VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

FRAMEWORK PRO ČÁSTEČNOU DYNAMICKOU REKONFIGURACI FPGA VIRTEX-5

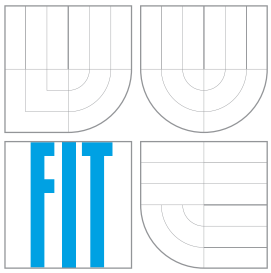
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB RAČEK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

FRAMEWORK PRO ČÁSTEČNOU DYNAMICKOU REKONFIGURACI FPGA VIRTEX-5

FRAMEWORK FOR DYNAMIC PARTIAL RECONFIGURATION OF VIRTEX-5 FPGA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB RAČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JIŘÍ MATOUŠEK

BRNO 2014

Abstrakt

Práce se zabývá návrhem a implementací frameworku částečné dynamické rekonfigurace pro FPGA architekturu Virtex-5. Částečná dynamická rekonfigurace umožňuje změnit funkci části FPGA, zatímco zbytek zařízení pracuje bez přerušení. Framework má usnadnit tvorbu aplikací s hardwarovými akcelerátory využívajících částečnou dynamickou rekonfiguraci. S využitím implementovaného frameworku byla vytvořena demonstrační aplikace pro pattern-matching nad příchozími síťovými pakety. Řízení procesu částečné dynamické rekonfigurace obstarává systém typu GNU/Linux, který běží na procesoru MicroBlaze. To navíc umožňuje běh méně náročných aplikací a zpracování paketů pomocí softwaru.

Abstract

The thesis is focused on design and implementiation of a framework for Dynamic Partial Reconfiguration for FPGA architecture Virtex-5. Dynamic Partial Reconfiguration allows to change the function of a part of a FPGA, while the rest of the device remains active and working. The aim of the framework is to simplify creating applications with hardware accelerators using Dynamic Partial Reconfiguration. Using an implemented framework, a demonstration application was created for pattern-matching incoming network packets. The process of Dynamic Partial Reconfiguration is controlled by GNU/Linux type operating system, which runs on MicroBlaze processor. This also allows to run less demanding applications and the processing of packets using software.

Klíčová slova

FPGA, Virtex-5, částečná dynamická rekonfigurace, akcelerace, framework.

Keywords

FPGA, Virtex-5, Dynamic Partial Reconfiguration, acceleration, framework.

Citace

Jakub Raček: Framework pro částečnou dynamickou rekonfiguraci FPGA Virtex-5, bakalářská práce, Brno, FIT VUT v Brně, 2014

Framework pro částečnou dynamickou rekonfiguraci FPGA Virtex-5

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Matouška. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Raček
16. května 2014

Poděkování

Rád bych poděkoval panu Ing. Matouškovi za poskytnuté informace a za to, že mi umožnil pracovat tak, jak jsem potřeboval.

Obsah

Kapitola 1

Úvod

V dnešní době existují technologie, jejichž využití má vysoké nároky na výpočetní systémy. Jde o aplikace jako je například pattern-matching, zpracování videa nebo zpracování obrazu pro rozeznání cíle. Univerzální procesory s běžným softwarovým vybavením nejsou v takových případech dostačující [39].

Pro pattern-matching nad síťovým provozem je rychlost zpracování velice důležitá. Pakety, které nelze včas zpracovat, jsou zahozeny [15]. Požadavky na systémy jsou zde zvyšovány s rostoucí rychlostí počítačových sítí. V současné době jsou specifikovány Ethernetové sítě do rychlosti 100 Gb/s [19]. Ethernetová zařízení o rychlosti 100 Gb/s jsou komerčně dostupná a probíhá další výzkum a vývoj.

Pro správné fungování aplikací s takovými požadavky je obvykle zapotřebí hardwarového akcelerátoru [39]. Pro hardwarovou akceleraci je možné použít aplikačně specifický integrovaný obvod (*Application-Specific Integrated Circuit*, dále jen „ASIC“) nebo programovatelné hradlové pole (*Field-Programmable Gate Array*, dále jen „FPGA“). Všechny obvody typu ASIC jsou rychlé a efektivní pro výpočty, pro které byly navrženy, ale výsledné obvody není možné změnit po jejich dokončení [29]. Většinu[1] FPGA lze naproti tomu rekonfigurovat a změnit jejich funkci. To činí FPGA vysoce flexibilní alternativou k ASIC. Rekonfigurovatelné obvody mají vyplnit mezeru mezi hardware a software řešeními.

Částečná dynamická rekonfigurace (*Dynamic Partial Reconfiguration* – DPR, někdy též *Partial Reconfiguration*) je technologie, která umožňuje za běhu měnit funkci části FPGA [29][41]. Mezi zařízení, která podporují DPR, patří rodina FPGA Virtex-5 firmy Xilinx. Cílem této práce je navrhnout a implementovat framework pro částečnou dynamickou rekonfiguraci FPGA Virtex-5. Navržený framework má usnadnit vytváření aplikací, které potřebují využívat částečnou dynamickou rekonfiguraci. Součástí implementace frameworku bude demonstrační aplikace.

Tato práce je rozdělena do několika kapitol. Kapitola 2 je věnována částečné dynamické rekonfiguraci na FPGA Virtex-5. Obsahuje seznámení s principy DPR, popis výhod a nevýhod, stejně jako informace o tom, jaké požadavky klade tato technologie na návrh. Popisem hardwarové platformy a vývojového prostředí se zabývá kapitola 3. Jsou uvedeny jednotlivé jednotky, které budou použity v rámci návrhu frameworku nebo které mohou být později užitečné pro návrh a implementaci demonstrační aplikace. Návrh frameworku je proveden v kapitole 4. Kapitola 5 je věnována implementaci frameworku a návrhu demonstrační aplikace. Je ukázáno, jakým způsobem byla tato aplikace vytvořena pomocí frameworku. Kapitola 6 shrnuje výsledky práce a obsahuje diskuzi možných omezení frameworku.

[1] Výjimku tvoří FPGA fungující na principu *anti-fuse*, které lze naprogramovat pouze jednou [24].

Kapitola 2

FPGA Virtex-5 a částečná dynamická rekonfigurace

Tato kapitola je věnována částečné dynamické rekonfiguraci a technologii architektury FPGA Virtex-5. Cílem je seznámit čtenáře obecně s technologií samotnou a poskytnout relevantní a specifické informace o DPR na FPGA Virtex-5. V podkapitole 2.1 je objasněno, jaké technické provedení FPGA umožňuje DPR. Tato podkapitola se zabývá částečnou dynamickou rekonfigurací z pohledu nízké úrovně abstrakce. Podkapitola 2.2 se zabývá DPR z pohledu vyšší úrovně abstrakce, která je bližší novému uživateli této technologie. Jsou uvedeny informace o zařazení této technologie a její dostupnosti na FPGA zařízeních. Také je vysvětlena základní terminologie, s kterou se čtenář bezpochyby setká při praktickém používání částečné dynamické rekonfigurace. Podkapitola 2.3 obsahuje informace o výhodách a nevýhodách této technologie. Zároveň jsou zde uvedeny některé konkrétní příklady použití DPR a dosažené výsledky. Pravidlům pro vhodný návrh projektu s částečnou dynamickou rekonfigurací je věnována podkapitola 2.4. Jejím cílem je seznámit čtenáře s omezeními, která technologie DPR klade na návrh. V podkapitole 2.5 jsou uvedeny možnosti řízení částečné dynamické rekonfigurace. Uvedená rozhraní jsou stručně popsána tak, aby si čtenář mohl udělat představu o tom, pro jaké účely jsou jednotlivá rozhraní vhodná.

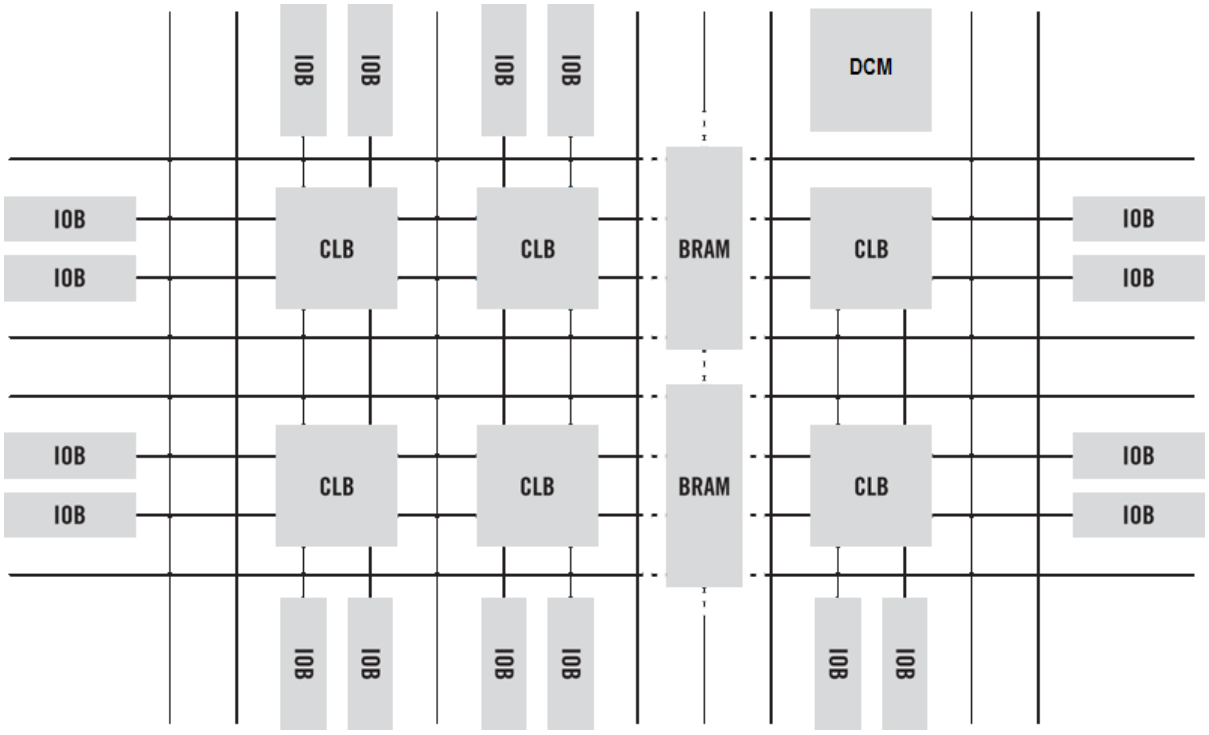
2.1 Technologie a architektura FPGA

Tato podkapitola popisuje architekturu a princip fungování FPGA, jenž společně umožňují částečnou dynamickou rekonfiguraci. Technické provedení je principiálně společné pro všechny FPGA podporující tuto technologii. Daná architektura se používá na rodině zařízení Virtex-5 i mnoha jiných. Informace o jednotlivých blocích integrovaných v FPGA jsou specifické pro Virtex-5.

Nejběžnější FPGA architektura se skládá z propojovacích vodičů (*routing channels*) a pole logických bloků (*Logic Cell Arrays* – LCA). Jedná se o takzvanou *island-style* architekturu; rozložení jednotlivých komponent připomíná ostrovy. [29] Obrázek 2.1 na straně 4 znázorňuje tuto architekturu a tabulka 2.1 na stejné straně popisuje jednotlivé bloky.

Většina moderních FPGA má konfigurační paměť založenou na statických RAM (SRAM) buňkách. Tyto čipy jsou označovány jako „SRAM programmable[1]“. Nevýhoda je, že integrovaný obvod ztratí svou konfiguraci při odpojení napájení.

[1] Termín SRAM je technicky nepřesný, protože konfigurační paměť nemusí podporovat náhodný přístup. Nicméně se jedná o obecně přijatý termín. [29]



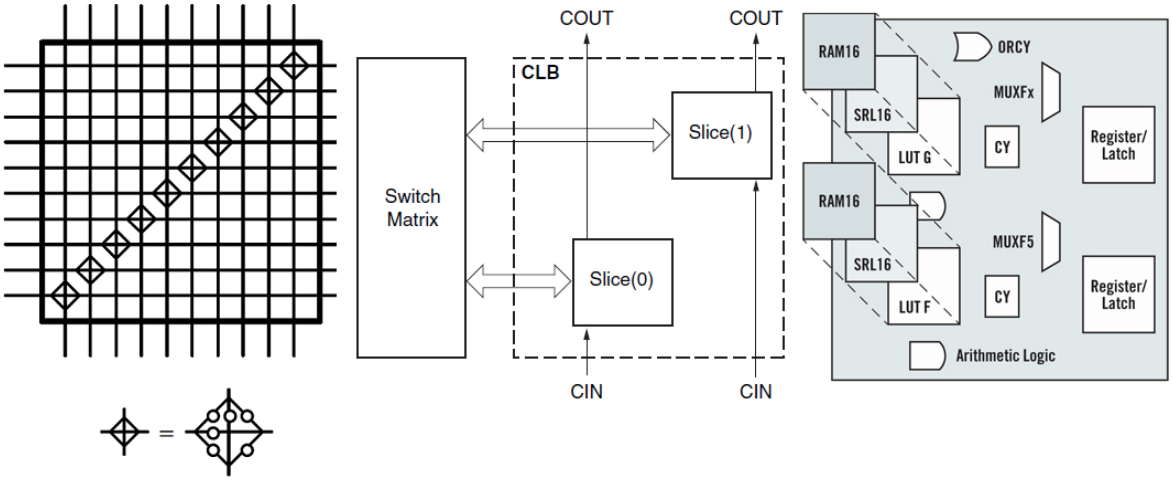
Obrázek 2.1: Island-style architektura s propojovacími vodiči a jednotlivými bloky [25]

Komponenta	**Označení**	**Stručný popis**
Vstupně/výstupní blok	IOB	IOB obsahuje konfigurovatelné zdroje umožňující vytvoření různých vstupně/výstupních rozhraní.
Konfigurovatelný logický blok	CLB	CLB je základní logická jednotka v FPGA. Obsahuje vyhledávací tabulky (LUT), selekční logiku (MUX) a klopné obvody.
Digital Clock Manager	DCM	DCM umožňuje pokročilé použití synchronizačních hodin. Frekvenci hodin je možné dělit nebo násobit a tím vytvořit novou frekvenci. Dále je možné provést fázový posun a řešit velké množství běžných záležitostí souvisejících se synchronizačními hodinami.
BlockRAM	BRAM	BRAM je konfigurovatelný paměťový modul. Celkových 36 kilobitů jedné BRAM paměti lze použít jako jednu RAM nebo jako dvě nezávislé (16 kilobitů každá). U každé RAM lze vybrat z více možností uspořádání (např. 16 Kb x 2 nebo 2 Kb x 8). BRAM používá synchronní zápisové a čtecí operace. Více informací o tom, co BRAM umožňuje, lze nalézt v [22].

Tabulka 2.1: Popis komponent běžně integrovaných v FPGA [25]

Výhodou je, že počet přeprogramování FPGA není omezen. [41] Vzhledem k této vlastnosti bylo na FPGA nahlíženo jako na zařízení vhodné pro rychlé prototypování. Bylo však také zjištěno, že volatilita použitých pamětí není značná nevýhoda, ale naopak klíčová vlastnost pro mnoho nových druhů aplikací [32]. Jinak řečeno, toto provedení umožňuje provádět částečnou dynamickou rekonfiguraci, která přináší mnoho výhod[2].

SRAM buňky určují konfiguraci jednotlivých logických bloků. Zároveň také umožňují směrování signálů a propojování jednotlivých částí [41]. Kdekoliv se stýkají horizontální a vertikální vodiče, tam je umístěn *switchbox*. Tvary připomínající diamant vlevo na obrázku 2.2 reprezentují jednotlivá propojení umožňující jakoukoliv permutaci spojení mezi čtyřmi vodiči.



Obrázek 2.2: Switchbox [32], struktura CLB [25] a struktura slice [22] (v tomto pořadí)

Všechny bloky uvedené v tabulce 2.1 jsou rekonfigurovatelné, s výjimkou DCM[3]. Konfigurovatelný logicky blok (*Configurable Logic Block* – CLB) lze do určité míry považovat za nejdůležitější z těchto bloků.

Jedná se o takřka ideální komponentu pro implementaci libovolného komplexního digitálního obvodu. [32] Jeho součásti jsou znázorněny uprostřed na obrázku 2.2. CLB se skládá z páru *slice*, které jsou znázorněny[4] vpravo na stejném obrázku. Tyto komponenty nemají jedna k druhé přímé spojení, pouze logiku pro rychlé přenosy do dalšího CLB (*carry chain*).

Pro FPGA Virtex-5 platí, že každý jeho slice obsahuje čtyřikrát vyhledávací tabulku (*Look-up Table* – LUT), čtyřikrát paměťový prvek a víceúčelové multiplexory. Některé slice umožňují další funkce: ukládání dat do distribuované RAM a posuv dat v 32bitovém registru. Více informací lze najít v [22].

FPGA Virtex-5 má vyhledávací tabulky s šesti nezávislými vstupy a dvěma nezávislými výstupy. Výhoda vyhledávací tabulky spočívá v rychlosti přístupu (vyhledání). Vyhledávací tabulka zde umožňuje implementovat libovolnou Booleovskou funkci s šesti vstupy.

S architekturou a technologickým provedením FPGA zařízení bylo různě experimentováno, ale i moderní rekonfigurovatelná zařízení stále fungují na principech LCA a SRAM, jenž byly použity v počátku FPGA zařízení. [29]

[2]Tyto výhody jsou popsány v podkapitole 2.3.

[3]Dokumenty [8] a [22] jsou ohledně rekonfigurovatelnosti DCM ve sporu. Podle souhrnu všech informací lze uvažovat, že DCM je možné rekonfigurovat, ale nesmí se jednat o globální logiku a lze měnit pouze výslednou frekvenci, kterou DCM vytváří násobením nebo dělením.

[4]Znázornění obsahu slice je schématické. Přesný počet jednotlivých komponent je uveden dále v textu.

2.2 Principy částečné dynamické rekonfigurace

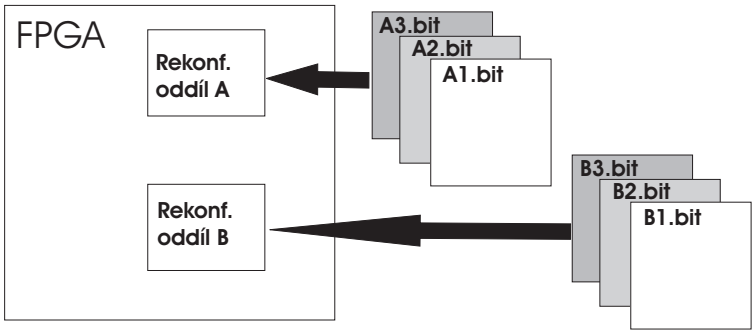
V odborné terminologii se používá termínu *rekonfigurovatelný systém* pro systémy, u nichž je změna hardwarové struktury součástí normální funkce [28]. FPGA jsou nejběžnější zařízení v oblasti rekonfigurovatelných systémů [1] a jejich význam se dále zvyšuje [32].

Částečná dynamická rekonfigurace je technologie, která umožňuje měnit funkci části FPGA během jeho činnosti [29][41]. Tato technologie značně zvyšuje flexibilitu FPGA [8]. Je třeba zdůraznit, že rekonfiguraci umožňuje každé FPGA, které lze konfigurovat více než jednou. Ne každé FPGA zařízení však umožňuje DPR [36]. Z [8] a [16] vyplývá, že většina moderních rodin FPGA technologii DPR podporuje.

FPGA zařízení lze programovat a přeprogramovat zapsáním bitů (konfigurace obvodu, „bitstream“) do vnitřní konfigurační paměti [43]. Tento princip je vždy stejný a uplatňuje se u všech způsobů rekonfigurace. Z [8] a [31] vyplývá, že FPGA zařízení společnosti Xilinx podporují dva způsoby rekonfigurace. Jeden je založený na rozdílech (*Diference-based Partial Dynamic Reconfiguration*) a jeden na modulárním návrhu (*Modular-based Partial Dynamic Reconfiguration*). První zmíněný způsob je vhodný, pokud je třeba v systému udělat drobnou změnu (např. změna bloků vyhrazené paměti). Druhý zmíněný způsob je vhodnější pro provádění větších změn v systému. [36]

FPGA zařízení může obsahovat různé oddíly. Rekonfigurovatelný oddíl (*Reconfigurable Partition* – RP) je uživatelem určená oblast FPGA, která obsahuje rekonfigurovatelnou logiku (*Reconfigurable Logic*). Tuto logiku je možné změnit za běhu zařízení, přičemž měnit lze konfiguraci jednotlivých komponent, i konfiguraci spojení mezi nimi. Instance RP bývá označována jako rekonfigurovatelný modul (*Reconfigurable Module* – RM). RP je tedy oblast FPGA, zatímco RM je konkrétní implementace obvodu. Implementaci obvodu určuje HDL popis nebo netlist. Statická logika (*Static Logic*) je neměnná a nemusí pro ni být explicitně vyhrazen oddíl. Není nikdy částečně rekonfigurována a je vždy aktivní, když jsou rekonfigurovatelné oddíly rekonfigurovány. [8]

Obrázek 2.3 ilustruje základní princip použití DPR. FPGA obsahuje rekonfigurovatelné oddíly, zbytek jsou statické logické obvody. Obvykle se nejprve načte bitstream, který nakonfiguruje celé zařízení. Poté lze díky DPR nahrávat částečné bitstreamy a tím (re)konfigurovat rekonfigurovatelné oddíly.



Obrázek 2.3: DPR a částečné bitstreamy [8]

V rámci této práce bude použita částečná dynamická rekonfigurace založená na modulárním návrhu. Z [8] vyplývá, že tento druh rekonfigurace je vhodný pro většinu uživatelských aplikací. Lze uvažovat, že je to do určité míry dáno tím, že se kapacita FPGA s postupem času dále navyšuje. FPGA mohou obsahovat větší moduly, v nichž je vhodné provádět větší změny částečnou dynamickou rekonfigurací.

2.3 Použití částečné dynamické rekonfigurace

Částečná dynamická rekonfigurace přináší určité výhody a nevýhody. Také klade určité požadavky na návrh jako takový a vyžaduje, aby návrhář byl poměrně dobře obeznámen s tvorbou obvodů v FPGA a uvědomoval si, co využití této technologie znamená. Tato podkapitola obsahuje základní přehled těchto informací.

Výhody DPR

1. Změna obvodu za běhu dle vstupních podmínek
2. Úspora prostoru na čipu
3. Zkrácení kritické cesty
4. Snížení spotřeby
5. Zvýšení odolnosti vůči poruchám
6. Zmenšení velikosti bitstreamu

Příkladem změny obvodu dle vstupních podmínek může být implementace násobičky, která provádí násobení pouze pomocí několika konstant (například 3, 7 a 13). Namísto plýtvání zdroji implementací obecné násobičky stačí připravit tři verze konstantní násobičky. Takovéto řešení zároveň ušetří místo na čipu, přičemž rychlost násobičky bude stejná nebo porovnatelná [42]. Tento příklad ukazuje, že obvod může být nahrazen jiným, méně obecným obvodem, přičemž je ve výsledku dosaženo stejné funkce.

Úspora prostoru na čipu vychází z možnosti využívat jednotlivé komponenty FPGA pro obvody, které mají v průběhu času různou funkci. V době, kdy není daného modulu zapotřebí, je nahrazen jiným, který provádí odlišnou, v danou chvíli potřebnou funkci. Tato výhoda může umožnit využít FPGA zařízení pro obvod, který by bez částečné dynamické rekonfigurace vyžadoval zařízení s větší kapacitou.

Kritická cesta (*critical path*) je cesta mezi dvěma sousedními sekvenčními prvky s největším zpožděním. Zpoždění na kritické cestě ovlivňuje délku cyklu hodin. Kritická cesta se změní při každé rekonfiguraci rekonfigurovatelného oddílu. Nahráním modulu s vhodnou implementací je možné kritickou cestu zkrátit. Zkrácení kritické cesty má pozitivní dopad na výkon celého obvodu. [30]

Snížení spotřeby je umožněno díky vypnutí části obvodu pomocí rekonfigurace. Alternativně může být namísto původního obvodu nahrána jeho méně výkonná varianta, která má nižší spotřebu.

Zvýšení odolnosti vůči poruchám spočívá ve využití obvodu umístěného jinde na čipu namísto poruchového obvodu. Pomocí částečné dynamické rekonfigurace je také možné uvést obvod do původního stavu opětovným nahráním konfigurace.

Snížení velikosti úplného bitstreamu spočívá v tom, že nepotřebné rekonfigurovatelné oddíly jsou označeny jako černá skříňka (*Black Box*) a ponechány prázdné. Konfigurace je do těchto oddílů nahrána až později pomocí DPR. [8] To umožňuje rychlejší konfiguraci a start systému.

S jednotlivými výhodami se lze blíže obeznámit studiem vhodných knih, článků a další materiálů v nich odkazovaných. Například práce [43] je věnována čistě zvýšení funkční hustoty (úspoře plochy na čipu) pomocí rekonfigurace.

Nevýhody DPR

1. Snížení frekvence obvodu
2. Doba rekonfigurace obvodu[5]
3. Zvýšená spotřeba energie během rekonfigurace [37]

Výše zmíněné výhody a nevýhody obecně platí pro všechna FPGA zařízení, která jsou založena na principech popsaných v podkapitole 2.1. Z těchto výhod a nevýhod také vyplývají určité požadavky pro efektivní použití této technologie. Návrhář si jich musí být vědom a při vytváření výsledného řešení by měl hledat vhodný kompromis. Níže jsou uvedeny dva nejdůležitější požadavky, které přímo souvisí s principem DPR.

Prvním požadavkem je, že FPGA obsahuje oblasti se statickou logikou, která se během činnosti zařízení nemění. [42] Jedná se o součásti, které jsou zapotřebí po celou dobu běhu systému. Jako příklad můžeme uvést řadič dynamické paměti, konfigurační rozhraní DPR nebo řídicí logiku. [8]

Druhý požadavek souvisí s dobou rekonfigurace obvodu. Ta zabírá určitý nezanedbatelný čas, během kterého není daná oblast FPGA používána. Je tudíž vhodné, aby doba rekonfigurace byla co možná nejkratší a následné urychlení zpracování vynahradilo časovou ztrátu (zpomalení zpracování), která vznikla rekonfigurací. [43] Alternativně není nutné vynahradit tuto časovou ztrátu, ale namísto toho během rekonfigurace provádět jiný výpočet. Záleží na konkrétním scénáři užití částečné dynamické rekonfigurace.

Výsledky

Částečná dynamická rekonfigurace byla použita pro různé aplikace. Vytvoření dobrého návrhu, maximální využití výhod DPR a dodržení výše zmíněných požadavků může mít pozitivní dopad na výsledný systém. Některé výsledky (příklady dosaženého urychlení) jsou uvedeny v seznamu níže. Další příklady urychlení lze nalézt v mnoha vědeckých článcích. Více odkazů lze najít v zde citovaných pracích.

1. šifrování v Xilinx Virtex XCV1000, urychlení 18x oproti univerzálnímu procesoru
2. implementace algoritmu prosévání čísel, urychlení více jak 28x oproti mikroprocesoru UltraSparc [29]

2.4 Částečná dynamická rekonfigurace a pravidla návrhu

Pro částečnou dynamickou rekonfiguraci je důležité, aby bylo možné opakovaně a spolehlivě dosáhnout stejných výsledků. To vyžaduje dodržování pravidel dobrého návrhu obvodu pro FPGA. Tato podkapitola obsahuje výběr pravidel pro návrh projektu s DPR. Uživatel by se s těmito pravidly měl seznámit před tím, než začne vytvářet návrh využívající DPR. Dodržení těchto pravidel má umožnit úspěšně vytvořit návrh, který bude možné implementovat a který bude splňovat očekávané požadavky. [8]

Doporučení jsou do značné míry svázána s nástroji a architekturou FPGA zařízení poskytovaných výrobcem. Uživatel se s těmito pravidly setká především v počáteční vývojové fázi (tj. během plánování) a při práci s nástrojem PlanAhead během procesu, který se označuje jako *floorplanning*.

[5] Na současných zařízeních se pohybuje v řádu milisekund – záleží na velikosti bitstreamu. Rozhraní ICAP společnosti Xilinx má propustnost až 3,25 Gbps.

Floorplanning je proces manuálního umisťování jednotlivých bloků logiky v FPGA, výběru nejlepšího seskupení a propojení této logiky. Uživatel DPR nemusí tento proces dokonale znát; postačí, pokud zná níže uvedená základní pravidla a dodržuje je.

Hierarchie návrhu

Návrh by měl být vhodně logicky rozdělený. Je třeba vědět, že hranice rekonfigurovatelného oddílu jsou překážkou pro optimalizace. Statická a rekonfigurovatelná logika musí být oddělena. Zároveň ale mezi statickou a rekonfigurovatelnou logikou musí existovat nějaké rozhraní (pevný bod známý ve všech konfiguracích). K tomuto účelu slouží *Partition Pin* a proxy logika (*Proxy Logic*). Partition Pin je logické a fyzické spojení, které nástroje automaticky vytvoří pro všechny porty rekonfigurovatelného oddílu. Proxy logika je jeden LUT1 prvek, který přenáší bitovou hodnotu, kterou obsahuje Partiton Pin. [8]

Partition Pin a proxy logika je sice vložena nástroji automaticky na správná místa, ale může mít za následek suboptimální výsledky nebo cesty s časováním, kterého není možné dosáhnout. Obvod, který vykonává nějakou funkci, by měl být rozdělen na statickou a rekonfigurovatelnou část tak, aby došlo k co možná nejmenší degradaci časování. [8]

Ne všechny prvky lze rekonfigurovat při běhu zařízení. Globální logika a prvky související se synchronizačními hodinami (PLL, DCM) musí být použity jako statická logika. Samotné používání a rozvod synchronizačních signalů je větší téma, s kterým se lze seznámit v [8]. Obecně lze uvažovat, že je nejlepší, pokud je daný prvek i jeho zátěž ve stejném oddílu, ať už statickém nebo rekonfigurovatelném.

Výkon a efektivita

- Může dojít k problémům se směrováním signálů, pokud je rekonfigurovatelná oblast příliš malá nebo pokud není vytvořena z obdélníkových tvarů.
- Lze očekávat 10% snížení frekvence synchronizačních hodin.
- Lze očekávat, že hustota zaplnění prvků slice nepřesáhne 80%. [8]

Používání nástrojů

- Uživatel by se měl řídit dle *Design Rule Checks* (DRCs). Jedná o sadu kontrol, které zjistí porušení pravidel DPR návrhu.
- Při spouštění implementace v nástroji by uživatel měl jako první vybrat tu konfiguraci, která obsahu prvky, jejichž implementace je nejnáročnější.
- Ve většině případů lze očekávat, že práce nástrojů bude trvat déle. Nejvíce bude ovlivněn MAP, ale také NGDBuild a PAR mohou projevovat známky, že zpracovávají návrh s DPR.

Další informace

Kompletní výčet doporučených postupů přesahuje rozsah této práce. Z těchto pravidel byly vybrány a výše uvedeny ty, jejichž znalost a dodržení umožní uživateli úspěšně implementovat návrh. Další informace, jako například výčet omezení vztahujících se k jednotlivým zařízením, lze najít v dokumentu [8].

Všechna výše zmíněná omezení souvisí s technologií částečné dynamické rekonfigurace a fakticky je není možné nijak ovlivnit. Tato omezení nemají nic společného s omezeními

(*constraints*) kladenými návrhářem/uživatelem na návrh obvodu v FPGA. Zkušený návrhář by měl být seznámen s dokumentem [4] a vědět, jak ovlivňují automatizovanou činnost softwarových nástrojů (např. MAP nebo PAR).

2.5 Možnosti řízení částečné dynamické rekonfigurace

Částečná dynamická rekonfigurace hraje v návrhu systému klíčovou roli. V této podkapitole budou popsány způsoby řízení částečné dynamické rekonfigurace. Kterékoliv z následujících rozhraní může být použito pro načtení částečného bitstreamu:

- JTAG
- Slave Serial
- Slave SelectMAP
- ICAP

Všechna uvedená rozhraní s výjimkou ICAP jsou vhodná k provedení rekonfigurace celého FPGA a částečné dynamické rekonfigurace přes stejné rozhraní [8].

Tato rozhraní mají různé použití a různou výkonnost. Maximální šířky pásem jsou uvedeny v tabulce 2.2. Uvedená rozhraní jsou v této podkapitole dále popsána s přihlédnutím k tomu, které z nich je z pohledu této práce nejvhodnější.

Rozhraní	**Max. frekvence hodin [MHz]**	**Datová šířka [b]**	**Max. šířka pásma [Gbps]**
JTAG	66	32	2,112
Slave Serial	100	1	0,1
Slave SelectMAP	100	32	3,2
ICAP	100	32	3,2

Tabulka 2.2: Maximální šířka pásma rozhraní [8]

JTAG

IEEE 1149.1 Test Access Port (TAP) a Boundary-Scan architektura, běžně zkracováno jako JTAG, je populární metoda pro testování. JTAG je zkratka pro Joint Test Action Group, součást technické komise zodpovědné za vyvíjení standardu. [3] Rozhraní JTAG umožňuje provést rekonfiguraci zařízení přes kabel pomocí nástroje iMPACT nebo ChipScope Analyzer. To může být výhodné pro počáteční seznámení s DPR, nebo pro testovací účely. [8] Všechna Virtex FPGA zařízení mohou automaticky využít tuto možnost řízení rekonfigurace. Boundary-Scan architektura je součástí FPGA zařízení firmy Xilinx [3]. Tabulka 2.3 na straně 11 obsahuje výčet a krátký popis nejdůležitějších pinů.

Slave Serial

Jednotlivé porty tohoto rozhraní jsou uvedeny v tabulce 2.4 na straně 11. Toto rozhraní je méně výkonné než varianty založené na SelectMAP, protože během každého přenosu se přesune jen jeden bit [38]. Výhoda spočívá v jednoduchosti fyzického spoje a možnosti řídit rekonfiguraci i z jednodušších zařízení.

Pin	Popis
TDI	Vstupní testovací data.
TDO	Výstupní testovací data.
TMS	Výběr módu testování.
TCK	Testovací hodiny.

Tabulka 2.3: Rozhraní JTAG [8]

Port	Směr	Popis
CCLK	Vstupní	Konfigurační hodiny.
$\overline{PROGRAM}$	Vstupní	Asynchronní reset konfigurační logiky aktivní v log. nule.
$\overline{INIT}$	Vstupní/výstupní	Vstup pro zpoždění konfigurace aktivní v log. nule. Indikuje stav, kdy je zařízení připraveno přijmout konfigurační data. Také indikuje konfigurační chyby.
DONE	Vstupní/výstupní	Vstup pro zpoždění startu zařízení. Indikuje stav, kdy zařízení provádí startovací sekvenci.
M[2:0]	Vstupní	Výběr konfiguračního módu.
DIN	Vstupní	Datový vstup sériové konfigurace.
DOUT	Výstupní	Datový výstup pro sériový *daisy chain* (někdy také označováno jako uzavřený cyklus), jenž umožňuje propojit více zařízení a provádět rekonfiguraci najednou.

Tabulka 2.4: Rozhraní Slave Serial [8]

Slave SelectMAP

Výhoda rozhraní SelectMAP spočívá v tom, že návrhář si může toto rozhraní dotvořit. Jedná se o vhodnou variantu, pokud je zapotřebí vysoká rychlost rozhraní, možnost provádět úplnou i částečnou (re)konfiguraci zařízení přes jedno rozhraní, ale pokud zároveň není z nějakého důvodu žádoucí využít rozhraní ICAP. V tabulce 2.5 jsou uvedeny porty tohoto rozhraní. Porty CCLK, $\overline{PROGRAM}$, $\overline{INIT}$, DONE a M[2:0] jsou zcela shodné s rozhraním Slave Serial (viz. tabulka 2.4). Šířka datové sběrnice může být 8 až 32 bitů; záleží na architektuře FPGA.

Port	Směr	Popis
D[0:7]	Vstupní	Datový konfigurační vstup o šířce jednoho bajtu.
$\overline{CS}$	Vstupní	Chip select aktivní v logické nule.
$\overline{WRITE}$ nebo $\overline{RDWR\_B}$	Vstupní	Write select / read select aktivní v logické nule.
BUSY	Výstupní	Handshake signál pro indikaci úspěšného přenosu dat (to samé jako DOUT v sériovém módu).

Tabulka 2.5: Rozhraní Slave SelectMAP [8]

Internal Configuration Access Port (ICAP)

ICAP je v podstatě interní verze SelectMAP rozhraní. Toto rozhraní musí být implementováno v FPGA, nebo přítomno jako hardwarový blok. Jedná se o vhodnou volbu pro většinu uživatelských návrhů a poskytuje největší výkon a flexibilitu. [8]

ICAP umožňuje nahrávat částečné bitstreamy a měnit celé rekonfigurovatelné oddíly, ale také měnit pouze jednotlivé LUT nebo klopné obvody. Rekonfigurace na takto nízké úrovni obvykle není zapotřebí, protože rekonfigurovatelná oblast je obvykle mnohem větší.

Základní popis jednotlivých portů je shodný s rozhraním SelectMAP. Další detaily nejsou uvedeny vzhledem k složitosti rozhraní. Lze je ale nalézt v [18].

Kapitola 3

Hardwarová platforma a vývojové prostředí

S návrhem frameworku logicky souvisí hardwarová platforma a softwarové prostředí. Hardware může být syntetizovaný (vytvořený pomocí součástek v FPGA) nebo přímo integrovaný na vývojové desce. Integrovaná vývojová prostředí firmy Xilinx obsahují velké množství nástrojů a konfigurovatelných jednotek hardware, které lze použít. Využití jednotlivých nástrojů a hardware usnadňuje práci a vytvoření vhodného hardwarového systému, který lze dále rozšiřovat. Použitím dalších nástrojů je možné připravit operační systém typu GNU/Linux, který je s takovým hardwarovým systémem kompatibilní. Na operačním systému je pak možno tvořit další nadstavbu. Celkově se jedná o obecné jednotky, jejichž použití je časté v mnoha návrzích. Tato kapitola poskytuje informace o hardwarových a softwarových prostředcích, jenž budou hrát ve frameworku důležitou roli.

3.1 Vývojová deska

Tato podkapitola je věnována výčtu vybraných hardwarových komponent, které lze najít na vývojovém prostředku[1]. Jsou uvedeny pouze ty komponenty, které jsou důležité z pohledu této práce. Všechny jsou ke zbytku systému připojeny přes FPGA.

- **FPGA**
  - Xilinx Virtex-5 XC5VLX110T-FF1136
- **Komunikace**
  - RS-232 sériový port
  - USB 2.0
  - 2x 10/100/1000 Ethernetový port
- **Paměť**
  - 256 MB DDR2 SODIMM modul

[1] http://www.em.avnet.com/en-us/design/drc/Pages/Xilinx-Virtex-5-LXTSXT-PCI-Express-Development-Kit.aspx

3.2 Použité IP cores

Tato podkapitola se věnuje popisu nejdůležitějších jednotek, které tvoří hardwarový systém. Jedná se o syntetizovaný hardware, protože je zcela vytvořen v FPGA. Díky tomu je možné tyto jednotlivé součásti do značné míry parametrizovat. Některé z těchto jednotek jsou nezbytné pro fungování operačního systému typu GNU/Linux, jiné umožňují uživateli komunikaci se systémem a usnadňují vývoj hardware a software. Jednotky jsou použitelné pro mnoho účelů a lze očekávat, že některé z nich budou užitečné pro návrh a implementaci frameworku.

Procesor MicroBlaze (`microblaze`)

MicroBlaze je 32bitový RISC procesor s *Harvardskou architekturou*. Tento procesor používá vícestupňové zřetězené zpracování instrukcí (*pipelining*). Stejně jako všechny další jednotky v této podkapitole je vytvořen pomocí součástek v FPGA. Označuje se tudíž jako tzv. *soft* procesor. Verze `8.40.b` má přes 70 uživatelských konfiguračních možností. Dle potřeby lze vybrat variantu, která je nejvhodnější vzhledem k požadavkům na systém.

Procesor může být konfigurován tak, aby používal *Little-Endian* nebo *Big-Endian* uspořádání. Rovněž je možné nastavit velikost paměti procesoru a paměti cache. MicroBlaze podporuje virtuální správu a pokročilou ochranu paměti. Tato konfigurační možnost je společně s jednotkou `mpmc` nutná pro systémy s operačním systémem Linux. [20] Mezi další dostupné konfigurační možnosti patří mimo jiné:

- parametrizace jádra pro urychlení zpracování
- podpora ladění v hardware
- instrukce pro plovoucí řádovou čárku a mocniny
- hardwarové výjimky
- optimalizace za účelem snížení obsazené oblasti na čipu
- detekce přetečení a podtečení zásobníku
- podpora odolnosti vůči chybám
- *byteswap* instrukce

Některé rysy tohoto procesoru jsou dány pevně. K procesoru MicroBlaze lze připojit periferie pomocí různých rozhraní. Mezi tato rozhraní patří například PLB, AXI, nebo OPB. Akcelerátory mohou být připojeny pomocí rozhraní FSL nebo AXI4-Stream. Paměť procesoru je připojena pomocí vyhrazené sběrnice Local Memory Bus (LMB). K dalším pevně daným rysům patří:

- třicet dva 32bitových registrů pro všeobecné použití
- 32bitové instrukce s třemi operandy a dvěma adresovacími módy
- 32bitová adresová sběrnice
- instrukce obvykle[2] trvá jeden hodinový cyklus

[2] V případě potřeby je procesor pozastaven (*stall*).

Multi-Port Memory Controller (`mpmc`)

Tento řadič paměti je nutností pro většinu návrhů využívajících MicroBlaze. Umožňuje jednodušší přístup k dynamické paměti systému. Verze `6.06a` umožňuje použít jeden až osm portů, přičemž u každého portu lze zvolit, jaké rozhraní bude používat. K dispozici jsou rozhraní NPI, PLB, XCL, LocalLink a další. Ne všechna rozhraní jsou podporována na každé architektuře; příkladem může být MCB, které je podporováno pouze na Spartan-6. Tato jednotka mimo jiné dále podporuje:

- celkem maximálně 2 gigabajty paměti
- přímý přístup do paměti *Soft Direct Memory Access* (SDMA)
- DDR/DDR2/DDR3/LPDDR a Single Data Rate (SDR) SDRAM paměťové moduly
- uživatelské nastavení způsobu arbitrace (pořadí obsluhování transakcí)
- různé druhy rozhraní
- sledování výkonu (*Performance Monitoring*)

Dokument [23] obsahuje velké množství informací včetně případů použití, odhadů výkonu jednotky, časových diagramů a funkčních popisů pro jednotlivá rozhraní.

LocalLink TEMAC (`xps_ll_temac`)

Tato jednotka (*Tri-Mode Ethernet Media Access Controller* – TEMAC) umožňuje připojení k Ethernetové síti. *Tri-Mode* označuje to, že jednotka může pracovat ve třech rychlostních režimech: 10 Mb/s, 100 Mb/s a 1000 Mb/s. Na FPGA čipech Virtex-4, Virtex-5 a Virtex-6 je k dispozici jako *hard block*. To znamená, že část hardware této jednotky je předem napevno integrována v FPGA. Ostatní čipy používají méně výkonnou *soft* variantu, která je zcela vytvořena pomocí běžných součástek v FPGA. Mezi základní rysy patří mimo jiné [13]:

- nezávislé TX a RX datové paměti FIFO pro umisťování rámců do front
- filtrování „špatných“ přijatých rámců
- podpora několika různých *PHY* rozhraní
- pouze režim *Full-Duplex*
- vestavěná podpora DMA a LocalLink rozhraní

Tato jednotka, stejně jako většina ostatních jednotek v této podkapitole, může být připojena k procesoru MicroBlaze pomocí PLB. To umožňuje jednodušší přístup k registrům a řízení jednotky pomocí softwarového ovladače.

LocalLink FIFO (`xps_ll_fifo`)

Toto zařízení umožňuje dekódovat LocalLink protokol a slouží jako rychlá vyrovnávací paměť. Je možné jej do určité míry parametrizovat a vyladit pro použití v určitém systému. Nejdůležitějším parametrem je šířka datové sběrnice a možnost konfigurace pro point-to-point komunikaci. Point-to-point konfigurace rozhraní snižuje latenci při komunikaci a šetří zdroje. Mezi charakteristické rysy patří mimo jiné [17]:

- Nezávislé vnitřní 2 kB TX a RX datové paměti FIFO
- Plně duplexní přenos
- Poskytuje přerušení pro mnoho chyb a stavových podmínek
- Podporuje rozhraní LocalLink

Jednotka `xps_ll_fifo` definuje celkem 11 registrů, z toho 8 je věnováno přijímání a odesílání dat. Zbylé tři slouží pro řízení (restart zařízení, nastavení a čtení stavu přerušení) samotného zařízení. Všechny registry jsou 32bitové, přičemž ne všechny jednotlivé bity jsou použity. Zbývající bity registrů jsou rezervovány pro budoucí použití. Jednotlivé registry jsou uvedeny a popsány v tabulce 3.1 na straně 17.

Pro správné použití tohoto zařízení je doporučeno seznámit se s obsahem dokumentu [17]. Zápis jednotlivých hodnot do registrů musí být proveden ve správném pořadí. Nedodržení tohoto pořadí má obvykle za následek chybu, jenž vyžaduje resetování, což má za následek ztrátu paketů uložených v zařízení.

Příkladem použití této jednotky může být komunikace s `xps_ll_temac` pro přesun velkých objemů dat bez nutnosti využít DMA.

Internal Configuration Access Port (ICAP) (`xps_hwicap`)

Toto zařízení umožňuje provádět částečnou dynamickou rekonfiguraci a bylo obecně popsáno v podkapitole 2.5. `xps_hwicap` je jednotka používaná pro FPGA Virtex-5, Spartan-6 a další rodiny FPGA zařízení. Využívá hardwarový blok, který je v těchto rodinách FPGA zařízení dostupný [18]. Předkompilované zdrojové soubory (*netlists*) tohoto zařízení jsou poskytovány firmou Xilinx, včetně ovladače pro operační systém typu GNU/Linux. [18]

Generátor synchronizačních hodin (`clock_generator`)

`clock_generator` je jednotka, která vytváří synchronizační hodinové signály požadované frekvence násobením nebo dělením frekvence synchronizačního signálu z krystalového oscilátoru. Konfiguračními možnostmi této jednotky jsou požadované parametry synchronizačních signálů. Výstupem je vygenerovaný obvod, který je specifický vůči dané architektuře. V případě, že vygenerování neuspěje, tak je vytvořena analýza vzniklé chyby. Jednotka podporuje 16 různých požadavků na synchronizační hodiny. [14]

MicroBlaze Debug Module (`mdm`)

Tento modul značně usnadňuje ladění aplikací pro procesor MicroBlaze. Připojení je realizováno přes rozhraní JTAG. Lze provádět ladění na až osmi MicroBlaze procesorech. Důležitou možností je připojení k jednotce ChipScope™ ICON, pomocí které je možno provádět ladění uživatelsky syntetizovaného hardware. [10]

Ethernet Lite Media Access Controller (`xps_ethernetlite`)

Tato jednotka umožňuje připojení k Ethernetové síti. Podporuje rozhraní Media Independent Interface (MII) a pracuje ve dvou rychlostních režimech – 10 a 100 Mbps. Existuje pouze v *soft* variantě a nevyužívá žádný hard-block. K systému je připojena pomocí sběrnice PLB, přes kterou jsou přístupné její registry. V systému GNU/Linux se pro ni používají jiné ovladače než pro `xps_ll_temac`. [11]

Registr	Počet bitů	Přístup	Popis
Řídicí			
ISR	9	R/W	Tento registr společně s IER definuje rozhraní pro jednotlivá přerušení. ISR používá jeden bit pro reprezentaci každé přerušitelné vnitřní podmínky.
IER	9	R/W	Tento registr rozhoduje, kterým zdrojům přerušení z ISR je povoleno generovat přerušení. V IER musí být nastaven bit odpovídající bitu v ISR, aby byl signál připojený k portu `IP2INTC_Irpt` nastaven do log. jedničky.
LLR	32	W	Tento registr slouží k provedení okamžitého resetu celého zařízení zapsáním specifické hodnoty. Zároveň bude aktivní výstup `Llink_rst`, který může být použit za účelem resetování zařízení na druhé straně LocalLink připojení.
Pro odesílání			
TDFR	32	W	Tento registr slouží k resetování TX FIFO. Reset nenastane, dokud se nedokončí poslední transakce přenosu.
TDFV	9	R	Tento registr slouží k zjištění obsazení TX FIFO.
TDFD	32	W	Tento registr slouží k zapsání dat pro odeslání. Lze odeslat paket o velikosti 13 až 2040 bajtů.
TLR	11	W	Tento registr se používá pro zadání velikosti celkového payloadu paketů připravených k odeslání.
Pro příjímání			
RDFR	32	W	Tento registr slouží k resetování RX FIFO. Reset nenastane, dokud se nedokončí poslední transakce přenosu.
RDFO	9	R	Tento registr slouží k zjištění obsazení RX FIFO.
RDFD	32	R	Tento registr slouží k přečtení přijatých dat. Lze přijmout paket o velikosti 13 až 2040 bajtů.
RLR	11	R	Tento registr se používá pro přečtení velikosti celkového payloadu paketů, které jsou připraveny k přijmutí.

Tabulka 3.1: Registry jednotky `xps_ll_fifo`

Řadič přerušení (`xps_intc`)

Řadič přerušení (`XPS Interrupt Controller`) agreguje přerušení od jednotlivých periferií v jeden výstup pro procesor. Registry pro kontrolu, spouštění a potvrzení přerušení jsou dostupné přes sběrnici PLB. Umožňuje připojit až 32 vstupů pro přerušení. Tuto jednotku je možné zapojit do kaskády a připojit tak i více než 32 zdrojů přerušení. Priorita jednotlivých požadavků je daná pozicí ve vektoru přerušení [6].

Timer/Counter (`xps_timer`)

Tato jednotka je důležitou součástí systému s procesorem. K procesoru MicroBlaze je připojena pomocí sběrnice PLB. Jednotka může obsahovat až dva čítače o šířce 32 bitů. Taková konfigurace je nezbytná pro systém GNU/Linux na procesoru MicroBlaze. Jednotka podporuje celkem tři různé módy: generování událostí (*generate mode*), záchyt hrany (*capture mode*) a pulzně šířkovou modulaci (*Pulse Width Modulation* – PWM). [7]

Rozhraní RS-232 (`xps_uartlite`)

UART (*Universal Asynchronous Receiver Transmitter*) umožňuje plně duplexní komunikaci přes rozhraní RS-232. Zařízení je řízeno pomocí přerušení a používá sběrnici PLB. Veškeré parametry této jednotky se nastaví před syntézou a poté je již není možné změnit. [5]

3.3 Použitá rozhraní

V rámci této práce je využito několik rozhraní různého druhu. Ta nejdůležitější jsou uvedena a popsána v této podkapitole.

Processor Local Bus (`plb_v46`)

Tato 128bitová sběrnice je často používána k připojení různých periferií k procesoru MicroBlaze. Adresová sběrnice je 32bitová. PLB je podporována na několika rodinách zařízení včetně Virtex-4 a Virtex-5. Na novějších rodinách zařízení ji nahrazuje sběrnice AXI. K této sdílené sběrnici lze připojit až 16 zařízení typu *master* a 16 zařízení typu *slave* pro přímou komunikaci. Sběrnice podporuje připojení *point-to-point*. Také umožňuje určitou parametrizaci, jež má za cíl využití pouze nezbytných zdrojů a nejlepší možný výkon. Jedná se o tzv. *system memory-mapped* sběrnici, která umožňuje přístup k ní připojeným jednotkám pomocí mapování registrů do paměti systému. To je užitečné pro řízení jednotlivých jednotek a značně usnadňuje tvorbu ovladačů. [12]

LocalLink

Toto rozhraní umožňuje vytvořit vysoce výkonné, synchronní, point-to-point spojení. Je možné přenášet data s jakýmkoliv druhem protokolu. Obecně se v tomto protokolu používají tři části: hlavička (*header*), paketová data (*payload*) a patička (*footer*). Bity jsou uspořádány ve formě Little-Endian. Komunikace může být kdykoliv jednoduše pozastavena pomocí signálů `SRC_RDY_N` a `DST_RDY_N` (tj. buď přijímající nebo odesílající stranou). Hlavní rysy jsou popsány níže.

- generická, uživatelem definovaná šířka dat (násobky bajtů)
- rámce libovolné délky
- efektivní použití šířky pásma
- kontrola parity a chyb [2]

Výčet jednotlivých povinných signálů a jejich popis je uveden v tabulce 3.2. Logické signály (všechny, s výjimkou prvních dvou uvedených) jsou aktivní v logické nule.

Pro ilustraci fungování přenosu dat pomocí tohoto protokolu je uveden časový diagram na obrázku 3.1 na straně 19.

Signál	Šířka [b]	Druh signálu	Význam
CLK	1	vstupní	Synchronizační hodiny
DATA	$n \times 8$	vstupní	Paketová data
SOF_N	1	vstupní	Začátek rámce
EOF_N	1	vstupní	Konec rámce
SRC_RDY_N	1	vstupní	Zdrojová strana připravena
DST_RDY_N	1	výstupní	Cílová strana připravena

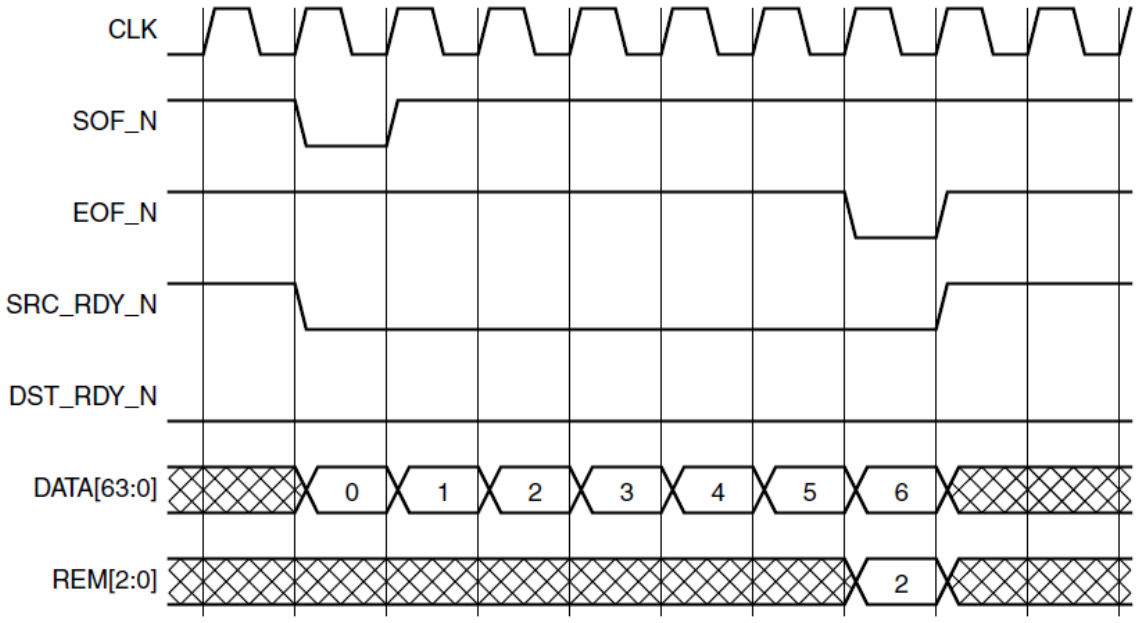
Tabulka 3.2: Povinné rozhraní LocalLink

Nepovinný signál `REM` má šířku $\log_2$N (kde N je šířka signálu `DATA`), v tomto případě to jsou 3 bity. Tento signál určuje pozici platného bajtu při přenosu posledního slova. Signál `DATA` má šířku 64 bitů.

Během znázorněného přenosu se posílají pouze paketová data (payload). Přenos začíná tím, že zdroj uvede signály `SOF_N` a `SRC_RDY_N` do logické nuly. Signál `DATA` musí být v této chvíli platný, provádí se první přenos dat. Při následující náběžné hraně synchronizačních hodin musí být signál `SOF_N` deaktivován. V případě potřeby může zdrojová nebo cílová strana pozastavit přenos deaktivací signálu `SRC_RDY_N` nebo `DST_RDY_N`.

V tomto konkrétním případě zůstává signál `DST_RDY_N` v logické nule. To znamená, že cílová strana nevkládá žádné čekací stavy a je schopná přijímat data bez prodlev. Signál `SRC_RDY_N` zůstává taktéž aktivní od začátku až do konce přenosu. To znamená, že zdrojová strana má data připravena a dokáže vysílat bez prodlev.

Na konci přenosu je aktivován signál `EOF_N`. Při následující náběžné hraně synchronizačních hodin musí být opět deaktivován. Zároveň se signálem `EOF_N` je deaktivován signál `SRC_RDY_N`. Tím končí celá operace. Z časového diagramu je patrné, kolikrát se provedl přenos po datové sběrnici a kolik se přeneslo bajtů.



Obrázek 3.1: Časový diagram protokolu LocalLink [2]

FrameLink

FrameLink je podmnožinou rozhraní LocalLink, má s ním společných mnoho rysů a je s ním kompatibilní. Od rozhraní LocalLink se liší především tím, že neumožňuje kontrolu parity a chyb. [40]

Úplné rozhraní FrameLink je shodné s povinným (minimálním) rozhraním LocalLink (viz. tabulka 3.2), přičemž by měly být definovány další tři signály, které jsou pro rozhraní LocalLink volitelné. Jedná se o vstupní signály `SOP_N`, `EOP_N` a `REM`. První z uvedených signálů značí začátek volitelné části. Druhý z uvedených signálů značí konec volitelné části. `REM` je signál o šířce $\log_2$N (kde N je šířka signálu `DATA`). Tento signál určuje pozici platného bajtu při přenosu posledního slova. Všechny logické signály jsou aktivní v logické nule.

V této práci bude tento protokol použit, aby bylo možné sjednotit způsob, kterým budou komunikovat akcelerátory s řídicí logikou. Takové sjednocení je nutné, protože jednotlivé řídicí registry (společně se související logikou) budou generovány jednotným způsobem automaticky.

3.4 Softwarové nástroje

Pro vytvoření návrhu s částečnou dynamickou rekonfigurací je zapotřebí použít mnoho nástrojů. Další softwarové prostředky jsou nutné pro vývoj aplikací na procesoru MicroBlaze. Tato podkapitola obsahuje minimální výčet a popis těchto nástrojů.

Embedded Development Kit (EDK)

Platform Studio a Embedded Development Kit je integrované vývojové prostředí pro vestavěné systémy. Obsahuje sadu nástrojů, které uživateli umožní vytvořit projekt pro danou hardwarovou platformu. Při použití grafického prostředí je uživatel proveden sérií dialogů. Automaticky jsou přidány základní jednotky (syntetizovaný hardware), které uživatel pravděpodobně využije.

Tento nástroj, stejně jako mnoho jiných nástrojů firmy Xilinx, je provázán s jinými nástroji. Součástí je studio SDK (Software Development Kit), které je důležité pro export popisu hardwarové platformy a knihoven pro použití dalšími nástroji. Zpracování tohoto popisu hardware umožňuje nastavit konfiguraci jádra operačního systému Linux tak, aby byly zahrnuty důležité součásti. Exportované knihovny obsahují kód pro řízení některých jednotek hardware nebo adresy registrů a další konstanty.

PlanAhead Design and Analysis Tool

PlanAhead je nástroj, který umožňuje uživateli zlepšit výkonnost obvodu definováním a analyzováním RTL (Register Transfer Level) zdrojových souborů v návrhu, syntetizovaných souborů netlist a výsledků implementace. Uživatel může experimentovat s různými implementačními volbami, zušlechťovat omezení časování a aplikovat fyzická omezení pomocí techniky floorplanning. Tato technika je nezbytná pro použití částečné dynamické rekonfigurace.

Nástroj umožňuje provádět mnoho různých úkonů, ale floorplanning je z pohledu částečné dynamické rekonfigurace nejdůležitější. Uživatel musí na čipu umístit jednotlivé rekonfigurovatelné oddíly. Dokument [21] obsahuje mimo jiné informace o důležitých technikách hierarchického návrhu a o tom, co vše nástroj umožňuje.

Xilinx Open Source Linux

Jedná se o operační systém[3] typu GNU/Linux. Nutnost využít operační systém tohoto typu vyplývá z požadavků na framework (resp. aplikace pomocí něho vytvořené). Poskytuje potřebnou flexibilitu a různé abstrakce (např. pseudozařízení nebo virtuální paměť). Jádro systému obsahuje mnoho ovladačů, které jsou specifické vůči syntetizovanému hardware.

Xilinx Toolchain 2.0

Tato sada nástrojů[4] je orientovaná na procesor MicroBlaze a umožňuje kompilovat C/C++ kód do souborů spustitelných na operačním systému typu GNU/Linux. Jedná se o nezbytnost pro vytváření a ladění aplikací. Obsahuje nástroje a knihovny, které se běžně používají na různých platformách (např. `gcc` a knihovna `stdio.h`).

MBSL/Crosstool

MicroBlaze Simple Linux Distribution[5] (MBSL) a Crosstool[4] jsou dva různé nástroje, které umožňují dosáhnout stejného výsledku. Hlavním účelem těchto nástrojů je vytvořit binární soubor, který obsahuje distribuci systému GNU/Linux, vhodné ovladače pro jednotlivá zařízení a aplikace.

MBSL je v současné době označován jako *obsolete* (zastaralý). Je jednodušší na použití než Crosstool a vhodný pro začínající uživatele nebo uživatele, kteří používají menší množství balíčků a aplikací. Ke své činnosti potřebuje Xilinx Toolchain 2.0 nebo podobnou sadu nástrojů.

Crosstool je rozvíjen a udržován větším množstvím vývojářů. Je vhodný při využití velkého množství balíčků. Bývá dodáván s novější sadou nástrojů, která plní stejnou funkci jako Xilinx Toolchain 2.0.

[3] https://github.com/xilinx
[4] http://www.li-pro.de/xilinx_mb/toolchain/start
[5] https://github.com/jviki/mbsl

Kapitola 4

Návrh frameworku

Aby byl framework užitečný, musí splňovat určité základní požadavky. Ty spočívají ve zjednodušení tvorby aplikací s DPR na FPGA Virtex-5 a zjednodušení přenosu dat do akcelerátorů, které jsou umístěny v rekonfigurovatelných oddílech. Podkapitola 4.1 obsahuje navržený koncept frameworku, který tyto požadavky splňuje. Následující podkapitoly se zabývají jednotlivými částmi návrhu podle uvedeného konceptu.

Částečnou dynamickou rekonfiguraci je možné řídit (provádět) pomocí různých rozhraní. Rekonfiguraci však není vhodné provést kdykoliv a je zapotřebí eliminovat či ošetřit nežádoucí stavy, ke kterým dochází při rekonfiguraci akcelerátorů. Těmito tématy se zabývá podkapitola 4.2.

Z požadavků předchozích podkapitol na hardware vychází podkapitola 4.3, kde je popsán návrh obálky rekonfigurovatelných akcelerátorů.

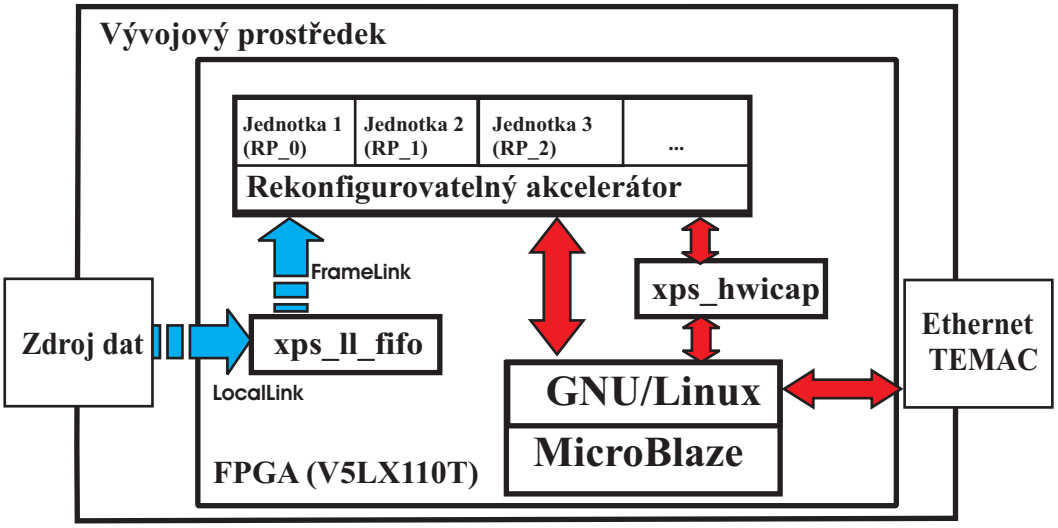
V podkapitole 4.4 jsou stručně popsány důvody pro použití softwarové nadstavby. Uvedený software umožňuje další zjednodušení tvorby uživatelských aplikací s částečnou dynamickou rekonfigurací. Návrh frameworku počítá s tím, že řízení rozhraní ICAP popsané v podkapitole 4.2 bude implementováno jako modul s použitím popsané nadstavby.

V podkapitole 4.5 je uvedeno krátké shrnutí této kapitoly v podobě seznamu hlavních úkonů, které je třeba provést pro implementaci návrhu frameworku. Zároveň je graficky naznačeno, jakým způsobem ovlivňuje framework práci s částečnou dynamickou rekonfigurací.

4.1 Koncept frameworku

Obrázek 4.1 na straně 23 ukazuje logické schéma frameworku. Jsou zvýrazněny jednotky a rozhraní, které jsou důležité z pohledu požadavků na framework. Tyto jednotky a rozhraní byly popsány v podkapitole 3.2 a 3.3. Modré čáry znázorňují přenos velkého objemu dat, zatímco červené čáry označují přítomnost více řídicích signálů. V případě, že je uvedena jednotka, která popsána nebyla, tak to znamená, že tuto jednotku bude třeba teprve implementovat.

Technologie DPR je často využívána pro aplikace, které vyžadují vysoce náročné výpočty. Mnohé takové aplikace používají akcelerátory, do kterých se přenáší velké množství dat k zpracování. Koncept frameworku počítá s možností takového scénáře. Zdrojem dat může být jakákoliv jednotka s rozhraním LocalLink, kterou uživatel potřebuje pro svou aplikaci. Tato jednotka komunikuje s vyrovnávací pamětí typu FIFO (`xps_ll_fifo`), která umí dekódovat LocalLink protokol.



Obrázek 4.1: Koncept frameworku

Součástí frameworku tedy bude demonstrační ovladač pro jednotku `xps_ll_fifo`. Zpracování bude možné provádět na univerzálním procesoru, nebo v akcelerátorech.

Rekonfigurovatelný akcelerátor je obálka (dále jen „wrapper“) pro rekonfigurovatelné akcelerátory (`RP_*`), které realizují danou funkci v hardware. Tento wrapper bude přímo umožňovat následující:

- Jednoduché přidání více akcelerátorů
- Řízení přenosů z vyrovnávací paměti FIFO
- Eliminace nežádoucích stavů hardware při rekonfiguraci
- Čtení stavů akcelerátorů
- Předání řízení přenosu dat

Výše zmíněný wrapper funguje do značné míry jako propojovací a řídicí logika. Jeho chování bude možné ovlivnit pomocí řídicích registrů. Společně se svým softwarovým ovladačem bude uživateli usnadňovat práci při použití částečné dynamické rekonfigurace.

Systém obsahuje další hardware, jehož účel a funkce byly popsány v podkapitole 3.2. Uživatel se systémem může komunikovat přes rozhraní Ethernet (umožněno jednotkou `xps_ethernetlite`) nebo RS-232 (umožněno jednotkou `xps_uartlite`). Řadič paměti (`mpmc`) obstarává správu paměti. Ostatní jednotky jsou buď nezbytné (např. řadič přerušení `xps_intc`), nebo usnadňují práci a vývoj (např. modul pro ladění `mdm`).

Zjednodušení tvorby aplikací

Lze uvažovat, že uživatelská aplikace s DPR se bude skládat z různých komponent. Jak bylo naznačeno, hardwarové jednotky budou řízeny pomocí softwarových ovladačů. Ty můžou na procesoru MicroBlaze běžet jako samostatná (*stand-alone*) softwarová aplikace, nebo v rámci operačního systému typu GNU/Linux. Použití operačního systému má oproti stand-alone aplikaci mnoho výhod. Ty nejdůležitější jsou uvedeny níže:

- Podpora víceúlohového zpracování (*multitasking*)
- Dostupnost ovladačů zařízení a aplikací pro usnadnění vývoje
- Dostupnost záplat pro existují balíčky
- Možnost používat a vytvářet abstrakce (oddělení implementace a rozhraní)

Těchto výhod je vhodné co nejvíce využít. Podkapitola 4.4 popisuje důvody pro využití složitější softwarové nadstavby nad GNU/Linux. Softwarová nadstavba a modul řízení částečné dynamické rekonfigurace uživateli značně zjednoduší tvorbu aplikací s DPR.

4.2 Návrh řízení rozhraní ICAP

Možnosti řízení procesu částečné dynamické rekonfigurace byly popsány v podkapitole 2.5. Bude použito rozhraní ICAP, konkrétně verze `xps_hwicap` určená pro Virtex-5. Tato podkapitola zmiňuje několik důležitých důvodů, proč je třeba, aby rozhraní ICAP bylo řízeno pomocí softwarového ovladače. Je popsáno, jak musí řízení tohoto rozhraní fungovat v souvislosti s rekonfigurovatelnými akcelerátory.

Eliminace nežádoucích stavů hardware při rekonfiguraci

Provedení rekonfigurace pomocí `xps_hwicap` je poměrně přímočaré a spočívá v zápisu bitů do odpovídajícího pseudozařízení.

```
cat bitstream.bit > /dev/icap
```

Takovýto příkaz však není vhodné provést zcela kdykoliv. Rekonfigurace aktivního akcelerátoru může vést k nežádoucímu chování. Jako příklad můžeme uvést jednotku s rozhraním FrameLink, která není připravena přijímat data a vkládá čekací stav nastavením signálu `DST_RDY_N` do logické jedničky. Pokud by byl tento akcelerátor rekonfigurován a hodnota signálu `DST_RDY_N` zůstala stejná, pak by zdroj dat čekal na akcelerátor donekonečna. Podobnému chování je třeba zamezit. Za tímto účelem je třeba sekvenčně provést několik kroků, jejichž výčet je uveden níže. Tyto kroky lze do značné míry považovat za univerzální, ale návrh jejich provedení je specifický vůči konceptu frameworku.

- Pozastavení jednotek, které budou rekonfigurovány
- Provedení rekonfigurace
- Restart jednotek, které byly rekonfigurovány

První krok vyžaduje určit vhodný časový interval, kdy je možné jednotku pozastavit. Vzhledem k navrženému konceptu přenosu dat ze zdroje dat do akcelerátorů lze uvažovat, že jednotku lze pozastavit, pokud existuje alespoň jedna jiná jednotka, která je schopna dokončit aktuální transakci. Díky tomu by měla být vyrovnávací paměť FIFO (`xps_ll_fifo`) schopná vždy dokončit přenos aktuálního paketu. Tato funkce bude součástí wrapperu a řídit ji bude softwarový ovladač.

Následující krok spočívá v provedení rekonfigurace. Po dobu rekonfigurace (tj. až po dokončení zápisu bitů do pseudozařízení) je nutné považovat veškeré vstupy a výstupy z rekonfigurovaného akcelerátoru vždy za neplatné. Wrapper musí ignorovat tyto neplatné vstupy a výstupy. I tato funkce bude řízena softwarovým ovladačem.

Po dokončení zápisu bitů do pseudozařízení lze jednotku považovat za úspěšně rekonfigurovanou. Poslední krok spočívá v aktivaci resetovacího signálu dané jednotky. Po dokončení resetu lze vstupy a výstupy daného rekonfigurovatelného akcelerátoru považovat opět za platné.

Určení vhodného časového intervalu pro provedení rekonfigurace konkrétních akcelerátorů z pohledu výkonnosti spočívá na uživateli. Naplánování rekonfigurace na správný čas souvisí s konkrétním scénářem použití.

Popis bitstreamů

Ovladač rozhraní ICAP (resp. pseudozařízení `/dev/icap`) umožňuje rekonfiguraci pomocí zápisu souboru bitstreamu v binárním formátu. Zápis chybného souboru do zařízení může způsobit nespecifikované chování. Aby se snížila pravděpodobnost takovéto nežádoucí akce, tak uživatel bude muset v konfiguračním souboru specifikovat jednotlivé bitstreamy a odpovídající rekonfigurovatelné oddíly. To bude zároveň užitečné pro scénáře, kdy bude o jednotlivých oddílech zapotřebí sbírat informace nebo vytvářet statistiky. Navíc bude možné přizpůsobit chování software podle konfigurace nahrané v FPGA. Konfigurační soubor bude vypadat následovně:

```
s0.bin z0.bin<LF>
s1.bin z1.bin<LF>
s2.bin z2.bin<LF>
```

V tomto případě existují tři RP, každý má dva moduly. Číslo řádku odpovídá číslu RP, číslo sloupce odpovídá číslu modulu. `<LF>` označuje zalomení řádku (*line feed*), kódování souboru je ASCII.

4.3 Návrh obálky rekonfigurovatelných akcelerátorů

V této podkapitole jsou popsány důležité části návrhu wrapperu vzhledem k požadavkům na framework.

Návrh rozhraní akcelerátorů

Pro přenos dat do akcelerátoru bude použito rozhraní FrameLink. Toto rozhraní bylo popsáno v podkapitole 3.3.

Ohledně akcelerátorů samotných není možno předem dělat mnoho závěrů. Je ale pravděpodobné, že budou využívat další porty, které nesouvisí s přenosem dat, ale s indikací stavu své činnosti. V tabulce 4.1 na straně 26 je navrhnuto jednoduché rozhraní pro indikaci stavu. Všechny porty mají šířku 1 bit a využívá je každý jednotlivý akcelerátor.

Výsledky operací budou uloženy do fronty typu FIFO (zařízení `rdpfifo_v4_02_a` [9]). Díky tomuto zařízení není zapotřebí provádět neustále v smyčce dotazování (*polling*) na stav operace. Možným vylepšením by bylo využít toto zařízení společně s přerušením, například při určitém zaplnění dostupné paměti.

Tento koncept frameworku počítá se scénářem, kdy všechny hardwarové akcelerátory zpracovávají stejná data a pracují synchronně. V případě, že by scénář použití frameworku byl jiný, tak je možno vytvořit více instancí (jednotky `xps_ll_fifo` a rekonfigurovatelných akcelerátorů `RP_*`) nebo framework lehce upravit.

Port	Druh portu	Význam
BITMAP	výstupní	Výsledek operace akcelerátoru, sémantika dle konkrétního uživatelského akcelerátoru
VLD	výstupní	Potvrzení platnosti hodnoty `BITMAP`, aktivní v logické jedničce
ACK	vstupní	Potvrzení přečtení hodnoty `BITMAP` řídící logikou wrapperu akcelerátorů

Tabulka 4.1: Doplněk rozhraní akcelerátorů

Řídicí registry

Jak bylo popsáno v podkapitole 4.2, během rekonfigurace musí být hodnoty signálů jednotlivých rekonfigurovatelných akcelerátorů ignorovány, protože jsou neplatné. Tato hardwarová logika bude ovlivněna nastavením řídicího registru `UR` (`unit_reconfigurating`). Odliší 2^N stavů, kde N odpovídá počtu rekonfigurovatelných oddílů. Pozice bitu v registru odpovídá číslu rekonfigurovatelného oddílu. Nejméně významný bit (LSB) odpovídá stavu rekonfigurovatelného oddílu číslo 0, nejvíce významný bit (MSB) odpovídá rekonfigurovatelnému oddílu $M-1$, kde M je šířka registru. Logická jednička bude označovat stav, kdy se jednotka rekonfiguruje; logická nula bude označovat stav, kdy se rekonfigurace neprovádí.

Jednotka `xps_ll_fifo` funguje jako rychlá vyrovnávací paměť. Jedná o frontu typu FIFO. V závislosti na rychlosti zpracování jsou z ní vyčítány pakety. Je žádoucí, aby úzkým hrdlem systému byla tato paměť. Pakety mají být co nejrychleji předány k zpracování. Vzhledem k požadavkům na framework je vhodné umožnit zpracování dat v akcelerátorech nebo alternativně na univerzálním procesoru. Zpracovávat data zároveň na univerzálním procesoru a zároveň v akcelerátorech nedává pro většinu scénářů žádný smysl.

Je tudíž třeba jasně odlišit, zdali zpracování probíhá v akcelerátorech, nebo na univerzálním procesoru. K tomuto účelu bude sloužit registr `AHC` (`accelerator_has_control`). Bude odlišovat dva stavy – zpracování v hardware (log. jednička v LSB) nebo v software (log. nula v LSB). To zároveň znamená, že jednotka `xps_ll_fifo` bude buď řízena pomocí řídicí logiky wrapperu, nebo pomocí softwarového ovladače pro operační systém GNU/Linux.

Řízení přenosu dat do akcelerátorů

Přenos dat z vyrovnávací paměti do akcelerátorů je důležitou součástí wrapperu. Řízení přenosu dat pomocí níže popsaného rozhraní je povoleno pouze tehdy, pokud je v LSB registru `AHC` zapsána logická jednička. Synchronizace činnosti vyrovnávací paměti s rekonfigurovatelnými akcelerátory bude vyžadovat drobné úpravy implementace jednotky `xps_ll_fifo`. Tabulka 4.2 na straně 27 obsahuje výpis portů, které takové řízení umožňují. Pro většinu obvyklých scénářů použití frameworku není zapotřebí, aby uživatel do tohoto rozhraní zasahoval.

Všechny signály připojené k jednotlivým portům jsou aktivní v logické jedničce. Porty `NUMBER_OF_BYTES_FOR_ACCELERATOR` a `DATA4ACCELERATOR` jsou 32bitové, ostatní porty mají šířku 1 bit.

Popis sémantiky signálů připojených k jednotlivým portům je uveden níže. Do značné míry souvisí s registry jednotky `xps_ll_fifo`. Řízení zařízení pomocí signálů v hardware a pomocí zápisu hodnot do registrů v software lze porovnat nahlédnutím do tabulky 3.1 v podkapitole 3.2.

Signál `ACCELERATOR_HW_ENABLE` je aktivní při předání řízení do hardware (zpracování dat

Port	Druh portu
ACCELERATOR_HW_ENABLE	výstupní
NUMBER_OF_BYTES_FOR_ACCELERATOR	vstupní
DATA4ACCELERATOR	vstupní
RXFIFO_READ_RLR	výstupní
RXFIFO_READ_DATA	výstupní
RXFIFO_EMPTY	vstupní
RXD_RX_ACK	vstupní

Tabulka 4.2: Interní rozhraní pro řízení jednotky `xps_ll_fifo`

pomocí akcelerátorů). Neaktivní signál umožňuje zpracování dat pomocí procesoru v software.

Signál `NUMBER_OF_BYTES_FOR_ACCELERATOR` určuje, kolik bajtů se bude přenášet z paměti do akcelerátorů. Značí velikost celého paketu v `xps_ll_fifo`. Hodnota tohoto signálu se s postupným vyčítáním bajtů z paměti nemění.

Signál `DATA4ACCELERATOR` obsahuje bajty dat pro akcelerátory. Při posledním přenosu může být počet platných bajtů nižší než 4. Počet platných bajtů je vypočítán v rámci logiky wrapperu.

Aktivní signál `RXFIFO_READ_RLR` způsobí přečtení registru RLR. Nejdříve v následujícím cyklu bude aktivován signál `RXD_RX_ACK`, který závazně potvrzuje úspěšné přečtení dat z paměti. Signál `NUMBER_OF_BYTES_FOR_ACCELERATOR` pak bude obsahovat velikost paketu v bajtech. Tento signál by měl být aktivován pouze jednou – na začátku přenosu, tj. před vyčtením jakýchkoliv dat z paměti. V jiném případě aktivace tohoto signálu způsobí nedefinované chování a bude pravděpodobně zapotřebí resetovat celé zařízení `xps_ll_fifo`.

Aktivní signál `RXFIFO_READ_DATA` způsobí přečtení registru RDFD. Nejdříve v následujícím cyklu bude aktivován signál `RXD_RX_ACK`, který závazně potvrzuje úspěšné přečtení dat z paměti. To pak značí, že signál `DATA4ACCELERATOR` obsahuje platná data.

Aktivní signál `RXFIFO_EMPTY` znamená, že paměť je prázdná a není z ní možné vyčíst žádná data.

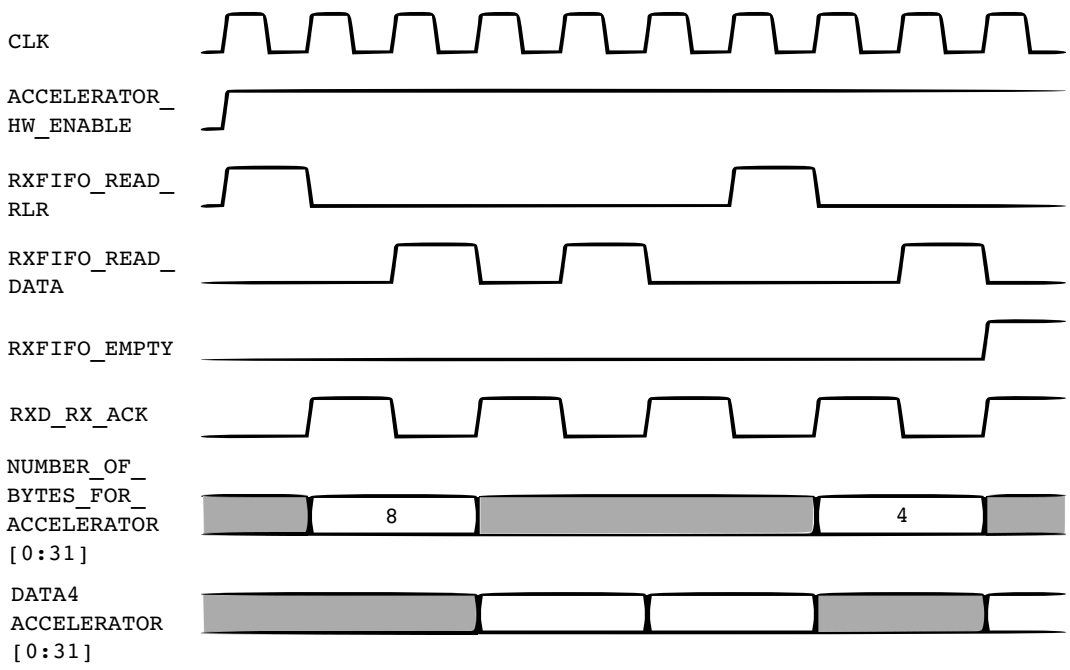
Aktivní signál `RXD_RX_ACK` potvrzuje úspěšné přečtení dat z paměti.

Obrázek 4.2 na straně 28 dále ilustruje sekvenci přenosu a sémantiku signálů. Během operace se provede přenos dvou paketů. Šedá barva označuje interval kdy signál neobsahuje platná data.

Poznámka: Tento příklad přenosu byl zjednodušen, aby byl diagram dobře čitelný. Každý paket v jednotce `xps_ll_fifo` bude mít minimálně 13 bajtů. Tomu bude odpovídat hodnota signálu `NUMBER_OF_BYTES_FOR_ACCELERATOR` a bude tudíž zapotřebí provést minimálně čtyři čtení pro získání celého paketu.

4.4 Softwarová nadstavba

V podkapitole 4.2 byly popsány úkony, které jsou zapotřebí v souvislosti s řízením částečné dynamické rekonfigurace. Výsledkem by mohla být spustitelná aplikace pro operační systém typu GNU/Linux, která by uživateli umožňovala provádět rekonfiguraci. Takovéto řešení by ale nebylo modulární a ani dobře znovupoužitelné. Přidávání jakýchkoliv dalších funkcí by vytvářelo systém, který má mezi jednotlivými částmi příliš silné vazby. To by uživateli



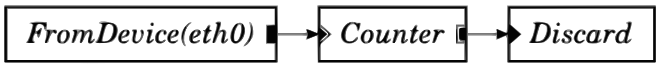
Obrázek 4.2: Časový diagram řídicího rozhraní wrapperu

neusnadňovalo práci s částečnou dynamickou rekonfigurací, protože takřka všechny úpravy by bylo nutné provádět v rámci C/C++ kódu aplikace.

Z těchto důvodů je lepší využít softwarové nadstavby nad GNU/Linux. Takovouto nadstavbou může být softwarový nástroj, který umožní modulární návrh aplikace. Příkladem takového nástroje je Click Modular Router[1] (dále jen „Click“), který je kompatibilní s procesorem MicroBlaze. Použití tohoto software poskytuje frameworku značné výhody pro síťové aplikace, ale není omezeno pouze na něj.

Elementy

Click označuje jednotlivé moduly jako elementy (*elements*). Elementy mohou ve své implementaci využívat funkcí, které Click poskytuje, ale komunikovat s jinými elementy mohou pouze pomocí definovaných vstupních a výstupních portů. Maximální počet portů není omezen. Porty mohou být různého typu a liší se způsobem předání dat. Obrázek 4.3 ukazuje propojení několika elementů.



Obrázek 4.3: Jednoduché propojení elementů [35]

Propojení mezi porty jednotlivých elementů, stejně jako konfiguraci elementů, definuje uživatelský skript. Ekvivalentní skript k obrázku 4.3 je uveden níže. Tento skript vyčítá pakety ze zařízení, počítá je a zahazuje.

```
FromDevice(eth0) -> Counter -> Discard;
```

[1] http://read.cs.ucla.edu/click/click

Změna konfigurace elementů a řízení rekonfigurace

Click používá pro čtení a změnu konfigurace elementů tzv. handlery[2] (*handlers*). Jejich hodnotu lze zjistit nebo upravit voláním funkce. Zároveň bývá vrácena jedna ze standardizovaných hodnot, která odpovídá výsledku operace. V případě, že Click běží jako modul jádra, pak jsou handlery také přístupné pomocí souborového systému `/proc` [35].

Handlery je možné měnit pomocí jakéhokoliv elementu. Framework bude využívat handlery pro změnu chování modulu a řízení částečné dynamické rekonfigurace. Zvláště vhodný je element `ControlSocket`[3], který taktéž umožňuje vzdálenou správu. Uživatel se k němu může připojit přes telnet a měnit handlery jednoduchými příkazy.

Za nejdůležitější druh příkazů budeme považovat ty, které řídí částečnou dynamickou rekonfiguraci. Níže je uveden příklad sekvence příkazů pro `ControlSocket`, která provede rekonfiguraci tří rekonfigurovatelných oddílů. `write` je příkaz pro zápis a `read` je příkaz pro čtení hodnoty handleru. Oddíl (jednotka – unit**X**, kde X je číslo jednotky) načte odpovídající modul (bitstream – bit**Y**, kde Y je číslo bitstreamu). Bitstreamy jsou značeny podle souboru s formátem popsaným v podkapitole 4.2. Rekonfigurace jednotlivých oddílů se nezačne provádět, dokud uživatel neodešle příkaz `reconfigurate`.

```
write unit1_bit1 1
write unit2_bit1 1
write unit3_bit1 1
reconfigurate
```

Tento přístup má výhodu, že protokol rozhraní se vytvoří dynamicky. Dodržení jednotného značení rekonfigurovatelných oddílů a jejich modulů usnadňuje práci programátorovi při úpravách/rozšiřování modulu řízení částečné dynamické rekonfigurace. Také je možné zjišťovat stav rekonfigurace pomocí příkazu `read`.

Pro uživatelskou aplikaci je také možné použít volání funkce pro změnu handleru (a tudíž i řízení rekonfigurace). Po změně hodnot odpovídajících handlerů je třeba zaslat procesu signál `SIGUSR1` pro provedení rekonfigurace.

4.5 Design flow a shrnutí

Tato podkapitola je věnována shrnutí kapitoly – stručnému výčtu nutných kroků pro vývoj frameworku. Zároveň je graficky znázorněn průběh návrhu aplikace s DPR pomocí frameworku a softwarových nástrojů.

Z předcházejících podkapitol vyplývá, že pro vytvoření frameworku je třeba provést následující úkony:

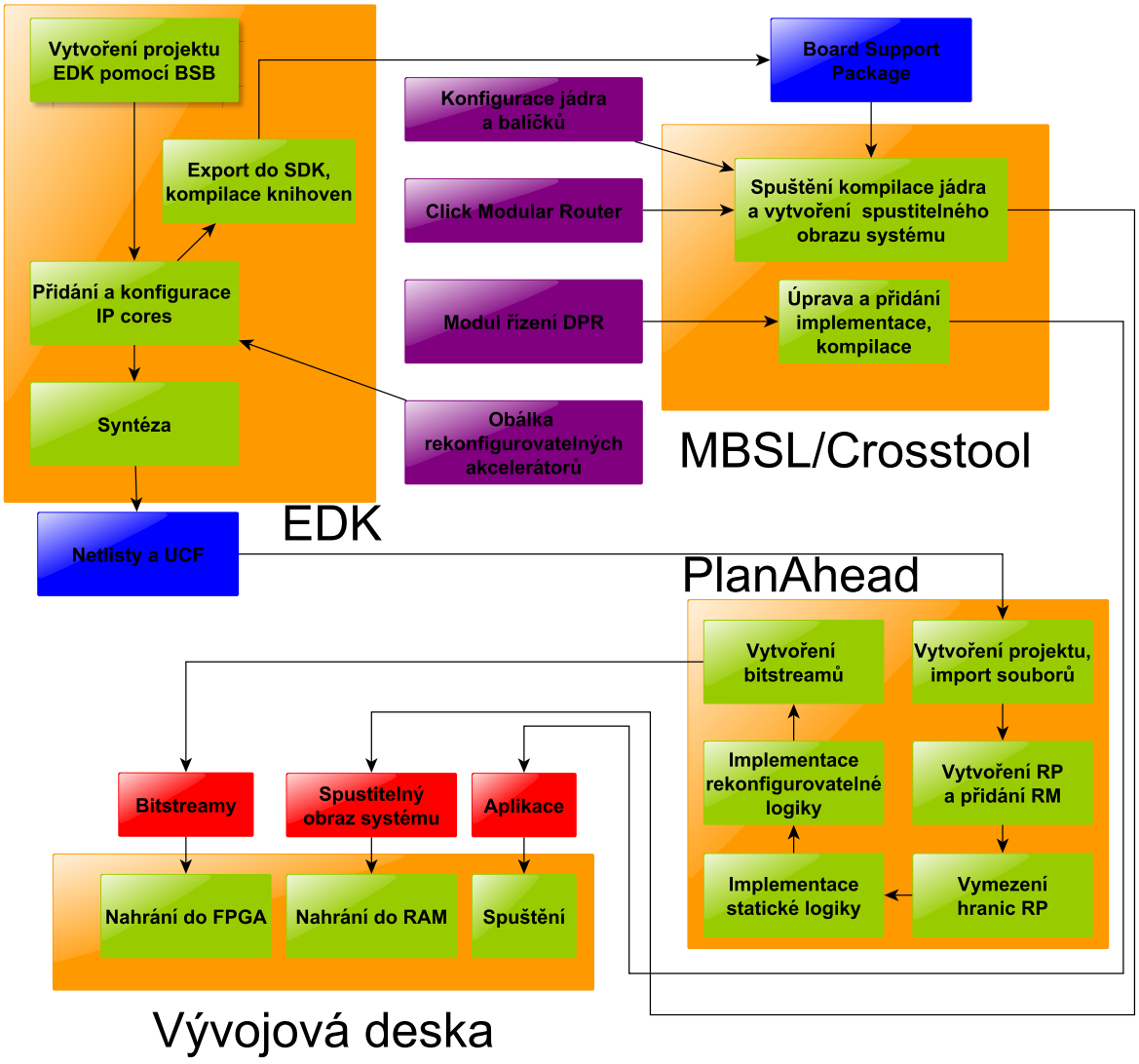
- Implementace hardware v jazyce VHDL
  - Obálka (wrapper) rekonfigurovatelných HW akcelerátorů
  - Úprava `xps_ll_fifo` pro synchronizaci s akcelerátory
  - Řídicí logika
- Implementace software v jazyce C/C++
  - Řízení částečné dynamické rekonfigurace

[2] http://read.cs.ucla.edu/click/element?s[]=handler
[3] http://read.cs.ucla.edu/click/elements/controlsocket

– Ovladač pro LocalLink FIFO (`xps_ll_fifo`)

- Vytvoření konfigurace jádra GNU/Linux a balíčků
- Integrace jednotlivých součástí s Click Modular Router

Na obrázku 4.4 je znázorněno, jak framework zasahuje do *design flow* částečné dynamické rekonfigurace. Oranžové bloky vyznačují hranice systému (jednotlivé softwarové nástroje). Zelené bloky v nich označují procesy, které buď provádí nástroj, nebo uživatel. Modré bloky znázorňují mezivýsledky z jednotlivých nástrojů. Framework je vyznačen pomocí fialové barvy. Červeně jsou pak vyznačeny konečné výsledky (bitstreamy, obraz systému, aplikace). Tento obrázek ukazuje, jakým způsobem framework ovlivňuje design flow částečné dynamické rekonfigurace.



Obrázek 4.4: Partial Reconfiguration Design Flow

Dokument [8] uvádí informace o jednotlivých procesech a interakcích s nástroji. Obsahuje mimo jiné obrázky zachycující důležité dialogy grafických prostředí nástrojů.

Kapitola 5

Implementace frameworku a návrh demonstrační aplikace

V rámci předchozí kapitoly byl navržen framework, který lze použít pro usnadnění tvorby aplikací s částečnou dynamickou rekonfigurací.

Podkapitola 5.1 se věnuje implementaci frameworku – tj. implementaci softwarových ovladačů (jazyk C/C++) a hardwarového wrapperu (jazyk VHDL).

Podkapitola 5.2 pak ukazuje použití implementovaného frameworku pro vytvoření aplikace. Nejdříve je popsána demonstrační aplikace, která byla vytvořena. Následně je pak vysvětleno, jakým způsobem framework vytvoření takovéto aplikace umožnil.

Ukázka použití vytvořené demonstrační aplikace je uvedena v podkapitole 5.3. Je popsán formát zpráv, kterými aplikace uživatele informuje o své činnosti. Jsou uvedeny krátké příklady výstupů a je znázorněno, jakým způsobem uživatel ovládá softwarový modul pro řízení rekonfigurace.

5.1 Implementace

Tato podkapitola obsahuje popis zajímavých částí implementace hardware a software, které jsou důležité pro framework.

Řízení přenosu dat

Řízení přenosu dat je implementováno pomocí konečného automatu. Tabulka přechodů 5.1 na straně 32 popisuje činnost tohoto automatu. Pro úplnost a přehlednost jsou vstupní podmínky uvedeny na stejné straně ve vlastní tabulce 5.2, stejně jako výstupy, které jsou uvedeny v tabulce 5.3. Syntax a sémantika výrazů v druhém sloupci tabulky 5.2 a tabulky 5.3 odpovídá syntaxi a sémantice výrazů ve VHDL.

Během stavu `IDLE` se neděje nic – není možné přenášet data do žádného akcelerátoru. Jedná se o počáteční stav. V případě, že žádný akcelerátor není schopen zpracovávat data (`acc_is_good` v log. nule), tak je aktuální paket zahozen a automat přejde do stavu `IDLE`. Toto chování je stejné i pro ostatní stavy. Pro přechod ze stavu `IDLE` do `START` musí být k dispozici alespoň jeden paket k přenosu ve vyrovnávací paměti a alespoň jeden akcelerátor musí být připraven tyto data přijímat. Zároveň musí být místo v paměti FIFO pro zápis výsledku. Stav `SOF` označuje začátek přenosu paketu do akcelerátorů. Použité akcelerátory mají datovou sběrnici o šířce 8 bitů. Slovo z vyrovnávací paměti (4 bajty) je tudíž třeba postupně „rozbalit“ (unpack) a přenést po bajtech do akcelerátorů.

Aktuální stav	Následující stav	Vstupní podmínka	Výstup
IDLE	IDLE	$\neg(x_1 \wedge x_2 \wedge x_3 \wedge x_4)$	y_1
IDLE	START	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	
START	IDLE	$\neg x_1$	
START	SOF	x_1	y_6
SOF	SOF	$\neg x_1$	
SOF	UNPACK_WORD_1	$x_1 \wedge x_5$	$y_2 \wedge y_3 \wedge y_4 \wedge y_5$
UNPACK_WORD_1	IDLE	$\neg x_1$	
UNPACK_WORD_1	IDLE	$x_1 \wedge x_6$	$y_3 \wedge y_7$
UNPACK_WORD_1	UNPACK_WORD_2	$x_1 \wedge \neg x_6$	$y_3 \wedge y_5 \wedge y_8$
UNPACK_WORD_2	IDLE	$\neg x_1$	
UNPACK_WORD_2	IDLE	$x_1 \wedge x_6$	$y_3 \wedge y_7$
UNPACK_WORD_2	UNPACK_WORD_3	$x_1 \wedge \neg x_6$	$y_3 \wedge y_5 \wedge y_9$
UNPACK_WORD_3	IDLE	$\neg x_1$	
UNPACK_WORD_3	IDLE	$x_1 \wedge x_6$	$y_3 \wedge y_7$
UNPACK_WORD_3	NEXT_WRITE	$x_1 \wedge \neg x_6$	$y_3 \wedge y_5 \wedge y_{10}$
NEXT_WRITE	IDLE	$\neg x_1$	
NEXT_WRITE	IDLE	$x_1 \wedge x_6$	$y_3 \wedge y_7$
NEXT_WRITE	UNPACK_WORD_1	$x_1 \wedge \neg x_6$	$y_3 \wedge y_4 \wedge y_5 \wedge y_6$

Tabulka 5.1: Tabulka přechodů konečného automatu řídícího přenos dat

Vstupní podmínka	Odpovídající výraz
x_1	`accelerator_is_good = '1'`
x_2	`part_dst_rdy /= 0`
x_3	`RFIFO2IP_Full = '0'`
x_4	`RXFIFO_EMPTY = '0'`
x_5	`RXD_RX_ACK = '1'`
x_6	`all_bytes_processed = '1'`

Tabulka 5.2: Výrazy odpovídající vstupním podmínkám v tabulce 5.1

Výstup	Odpovídající výraz
y_1	`local_reset <= '1'`
y_2	`accelerator_sof <= '1'`
y_3	`accelerator_src_rdy <= '1'`
y_4	`accelerator_input_sel <= "1000"`
y_5	`decrease_bytes_counter <= '1'`
y_6	`fifo_rdreq_cmb <= '1'`
y_7	`accelerator_eof <= '1'`
y_8	`accelerator_input_sel <= "0100"`
y_9	`accelerator_input_sel <= "0010"`
y_{10}	`accelerator_input_sel <= "0001"`

Tabulka 5.3: Výrazy odpovídající výstupům v tabulce 5.1

Tomu odpovídají stavy[1] `UNPACK_WORD_1`, `UNPACK_WORD_2` a `UNPACK_WORD_3`. Ve stavu `NEXT_WRITE` bylo úspěšně vyčteno a přeneseno jedno slovo z vyrovnávací paměti. Po potvrzení úspěšného přečtení je možné toto slovo také rozbalit a poslat do akcelerátorů. Jakmile jsou přečteny všechny bajty paketu, je možné vstoupit do stavu `IDLE` a cyklus automatu se může opakovat.

Ukládání výsledků akcelerátorů

Akcelerátory mohou oznámit svůj stav pomocí rozhraní navrženého v podkapitole 4.3. Výsledky jsou uloženy do paměti typu FIFO (jednotka `rdpfifo_v4_02_a`). Řízení této činnosti bylo implementováno pomocí konečného automatu. Tabulka přechodů 5.4 popisuje činnost tohoto automatu. Vstupní podmínky jsou uvedeny ve vlastní tabulce 5.5. Syntax a sémantika výrazů čtvrtém sloupci tabulky 5.4 a druhém sloupci tabulky 5.5 odpovídá syntaxi a sémantice výrazů ve VHDL.

Aktuální stav	Následující stav	Vstupní podmínka	Výstup
IDLE	IDLE	$\neg x_1 \vee x_4$	
IDLE	COMPLETITION	$x_1 \wedge \neg x_4 \wedge \neg x_2$	
IDLE	WR_REQ	$x_1 \wedge \neg x_4 \wedge x_2$	`fifo_wrreq_cmb <= '1'`
WR_REQ	COMPLETITION	x_3	`accelerator_ack <= '1'`
WR_REQ	WR_REQ	$\neg x_3$	
COMPLETITION	IDLE	x_4	
COMPLETITION	COMPLETITION	$\neg x_4$	`accelerator_ack <= '1'`

Tabulka 5.4: Tabulka přechodů konečného automatu řídícího ukládání výsledků

Vstupní podmínka	Odpovídající výraz
x_1	`RFIFO2IP_Full = '0'`
x_2	`rp_status(i*3 + 1) and result_valid(i)) = '1'`
x_3	`RFIFO2IP_WrAck = '1'`
x_4	`result_valid = 0`

Tabulka 5.5: Výrazy odpovídající vstupním podmínkám v tabulce 5.4

Stav `IDLE` značí nečinnost a jedná se o počáteční stav. Pro pokračování do stavu `WR_REQ` musí být v paměti místo a zároveň musí být alespoň jeden výsledek platný. Zároveň musí být alespoň jeden platný výsledek značit úspěšnou operaci. Ve stavu `WR_REQ` musí být potvrzeno zapsání výsledku operace do paměti FIFO. Po potvrzení zápisu následuje přechod do `COMPLETITION`. Akcelerátorům je zaslán signál `ACK`. Akcelerátor by měl následovně deaktivovat signál `VLD` – signál `result_valid` by tudíž měl být v log. nule. Pak je možné opět přejít do stavu `IDLE` a cyklus se může opakovat.

Je třeba poznamenat, že podmínka x_2 se vyhodnocuje i vícekrát – to odpovídá použití více než jednoho akcelerátoru (parametr i).

[1] Z tabulky přechodů vyplývá, že automat by bylo možné optimalizovat a některé stavy sloučit.

Nastavení počtu akcelerátorů

Počet rekonfigurovatelných akcelerátorů musí být možno jednoduše ovlivnit. V rámci popisu architektury jednotky wrapperu (`user_logic.vhd`) se instance pro jednotlivé akcelerátory generují automaticky. Počet závisí na hodnotě generického parametru `C_MY_RP_COUNT`.

Implementace ovladačů

PLB je tzv. system memory-mapped sběrnice, která umožňuje přístup k registrům jednotlivých zařízení přes stránky paměťového prostoru systému. K tomu je zapotřebí znát bázovou adresu zařízení k jehož registrům chceme přistupovat. Pro čtení a zápis různých registrů je třeba znát jejich offset vzhledem k bázové adrese zařízení. Přístup k paměti systému umožňuje pseudozařízení `/dev/ram`. Získání ukazatele na požadovanou stránku paměťového prostoru je možné pomocí funkce `mmap`.

Celý algoritmus pro připojení správné stránky do paměťového prostoru běžícího procesu lze najít v souboru `DPRControl.cc`. Tento algoritmus funguje pouze tehdy, pokud je paměťová struktura pevně daná.

Struktura paměti

S implementací ovladačů hardware souvisí struktura paměti obálky rekonfigurovatelných akcelerátorů. Tato obálka zpřístupňuje několik zařízení, které mají mnoho registrů. Modul řízení částečné dynamické využívá jen některé z nich. Tabulka 5.6 obsahuje výčet jednotlivých offsetů s informací o tom, co obsahují.

Offset	Obsah
0x00000000	Registry jednotky `xps_ll_fifo`
0x00000100	Stavové registry jednotky `rdpfifo_v4_02_a`
0x00000200	Datové registry (pro čtení dat) jednotky `rdpfifo_v4_02_a`
0x00000300	Řídící registry wrapperu (`user_logic.vhd`)

Tabulka 5.6: Struktura paměti wrapperu

Pro přístup ke správnému registru je třeba připočítat offset registru. Popis použitých registrů a jejich offsety lze najít v tabulce 5.7 na straně 35.

Uživatel k těmto registrům nemusí přistupovat – řídicí činnost je zcela obstarána modulem částečné dynamické rekonfigurace.

Použité datové struktury

Nejzajímavější vytvořenou datovou strukturou je `rPartition`, která se používá v modulu řízení DPR. Deklaraci této struktury lze najít v souboru `DPRControl.hh`. Jejím účelem je seskupovat logicky související informace o rekonfigurovatelném oddílu (resp. o akcelerátoru, který daný oddíl obsahuje). Odráží aktuální stav rekonfigurovatelného oddílu.

Číslování oddílů odpovídá značení dle formátu popsaném v podkapitole 4.2. Všechny další související datové struktury používají stejné číslování. Jako příklad související struktury lze uvést statistiky výsledků operací akcelerátorů, které jsou uchovávány pro každý oddíl.

Příkazy rozhraní pro rekonfiguraci RP jsou vytvořeny dynamicky a uchovávány v této datové struktuře. Řídicí element `ControlSocket` tudíž umožňuje provést nahrání pouze těch bitstreamů, které byly specifikovány.

Název registru	Offset	Popis
Jednotka `xps_ll_fifo` (viz. 3.2)		
LLR	0x00000028	Jedná se o řídicí registr, který umožňuje resetovat jednotku.
Jednotka `rdpfifo_v4_02_a` [9]		
RPDR	0x00000000	Pomocí tohoto datového registru se vyčítají uložené hodnoty z paměti.
RPSR	0x00000004	Tento stavový registr jednotky slouží k zjišťování zaplnění zařízení.
Wrapper (viz. 4.3)		
AHC	0x00000000	Jedná se o řídicí registr, který umožňuje přepínat řízení zpracování mezi akcelerátory a procesorem.
UR	0x00000004	Tento řídicí registr umožňuje určovat rekonfigurované jednotky.

Tabulka 5.7: Offsety registrů

V struktuře je uložena informace o tom, jaká konfigurace je v oddílu zrovna nahrána. To může být užitečné, pokud by uživatel ve své aplikaci chtěl provádět operace, které závisí na aktuální hardwarové konfiguraci rekonfigurovatelných oddílů.

Modul také udržuje v této datové struktuře aktuální informaci o tom, zdali se daný oddíl rekonfiguruje.

5.2 Návrh demonstrační aplikace

Popis aplikace

Demonstrační aplikace ukazuje akceleraci úlohy pattern-matching nad příchozím síťovým provozem s využitím DPR. Pattern-matching je prováděn nad každým paketem a to včetně jeho hlavičky (header) i dat (payload). Dle potřeby je možné pattern-matching buď provádět na procesoru MicroBlaze s využitím knihovny PCRE [33], nebo pomocí akceleračních jednotek. Tyto akcelerační jednotky byly vytvořeny pomocí jedné z implementací algoritmu CLARK_NFA [34]. Pro demonstraci jsou využity tři rekonfigurovatelné akcelerátory s různou implementací.

Předzpracování paketů (např. zahození Ethernetové hlavičky) je taktéž prováděno v software i v hardware. V software je předzpracování provedeno pomocí dostupných elementů pro Click. Předzpracování v hardware probíhá přímo pomocí jednotky provádějící maskování signálů.

Řízení částečné dynamické rekonfigurace je možné provádět pomocí upraveného elementu `ControlSocket`, který reaguje na příkazy uživatele. Při rekonfiguraci akcelerátorů jsou zahazovány chybné výsledky z akcelerátorů.

Při běhu aplikace se sbírají jednoduché statistiky. Tento sběr statistik se provádí pro pattern-matching jak v software, tak v hardware. Zaznamenává se celkový počet přijatých paketů a celkový počet paketů, pro které neuspěl žádný vzor. Pro jednotlivé hardwarové jednotky (nebo jednotlivé vzory v software) se zaznamenává počet úspěchů vyhledání daného vzoru. Tyto statistiky je možné nechat vypsat zadáním příkazu uživatele. Případně lze aplikovat jednoduchý filtr, který vypíše statistiky pouze pro jednotky logicky odpovídající intervalu čísel.

Jádrem demonstrační aplikace je element `ClickFifo`, který je schopen provádět řízení hardware. Má konfigurační volby, přičemž některé je možno měnit za běhu. Tyto konfigurační možnosti jsou komentovány na začátku hlavičkového souboru[2] `ClickFifo.hh`.

Jednou z těchto konfiguračních možností je výběr, zdali statistiku pro danou jednotku při rekonfiguraci automaticky vynulovat, nebo ponechat. V software je také možné měnit vzory pro vyhledávání při běhu aplikace (*on-the-fly*).

Další zajímavou možností je vynutit hardwarovou konfiguraci. Uživatel nemusí řešit, který úplný bitstream nahraje do FPGA. Softwarový modul řízení DPR nahraje částečné bitstreamy do FPGA tak, aby počáteční konfigurace odpovídala té, kterou uživatel očekává.

Výsledky vyhledání vzorů jsou označeny časovými značkami (*timestamps*). Tyto časové značky odpovídají počtu sekund od startu aplikace. Nejsou součástí statistik a nejsou tudíž ukládány; zprávy jsou pouze vypisovány na standardní výstup.

Obrázek 5.1 na straně 37 ukazuje demonstrační uživatelský skript pro Click. V tomto případě elementy klasifikují provoz a odstraňují nepotřebné hlavičky paketů. Na dané hardwarové architektuře není možné přistupovat k nezarovnaným (*unaligned*) datům. Zdánlivě nadbytečné, vícekrát uvedené elementy (např. `Discard`) byly přidány úpravou skriptu pomocí nástroje `click-align`.

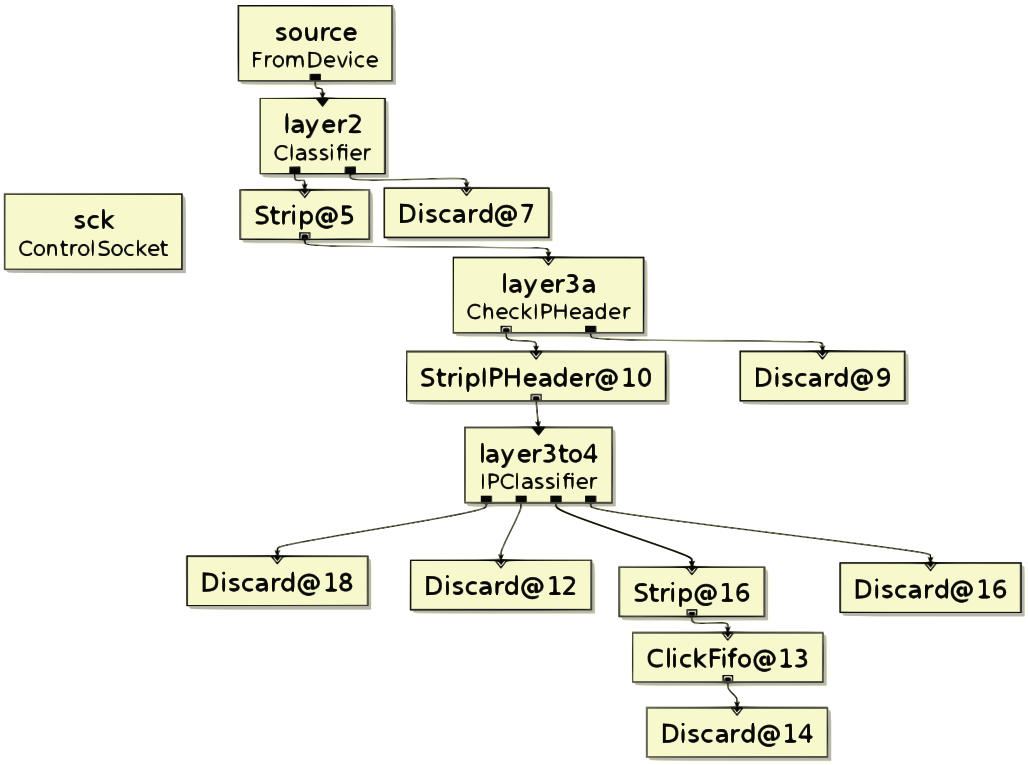
Při testování/ukázce demonstrační aplikace je vhodné použít nástroj Wireshark [27] a Tcpreplay [26]. Aby se komunikace se systémem nemísila s testovacími pakety, jsou použity dva různé Ethernetové porty. Port pro komunikaci využívá jednotku `xps_ethernetlite`, zatímco port pro přenášení paketů do FIFO paměti používá jednotku `xps_ll_temac` s rozhraním LocalLink.

Použití frameworku pro vytvoření aplikace

Níže je uvedeno, jakým způsobem byla výše popsaná aplikace vytvořena pomocí frameworku z pohledu uživatele. Uvažujeme, že uživatel má hardwarový systém vytvořený pomocí EDK pro FPGA Virtex-5, ve kterém chce použít technologii DPR. Uživatel musí provést následující kroky:

- Přidat do projektu
  - Jednotku `xps_ll_temac` (zdroj dat)
  - Upravenou[3] jednotku `xps_ll_fifo` (vyrovnávací paměť FIFO)
  - Jednotku `xps_hwicap` (rozhraní ICAP)
- Nastavit počet rekonfigurovatelných akcelerátorů
- Připojit výše uvedené jednotky ke sběrnicím a portům
- Připravit jednotky akcelerátorů

[2] Jde o styl dokumentace specifický pro Click.
[3] Pro snadné použití je wrapper rekonfigurovatelných akcelerátorů součástí této jednotky.



Obrázek 5.1: Demonstrační uživatelský skript pro Click

– Provést nízkoúrovňovou syntézu zdrojových souborů
– Jeden[4] z vzniklých netlistů nakopírovat do adresáře `implementation` projektu

Doporučené jednotky pro komunikaci se systémem a pro zjednodušení vývoje aplikací byly uvedeny a popsány v podkapitole 3.2.

Využití částečné dynamické rekonfigurace nijak jinak neovlivňuje přípravu hardwarového systému pro operační systém typu GNU/Linux. Uživatel exportuje popis hardwarové platformy, jenž je potřebný pro vytvoření binárního obrazu systému pomocí nástroje MBSL nebo Crosstool. Na přiloženém DVD je k dispozici doporučená konfigurace balíčků a jádra operačního systému.

V nástroji PlanAhead je třeba importovat implementační výsledky z EDK. Uživatel vytvoří rekonfigurovatelné oddíly s různými moduly a určí jejich umístění v FPGA. Dokončený implementační proces je nutné povýšit (*promote*). Pak je možné spustit implementaci jednotlivých modulů a vygenerování částečných bitstreamů. Výsledky implementace pro statickou logiku budou importovány.

Uživatel následně zkompiluje Click Modular Router s modulem pro řízení částečné dynamické rekonfigurace. Pro mnoho aplikací je možné přímo použít spustitelný binární soubor, který byl předpřipraven. Uživatel může implementovat další funkce v software a rozdělit je do jednotlivých modulů. Komunikaci (přenos paketů) mezi těmito moduly definuje uživatelský skript.

Na dané hardwarové architektuře není možné přistupovat k nezarovnaným (*unaligned*) datům. Vytvoří-li uživatel nový skript, tak jej musí zpracovat pomocí nástroje `click-align`,

[4]Účelem je pouze to, aby nástroje našly nějakou implementaci rek. akcelerátorů. Jednotlivé moduly se definují pomocí nástroje PlanAhead.

který automaticky doplní do uživatelského skriptu správné anotace, aby bylo možné skript spouštět.

Demonstrační aplikace umožňuje provádět pattern-matching na univerzálním procesoru pomocí knihovny PCRE nebo v rekonfigurovatelných akcelerátorech. Částečnou dynamickou rekonfiguraci je možné řídit pomocí definovaného rozhraní. Stav akcelerátorů (v tomto případě výsledek vyhledání vzoru) je ukládán do zvláštní paměti FIFO (nesouvisí s `xps_ll_fifo`). Wrapper rekonfigurovatelných akcelerátorů eliminuje nežádoucí stavy a zahazuje neplatné výsledky.

5.3 Ukázka demonstrační aplikace

Funkcionalita odpovídá popisu v podkapitole 5.2. Chování elementu je možné parametrizovat a dynamicky měnit za běhu. Výsledky vyhledání vzorů jsou ve výchozím nastavení vypsány na standardní výstup hned jakmile jsou dostupné. Způsob řízení částečné dynamické rekonfigurace byl navržen a popsán v podkapitole 4.4. Syntax a sémantika příkazů je přesně dodržena.

Propojení použitých elementů odpovídá diagramu 5.1 na straně 37. Konfigurace elementu je uvedena v uživatelském skriptu, který lze najít na přiloženém DVD v adresáři `/Sources/Demo/click-script/`.

Formát výstupu

Pro výpis výsledků operací se používá pevně stanovený formát. Výpis výsledků se skládá ze tří hodnot uzavřených v hranatých závorkách, přičemž skupiny závorek jsou odděleny mezerou (ASCII kód `0x20`). První skupina závorek obsahuje aktuální časovou značku[5]. Druhá skupina závorek určuje druh zprávy, třetí skupina závorek nese obsah zprávy. Výčet a popis jednotlivých možných druhů zpráv je uveden v příloze v tabulce B.1. Některé zprávy jsou dostupné pouze při zpracování v hardware, jiné pouze při zpracování v software. Ostatní jsou společné. Pokud je součástí zprávy číslo, tak je vždy 32bitové (a bez znaménka).

Ukázka výstupu

Veškeré ukázky výstupu jsou uvedeny v příloze B.2. Je uveden výstup při průběžném vyhledávání vzorů. Může uspět jeden vzor, více vzorů najednou, nebo žádný. Dále je ukázán příklad výstupu řízení rekonfigurace. Uživatel se k řídicímu elementu připojí pomocí protokolu `telnet` a zadává jednotlivé příkazy. Na standardním výstupu není provedení rekonfigurace nijak oznámeno. Uživatel připojený k elementu `ControlSocket` však výsledek provedení příkazu vidí (kód 200 odpovídá úspěšnému provedení operace). Zadáním příkazu `stats` přes rozhraní elementu `ControlSocket` se provede vypsání statistik.

[5] http://read.cs.ucla.edu/click/doxygen/class_timestamp.html

Kapitola 6

Diskuze omezení a závěr

V rámci této práce byl navržen framework, který usnadňuje tvorbu aplikací s DPR. Na závěr je třeba zhodnotit výsledek a zamyslet se nad možnými omezeními frameworku. Podkapitola 6.1 obsahuje diskuzi těchto omezení. V podkapitole 6.2 jsou uvedeny některá možná vylepšení. Výsledky práce jsou shrnuty v podkapitole 6.3.

6.1 Omezení frameworku

Obálka rekonfigurovatelných akcelerátorů (wrapper)

Ve výchozí konfiguraci je bez jakýchkoliv úprav možné použít až 32 rekonfigurovatelných jednotek. Tento limit je teoreticky možné jednoduše navýšit několika drobnými úpravami kódu.

Je třeba poznamenat, že vytvoření obvodu v FPGA, který využívá více jak 32 rekonfigurovatelných oddílů, by bylo pravděpodobně značně náročné. Při takto vysokém počtu RP mohou například vznikat problémy se směrováním signálů a splněním časování jednotlivých cest. Vzhledem k tomu lze uvažovat, že tento limit prozatím není omezující.

Řízení konfiguračního rozhraní ICAP

Řízení rozhraní je implementováno jako softwarový ovladač. Implementace v hardware by byla výkonnější, ale méně flexibilní. Úpravu softwarového modulu řízení částečné dynamické rekonfigurace by měl být schopný zvládnout i programátor, který se nespecializuje na vývoj hardware. Tento bod tudíž nelze vnímat zcela jako omezení – souvisí značně s filozofií návrhu frameworku.

Floorplanning

Framework uživateli nijak nezjednodušuje umístění rekonfigurovatelných oddílů. Vzhledem k požadavkům kladeným touto technologií na návrh obvodu (viz. podkapitola 2.4) je řešení takové úlohy velice náročné. V současné době existuje v nástrojích firmy Xilinx podpora pro úpravy a změnu již existujících rekonfigurovatelných oddílů pomocí TCL/Tk skriptů.

Kompatibilita s jinými platformami než Virtex-5

Framework obsahuje následující platformě závislé součásti:

- Sběrnici `PLB`

- Instanciaci BlockRAM (komponenta RAMB16_S36_S36)
- Jednotku `xps_hwicap`

Implementace ovladačů spoléhá na to, že registry zařízení jsou přístupné přes systémovou sběrnici. Sběrnici `PLB` využívá mnoho jednotek, včetně vyrovnávací paměti `xps_ll_fifo`. Na novějších rodinách FPGA zařízení (např. Virtex-7) je tato sběrnice nahrazena sběrnicí AXI.

Implementace vyrovnávacích pamětí FIFO používají komponentu RAMB16_S36_S36. Instanciace této BlockRAM paměti se může lišit u novějších rodin FPGA zařízení.

Modul řízení částečné dynamické rekonfigurace je možné použít bez omezení. Je ale třeba uvažovat, že byl vyvinut pro jednotku `xps_hwicap` a nemusí fungovat pro jiné druhy zařízení ICAP.

Framework tedy využívá minimum součástí závislých na platformě. Je pravděpodobné, že použití frameworku na platformách Virtex-4 a Virtex-6 by vyžadovalo jen drobné, nebo žádné úpravy. Vzhledem k tomu, že framework byl navržen především pro platformu Virtex-5, tak lze uvažovat, že toto omezení není závažné.

6.2 Možná vylepšení

Níže jsou uvedena možná vylepšení, která by měla pozitivní dopad na framework jako takový.

Software

- použití vláken (*threads*) v modulu DPR (`DPRControl.cc`)
- reimplementace poruchového ovladače jednotky `xps_ll_temac`, který je součástí jádra operačního systému

Hardware

- oznámení zaplnění jednotky `rdpfifo_v4_02_a` pomocí přerušení
- stanovení omezení (constraints) za účelem zvýšení výkonnosti
- zvýšení kmitočtu některých synchronizačních hodin

Další vylepšení

Použití částečné dynamické rekonfigurace logicky souvisí s aplikací, která má být vytvořena. Díky tomu by bylo možné uvést mnoho dalších návrhů pro vylepšení.

Příkladem takového vylepšení může být efektivní plánování částečné dynamické rekonfigurace. Jedná se obsáhlé téma, které značně přesahuje rozsah této práce. Plánování částečné dynamické rekonfigurace není v rámci frameworku (resp. řídícího modulu) podporováno, ale vzhledem k návrhu modulu si jej uživatel může v nejzákladnější podobě jednoduše dotvořit.

6.3 Shrnutí

Navržený a implementovaný framework usnadňuje tvorbu aplikací s částečnou dynamickou rekonfigurací. Použitý koncept hardwarové struktury je vhodný pro přenos velkého objemu dat do rekonfigurovatelných akcelerátorů. Operační systém umožňuje použít softwarovou nadstavbu – Click Modular Router. Použití tohoto softwaru poskytuje frameworku značné výhody pro síťové aplikace, ale není omezeno pouze na něj. Modul částečné dynamické rekonfigurace je přístupný přes definované rozhraní a skrývá implementaci řízení hardwarové obálky rekonfigurovatelných akcelerátorů.

Pomocí frameworku byla vytvořena demonstrační aplikace. Tato aplikace ukazuje pattern-matching v hardware s částečnou dynamickou rekonfigurací. Vzory je možné měnit nahráním připravených bitstreamů. Pattern-matching je taktéž možné provádět na univerzálním procesoru.

Framework lze jednoduše přizpůsobit a využít jej pro různé účely.

Literatura

[1] *Reconfigurable Hardware* [online]. The Phoenix Project, 2001 [cit. 2014-01-16]. Dostupné z: http://www.cs.cmu.edu/~phoenix/reconfigurable.html.

[2] *LocalLink Interface Specification* [online]. SP006 (2.0). Xilinx, 2005-07-25 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/aurora/aurora_member/sp006.pdf.

[3] *Configuration and Readback of Virtex FPGAs Using JTAG Boundary-Scan* [online]. XAPP139 (1.7). Xilinx, 2007-02-14 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/application_notes/xapp139.pdf.

[4] *Constraints Guide* [online]. Xilinx, 2008-01-21 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/itp/xilinx10/books/docs/cgd/cgd.pdf.

[5] *XPS UART Lite* [online]. DS571 (1.01.a). Xilinx, 2009-12-02 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/xps_uartlite.pdf.

[6] *LogiCORE IP XPS Interrupt Controller* [online]. DS572 (2.01.a). Xilinx, 2010-04-19 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/xps_intc.pdf.

[7] *LogiCORE IP XPS Timer/Counter* [online]. DS573 (1.02a). Xilinx, 2010-04-19 [cit. 2014-02-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/xps_timer.pdf.

[8] *Partial Reconfiguration User Guide* [online]. UG702 (12.1). Xilinx, 2010-05-03 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/ug702.pdf.

[9] *LogiCORE Read Packet FIFO* [online]. DS514 (4.02a). Xilinx, 2010-05-27 [cit. 2014-02-16]. Dostupné z: http://www.cs.indiana.edu/hmg/le/project-home/xilinx/ise_13.2/ISE_DS/EDK/hw/XilinxProcessorIPLib/pcores/rdpfifo_v4_02_a/doc/rdpfifo.pdf.

[10] *MicroBlaze Debug Module* [online]. DS641 (2.00.a). Xilinx, 2010-07-23 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/mdm.pdf.

[11] *LogiCORE IP XPS Ethernet Lite Media Access Controller* [online]. DS580 (4.00.a). Xilinx, 2010-09-21 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/xps_ethernetlite.pdf.

[12] *LogiCORE IP Processor Local Bus (PLB) v4.6* [online]. DS531 (1.05a). Xilinx, 2010-09-21 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/plb_v46.pdf.

[13] *LogiCORE IP XPS LL TEMAC* [online]. DS537 (2.03a). Xilinx, 2010-12-14 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/xps_ll_temac.pdf.

[14] *LogiCORE IP Clock Generator* [online]. DS614 (4.01.a). Xilinx, 2010-12-14 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/clock_generator.pdf.

[15] *Snort* [online]. Sourcefire, 2010 [cit. 2014-01-16]. Dostupné z: http://www.snort.org/.

[16] *FPGA Run-Time Reconfiguration: Two Approaches* [online]. Altera, 2011-03-01 [cit. 2013-01-25]. Dostupné z: http://www.altera.co.jp/literature/wp/wp-01055-fpga-run-time-reconfiguration.pdf.

[17] *LogiCORE IP XPS LL FIFO* [online]. DS568 (1.02a). Xilinx, 2011-03-01 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/xps_ll_fifo.pdf.

[18] *LogiCORE IP XPS HWICAP* [online]. DS586. Xilinx, 2011-06-22 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/xps_hwicap/v5_01_a/xps_hwicap.pdf.

[19] IEEE Standard for Ethernet - Section 1. *IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008)*, 2012: s. 1–0. Dostupné z: http://standards.ieee.org/getieee802/download/802.3-2012_section1.pdf

[20] *LogiCORE IP MicroBlaze Micro Controller System* [online]. DS865 (1.1). Xilinx, 2012-04-24 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ds865_microblaze_mcs.pdf.

[21] *PlanAhead User Guide* [online]. UG632 (14.1). Xilinx, 2012-04-24 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/PlanAhead_UserGuide.pdf.

[22] *Virtex-5 FPGA User Guide* [online]. UG190 (5.4). Xilinx, 2012-05-16 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/user_guides/ug190.pdf.

[23] *LogiCORE IP Multi-Port Memory Controller* [online]. DS643 (6.06.a). Xilinx, 2013-02-22 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/ip_documentation/mpmc/v6_06_a/mpmc.pdf.

[24] *Axcelerator FPGAs* [online]. Microsemi, 2013 [cit. 2014-01-16]. Dostupné z: http://www.microsemi.com/products/fpga-soc/antifuse-fpgas.

[25] *What is a FPGA?* [online]. Xilinx, 2014 [cit. 2014-01-16]. Dostupné z: http://www.origin.xilinx.com/fpga/.

[26] Tcpreplay. 2014 [cit. 2014-02-16]. Dostupné z: http://tcpreplay.synfin.net/.

[27] Wireshark. 2014 [cit. 2014-02-16]. Dostupné z: http://www.wireshark.org/.

[28] Bauer, L.; Braun, C.; Imhof, M. E.; aj.: OTERA. In *2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2012. Dostupné z: http://ces.itec.kit.edu/ bauer/pdf/OTERA_AHS12.pdf

[29] Compton, K.; Hauck, S.: Reconfigurable computing: a survey of systems and software. *ACM Comput. Surv.*, ročník 34, č. 2, Červen 2002: s. 171–210, ISSN 0360-0300. Dostupné z: http://doi.acm.org/10.1145/508352.508353

[30] (eds)., M. H. .; Farisi, B. A.; Heirman, W.; aj.: *Proceedings of the 5th International Workshop on Reconfigurable Communication-centric Systems on Chip 2010 - ReCoSoC'10 May 17-19, 2010, Karlsruhe, Germany*. Karlsruhe: KIT Scientific Publishing, print on demand vydání, 2010.

[31] Eto, E.: *Difference-Based Partial Reconfiguration* [online]. XAPP290 (2.0). Xilinx, 2007-12-03 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/application_notes/xapp290.pdf.

[32] Hauck, S.: The roles of FPGAs in reprogrammable systems. In *Proceedings of the IEEE*, ročník vol. 86, 1998, s. 615–638. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=663540

[33] Hazel, P.: *PCRE - Perl Compatible Regular Expressions* [online]. 2014 [cit. 2014-01-16]. Dostupné z: http://www.pcre.org/.

[34] Kořenek, J.; Košař, V.: NFA split architecture for fast regular expression matching. In *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems - ANCS '10*, ACM Press, 2010, s. 1–. Dostupné z: http://portal.acm.org/citation.cfm?doid=1872007.1872024

[35] Kohler, E.; Morris, R.; Chen, B.; aj.: The click modular router. In *ACM Transactions on Computer Systems*, ročník vol. 18, 2000, s. 263–297. Dostupné z: http://pdos.csail.mit.edu/papers/click:tocs00/paper.pdf

[36] Lie, W.; Feng-yan, W.: Dynamic Partial Reconfiguration in FPGAs. In *2009 Third International Symposium on Intelligent Information Technology Application*, 2009. Dostupné z: http://www.eng.ucy.ac.cy/theocharides/Courses/ECE664/05369525.pdf

[37] Liu, S.; Pittman, R. N.; Forin, A.: Energy reduction with run-time partial reconfiguration (abstract only). In *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays - FPGA '10*, ACM Press, 2010, s. 292–. Dostupné z: http://portal.acm.org/citation.cfm?doid=1723112.1723189

[38] Peattie, M.: *Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode* [online]. XAPP502 (1.6.1). Xilinx, 2009-08-24 [cit. 2014-01-16]. Dostupné z: http://www.xilinx.com/support/documentation/application_notes/xapp502.pdf.

[39] Sundararajan, P.: High Performance Computing Using FPGAs. Technická zpráva, Xilinx, Září 2010. Dostupné z: http://www.xilinx.com/support/documentation/white_papers/wp375_HPC_Using_FPGAs.pdf

[40] Tobola, J.: Platforma pro rychlý vývoj síťových zařízení. 2007. Dostupné z: http://www.fit.vutbr.cz/study/DP/DP.php?id=5944&y=2006

[41] Trimberger, S.: A Reprogrammable Gate Array and Applications. In *Proceedings of the IEEE*, ročník 81, 1993, s. 1030–1041. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00231341

[42] Vaidyanathan, R.: *Dynamic reconfiguration*. Kluwer Academic, c2003.

[43] Wirthlin, M.; Hutchings, B.: Improving functional density using run-time circuit reconfiguration [FPGAs]. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, ročník vol. 6, 1998, s. 247–256. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=678880

Příloha A

Obsah DVD

/Binaries

Adresář obsahuje knihovny, binární soubory a nástroje, které byly použity při implementaci demonstrační aplikace.

/Refs

Obsahuje většinu dokumentů uvedených v seznamu literatury.

/Results

Zde lze najít mezivýsledky (projekty EDK a PlanAhead) vytváření demonstrační aplikace.

/Sources

Adresář obsahuje zdrojové soubory vytvořené autorem této práce. Jsou odděleny zdrojové soubory demonstrační aplikace a frameworku samotného.

/Testrun

Adresář s vesměs binárnimi soubory, které lze použít pro otestování demonstrační aplikace na použitém přípravku. Spuštění souboru `!run.bat` na operačním systému Windows 7 automaticky obstará nahrání úplného bitstreamu a obrazu systému do vývojového přípravku.

Podadresář `App` obsahuje binární soubor aplikace Click pro ozkoušení, částečné bitstreamy a uživatelské skripty pro Click. Krátký popis ke všem souborům lze najít v `README`.

Podadresář `test-pcaps` obsahuje soubory s použitými testovacími pakety (ve formě `pcap`). Ty jsou užitečné pro testování demonstrační aplikace pomocí `tcpreplay`.

/Text

V tomto adresáři jsou obrázky, zdrojové soubory a text této práce.

Příloha B

Ukázka demonstrační aplikace

B.1 Formát výstupu

Druh zprávy	Popis zprávy
Společné	
`SUCC`	Nastala shoda alespoň jednoho vzoru s paketem. Při zpracování v hardware obsahuje číselnou hodnotu registru výsledků.
`FAIL`	Nenastala shoda vzoru s paketem. Při zpracování v hardware obsahuje číselnou hodnotu registru výsledků.
`FATAL`	Došlo k chybě, ze které se není možné zotavit.
`CMD`	Volání systémové aplikace vrátilo neočekávaný výsledek. To může indikovat problém s nějakou nezbytnou systémovou aplikací.
`STATS-BEGIN`	Začíná se vypisovat zaznamenaná statistika. Je uvedeno číslo, které určuje počet řádků, které budou vypsány.
`PATTERN:X`	Byl vypsán počet úspěchů pro daný vzor, kde **X** je číslo vzoru nebo RP.
`STATS`	Byl vypsán celkový počet zpracovaných paketů.
`STATS-END`	Byl vypsán celkový počet paketů v kterých se nepovedlo vyhledat žádný vzor.
SW zpracování	
`SUCC-AGAIN`	Uspělo vyhledání podskupiny. Obsahuje offset, který vyjadřuje pozici začátku podskupiny, která uspěla.
`INFO`	Byla vypsána informační, nebo varovná zprávy související se zpracováním v softwaru. Při ladění může obsahovat hodnotu `IME_MSG` z konfiguračního uživatelského skriptu.
`PACKET`	Byl vypsán obsah paketu. Element je možné konfigurovat tak, aby modul vypisoval přijatý paket (omezeno výchozí, nebo zadanou délkou zprávy).

Tabulka B.1: Formát výstupu demonstrační aplikace

B.2 Ukázka výstupu

Výpis průběžného vyhledávání vzorů s použitím akcelerátorů

```
[40.156363032] [SUCC] [Match. RFIFO read is: 1]
[62.867046800] [SUCC] [Match. RFIFO read is: 3]
[80.959225768] [FAIL] [No match. RFIFO read is: 0]
```

Připojení k řídicímu rozhraní a provedení rekonfigurace

```
Click::ControlSocket/1.3
write unit1_bit2 1
200 Write handler 'unit1_bit2' OK
reconfigurate
200-Reconfigurating by sending USR1 to self.
```

Výpis statistik

```
[94.583078360] [STATS-BEGIN] [5 more lines will be outputed.]
[94.583078360] [PATTERN:1] [2 match successes.]
[94.583078360] [PATTERN:2] [1 match successes.]
[94.583078360] [PATTERN:3] [0 match successes.]
[94.583078360] [STATS] [108 packets processed in total.]
[94.583078360] [STATS-END] [106 matches failed in total.]
```