarchii počítače

Takový systém bude zajišťovat jak komunikaci mezi komponentami v rámci FPGA, tak mezi interními komponentami a prvky hostitelského systému. Zároveň bude důležité zajistit možnost iniciovat přenosy z obou stran a podporovat tak tzv. master přenosy. Spoje budou realizovány pomocí vysokorychlostních plně duplexních sběrnic postavených na paketové komunikaci, aby byl zajištěn současný přenos v obou směrech a byla dosažena co nejvyšší propustnost. Tomu napomůže i snížení citlivosti na vzdálenost, které umožní zvýšení pracovní frekvence. Zároveň bude implementován model rozdělených transakcí, aby při dlouhé latenci čtecích operací nedocházelo ke zbytečnému blokování sběrnice a snižování výkonnosti.

Celý systém bude flexibilní a dobře škálovatelný, nejen z pohledu datové šířky či množství připojených komponent. Kromě základní funkcionality bude totiž systém podporovat mnoho dalších přídavných vlastností jako např. konfigurovatelné buffery, striktní či uvolněné řazení transakcí, dekódování adresy či zarovnání dat. Příslušné vlastnosti bude možné genericky povolit či zakázat, což mimo jiné sníží množství využitých zdrojů na čipu. Návrhář tak získá větší prostor pro vlastní aplikaci.

Při návrhu takového systému se vycházelo z již existujících přístupů a principů, které byly popsány v kapitole 2. Na základě analýzy dostupných řešení shrnutých v kapitole 3 bylo dále možné identifikovat jejich nedostatky či chyby a při návrhu se jich vyvarovat. Získané poznatky byly konfrontovány se specifikovanými požadavky a vlastnostmi a na základě toho byla navržena konkrétní topologie propojovacího systému a její arbitrační schéma, architektura jednotlivých komponent, komunikační protokol a typy a formát transakcí, které bude systém interních sběrnic podporovat.

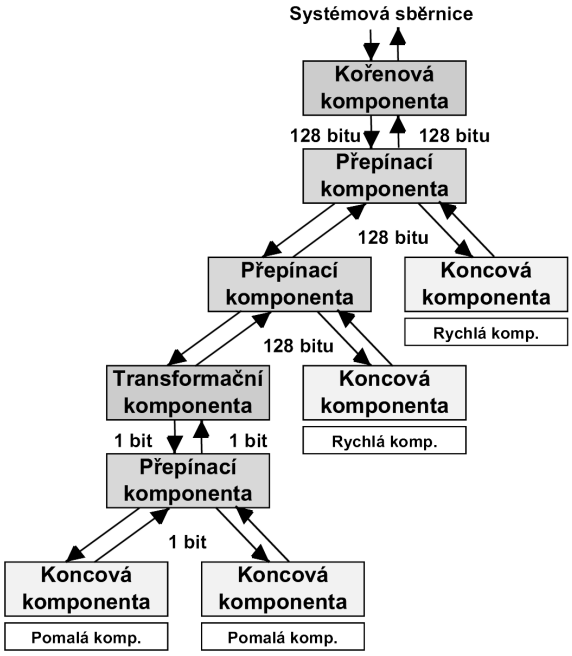
4.1 Topologie systému

V kapitole 2.1 byly popsány čtyři konkrétní sběrnicové topologie, které jsou díky svým vlastnostem použitelné pro cílovou technologii FPGA. Ukázali jsme si, že každá architektura může být modifikována pomocí různých speciálních komponent (např. registrovací nebo transformační komponenty). Cílem těchto úprav je většinou vylepšení výkonnostních parametrů nebo přidání další funkcionality. Každá z uvedených sběrnicových topologií může být tedy upravena tak, aby co nejlépe vyhovovala specifikovaným požadavkům.

Na základě analýzy požadavků kladených na systém (propustnost, podpora veškeré funkcionality, množství spotřebovaných zdrojů atd.) byl zvolen koncept přepínané topologie se zapojením do stromu. Místo centrálního způsobu arbitrace je využíváno efektivního distribuovaného směrovacího schématu. Prvky, které data pouze přenáší, jsou odděleny od těch, které zajišťují připojení samotných uživatelských komponent. Jejich počet není díky hierarchickému rozložení omezen. Stromová architektura navíc umožňuje nezávislou komunikaci komponent v nižších větvích stromu, aniž by bylo plýtváno šířkou pásma sběrnic na vyšší úrovni.

Základními prvky obecné stromové struktury jsou tzv. uzly. Rozlišujeme tři typy stromových uzlů: kořen, vnitřní uzly a koncové uzly (listy). Jak je možné vidět na obrázku 4.2, v našem kontextu odpovídají jednotlivé uzly příslušným komponentám propojovacího systému. Kořenová komponenta poskytuje vstup-výstupní rozhraní k hostitelskému systému, ke kterému je interní sběrnice připojena. Tato komponenta je platformově závislá a část její architektury je specifická pro každou systémovou sběrnici. Vlastní nezávislý propojovací systém se skládá z přepínací, koncové a transformační komponenty. Nejsložitější z nich je koncová komponenta. Jejím úkolem je řídit komunikační protokol interní sběrnice a transformovat transakce určené uživatelským komponentám na jednodušší paměťové rozhraní. Tří-portová přepínací komponenta provádí směrování transakcí a na základě cílové adresy přeposílá příchozí pakety ze vstupního na příslušné výstupní rozhraní.

Jednou z nejdůležitějších vlastností navrhovaného systému je genericky nastavitelná datová šířka a to individuálně pro každou její část (větev). Transformace mezi datovými šířkami je realizována pomocí transformační komponenty. Jak je patrné z obrázku 4.2, největší datová šířka je většinou u kořene stromu a připojuje rychlé komponenty. To se týká např. vyrovnávacích pamětí pro DMA (přímé přístupy do paměti) nebo řadičů dynamické paměti, jež mají široká rozhraní a vyžadují vysokorychlostní datové přenosy. Komponenty, které komunikují pouze zřídka a nevyžadují vysokou propustnost, jsou umístěny na niž-



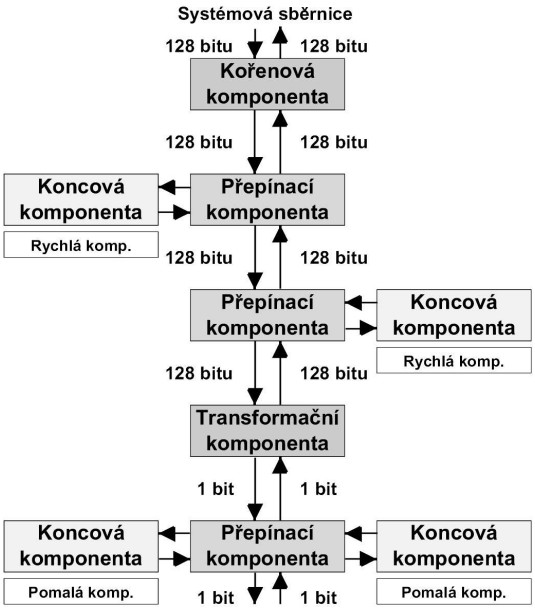
Obrázek 4.2: Příklad konkrétního zapojení stromové architektury

ších úrovních s menší datovou šířkou. Z té také vyplývá menší množství spotřebovaných zdrojů. Při návrhu konkrétního zapojení pro použití systému v dané aplikaci je nutné, aby její návrhář dbal na správné rozložení datových šířek ve stromu a byla tak zajištěna maximální propustnost a minimální množství využitých zdrojů na čipu. S ohledem na současné možnosti technologie FPGA a jednoduchost implementace je zvolen rozsah šířky datové sběrnice 1-128 bitů, přičemž konkrétní hodnoty musí být mocniny dvou.

Díky konfigurovatelné datové šířce a použití transformační komponenty je tak možné vybudovat jednotný hierarchický systém interních sběrnic s různými datovými šířkami, který umožňuje propojení různě rychlých komponent. Není potřeba spravovat mnoho typů sběrnic a komunikačních protokolů a celý propojovací systém je složen pouze ze čtyř univerzálních generických komponent (kořenová, přepínací, koncová a transformační komponenta).

Obrázek 4.3 ukazuje, že příklad stromové architektury z obrázku 4.2 je možné překreslit také jako sběrnici postupně registrovanou jednotlivými stupni ve formě přepínacích komponent. Tato vlastnost je v kontextu technologie FPGA velmi důležitá, protože široké sběrnice jsou obvykle citlivé na vzdálenost a nižší frekvence by pak znamenala i nižší propustnost.

S topologií sběrnicového systému souvisí i výběr vhodného schématu arbitrace. U stromové architektury se přímo nabízí model směrování transakcí, který je podobný směrování paketů v počítačových sítích. Pakety postupně prochází systémem od přepínače k přepínači. Každá přepínací komponenta zná adresové rozsahy svých portů a na základě této informace a cílové adresy provádí směrování. Toto schéma nepotřebuje k řízení žádný centrální arbiter, díky čemuž je takový systém velmi dobře škálovatelný a umožňuje připojení obecně libovolného množství komponent.



Obrázek 4.3: Příklad stromové architektury překreslené jako sběrnice

4.1.1 Vliv podpory master přenosů na rozmístění komponent

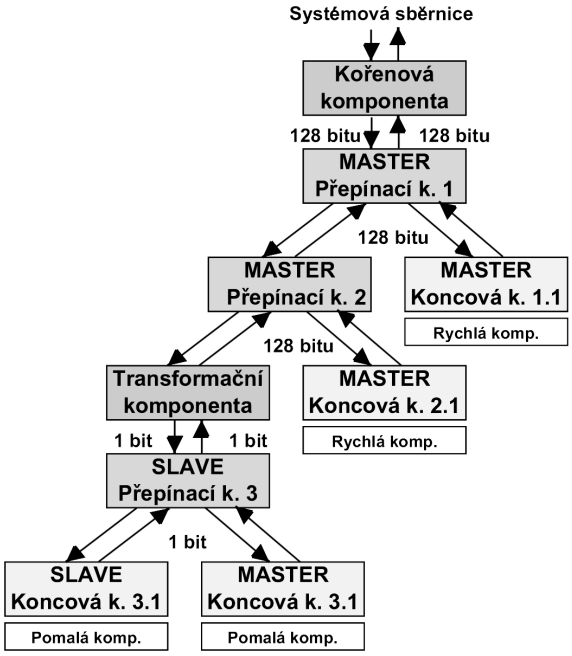
Velmi důležitou vlastností propojovacího systému je mimo jiné i schopnost komponent iniciovat datové přenosy (tzv. master přenosy). Tato přídavná funkcionalita však různou měrou komplikuje jejich architekturu a má za následek větší množství spotřebovaných zdrojů na čipu. Vzhledem k tomu, že podpora těchto přenosů není potřeba vždy a u všech komponent, je vhodné ji implementovat jako genericky nastavitelnou, aby mohla být podle potřeby povolena či zakázána. Komponenty tak mají dvě varianty: jestliže zahrnují podporu master přenosů, označují se jako master, v opačném případě pak jako slave.

Koncová komponenta poskytuje uživateli pro iniciování master přenosů speciální rozhraní. V případě slave varianty je toto rozhraní nedostupné a s ním se redukuje i logika, která měla iniciování master přenosů na starosti. Podobně je tomu i u kořenové komponenty, která rovněž obsahuje toto rozhraní.

Master a slave varianta přepínací komponenty se liší mnohem výrazněji. Master přepínač umožňuje směrování mezi všemi třemi porty a zabírá nezanedbatelné množství zdrojů na čipu. Slave varianta úplné směrování neprovádí. Pakety přicházející z vyšší úrovně přeposílá k oběma následníkům, zatímco všechny pakety jdoucí od následníků posílá do vyšší větve stromu. Vzhledem k tomu, že není třeba směrovat na základě cílové adresy, je logika mnohem jednodušší a úspora zdrojů dosahuje až 70 %. Podrobnější informace o přepínací komponentě a směrování jsou popsány v kapitole 4.5.2.

Poslední, transformační komponenta má vzhledem k tomu, že mění šířku dat bez ohledu na jejich obsah, pouze jednu variantu.

Skutečnost, zda komponenta podporuje či nepodporuje master přenosy, má vliv i na vhodné rozmístění komponent ve stromové struktuře. Jejich správným uspořádáním je totiž možné výrazně eliminovat množství spotřebovaných zdrojů na čipu. Rozhodující je použití master (směrování mezi všemi porty) či slave (pouhé přeposílání shora dolů a naopak) variant přepínacích komponent.



Obrázek 4.4: Příklad stromové architektury se slave i master komponentami

Návrhář konkrétní aplikace by se měl řídit podle následujícího základního pravidla: do větve za slave přepínač není možné umístit dvě koncové komponenty, které spolu komunikují. Je nutné je připojit za master variantu přepínače.

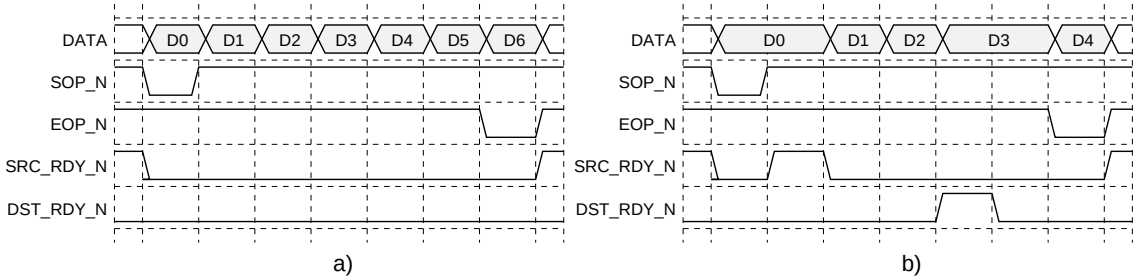
Správnost rozmístění komponent budeme demonstrovat na příkladu z obrázku 4.4. Znázorněná architektura obsahuje dvě master a jednu slave přepínací komponentu. Za slave přepínač č. 3 je připojena koncová komponenta 3.1. Ta může komunikovat např. s koncovou komponentou 2.1, protože obě se nachází ve větvi za společným master přepínačem č. 2, který provádí úplné směrování. Podobným způsobem může komunikovat i s koncovou komponentou 1.1. Naproti tomu není možná komunikace koncových komponent 3.1 a 3.2, protože se nachází ve větvi za společným slave přepínačem. Ten neprovádí směrování a neumožňuje přeposílání paketů mezi porty následníků.

4.2 Komunikační protokol

Pro každé rozhraní, které se v rámci propojovacího systému vyskytuje, je potřeba přesně definovat komunikační protokol, který určuje, jakým způsobem probíhá výměna dat. Nejdůležitější z nich je rozhraní vlastní interní sběrnice. Ta je tvořena dvěma plně duplexními směry a každý z nich je řízen pomocí paketově-orientovaného komunikačního protokolu.

Každý paket se skládá z řídící hlavičky a volitelně může obsahovat i vlastní užitečná data. Hlavička nese nezbytné kontrolní informace, na základě kterých se provádí např. směrování nebo generování čtecích a zápisových požadavků.

Časové diagramy komunikačního protokolu jsou znázorněny na obrázku 4.5. Jednotlivé pakety jsou označeny počáteční (signál SOF_N) a ukončovací značkou (signál EOF_N). Přenos dat je řízen pomocí signálů SRC_RDY_N a DST_RDY_N, které umožňují pozastavení komunikace buď ze strany příjemce, nebo vysílače.



Obrázek 4.5: Časové diagramy komunikačního protokolu interní sběrnice: (a) bez vkládání čekacích stavů, (b) s vkládáním čekacích stavů

Jednou z výhod tohoto komunikačního protokolu je, že vychází ze základního módu Xilinx Local Link protokolu [25]. Tento protokol je poměrně rozšířený a díky vzájemné kompatibilitě je tak možné na interní sběrnici přímo napojit různé komerční moduly (tzv. IP Core), které jej podporují.

Mezi další rozhraní vyskytující se v rámci propojovacího systému patří zápisové, čtecí a master rozhraní, která slouží k připojení uživatelských komponent. Tato rozhraní poskytuje koncová komponenta a jsou podrobněji popsána v kapitole 4.5.4.

4.3 Typy transakcí

V rámci adresového prostoru počítače probíhá veškerá výměna dat prostřednictvím zápisů a čtení z jednotlivých adres. Vysokorychlostní plně duplexní interní sběrnice umožňuje současný přenos dat v obou směrech. Aby výkonnost nebyla degradována dlouhou latencí čtecích operací a nedocházelo ke zbytečnému blokování sběrnice, je implementován tzv. model rozdělených transakcí. Mezi základní typy tak kromě zápisové a čtecí transakce patří i tzv. dokončovací transakce. Ta je generována jako odpověď na čtení a kromě požadovaných dat nese i příznak, který ji s původní čtecí transakcí spojuje.

Transakce interní sběrnice lze dále rozdělit do dvou základních skupin. První skupina zahrnuje lokální transakce, které realizují operace v rámci FPGA, druhou skupinou jsou transakce globální, které se užívají pro komunikaci s komponentami nadřazeného systému skrze platformově závislou kořenovou komponentu.

Lokální transakce slouží ke čtení či zápisu dat v rámci lokálního adresového prostoru. Typickým příkladem použití těchto transakcí jsou přenosy mezi interními komponentami v FPGA. Další jejich použití je v případě, že hostitelský systém iniciuje zápisovou nebo čtecí operaci. Pak je kořenová komponenta zodpovědná za jejich překlad právě na příslušné lokální transakce.

Globální transakce slouží k výměně dat mezi lokálním a globálním adresovým prostorem. Typickým příkladem použití těchto transakcí je případ, kdy uživatelská komponenta připojená k master verzi koncové komponenty potřebuje číst nebo zapisovat data do paměti hostitelského systému.

V kapitole 2 byly nastíněny některé problémy, s kterými je možné se v souvislosti se sběrnicovými systémy potkat. Mezi ně patří i problematika potvrzování transakcí. Jak je patrné, mezi navržené typy transakcí není zahrnuta zápisová potvrzovací (dokončovací) transakce. Obecně platí, že každý další mechanismus pro ošetřování chybových stavů vede k zesložitení logiky a větším požadavkům na spotřebované zdroje, což nemusí být vždy

odpovídající přidané hodnotě daného mechanismu. Vzhledem ke snaze vytvořit jednoduchý, lehce spravovatelný, ale také výkonný a na zdroje nenáročný systém není zápisová potvrzovací transakce implementována.

Dalším typem transakce, který by bylo možné potencionálně přidat, je paměťová bariéra, která by představovala hardwarovou podporu pro programové řešení pořadí čtecích a zápisových operací. Systém je navržen tak, aby bylo možné v případě potřeby tuto podporu jednoduše doimplementovat.

4.4 Formát paketu

Jak již bylo zmíněno, každý paket se skládá z hlavičky nesoucí řídící informace a volitelně může obsahovat i užitečná data zarovnaná podle cílové adresy. Na základě specifikace všech typů transakcí, které budou na interní sběrnici podporovány, je možné přesně definovat, jaké budou jednotlivé položky paketu a jak budou v hlavičce uspořádány. To je znázorněno na obrázku 4.6. Následuje popis významu jednotlivých položek.

Typ (4 bity) transakce určuje, jaké vlastnosti přenášený paket má a jakou operaci reprezentuje. Následující výčet udává zakódování typu a význam jednotlivých bitů:

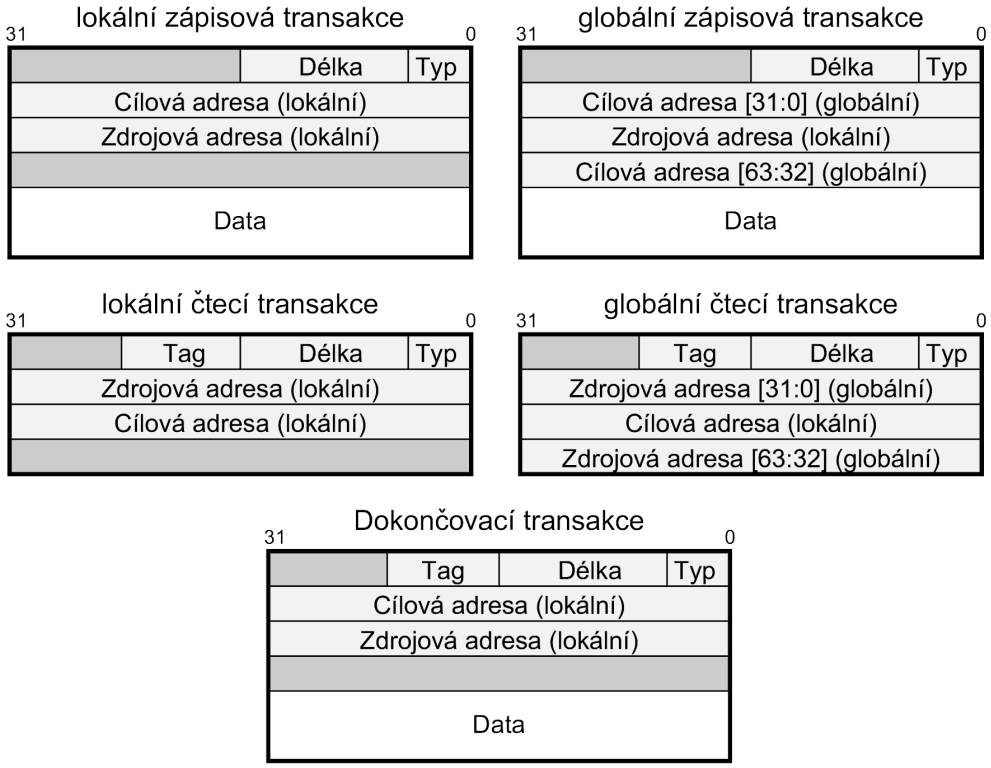
- 0. bit určuje, zda paket nese užitečná data,
- 1. bit určuje, zda se jedná o lokální (0) či globální (1) transakci,
- 2. bit určuje, zda se jedná o dokončovací transakci,
- 3. bit určuje, zda se jedná o poslední dokončovací transakci.

Poslední bit typu je nutný proto, protože odpověď na čtení se může obecně skládat z jedné a více dokončovacích transakcí.

Zdrojová a cílová adresa (32/64 bitů) určuje odkud a kam bude daný paket po sběrnici putovat. Globální adresa je 64 bitů široká stejně jako prostor celého systému počítače. Pokud by byl i lokální prostor na čipu FPGA stejně široký, musela by se zvýšit délka hlavičky. Situaci by zkomplikoval i fakt, že za běhu hostitelského systému se obecně mohou globální bázové adresy dynamicky měnit. Proto je lokální adresa vždy 32 bitů široká a stejně tak lokální adresový prostor propojovacího systému. Pokud by tento prostor přece jen nedostačoval, je možné rezervovat některé nevyužité bity hlavičky pro případné rozšíření lokální adresy. Při uspořádávání položek hlavičky se vycházelo mimo jiné z toho, že cílová adresa musí být vždy v jednom z prvních slov, aby byly přepínače schopny identifikovat cílový port pro směrování co nejdříve.

Tag (8 bitů) reprezentuje příznak, který jednoznačně identifikuje konkrétní čtecí požadavek a k němu příslušející odpověď. Protože mezi podporované typy transakcí není zahrnuta zápisová dokončovací transakce, nemá položka tagu pro zápis žádný význam. Proto může být ušetřené místo v hlavičce využito k případným dalším rozšířením.

Délka (12 bitů) udává počet bajtů užitečných dat, která jsou v paketu přenášena (v případě zápisové a dokončovací transakce) nebo která mají být přečtena (v případě čtecí transakce). Hodnota 0x000 udává maximální délku paketu, která je 4 KB. Tato délka odpovídá jedné stránce paměti počítače.



Obrázek 4.6: Formát paketů interní sběrnice

Vzhledem k uvedeným nezbytným řídícím informacím, které musí hlavička obsahovat, je její celková délka stanovena na fixních 128 bitů. Její šířka závisí na aktuální datové šířce interní sběrnice. Uvedený obrázek 4.6 představuje příklad hlaviček pro 32-bitovou konfiguraci šířky.

4.5 Architektura komponent

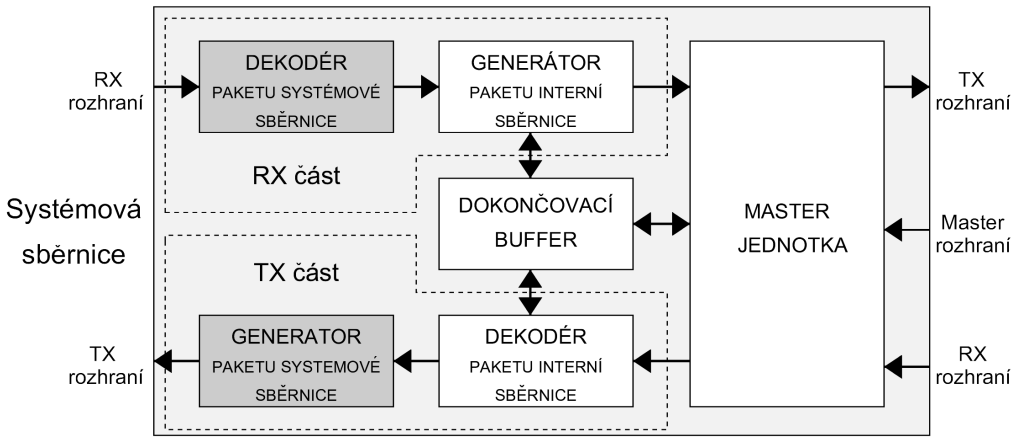
Navržený systém interních sběrnic umožňuje propojení libovolného počtu různě rychlých uživatelských komponent, které mohou mít různé požadavky. I přes dobrou škálovatelnost, flexibilitu a početné možnosti je propojovací systém složen pouze ze čtyř univerzálních generických komponent (kořenová, přepínací, koncová a transformační komponenta). Tato kapitola podrobněji popisuje architekturu jednotlivých komponent, specifikuje komunikační protokol jejich rozhraní a vysvětluje principy, které se při návrhu a implementaci uplatňují.

4.5.1 Kořenová komponenta

Nejvyšším uzlem stromové architektury je kořenová komponenta, která propojuje systém interních sběrnic s nadřazeným hostitelským systémem. Jejich prostředníkem je pak ještě již zmiňovaná speciální jednotka, která řídí komplexní nízkoúrovňový komunikační protokol systémové sběrnice a propojovacímu systému poskytuje jednodušší rozhraní na úrovni transakcí. Na základě analýzy dostupných komerčních modulů, které by pro tento účel mohly být využity, a vzhledem k požadované propustnosti u kořene stromu byla zvolena datová šířka interní sběrnice vedoucí z/do kořenové komponenty na fixní hodnotu 64 bitů.

Hlavní úlohou kořenové komponenty je obousměrný překlad mezi transakcemi interní sběrnice a nadřazeného systému. Z hostitelské strany jsou přijímány pakety systémové sběrnice. Z hlavičky paketu se extrahují jednotlivé řídící informace a transformují se do podoby, která odpovídá položkám hlavičky interní sběrnice. Provede se např. překlad globálních adres na lokální nebo adaptace délky paketu. Nakonec se dle formátu interních transakcí poskládá výsledný paket a je odeslán na interní sběrnici. V druhém směru jsou ze strany propojovacího systému přijímány pakety interní sběrnice. Ty jsou podobným způsobem jako v předchozím případě transformovány na pakety systémové sběrnice.

Další možností, kterou kořenová komponenta disponuje, je iniciování master přenosů ze strany čipu FPGA směrem k jiným zařízením namapovaným do paměti hostitelského systému. Cílovou adresou v případě čtení a zdrojovou adresou v případě zápisu je libovolná adresa globálního adresového prostoru. S ohledem na požadavky a účel konkrétní aplikace je zde dále možné implementovat například generování přerušení či jiné kontrolní a servisní mechanismy.



Obrázek 4.7: Architektura kořenové komponenty

Architektura kořenové komponenty je znázorněna na obrázku 4.7. Skládá se z přijímací (RX) a vysílací (TX) části, dokončovací vyrovnávací paměti a volitelné master jednotky.

Jestliže má být možné připojit propojovací systém interních sběrnic pomocí kořenové komponenty k libovolné sběrnici hostitelského systému (např. PCI, PCI-X, PCI Express a další), musí být některé části kořenové komponenty platformově závislé. Tyto části jsou na obrázku 4.7 zobrazeny tmavou barvou. Jedná se o jednotky, které jsou zodpovědné za rozkládání a sestavování paketů systémové sběrnice. Aby bylo možné celý navržený propojovací systém otestovat, byly jako platformově závislá část kořenové komponenty implementovány jednotky pro práci s PCIe pakety. Díky tomu je možné systém testovat např. na kartě ML555, která disponuje PCIe konektorem.

Bloky, které slouží k rozkládání a sestavování paketů interní sběrnice, patří do platformově nezávislé části kořenové komponenty. Tyto jednotky komunikují s tzv. dokončovací vyrovnávací pamětí (buffrem), která představuje komunikační bod pro přijímací a vysílací část. Dokončovací buffer slouží k ukládání identifikačních informací, které přísluší jednotlivým čtecím transakcím. V okamžiku, kdy přijde odpovídající dokončovací transakce, se na základě tagu uložené informace vyzvednou a použijí se při překladu mezi transakcemi interní sběrnice a hostitelského systému. Tento princip funguje jak při čtení z lokálního, tak globálního adresového prostoru.

Poslední platformově nezávislou komponentou je master jednotka zajišťující iniciování přenosů skrze master rozhraní. To je postaveno na stejném komunikačním protokolu jako rozhraní interní sběrnice a je podrobněji popsáno v kapitole 4.5.4. Master jednotka tak v obou směrech provádí pouze multiplexování mezi pakety interní a master sběrnice. Slave a master varianta kořenové komponenty je rozlišena právě existencí této jednotky a jejího rozhraní.

4.5.2 Přepínací komponenta

Základní úlohou přepínací komponenty je směrování paketů stromovou topologií. Směrovací schéma arbitrace umožňuje postupné distribuování paketu skrze jednotlivé přepínače až do cílového místa. Díky přepínací komponentě je možné stromovou strukturu libovolně větvit, což poskytuje dobře škálovatelné řešení. Rozhraní představují tři porty. Jeden slouží k připojení předchůdce (tzv. upstream port) a dva k připojení potomků (tzv. downstream porty).

Přepínací komponenta je navržena ve dvou variantách: master a slave. Jednotlivé varianty se od sebe podstatně liší a to především různým chováním (směrovací strategií), složitostí architektury a potažmo i výrazným rozdílem v množství spotřebovaných zdrojů na čipu FPGA. Tuto úsporu prostředků lze v rámci celého systému mnohonásobně navýšit, jestliže jsou vybrané master a slave přepínače vhodně rozmístěny ve stromové struktuře. To již bylo podrobněji popsáno v kapitole 4.1.1. Vzhledem k výrazné odlišnosti obou variant bude master i slave přepínač popsán samostatně.

Master varianta

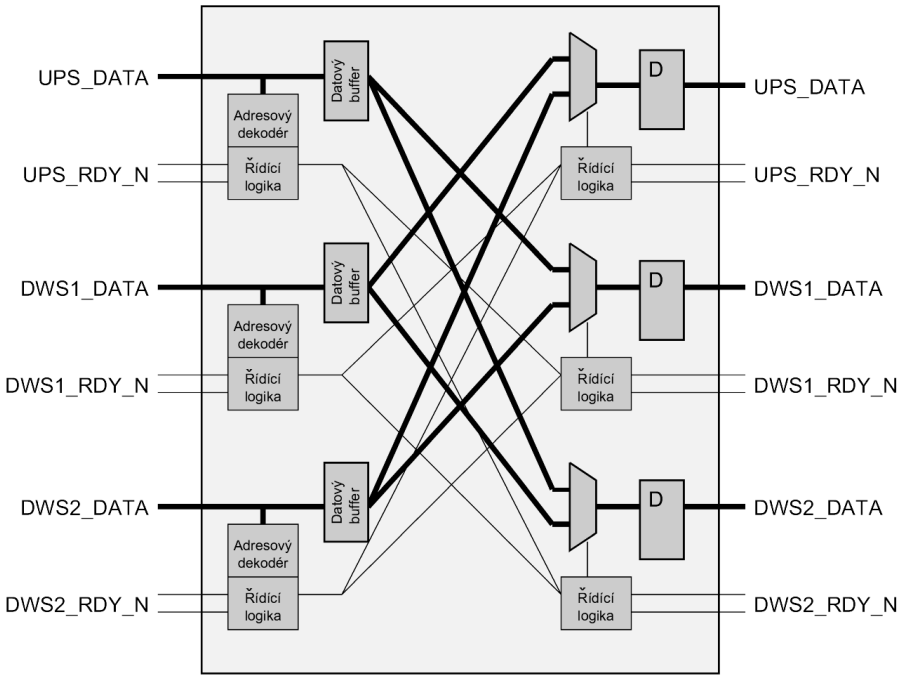
Aby mohla být zajištěna komunikace mezi komponentami, které se nachází ve stejném podstromu a mohly tak být mezi nimi realizovány master přenosy, musí se v kořeni dané větve nacházet master přepínač, který provádí úplné směrování mezi všemi třemi porty. To znamená, že příchozí paket ze vstupního rozhraní je analyzován a na základě cílové adresy je určen výstupní port, na který je paket přeposlán. V případě, že se jedná o globální transakci přicházející z downstream portu, je paket automaticky směrován na upstream port. Vzhledem k tomu, že výstupní rozhraní je vybíráno mimo jiné i na základě cílové adresy, je nutné, aby přepínač znal rozsahy adresových prostorů na jednotlivých portech. Tyto rozsahy jsou komponentě předány prostřednictvím generických parametrů. Přesná směrovací strategie, která je implementována v master variantě přepínací komponenty je následující:

Příchod paketu z downstream portu:

- Pokud cílová adresa paketu spadá do rozsahu sousedního downstream portu, pak je transakce směrována na tento port.
- Pokud cílová adresa paketu spadá do rozsahu upstream portu nebo se jedná o globální transakci nesoucí adresu globálního prostoru, pak je transakce směrována na upstream port.
- V jiných případech je paket zahozen. Nemůže být směrován na neexistující adresu ani na stejný port, ze kterého přišel.

Příchod paketu z upstream portu:

- Pokud cílová adresa paketu spadá do rozsahu některého z downstream portů, pak je transakce směrována na tento port.
- V jiných případech je paket zahozen. Nemůže být směrován na neexistující adresu ani na stejný port, ze kterého přišel.



Obrázek 4.8: Architektura master varianty přepínací komponenty

Na obrázku 4.8 je znázorněna architektura master varianty přepínací komponenty. U každého vstupního rozhraní se nachází vyrovnávací paměť, do které je příchozí paket postupně nahráván. V průběhu ukládání jsou data monitorována adresovým dekodérem, jehož úkolem je určit, na jaký výstupní port bude paket přeposlán.

Vzhledem k tomu, že šířka vstupní sběrnice může být v rozsahu 1-128 bitů, není vždy zaručeno, že celá cílová adresa bude k dispozici v jednom slově. V takovém případě je adresa přenášena po částech a postupně je porovnávána s příslušnými bity adresových konstant omezujících adresové prostory jednotlivých portů. Díky průběžnému udržování informace o výsledku porovnání předchozí části je možné rozhodnout, do rozsahu jakého výstupního portu cílová adresa spadá. Sekvenční porovnávání po menších částech navíc umožňuje dosáhnout vyšší frekvenci a není náročné na spotřebované zdroje.

Na základě dekódování a porovnání cílové adresy a příznaku určujícího, zda se jedná o globální transakci, je podle výše uvedené směrovací strategie vybrána akce, která se provede. Tu pak vykonává výstupní část cílového portu přepínače. První možností je, že příchozí paket má být zahozen. Druhou možností je výběr paketu z jednoho ze dvou rozhraní a jeho odbavení na výstupní port. Strategie výběru mezi dvěma rozhraními implementovaná ve výstupní řídící jednotce je založena na cyklické prioritě (algoritmus Round Robin).

Slave varianta

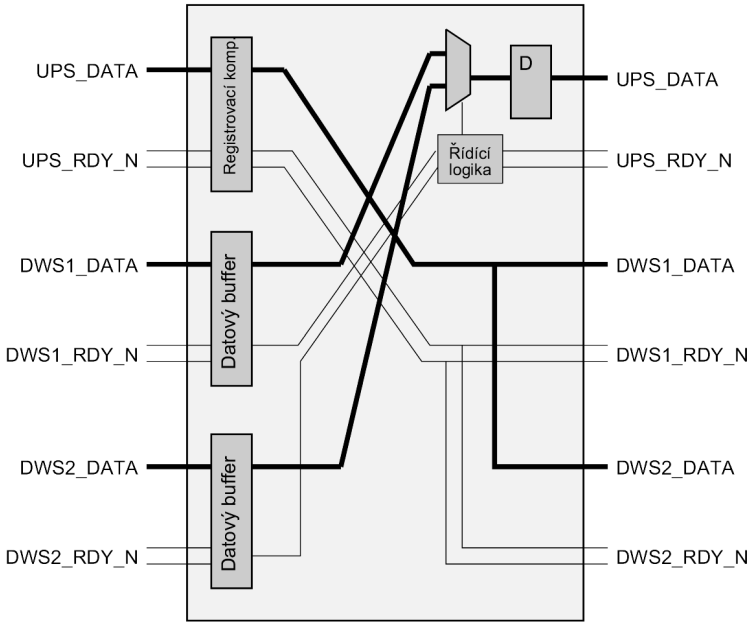
Jestliže není potřeba, aby spolu v daném podstromu mohly jednotlivé komponenty komunikovat, může být v kořeni této větve umístěna pouze slave varianta přepínací komponenty. Takový přepínač neprovádí úplné směrování mezi všemi porty a ke svému rozhodování nepotřebuje žádné řídící informace z hlavičky paketu. Díky tomu odpadá nutnost monitorovat přijímaná data, extrahovat z nich cílovou adresu a porovnávat ji s adresovými konstantami. Není tak ani nutné, aby přepínač znal rozsahy adresových prostorů na jednotlivých portech. Přesná směrovací strategie, která je implementována ve slave variantě přepínací komponenty je následující:

Příchod paketu z downstream portu:

- Jestliže paket přichází z jednoho z downstream portů, pak je transakce automaticky směrována na upstream port.
- V případě, že v jeden okamžik čekají na příjem pakety na obou downstream portech, pak je jeden z nich vybrán na základě strategie cyklické priority.

Příchod paketu z upstream portu:

- Jestliže paket přichází z upstream portu, pak je transakce automaticky směrována na oba downstream porty.



Obrázek 4.9: Architektura slave varianty přepínací komponenty

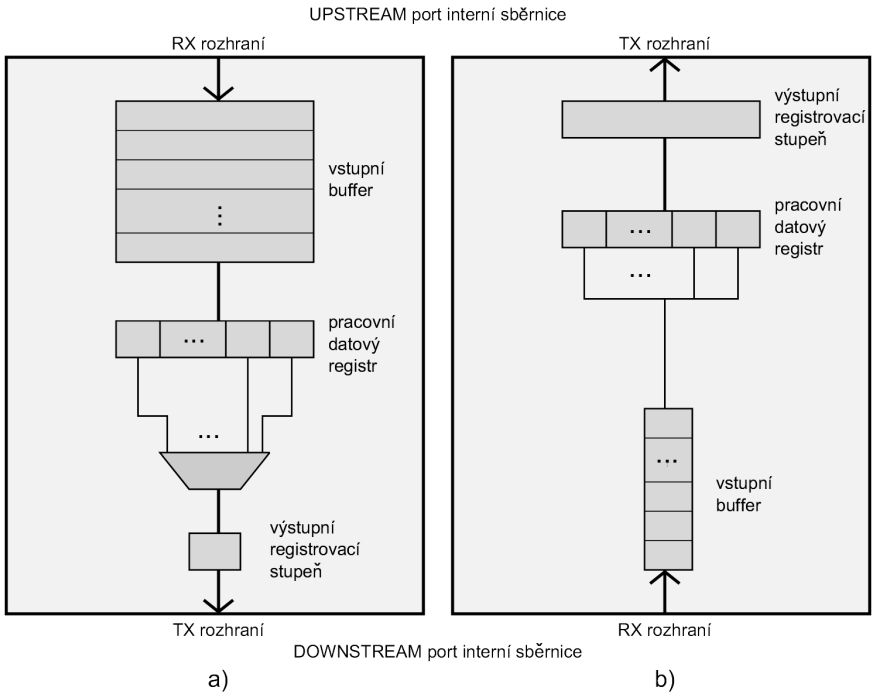
Implementované směrovací schéma umožňuje přenosy iniciované z komponent ve vyšších částech stromu (nebo mimo lokální systém na čipu), které směřují ke komponentám umístěným na nižších úrovních. Stejně tak je možné vysílat pakety i druhým směrem, z níže umístěných komponent do vyšších úrovní (nebo mimo propojovací systém na čipu). Není

možná pouze vzájemná komunikace v rámci daného podstromu. Je patrné, že i při použití slave přepínacích komponent je možné realizovat master přenosy. Je však důležité, jakým způsobem jsou komunikující komponenty rozmístěny.

Na obrázku 4.9 je znázorněna architektura slave varianty přepínací komponenty. Na první pohled je patrné, že ve srovnání s master variantou obsahuje podstatně méně logiky. Ve vstupní části odpadají adresové dekodéry a řídící jednotky, zůstávají pouze vyrovnávací paměti (buffery) na downstream portech a buffer pro upstream rozhraní je nahrazen registrovacím stupněm. Ve výstupní části zcela chybí veškerá logika pro downstream porty, logika výstupní části upstream rozhraní zůstává zachována. Díky těmto redukcím architektury, které má za následek především absence úplného směrovacího schématu, dosahuje úspora zdrojů pro některé datové šířky při použití slave přepínače až 70 %.

4.5.3 Transformační komponenta

Transformační komponenta provádí konverzi datové šířky interní sběrnice. V rámci propojovacího systému je jedinou komponentou, jejíž vstupem a výstupem je rozhraní s odlišnou datovou šířkou. To je výhodné zejména z pohledu modulárnosti a škálovatelnosti celého systému, ale také z hlediska jednoduchosti návrhu a množství využitých zdrojů u jednotlivých komponent. Kořenová, přepínací i koncová komponenta má tedy vstupy i výstupy vždy na stejné datové šířce. V případě, že návrhář konkrétní aplikace potřebuje založit podstrom sběrnice na nižší datové šířce, zapojí před vlastní přepínač transformační komponentu s příslušně nastavenými generickými parametry.



Obrázek 4.10: Architektura transformační komponenty:
(a) směr dolů, (b) směr nahoru

Použití transformační komponenty je z pohledu celé hierarchie systému velmi důležité. Díky konfigurovatelné datové šířce je možné vybudovat jednotný hierarchický systém inter-

ních sběrnic s různými datovými šířkami, který umožňuje propojení různě rychlých komponent. Celý systém je tedy možné nastavit tak, aby splňoval požadovanou propustnost a šířku pásma a zároveň výrazným způsobem eliminoval množství spotřebovaných zdrojů na čipu.

Vzhledem k tomu, že interní sběrnice je plně duplexní a má dva nezávislé směry, se architektura transformační komponenty znázorněná na obrázku 4.10 skládá ze dvou základních jednotek. Jedna slouží pro převod (postupné multiplexování) větší datové šířky na menší a v kontextu stromové topologie můžeme hovořit o směru dolů. Druhá zajišťuje konverzi (postupné seskládávání) menší datové šířky na větší a zpracovává tak pakety, které jdou směrem nahoru.

Důležitou součástí obou jednotek jsou vstupní vyrovnávací paměti. Různým datovým šířkám totiž přísluší také odpovídající šířky pásma. Při transformaci rychlejší sběrnice na pomalejší je nutné přicházející transakce ukládat, aby nebyla rychlejší sběrnice zbytečně blokována. Stejně tak ve směru opačném je potřeba vstupní slova uložit, aby mohla být celá transakce nepřerušovaně předána na rychlejší sběrnici. Velikost obou vyrovnávacích pamětí je konfigurovatelná a měla by být nastavena na hodnotu maximální velikosti transakce, která může transformační komponentou v daném podstromu procházet.

Na výstupních rozhraních je možné genericky povolit či zakázat registrovací stupeň. Ten v případě potřeby zlepší časování při napojení transformační komponenty k ostatním částem systému.

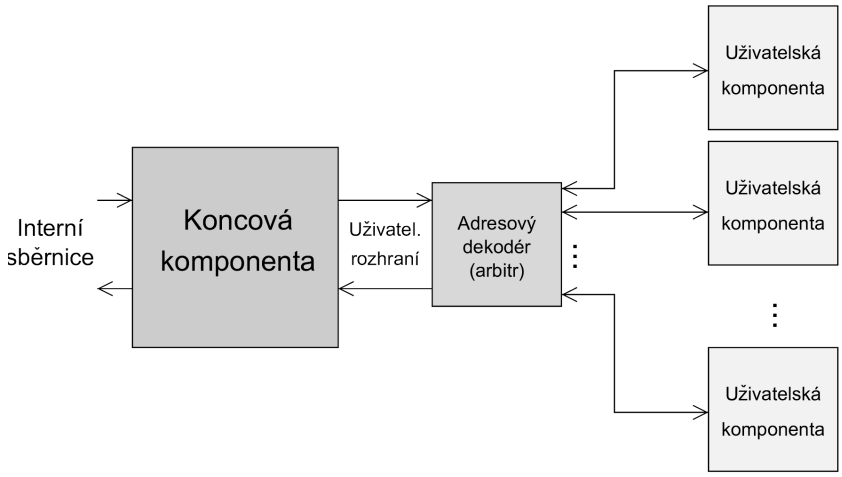
4.5.4 Koncová komponenta

Koncový uzel (list) stromové topologie je v rámci propojovacího sběrnicového systému reprezentován tzv. koncovou komponentou. Jedná se o relativně složitou jednotku, která slouží k připojení uživatelských komponent k propojovacímu systému. Jejím úkolem je řídit sériový komunikační protokol interní sběrnice a transformovat paketové transakce určené uživatelským komponentám na jednodušší paralelní paměťové rozhraní. Tato rozhraní (stejně jako rozhraní interní sběrnice) podporují rozdělený model transakcí, aby mohlo být dosaženo co nejvyšší propustnosti a byla eliminována případná velká latence čtecích operací. Důležitá je také schopnost koncové komponenty iniciovat master přenosy.

Připojované uživatelské komponenty mají většinou velmi rozdílné požadavky např. na rychlost zpracování, propustnost nebo zarovnanost paměťových přístupů. Zároveň se vyznačují různými vlastnostmi (latence čtecích operací, potřeba iniciovat datové přenosy atd.), které určují, jaká funkcionalita by měla být v přípojném bodě implementována. Ne vždy je však nutné podporovat vlastnosti všechny nebo s těmi nejvyššími výkonnostními parametry, a proto je žádoucí, aby mohly být jednotlivě povoleny, zakázány či nastaveny na odpovídající hodnotu. To je výhodné zejména z pohledu množství spotřebovaných zdrojů.

Mezi nastavitelné vlastnosti koncové komponenty patří např. šířka datové a adresové sběrnice nebo velikost vyrovnávacích pamětí. Jejich výběr závisí především na požadavcích na rychlost zpracování. Dále lze rozlišovat mezi dvěma typy čtecích rozhraní. První slouží k jednorázovému okamžitému odbavení požadavku, zatímco druhé vyřizuje čtecí požadavek postupně a adresa se inkrementuje po jednotlivých slovech. Podrobnější popis čtecího rozhraní je zmíněn v sekci Uživatelská rozhraní.

Důležitou nastavitelnou vlastností je také podpora master transakcí, která rozlišuje mezi master a slave variantou koncové komponenty. V případě master varianty je kromě čtecího a zápisového rozhraní k dispozici rovněž tzv. master rozhraní, skrze které je možné iniciovat datové přenosy směřované k ostatním komponentám na čipu nebo mimo FPGA.



Obrázek 4.11: Připojení více uživatelských komponent k jedné koncové komponentě

Data transakcí interní sběrnice jsou zarovnaná na cílovou adresu. V případě zápisové transakce tak není nutné data přerovnávat a stačí pouze generovat příslušné povolovací signály pro jednotlivé bajty (tzv. byte enably), jejichž počet vychází z šířky datové sběrnice. U čtecích operací jsou data čtena ze zdrojové adresy, a proto je třeba je přerovnat na adresu cílovou. Některé uživatelské komponenty podporují pouze zarovnané přístupy, a proto je umožněno generické odpojení či připojení přerovnávacího obvodu.

Na úrovni koncové komponenty je rovněž vhodné řešit problém spojený s pořadím čtecích a zápisových operací. S ohledem na jednoduchost a množství zabraných zdrojů byl zvolen tzv. striktní model. Ten neumožňuje současné provádění čtecích a zápisových operací. Začátek následující operace tak nemůže začít dříve, než skončí operace předchozí. Jednotku kontrolující pořadí jednotlivých transakcí lze buď povolit (striktní model) nebo zakázat (uvolněný model). Další variantou, která by vzhledem ke složitosti a náročnosti na zdroje přicházela v úvahu, je podpora pro mechanismus paměťových bariér. Ten je v případě potřeby možné relativně jednoduše doimplementovat.

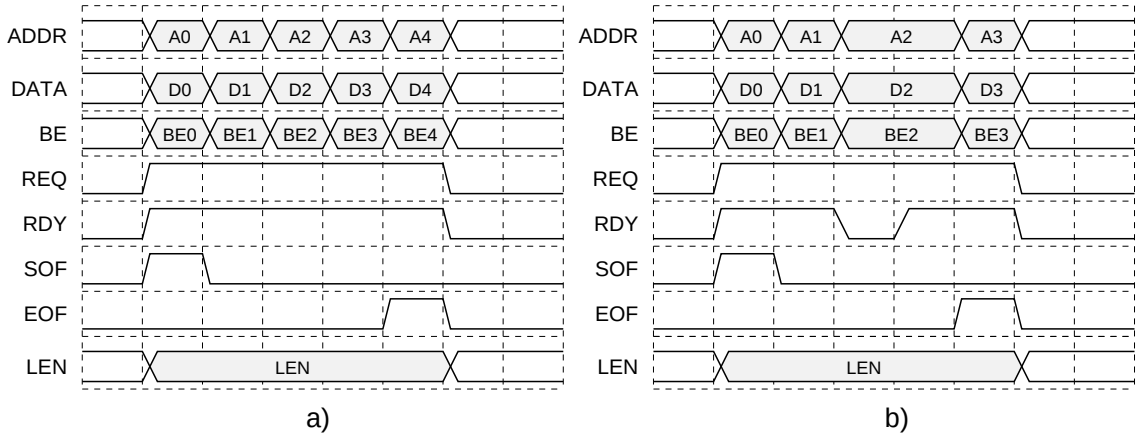
Vypínatelnou jednotkou koncové komponenty je také adresový dekodér. Ten je nutné povolit v případě, kdy se komponenta nachází v podstromu, v jehož kořeni je slave varianta přepínací komponenty. Ta neprovádí směrování na základě cílové adresy a tudíž adresový dekodér neobsahuje. Transakce jsou z upstream portu přeposílány na oba downstream porty a je až na koncových komponentách, aby příchozí paket zpracovaly nebo zahodily (v případě, že cílová adresa nespadá do jejich adresového prostoru).

V souvislosti s různými požadavky uživatelských komponent je ještě vhodné zmínit, že více komponent, které nevyžadují rychlé datové přenosy, je možné umístit za jednu koncovou komponentu. Stačí přidat pouze jednoduchý adresový dekodér, který zajišťuje distribuci požadavků jednotlivým komponentám. Tento případ znázorňuje obrázek 4.11. Můžeme tak říci, že na koncovou komponentu je napojená jednoduchá klasická sběrnice s centrálním arbitrem ve formě adresového dekodéru. Takové řešení rovněž vede k výrazné redukci spotřebovaných zdrojů.

Uživatelská rozhraní

Koncová komponenta poskytuje tři základní uživatelská rozhraní, která slouží pro připojení uživatelských komponent. Jedná se především o čtecí a zápisové rozhraní. Jejich datová šířka je omezena na rozsah 8-128 bitů, protože nejmenší adresovatelnou jednotkou je 1 bajt (8 bitů). Master verze koncové komponenty poskytuje dále speciální stejně široké rozhraní pro iniciování master přenosů, tzv. master rozhraní.

Zápisové rozhraní umožňuje zápis dat do adresového prostoru uživatelské komponenty. Zápisová transakce začíná aktivováním žádosti pomocí signálu REQ. V případě, že je tento signál aktivní, jsou platná i data, adresa a povolovací signály (BE). V okamžiku, kdy je uživatelská komponenta schopna přijímat data, nastavuje signál RDY. V případě, že není tento signál nastaven okamžitě, dochází ke vkládání tzv. čekacích stavů. K přenosu dat dochází pouze v případě, že je jak signál REQ, tak RDY aktivní. Zápisová operace je ohraničena signály pro začátek (SOF) a konec transakce (EOF). Po celou dobu jejího průběhu je uživatelské komponentě poskytována informace o délce zapisovaných dat v bajtech (LEN). Příslušné časové diagramy jsou znázorněny na obrázku 4.12.



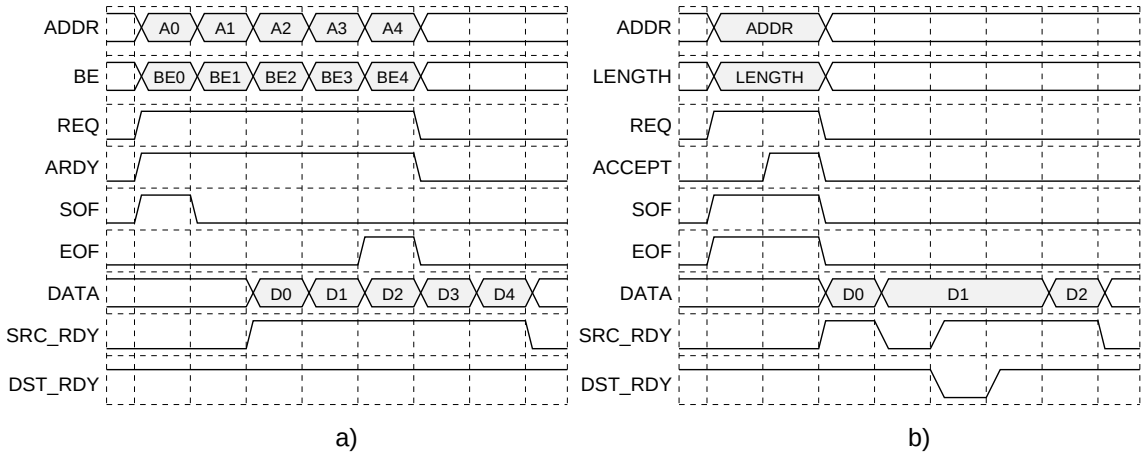
Obrázek 4.12: Časové diagramy zápisové transakce:
(a) bez vkládání čekacích stavů, (b) s vkládáním čekacím stavů

Čtecí rozhraní umožňuje čtení dat z adresového prostoru uživatelské komponenty. Vzhledem k tomu, že navržený propojovací systém podporuje rozdělený model transakcí, je čtecí (generování čtecího požadavku) a dokončovací (příjem přečtených dat) část rozhraní naprosto nezávislá a komunikace může probíhat souběžně.

Aby se systém vypořádal s dlouhými latencemi některých uživatelských komponent, rozlišujeme dále dva druhy čtení: kontinuální a rozdělené. U kontinuálního čtení jsou čtena data postupně, z inkrementující se adresy. V případě, že uživatelská komponenta má velkou latenci přečtených dat, je vhodná varianta rozděleného čtení, kdy je požadavek ihned odbaven, interní sběrnice není blokována kvůli postupnému inkrementování adresy a další čtení nebo zápis může okamžitě následovat. Každá koncová komponenta umožňuje generické nastavení jedné či druhé varianty.

Čtecí transakce začíná aktivováním žádosti pomocí signálu REQ. V případě, že je tento signál aktivní, je platná i adresa a povolovací signály (BE). V okamžiku, kdy je uživatelská komponenta schopna přijímat data, nastavuje signál ARDY (kontinuální čtení) nebo ACCEPT (paketové čtení). Pokud není tento signál nastaven okamžitě, dochází ke vklá-

dání tzv. čekacích stavů. Čtecí požadavek je ohraničen signály pro začátek (SOF) a konec transakce (EOF). Příchozí přečtená data jsou indikována nastaveným signálem SRC_RDY a mohou být pozdržena aktivováním signálu DST_RDY. Po celou dobu průběhu transakce je uživatelské komponentě poskytována informace o délce zapisovaných dat v bajtech (LEN). Příslušné časové diagramy jsou znázorněny na obrázku 4.13.



Obrázek 4.13: Časové diagramy čtecí transakce:
(a) kontinuální čtení, (b) paketové čtení

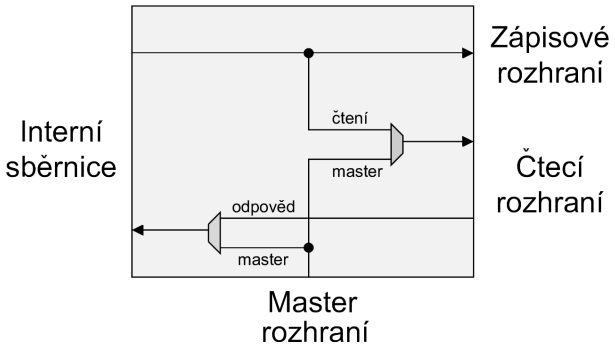
Master rozhraní umožňuje iniciování přenosu dat (čtení nebo zápisu) mezi uživatelskou komponentou, která je ke koncovému bodu připojena, a mezi libovolnou komponentou v rámci globálního adresového prostoru celého systému (paměti počítače). Jestliže je cílem transakce interní komponenta propojovacího systému, nazýváme ji lokální, v případě, že směřuje mimo čip FPGA, jedná se o globální transakci.

Komunikační protokol tohoto rozhraní je totožný s protokolem interní sběrnice popsaným v kapitole 4.5. Jednotlivé typy master transakcí a jejich položky navíc přesně odpovídají hlavičkám příslušných transakcí interní sběrnice. Formáty následujících dvojic transakcí nebo jejich částí jsou si tedy rovny:

- master lokální čtecí transakce a lokální čtecí transakce interní sběrnice,
- master lokální zápisová transakce a hlavička lokální zápisové transakce interní sběrnice,
- master globální čtecí transakce a globální čtecí transakce interní sběrnice,
- master globální zápisová transakce a hlavička globální zápisové transakce interní sběrnice.

Díky tomu, že jak komunikační protokol, tak formát master transakcí je stejný jako u rozhraní interní sběrnice, stává se návrh a implementace koncové komponenty výrazně jednodušší. Situaci demonstrujeme na obrázku 4.14. Nákres znázorňuje zjednodušenou blokovou strukturu koncové komponenty a jednotlivá rozhraní. Čtecí master transakce může díky zvolenému formátu koncovou komponentou projít rovnou na interní sběrnici. Jedinou překážkou je logika, která vybírá na základě strategie Round Robin mezi odpověďmi na čtení a master čtecími transakcemi. V případě zápisové master transakce je situace jen

o něco složitější. Za její hlavičku je navíc nutné připojit zapisovaná data, která se před odchodem transakce na interní sběrnici přečtou z uživatelské komponenty.



Obrázek 4.14: Zjednodušená bloková struktura koncové komponenty

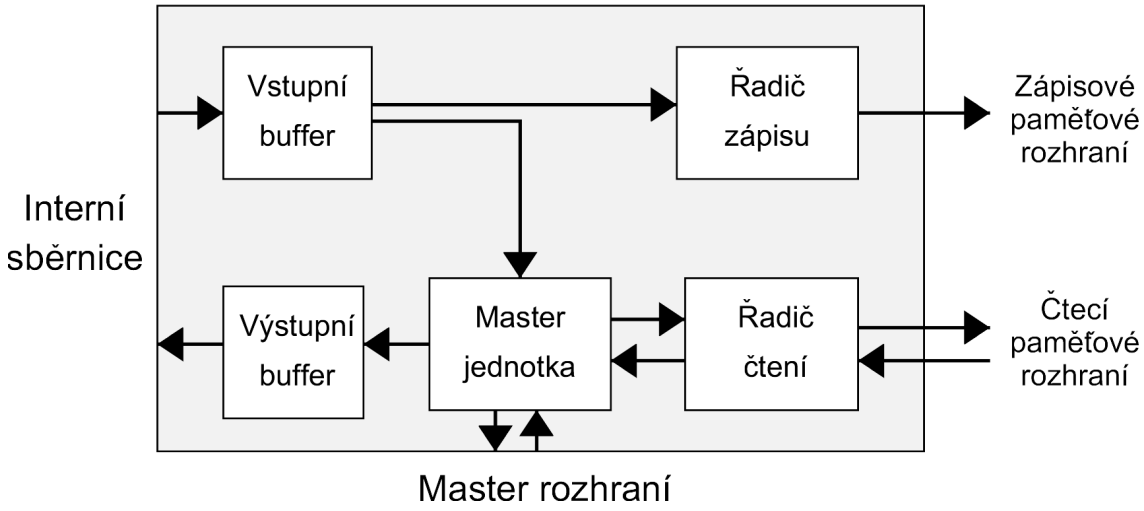
Bloková struktura

Na obrázku 4.15 je znázorněna bloková struktura koncové komponenty. Vzhledem k tomu, že představuje nejsložitější komponentu propojovacího systému, byl kladen velký důraz na modularitu, ale také množství spotřebovaných zdrojů. Navržená architektura je složena z pěti hlavních funkčních jednotek: vstupní a výstupní vyrovnávací paměť, řadič čtení a zápisu a také master jednotka. Velkou výhodou je, že všechny bloky spolu komunikují prostřednictvím stejného paketové protokolu jako je protokol interní sběrnice. Datové cesty lze jednoduše registrovat a nízkým šířkám odpovídá také menší množství zdrojů spotřebovaných na jednotlivé registrovací stupně. Data se tak v co největší míře udržují v paketové formě a provádí se extrakce jen těch řídících položek hlavičky, které jsou nezbytně nutné pro provedení čtecí nebo zápisové operace.

Vstupní vyrovnávací paměť slouží primárně k ukládání příchozích požadavků do vstupní fronty, aby nebyla interní sběrnice zbytečně blokována. Dále je tok transakcí rozdělen na dva proudy: čtecí a zápisový. Formát čtecích transakcí je pak transformován na formát hlavičky příslušných dokončovacích transakcí. Jedná se pouze o změnu typu transakce a o prohození položek zdrojové a cílové adresy. Součástí tohoto vstupního bloku může být volitelně i adresový dekodér a tzv. striktní jednotka. Adresový dekodér určuje, zda cílová adresa příchozí transakce skutečně spadá do adresového prostoru koncové komponenty. Pokud tomu tak není, transakce se zahodí. Povolená striktní jednotka monitoruje čtecí a zápisový proud transakcí a zajišťuje dodržování striktního pořadí.

Řadič zápisu provádí transformaci příchozích zápisových transakcí na zápisové požadavky pro uživatelskou komponentu. V průběhu příjmu hlavičky jsou extrahovány potřebné řídící položky (adresa a délka). Data jsou pak distribuována skrze řadič přímo na zápisové rozhraní a spolu s nimi také inkrementující se adresa, délka a generované povolovací signály pro jednotlivé bajty dat.

Nejsložitější jednotkou koncové komponenty je řadič čtení. Vzhledem k implementovanému modelu rozdělených transakcí se skládá ze dvou nezávislých částí: čtecí a dokončovací. Čtecí část extrahuje z příchozích transakcí potřebné řídící informace pro generování příslušného čtecího požadavku k uživatelské komponentě a zároveň tyto informace ukládá. Spolu s nimi je uložena i příchozí paketová transakce (ve vstupní vyrovnávací paměti již transformovaná na formát hlavičky dokončovací transakce). Dokončovací část přijímá přečtená



Obrázek 4.15: Bloková struktura koncové komponenty

data a v případě potřeby je na základě uložených řídících informací z hlavičky přerovnává. Pak je připojuje za příslušnou dokončovací hlavičku a celou transakci posílá na výstup.

Master jednotka je součástí pouze master varianty koncové komponenty a zajišťuje podporu přenosů iniciovaných ze strany připojené uživatelské komponenty. Tento blok se vloží mezi vstupní a výstupní vyrovnávací paměť a řadič čtení. Jeho úkolem je multiplexování paketové datové sběrnice ve dvou směrech: jednak směrem k řadiči čtení mezi čtecími požadavky ze vstupního bloku a master zápisovými transakcemi (které také potřebují přečíst data) a jednak směrem k výstupnímu bloku mezi dokončovacími transakcemi a master čtecími požadavky (čte se ze vzdálené komponenty). Tento scénář byl již popsán v souvislosti s master rozhraním a byl demonstrován na obrázku 4.14.

Poslední hlavní jednotkou koncové komponenty je výstupní vyrovnávací paměť. Jedinou její úlohou je ukládat odchozí transakce do výstupní fronty.

Kapitola 5

Dosažené výsledky

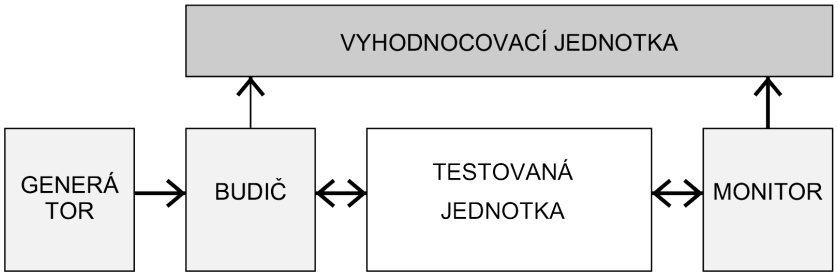
Aby bylo možné zhodnotit výsledky, kterých navržený a implementovaný systém dosahuje, a mohli jsme je porovnat dle různých parametrů s dostupnými řešeními, musely být nejprve řádně otestovány jednotlivé dílčí komponenty systému. Základní funkčnost kořenové, přepínací, transformační a koncové komponenty byla nejprve ověřena v simulacích. Na ně navázaly komplexní verifikační testy zaručující správnost chování komponent za všech možných podmínek. Nakonec byla vytvořena ukázková architektura sběrnicového systému, která realizovala propojení několika uživatelských komponent. Tato architektura byla syntetizována do podoby konfiguračního souboru pro programovatelné hradlové pole FPGA, v kterém pak byla funkčnost obvodu znovu ověřena.

Celý systém byl implementován v jazyce VHDL [10], zatímco simulace byly programovány pomocí relativně nového a zatím málo rozšířeného jazyka SystemVerilog [8]. Tento jazyk využívá objektově-orientovaný model a mnoho dalších moderních vlastností a přístupů (např. Constrained Random Variables, Coverage a další), jejichž aplikace je vysoce efektivní zejména pro účely simulace a verifikace.

Pro simulaci každé komponenty bylo vytvořeno speciální simulační prostředí, které se kromě vlastní testované jednotky skládá ještě z generátoru, budiče, monitoru a vyhodnocovací jednotky. Tuto testovací architekturu zachycuje obrázek 5.1. Vstupem budiče je tzv. transakce, kterou tato komponenta převádí na sled signálů odpovídající protokolu testované jednotky. Transakce je dále zpracována a na výstupu jednotky se objeví odpovídající výsledek. Ten je přijat monitorovací jednotkou, která zachycený sled signálů převádí zpět na objekt transakce.

Důležitou roli při simulaci hraje vyhodnocovací jednotka, která implementuje stejný algoritmus jako jednotka testovaná (avšak pomocí vyššího programovacího jazyka). Vyhodnocovací jednotka přijímá kopie vstupních transakcí, které jdou z budiče, a převádí je podle implementovaného algoritmu na transakce výstupní. Ty jsou pak ukládány do tabulky. Monitor zachytává odpovědi, které generuje testovaná jednotka, a předává je vyhodnocovací jednotce. Ta pak porovnává, zda se skutečná výstupní transakce rovná očekávané.

Existují dva způsoby vytváření transakcí. V prvním případě je vymýšlí programátor sám, což je vhodné zejména pro testování různých hraničních podmínek. Druhou možností je jejich automatické generování pomocí tzv. generátoru, který je součástí simulačního prostředí. Na tomto způsobu jsou pak založeny komplexní verifikační testy, jejichž cílem je zbavit jednotku všech chyb. Programátor pouze nastavuje parametry generátoru, čímž řídí celý proces verifikace. Dalšími aplikovanými verifikačními technikami byla např. kontrola protokolu pomocí výroků temporální logiky nebo pokrývání všech možných stavů, transakcí či pokrývání jednotlivých částí zdrojového kódu.



Obrázek 5.1: Architektura simulačního prostředí testované komponenty

Odladěný propojovací systém byl syntetizován programem Xilinx ISE. Jako cílové zařízení byl vybrán čip xc5vlx50t, který je jedním z nejmenších z rodiny čipů s technologií Virtex 5 [24]. Jejich součástí je i vystavěný modul, který převádí komplexní komunikační protokol systémové sběrnice PCI Express na jednodušší uživatelské rozhraní. Na něj je napojena kořenová komponenta propojovacího systému. Cílovou platformou, na které byl systém testován, je vývojová deska ML555 od firmy Xilinx.

Propojovací systémy mohou být hodnoceny na základě mnoha hledisek. Tato práce se zabývá především analýzou množství spotřebovaných zdrojů na čipu a pracovní frekvencí s ohledem na vybrané datové šířky 8, 16, 32 a 64 bitů. Kořenová komponenta byla pro zjednodušení napsána pouze na 64 bitech, aby mohla být jednoduše napojena na dostupný modul pro převod rozhraní PCI Express. Vzhledem k mnoha možným variantám transformační komponenty byla jako příklad ohodnocena pouze alternativa konvertující mezi šířkami 8 a 64 bitů. Všechny komponenty (vyjma transformační) byly vybrány ve variantách master i slave. Tabulka 5.1 shrnuje výsledky procesu syntézy. Pro každou verzi komponenty a datovou šířku je uveden počet spotřebovaných registrových bitů, počet využitých LUT, využití čipu FPGA a pracovní frekvence.

	8 bitů			**16 bitů**			**32 bitů**			**64 bitů**		
	Lut/Reg.	%	MHz	Lut/Reg.	%	MHz	Lut/Reg.	%	MHz	Lut/Reg.	%	MHz
Kořk. (M)	-	-	-	-	-	-	-	-	-	980/986	3.43	207
Kořk. (S)	-	-	-	-	-	-	-	-	-	898/971	3.38	221
Přk. (M)	261/84	0,91	221	339/108	1,18	221	519/207	1,8	210	714/303	2,48	203
Přk. (S)	69/18	0,24	287	101/26	0,35	285	165/42	0,57	281	293/74	1,02	276
Kon. k. (M)	302/160	1,05	228	379/163	1,32	229	405/183	1,41	216	568/180	1,97	201
Kon. k. (S)	236/145	0,82	254	272/148	0,94	255	284/168	0,99	234	383/165	1,33	208
Transf. k.	varianta 8-64 bitů: 157 Lut, 147 Reg., 0.55 % plochy čipu, 225 MHz											

Tabulka 5.1: Množství spotřebovaných zdrojů a pracovní frekvence jednotlivých komponent s ohledem na zvolené datové šířky

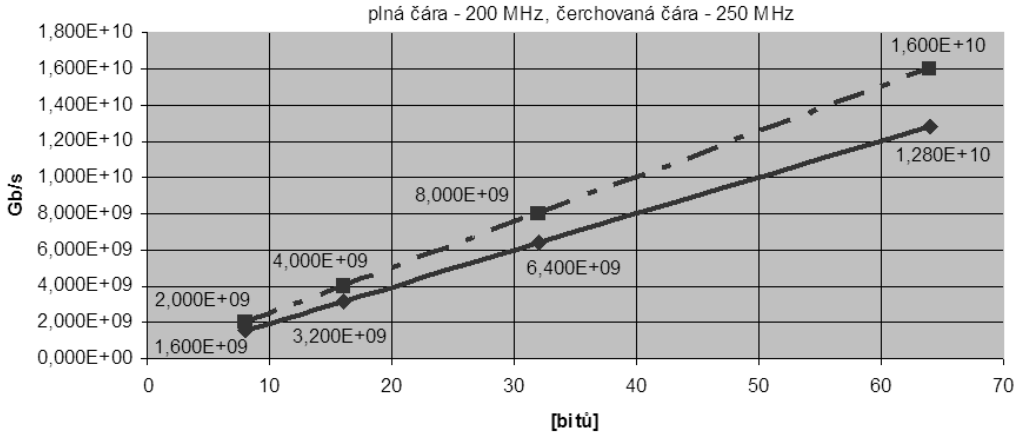
Z výsledků je patrné, že největší množství zdrojů zabírá kořenová komponenta. To je dáno především následkem různých formátů paketu interní a systémové sběrnice. Jednotlivé položky hlavičky a datová slova tak musí být ukládána do registrů, aby mohla být následně přeskládána na cílový formát. Vzhledem k tomu, že je kořenová komponenta v systému nasazena vždy maximálně jednou, je dosažený výsledek vyhovující. Master a slave varianta se výrazným způsobem neliší, rozdíl je dán pouze menším množstvím převážně kontrolní

logiky. Pracovní frekvence dosahuje hodnot 207 MHz pro master a 221 MHz pro slave variantu.

Výsledky přepínací komponenty ukazují, že množství spotřebovaných zdrojů výrazným způsobem stoupá se zvětšující se datovou šířkou (v rozsahu 0.9-2,5 % plochy čipu). To je způsobeno především tím, že každý port přepínací komponenty obsahuje dva registrovací stupně (vstupní a výstupní vyrovnávací paměť). Řídící logika není složitá, a proto má datová šířka na množství spotřebovaných zdrojů největší vliv. Slave varianta je mnohem jednodušší, protože se neskládá z tolika registrovacích stupňů, a zabírá o 60-70 % zdrojů méně než master varianta (0.25-1 % plochy čipu). Pracovní frekvence je v rozsahu 203-211 MHz pro master a 276-287 MHz pro slave variantu.

Na koncovou komponentu nemá datová šířka takový vliv (1-2 % plochy čipu). Nejvýznamnější část dílčích jednotek tak představuje řídící logika. Rozdíl mezi slave a master variantou koncové komponenty je dán přítomností tzv. master jednotky. Využití zdrojů se liší přibližně o 20 %. Pracovní frekvence se pohybuje v rozsahu 201-228 MHz pro master a 208-254 MHz pro slave variantu.

Nejmenší množství zdrojů na čipu zabírá transformační komponenta. Řídící logika je pro všechny šířky stejná a zabírá okolo 150 LUT (asi 0.5 % plochy čipu). Počet registrových bitů je dán dvojnásobkem větší z transformovaných šířek (pro každý směr transformace jeden registrový stupeň) a dalšími zhruba 20 řídícími bity. Pro transformaci mezi datovými šířkami 8 a 64 bitů je tak počet registrů roven 147 (asi 0.51 % čipu). Pracovní frekvence této konkrétní varianty je rovna 225 MHz.



Obrázek 5.2: Propustnost propojovacího systému v závislosti na datové šířce

Frekvence obvodu představuje velmi důležitý parametr, který je úzce spojen s propustností propojovacího systému. Pro všechny zvolené šířky a komponenty uvedené v tabulce přesahuje pracovní frekvence 200 MHz. Jak je patrné z obrázku 5.2 znázorňujícího graf propustnosti sběrnice v závislosti na její datové šířce (pro 200 a 250 MHz), odpovídá této frekvenci propustnost v rozsahu 1.6-12.8 Gb/s. Pracovní frekvence některých komponent však přesahují při vybraných datových šířkách i 250 MHz. V rámci možností dalšího pokračování projektu bude vhodné soustředit se na to, aby všechny komponenty systému mohly běžet na této frekvenci. Díky tomu bude možné dosáhnout propustnosti rovnající se 16 Gb/s, což je v současné době maximum dostupného vestavěného modulu pro připojení sběrnice PCI Express.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a implementovat propojovací systém interních sběrnic pro čipy s technologií FPGA. Systém zajišťuje jak komunikaci mezi interními komponentami v rámci čipu, tak jejich komunikaci s ostatními prvky počítače, které jsou namapovány do paměti hostitelského systému. Při návrhu byl kladen důraz především na propustnost a škálovatelnost systému, ale také na množství využitých zdrojů.

Pro pochopení zkoumané problematiky byly nastudovány potřebné materiály, které se zabývají technologií programovatelných hradlových polí FPGA, existujícími sběrnicovými systémy a způsoby přenosu dat mezi procesorem a adaptérem periferního zařízení. Hlavním předmětem zájmu byly především sběrnicové topologie, komunikační protokoly a další principy, které se v propojovacích systémech uplatňují. Načerpané znalosti byly shrnuty v teoretickém rozboru a v kapitole pojednávající o existujících sběrnicových systémech.

Hlavní část práce představoval návrh celého systému interních sběrnic. Tento systém vychází ze stromové topologie a využívá vysokorychlostní plně duplexní sběrnice založené na paketové komunikaci. Šířka sběrnice je genericky nastavitelná a to individuálně pro každou její část. Díky tomu je možné vybudovat jednotný hierarchický systém sběrnic na různých rychlostech umožňující připojení komponent s různými výkonnostními a funkčními požadavky. Zohledněna je podpora přenosů inicovaných ze strany uživatelských komponent, realizace modelu rozdělených transakcí a další požadavky. I přes bohatou podporu různých vlastností a vysokou propustnost je systém díky jednomu typu sběrnice (s různou datovou šířkou) velmi dobře spravovatelný a je složen z malého počtu komponent. Možnost zřetězení interní sběrnice navíc snižuje její citlivost na vzdálenost.

Navržený propojovací systém byl implementován v jazyce VHDL. Jeho funkce byla ověřena nejprve v simulacích a poté přímo na vývojové kartě ML555 a COMBO6X. Nad rámec zadání této diplomové práce byla dále provedena komplexní verifikace jednotlivých komponent pomocí jazyka SystemVerilog, který integruje podporu pro některé efektivní verifikační techniky.

Celá práce byla zhodnocena v kapitole 5 popisující dosažené výsledky. Byla provedena podrobná analýza množství spotřebovaných zdrojů a pracovní frekvence s ohledem na vybrané datové šířky. Z výsledků je patrné, že množství spotřebovaných zdrojů jednotlivých komponent (zvláště přepínače) se zvětšuje s rostoucí datovou šířkou. Výběrem přiměřené datové šířky a vhodné varianty (slave/master) jednotlivých komponent je možné dosáhnout takového obsazení plochy čipu, které dává návrháři dostatečný prostor pro vlastní aplikaci. Frekvence všech komponent přesahuje pro největší zkoumanou datovou šířku (64 bitů) hranici 200 MHz, což odpovídá maximální propustnosti 12.8 Gb/s.

Nakonec byly analyzovány možnosti dalšího pokračování projektu. V první řadě je

možné zvýšit pracovní frekvenci na cílových 250 MHz, což umožní dosažení propustnosti až 16 Gb/s. Dále je vhodné se zamyslet nad možnostmi využití volných bitů v hlavičce paketu, například s ohledem na zvětšení lokálního adresového prostoru. Dalším rozšířením může být vytvoření sady platformově závislých kořenových komponent pro další nejpoužívanější systémové sběrnice, jako například PCI nebo PCI-X. Na závěr je vhodné vzít rovněž v potaz, že díky nezávislosti jazyka VHDL na cílové technologii je možné uvažovat nejen FPGA, ale i jiné technologie (např. ASIC), do kterých by bylo možné implementovaný systém přeportovat.

Literatura

[1] ARM. *AMBA Specification, Revision 2.0*, May 1999.

[2] Sauer C., Gries M., Gomez J., and Keutzer K. Towards a flexible network processor interface for rapidio, hypertransport, and pci-express. In *3rd Workshop on Network Processors (NP3) at the 10th International Symposium on High Performance Computer Architecture (HPCA10)*, pages 26–39, February 2004.

[3] Coffman E. G., Elphick M., and Shoshani A. *System Deadlocks*. ACM, 1971. ISSN 0360-0300.

[4] IBM. *CoreConnect Bus Architecture*, September 1999. Dokument dostupný na http://www-01.ibm.com/chips/techlib/techlib.nsf/products/CoreConnect_Bus_Architecture (květen 2008).

[5] IBM. *On-Chip Peripheral Bus*, January 2001. Dokument dostupný na http://www-01.ibm.com/chips/techlib/techlib.nsf/products/CoreConnect_Bus_Architecture (květen 2008).

[6] IBM. *Device Control Register Bus 3.5 Architecture Specification*, January 2006. Dokument dostupný na http://www-01.ibm.com/chips/techlib/techlib.nsf/products/CoreConnect_Bus_Architecture (květen 2008).

[7] IBM. *Processor Local Bus (128-bit)*, May 2007. Dokument dostupný na http://www-01.ibm.com/chips/techlib/techlib.nsf/products/CoreConnect_Bus_Architecture (květen 2008).

[8] Accellera Organization, Inc. *SystemVerilog 3.1a Language Reference Manual*, March 2004. Dokument dostupný na http://www.eda.org/sv/SystemVerilog_3.1a.pdf (květen 2008).

[9] Xilinx, Inc. *Virtex-5 ML555 Development Kit for PCI and PCI Express Designs User Guide*, June 2007. Dokument dostupný na http://www.xilinx.com/support/documentation/boards_and_kits/ug201.pdf (květen 2008).

[10] Ashenden P. J. *The VHDL Cookbook*, July 1998. Dokument dostupný na http://tech-www.informatik.uni-hamburg.de-vhdl/doc/cookbook/VHDL-Cookbook.pdf (květen 2008).

[11] Novotný J., Fučík O., and Kokotek R. *Schematics and PCB of COMBO6*. Technical report, CESNET, Leden 2002.

[12] Rushby J. Bus architectures for safety-critical embedded systems. In Tom Henzinger and Christoph Kirsch, editors, *EMSOFT 2001: Proceedings of the First Workshop on Embedded Software*, volume 2211, pages 306–323, Lake Tahoe, CA, oct 2001. Springer-Verlag.

[13] OpenCores. *WISHBONE System-on-Chip Interconnection Architecture for Portable IP Cores*, September 2002. Dokument dostupný na http://www.opencores.org/projects.cgi/web/wishbone/wbspec_b3.pdf (květen 2008).

[14] Brassington M. P. Semi-custom ASIC technology trends. *Proceedings of Custom Integrated Circuits Conference*, pages 47–54, March 1995.

[15] PCI-SIG. *PCI Express Base Specification, Revision 2.0*, December 2006.

[16] WWW stránky. *Liberouter project - Programmable hardware.* http://www.liberouter.org (květen 2008).

[17] WWW stránky. *PCI-SIG.* http://www.pcisig.com (květen 2008).

[18] Martinek T. and Tobola J. *Interconnection System for the NetCOPE Platform.* Technical report, CESNET, Leden 2006.

[19] RapidIO Trade Association. *RapidIO Interconnect Specification Part 3: Common Transport Specification*, June 2005. Dokument dostupný na http://www.rapidio.org/specs/disclaimer?specfile=/zdata/specs/cmn_trnspt.pdf (květen 2008).

[20] RapidIO Trade Association. *RapidIO Interconnect Specification Part 8: Error Management Extensions Specification*, June 2005. Dokument dostupný na http://www.rapidio.org/specs/disclaimer?specfile=/zdata/specs/errspec.pdf (květen 2008).

[21] RapidIO Trade Association. *RapidIO Interconnect Specification Part 9: Flow Control Logical Layer Extensions Specification*, June 2005. Dokument dostupný na http://www.rapidio.org/specs/disclaimer?specfile=/zdata/specs/fcsspec.pdf (květen 2008).

[22] Trimberger S. *Field-Programmable Gate Array Technology.* Kluwer Academic Publishers, 1994.

[23] Trodden J. and Anderson D. *HyperTransport System Architecture.* Addision-Wesley, 2003.

[24] Xilinx. *Virtex-5 FPGA User Guide*, April 2008. Dokument dostupný na http://www.xilinx.com/support/documentation/virtex-5.htm (květen 2008).

[25] Xilinx Inc. 2100 Logic Drive. *LocalLink Interface Specification*, September 2006.