VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZPRACOVÁNÍ OBRAZU V FPGA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE BRONISLAV PŘIBYL
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZPRACOVÁNÍ OBRAZU V FPGA

IMAGE PROCESSING IN FPGA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — BRONISLAV PŘIBYL
AUTHOR

VEDOUCÍ PRÁCE — Doc. Dr. Ing. PAVEL ZEMČÍK
SUPERVISOR

BRNO 2008

Abstrakt

Obrazy pro vědecké účely jsou snímány optickými zařízeními, jako např. mikroskopy, kamerami apod., které se obvykle skládají ze soustavy optických čoček a optického senzoru. Většina čoček trpí různými vadami. Geometrické zkreslení je jednou z vad, které mohou být překážkou pro přesné měření vzdáleností v obraze. Tato práce popisuje rychlý algoritmus pro separabilní převzorkování obrazu, jenž je schopen korigovat jemné geometrické vady způsobené čočkami. Je také popsána implementace tohoto algoritmu v FPGA.

Klíčová slova

Biomarker, FPGA, VHDL, zpracování obrazu, geometrické zkreslení, vzorkovací teorém, alias, konvoluční filtr, převzorkování

Abstract

Images for scientific purposes are taken by optical devices, such as microscopes, cameras etc., which usually consist of system of lenses and optical sensor. Most lenses suffer from various distortions. Geometrical distortion may not be acceptable for example for precise distance measurement in the image. This thesis describes fast algorithm for separable image resampling capable of correcting smooth geometrical distortions caused by lenses. An FPGA implementation of this algorithm is also described.

Keywords

Biomarker, FPGA, VHDL, image processing, geometrical distortion, sampling theorem, alias, convolution filter, resampling

Citace

Bronislav Přibyl: Zpracování obrazu v FPGA, bakalářská práce, Brno, FIT VUT v Brně, 2008

Zpracování obrazu v FPGA

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Dr. Ing. Pavla Zemčíka a uvedl jsem všechny použité prameny a literaturu.

. . . . . .
Bronislav Přibyl
14.5.2008

Poděkování

Děkuji vedoucímu bakalářské práce Doc. Dr. Ing. Pavlu Zemčíkovi za odborné vedení a Ing. Michalu Seemanovi za konzultace a implementaci podpůrného softwaru.

Obsah

Kapitola 1

Úvod

Proces snímání a zpracování obrazu je důležitou součástí mnoha vědních disciplín. Nejinak je tomu v biologii a medicíně, kde výzkumníci zpravidla nejdříve odeberou vzorky, které si v laboratoři podle potřeby upraví a nasnímají. Teprvé poté jsou na nich prováděna měření. Zde do hry vstupují informační technologie, které přispívají k vysoké automatizaci zpracování vzorků.

V projektu Biomarker, který se zabývá diagnostikou nádorových onemocnění pomocí automatizovaného zpracování biomedicínských obrazů, se pracuje mj. se snímky buněk pořízenými konfokálním fluorescenčním mikroskopem. Objektiv tohoto mikroskopu trpí velkou chromatickou aberací, takže geometrické zkreslení nasnímaného obrazu je pro každou vlnovou délku světla odlišné. Výsledkem je rozmazaný a znehodnocený obraz. Je proto nutné nasnímat jednotlivé barevné složky obrazu zvlášť, tyto dílčí snímky odděleně geometricky zkorigovat a poté je sloučit zpět do výsledku.

Geometrické zkreslení nasnímaného obrazu je možné odstranit pouze převzorkováním, což je obecně poměrně drahá operace. Při zpracovávání velkých objemů dat, se kterými se v biomedicíně běžně pracuje, přestává výpočetní výkon klasických procesorů stačit a je nutno uvažovat o rychlejším řešení. Z tohoto důvodu se implementace převzorkovacích algoritmů stále častěji přesouvá do aplikačně specifických obvodů nebo programovatelných hradlových polí (FPGA).

Tato práce se zabývá návrhem a realizací optimalizovaného algoritmu pro převzorkování obrazu, který je implementován právě v FPGA. Algoritmus je vhodný pouze pro korekci jemných geometrických zkreslení ve dvourozměrných datech, kdy je možno zcela opominout lokální změny měřítka a případné rotace, ale na druhou stranu je nutné provádět posuny pixelů se značnou přesností. Je také popsána implementace algoritmu na speciální desce osazené digitálním signálovým procesorem (DSP) a FPGA. Tato deska je prostřednictvím další PCI karty připojena do klasického osobního počítače (PC).

Dokument je členěn do několika částí. V kapitole 2 je čtenář seznámen se základními principy a přístupy k problematice převzorkování, v kapitole 3 je mu pak stručně představena oblast programovatelného hardwaru. Ve 4. kapitole je diskutován současný stav akcelerace převzorkování v FPGA. V kapitole 5 je rozebrán návrh a implementace jednotky pro převzorkování obrazu. Závěrečnou kapitolu tvoří zhodnocení dosažených výsledků a možné pokračování práce v budoucnu.

Kapitola 2

Převzorkování

Tato kapitola si neklade za cíl být vyčerpávajícím zdrojem informací o problematice převzorkování. Z důvodu rozsahu se omezuje pouze na témata relevantní vzhledem k zaměření práce.

2.1 Vady optických soustav

Optické soustavy jsou složeny z optických čoček (případně hranolů) a optického senzoru. Každý z těchto prvků trpí řadou vad, z nichž některé nám mohou být pro specifické účely na obtíž. Mezi základní vady optických prvků, jak se píše v [36], [35] nebo [11] patří následující:

- **Aberace** – barevná vada, je důsledkem závislosti indexu lomu skla čočky na vlnové délce procházejícího světla. Objevuje se ve všech objektivech pracujících s lomem na optických plochách, proto je výrazně eliminována v zrcadlových objektivech. Projevuje se posunem barev do modra na ostrých rozhraních snímku, zejména v jeho okrajových částech.

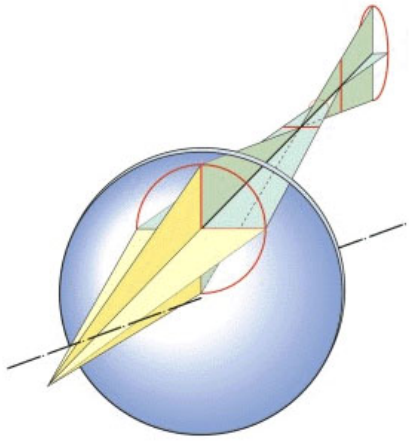
- **Absorpce světla** – všechny optické prvky, kterými světlo prochází, pohlcují určitou část světla. S rostoucím počtem těchto prvků roste také ztráta světla v objektivu. Tento jev je do značné míry ovlivnitelný kvalitou použitého skla, průměrem čoček a typem konstrukce objektivu.

- **Astigmatismus** (viz obr. 2.1) – projevuje se při promítání bodů ležících mimo optickou osu. Dva body ležící v jedné rovině kolmé k optické ose se zobrazí do dvou různých rovin, takže svazek paprsků má místo kruhového průřezu průřez elipsovitý. Koriguje se spojením dvou soustav se stejně velkým, ale opačným astigmatickým rozdílem.

- **Bokeh** – popisuje způsob rozmazání nezaostřených předmětů, nejvíce se projevuje na světlých předmětech. Jde o rozptylové kroužky jednotlivých předmětových bodů, které se zvětšují s rozostřením. Mohou mít tvar víceúhelníků, kruhů nebo prstenců.

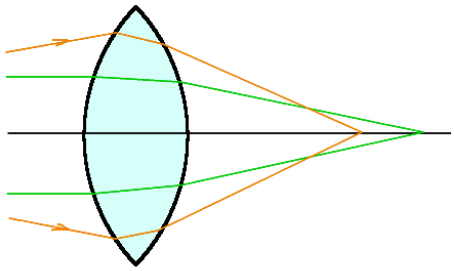
- **Koma** – asymetrická vada vznikající průchodem velmi šikmých paprsků blízko u okraje čočky. Tyto paprsky se lámou značně nepravidelně a jimi tvořený obraz je tak jinak velký než obraz tvořený paprsky procházejícími středem objektivu. Odstraňuje se vhodným zacloněním okrajů objektivu.

- **Sférická (otvorová) vada** (viz obr. 2.2) – vyskytuje se u čoček se sférickými nebo rovnými povrchy. Projevuje se při zobrazování bodů ležících blízko optické osy, kdy



Obrázek 2.1: Astigmatismus, převzato z [20].

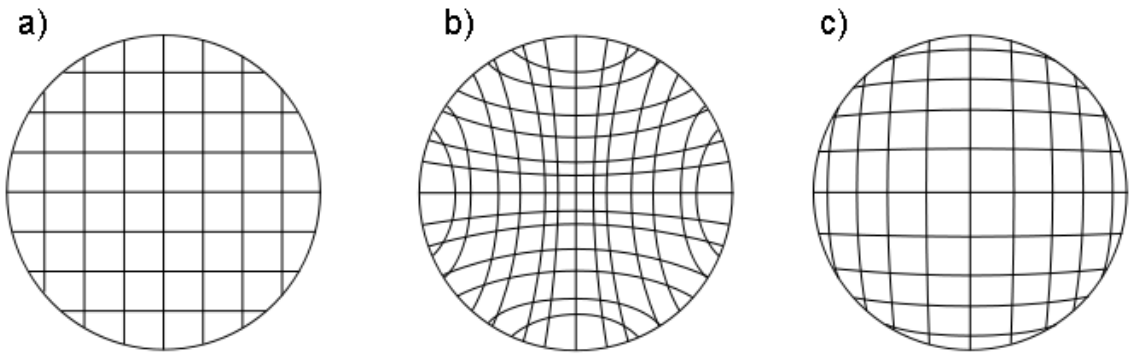
paraxiální paprsky (procházející blízko optické osy) zobrazí bod ve větší vzdálenosti než paprsky procházející dále od optické osy. Projevuje se na okrajích spojných čoček jako „rozzářená“ ostrá kresba. Tato vada se dá tlumit zacloněním objektivu, opačnou orientací ploskovypuklé čočky (vypuklou stranou ven) nebo spojením spojné čočky s čočkou rozptylnou, která má průběh sférické vady zcela opačný.



Obrázek 2.2: Sférická vada, převzato z [24].

- **Vinětace** – pokles osvětlení na krajích obrazu. Projevuje se u objektivů s velkým zorným úhlem a u čoček s velkým zakřivením. Někdy se koriguje přídavnými optickými členy.
- **Zkreslení** (zhroucení kresby) – patrné hlavně u okrajů snímku. Vzniká vlivem různě velkého zvětšení ve středu a na krajích obrazu. Rozlišujeme dva typy zkreslení – viz obr. 2.3. Jeho typ závisí na tom, zda je clona umístěna před spojnou nebo rozptylnou čočkou.

Jak se píše v [18], největší překážkou pro měření vzdáleností nebo tvarů v obraze jsou geometrické vady, zvláště pak zkreslení.



Obrázek 2.3: Zkreslení: a) původní rastr, b) poduškovité zkreslení, c) soudkovité zkreslení.

2.2 Základy převzorkování

V části 2.1 bylo zmíněno, že jediným známým způsobem korekce geometrických vad v digitálním obraze je jeho převzorkování. Jde o proces měnící vzorkovací frekvenci nebo rozměry digitálních obrazových nebo zvukových signálů pomocí časové nebo prostorové analýzy a vzorkování původních dat.

Základem převzorkování je tedy proces vzorkování. Omezíme-li se na oblast zpracování digitálního obrazu, můžeme proces vzorkování definovat jako v [7]: Vzorkování je proces transformace spojité funkce dvou proměnných na množinu hodnot vázaných k diskrétní dvourozměrné mřížce.

V průběhu tohoto procesu může přirozeně docházet ke ztrátě informace. Tuto skutečnost však můžeme do jisté míry ovlivnit tím, zda dodržíme tzv. vzorkovací teorém[1], který nám podle [25] udává, jakou nejmenší vzorkovací frekvenci je třeba použít, aby ke ztrátě informace nedošlo: Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byl vzorkován frekvencí alespoň dvakrát vyšší než je maximální frekvence rekonstruovaného signálu.

Jinými slovy, vzorkovací frekvence f_s musí být alespoň dvakrát větší, než největší frekvence f_{max} obsažená ve vzorkovaném signálu, jak popisuje následující rovnice:

$$f_s > 2f_{max} \tag{2.1}$$

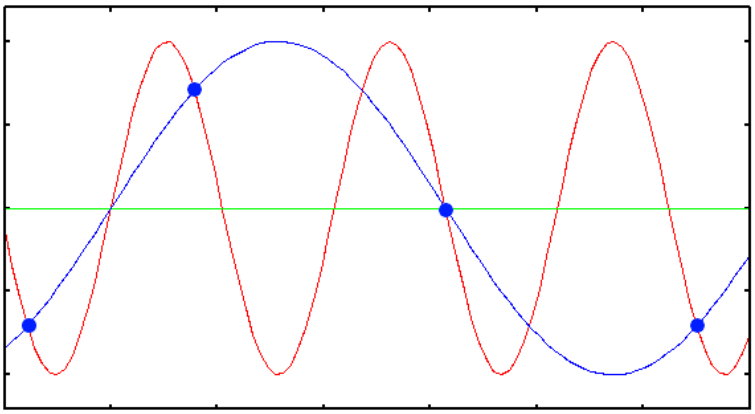
Pokud tato podmínka splněna není, dochází k jevu známému jako alias nebo aliasing. V [5] se říká, že aliasing je jev způsobující skutečnost, že různé spojité signály jsou po navzorkování nerozlišitelné (sada vzorků je stejná).

Tento případ ilustruje obrázek 2.4. V obrazových datech se aliasing projevuje zejména v oblastech s jemnou opakující se kresbou vznikem artefaktů nebo tvz. moaré (obr. 2.5).

Vzniku aliasingu se dá zabránit různými metodami, které můžeme souhrnně nazvat anti-aliasing. Podle [8] je anti-aliasing odstranění těch komponent signálu, které mají vyšší frekvenci, než jaká může být zachycena při procesu vzorkování.

Obecně tak můžeme říci, že anti-aliasing způsobí rozmazání obrazu. Existují různé metody pro potlačení aliasingu v obrazu a každá z nich obraz rozmazává odlišným způsobem.

[1] Nebo také Shannonův vzorkovací teorém, Nyquist-Shannonův vzorkovací teorém.



Obrázek 2.4: Vznik aliasu: Sada vzorků (modré kuličky) je při nízké vzorkovací frekvenci stejná pro červený i modrý signál. Převzato z [23].

Hlavní metrikou pro hodnocení dokonalosti anti-aliasingových metod je podobnost aproximovaného signálu s Fourierovou řadou. Avšak vzhledem k tomu, že naše znalosti o lidském vnímání obrazu jsou omezené, hodnotíme většinou anti-aliasingové metody subjektivně. Základní metody pro potlačení aliasingu jsou:

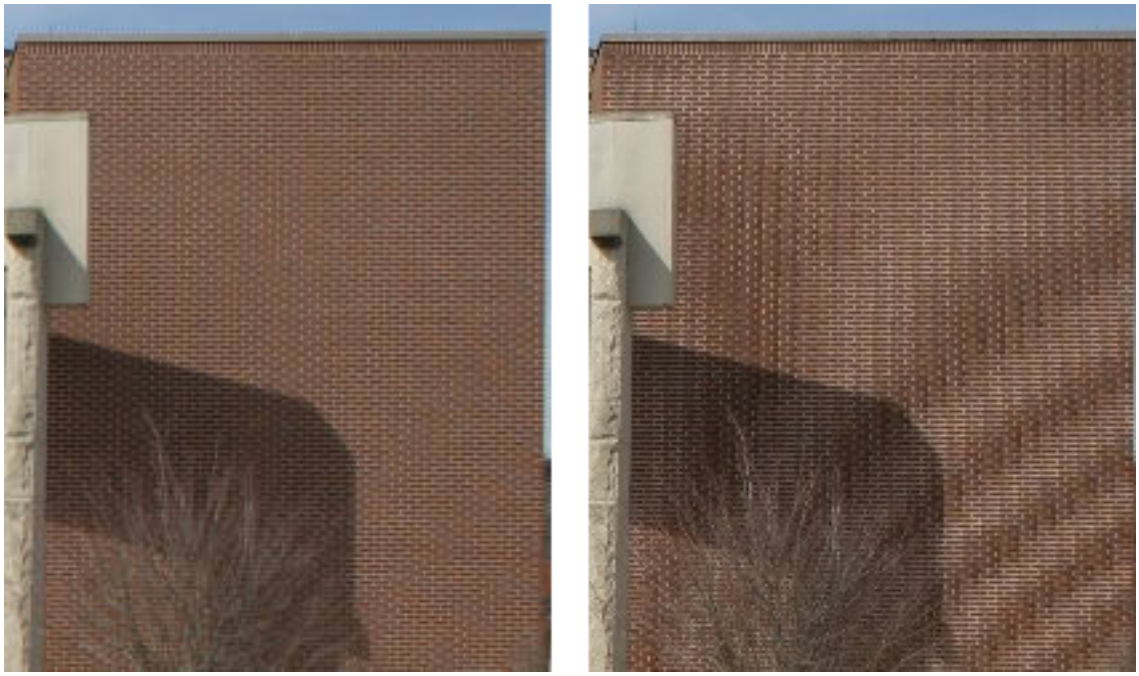
- **Supersampling** (viz [9]) – pro každý pixel výsledného obrazu se odebere několik vzorků, ze kterých se vypočítá hodnota pixelu jako aritmetický průměr hodnot těchto vzorků. Rozložení těchto vzorků uvnitř pixelu může být různé (obr. 2.6). Tento algoritmus vyžaduje řádově několikanásobně větší množství paměti než by bylo potřeba pro uchování výsledného obrazu. Částečným řešením tohoto problému je použití multisamplingu.
- **Multisampling** (viz [12]) – adaptivní supersampling, kdy se vzorkování s větší frekvencí používá pouze na hranicích vykreslovaných objektů. Uvnitř objektů či monotónních ploch se pixely vyhodnocují pouze pro jeden vzorek. Pro ukládání mezivýsledků částečně použitých vzorků se používá speciální buffer.
- **Vyhlazení obrazu** – oproti předchozím metodám nezískává přidanou informaci pomocí více vzorků pořízených při vyšším rozlišení, ale až ze sousedních pixelů výsledného obrazu. Dochází tak k mnohem větší ztrátě informace, respektive rozmazání. Hodnoty pixelů se určují některým z interpolačních algoritmů, o kterých se čtenář dozví v následující části 2.3.

2.3 Interpolace obrazových dat

V [22] nalezneme několik definic interpolace, pro účely této práce se však nejlépe hodí následující:

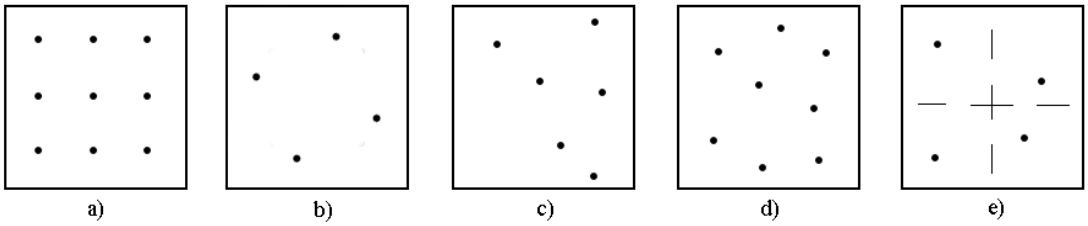
> Interpolace je na modelech založená rekonstrukce spojitých dat z diskrétních dat na známém (na rozdíl od extrapolace) intervalu první souřadnice.

Interpolace je tedy metoda pro výpočet hodnot obrazové funkce v „neměřených“ souřadnicích, jak se píše v [16]. Vzhledem k tomu, že hodnoty v takových bodech se získávají



Obrázek 2.5: Správně navzorkovaný obrázek cihlové zdi (vlevo) a prostorový alias vzniklý vzorkováním na nízké frekvenci (vpravo), převzato z [4].

.



Obrázek 2.6: Typy supersamplingu podle rozložení vzorků uvnitř pixelu: a) mřížka, b) otočená mřížka, c) náhodný, d) Poissonův disk (náhodné vzorky s dodrženou minimální vzdáleností), e) jitter (kombinace a) a d)). Převzato z [33].

.

z hodnot okolních bodů, se do obrazu nikdy nepřidá nová informace. Spíše naopak dochází k její ztrátě a tím i degradaci obrazu (viz dále popis jednotlivých metod). Výsledný obraz je vždy pouhou aproximací původního obrazu. Mezi nejpoužívanější metody interpolace obrazu patří následující:

- **Metoda nejbližšího souseda** (viz [15], [16]) – nejjednodušší z interpolačních metod, v čemž spočívá její síla i slabost. Hodnota výsledného pixelu se určí jako hodnota nejbližšího pixelu ve zdrojovém obraze. Výsledný obraz je většinou kostrbatý, na pohled nepěkný, avšak výpočetní čas je minimální. Interpolace touto metodou je jako jediná použitelná i pro obrazy s indexovanými barvami a pro černobílé obrazy.

- **Bilineární interpolace** (viz [15]) – pokročilejší metoda, hodnota výsledného pixelu se spočítá jako vážený průměr hodnot pixelů z okolí o rozměru 2×2. Striktně matematicky vzato jde o dvojici po sobě jdoucích lineárních interpolací ve směru os x a y. Na pořadí těchto operací nezáleží – algoritmus je tedy separabilní. Jeho výsledkem je mnohem hladší obraz než při použití metody nejbližšího souseda.

- **Bikubická interpolace** (viz [15], [3]) – pracuje ještě o krok dále než bilineární interpolace – hodnota výsledného pixelu se počítá z okolí o rozměru 4×4 pixely. Opět jde o vážený průměr, bližší pixely mají přirozeně větší váhu. Tento algoritmus je výpočetně náročnější než předchozí dva, nicméně produkuje o poznání ostřejší výstupy. Podle [15] je ideálním kompromisem mezi výpočetní dobou a výstupní kvalitou. Proto je implementován v mnoha grafických editorech, ovladačích tiskáren i digitálních kamerách a fotoaparátech.

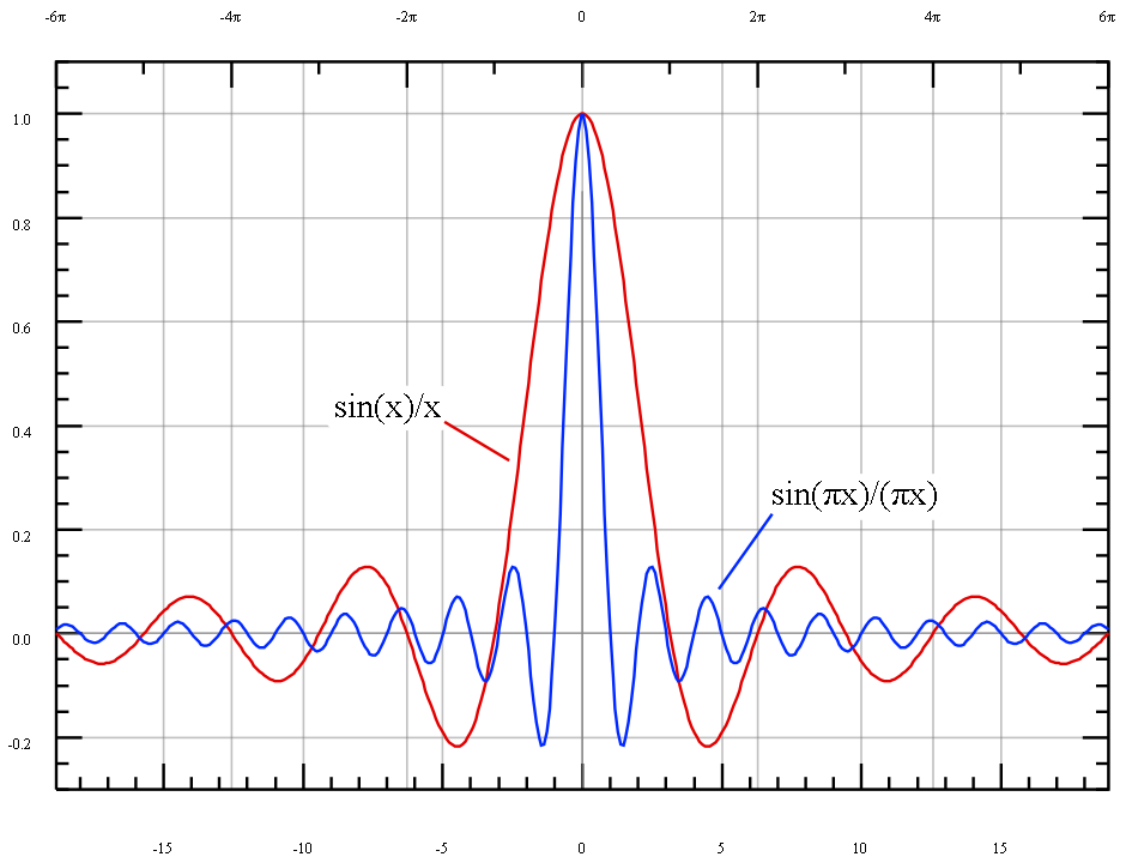
- **Metody založené na konvolučních filtrech** – výsledný obraz je získán konvolucí vstupního obrazu s maticí filtru, jehož koeficienty zpravidla aproximují nějakou funkci. Tato práce se dále zabývá jen filtrem aproximujícím funkci sinc (viz rovnice 2.2 a obr. 2.7).

$$\mathrm{sinc}(x) = \frac{\sin(x)}{x} \tag{2.2}$$

  Podle [19] je sinc filtr ideálním filtrem typu dolní propusť, navíc s lineární fází. V časové doméně je jeho impulzní odezvou vlastní funkce sinc, ve frekvenční doméně je to obdélníková funkce. Protože je impulzní odezva ideálního sinc filtru nekonečně dlouhá, jsou reálné filtry pouze jeho aproximací.

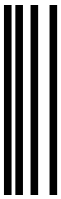
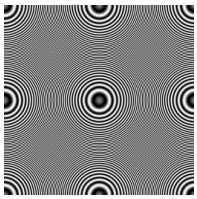
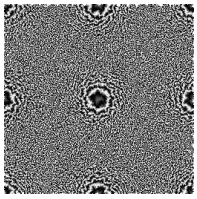
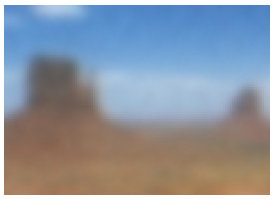
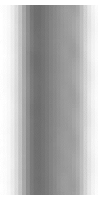
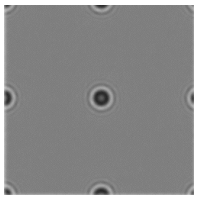
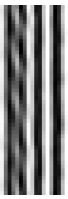
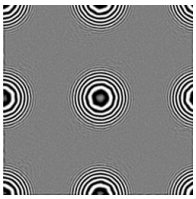
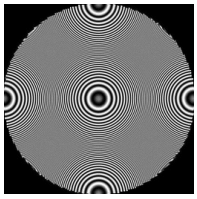
  V praxi se funkce sinc omezuje různými okny a vzniká tak „windowed-sinc“ filtr. Nejpoužívanějšími okny jsou Lanczosovo, Hammingovo, Hamming-Hanovo nebo Blackmanovo. Takovým oknem se rozumí spojitá sudá nezáporná funkce s jediným maximem, jejíž funkční hodnoty jsou nenulové pouze na intervalu $(-d; d)\,, d \in R$. Kýženého efektu omezení se dosáhne prostým vynásobením s funkcí sinc. Ta pak nabývá vně intervalu $(-d; d)\,, d \in R$ rovněž nulových hodnot. Tvar výsledných funkcí je možno vidět v [18] nebo [13].

V tabulce 2.1 je k dispozici srovnání uvedených interpolačních metod.



Obrázek 2.7: Normalizovaná ($\frac{\sin(\pi x)}{\pi x}$, modře) a nenormalizovaná ($\frac{\sin(x)}{x}$, červeně) funkce sinc vykresleny do stejného grafu na intervalu $x \in (-6\pi; 6\pi)$. Převzato z [32].

.

	a	b	c
Originál			
Interpolace metodou nejbližšího souseda			 1
Bilineární interpolace			 2
Bikubická interpolace			 8,5
Interpolace sinc filtrem o velikosti 8 x 8			 20

Tabulka 2.1: Srovnání interpolačních metod: Byly použity tři testovací obrázky (**a**, **b**, **c**). Test spočíval v 36-násobné rotaci originálů o 5°, kdy po každé dílčí rotaci byly obrázky interpolovány příslušnými metodami. Nakonec bylo provedeno otočení o 180° zpět, což je bezeztrátová operace. Výsledky jsou určeny především k subjektivnímu porovnání, u obrázku **c** je navíc uveden počet zachovaných kroužků. Obrázky a výsledky testu byly převzaty z [6].

Kapitola 3

Programovatelný hardware

Problematika programovatelného hardwaru je velmi komplexní, tato práce si však neklade za cíl pokrýt ji celou; z důvodu rozsahu se omezuje pouze na fakta vztahující se k tématu práce.

3.1 Software vs. hardware

Obecně se dá říci, že algoritmus musí být někým nebo něčím vykonán, aby měl efekt. V širším slova smyslu to však nemusí být jen klasický procesor, který se používá v PC, ale například i člověk, virtuální stroj nebo speciální hardwarový obvod. V oblasti informačních technologií se implementátor nejčastěji rozhoduje mezi realizací algoritmu v softwaru, programovatelném hardwaru nebo aplikačně specifickém hardwaru. Každá z těchto voleb má přirozeně svá pro i proti, proto je nutno důkladně zvážit účel aplikace, finanční možnosti i jiné aspekty. Následuje stručné porovnání jednotlivých způsobů implementace.

- **Software** – pravděpodobně nejpoužívanější přístup. Vývoj softwaru je poměrně rychlý, lechce se modifikuje a většinou je přenositelný. Oprava chyb je také rychlá a bez dalších větších nákladů. Nevýhodou softwarových implementací je jejich relativní pomalost a neefektivnost. Výkon je omezen zařízením, na kterém je program vykonáván.

- **Programovatelný hardware** – je jakýmsi mezistupněm mezi softwarem a skutečným aplikačně specifickým hardwarem. Z hlediska provozu se taktéž jedná o aplikačně specifický HW, je zde ale možnost rekonfigurace a použití pro jiný účel (více viz následující část 3.2). Programovatelný hardware se stává v poslední době stále oblíbenějším a to hned z několika důvodů: Ceny programovatelných čipů v minulé dekádě znatelně poklesly, zvyšuje se abstrakce od nejnižší vrstvy HW, což klade menší požadavky na vývojáře, a neopominutelným faktem je také vysoký výkon vyvinutých aplikací. Ve srovnání se SW trvá vývoj aplikací déle a vývojář je nucen porozumět SW i HW. Vývojových nástrojů extistuje mnohem méně a jsou řádově dražší, oprava chyb je náročnější. Negativem je také závislost na cílové platformě, i když v dnešní době už je v omezené míře možno kód aplikací přenášet dokonce mezi různými typy čipů.

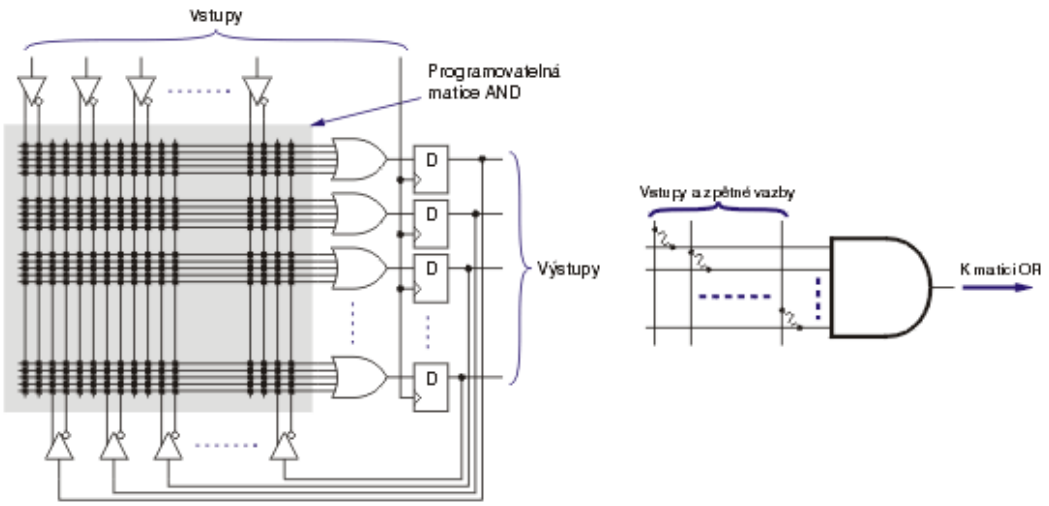
- **Aplikačně specifický hardware** – někdy také ASIC (z anglického „Application Specific Integrated Circuit“). Jedná se o řešení, které bývá nejvýkonnější, ale také nejdražší. Specializace integrovaných obvodů na jedinou úlohu umožňuje využít HW prostředky beze zbytku, což vede k velmi vysokému výkonu. Na vývojáře, který musí

dokonale znát cílovou technologii, jsou však kladeny velké nároky. Vývoj aplikace trvá extrémně dlouho a oprava chyb je nemožná. Vzhledem k vysoké ceně za vývoj aplikace se použití tohoto přístupu vyplatí pouze při masové produkci zařízení, výroba několika desítek či stovek kusů je nerentabilní.

3.2 Typy programovatelného hardwaru

Pod pojmem programovatelný hardware se většinou rozumí tzv. programovatelná hradlová pole (někdy také PLD, z anglického Programmable Logic Device). PLD je elektronická součástka pro implementaci rekonfigurovatelných číslicových obvodů. Na rozdíl od zařízení typu ASIC (viz předchozí část 3.1) nemají PLD při výrobě určenu svou funkci. Tu získají až po naprogramování.

Ještě před vynálezem PLD jejich funkci zastávaly paměti typu ROM. Uvažujeme-li paměť o 2^m paměťových buňkách o šířce n bitů, kterou adresuje m nezávislých logických proměnných, můžeme na výstupních pinech snímat realizaci n logických funkcí (bohužel pro m signálů takových funkcí existuje teoreticky 2^m). Taková paměť se pak stává ekvivalentem n samostatných kombinačních obvodů, z nichž každý generuje zvolenou funkcí m proměnných. Jejich výhodou je, že kýženou logickou funkci můžeme generovat na kterýkoliv z výstupních pinů, což činí z ROM pamětí nejvšestranější kombinatorické zařízení. Přepisovatelné paměti typu PROM[1], EPROM[2] nebo EEPROM[3] mohou být navíc přeprogramovány. Nevýhodou tohoto řešení je malá rychlost pamětí ve srovnání se specializovanými integrovanými obvody, větší spotřeba energie a nedefinované výstupy v přechodech mezi dvěma stavy. Ve většině aplikací také není plně využita jejich kapacita.



Obrázek 3.1: Charakteristická vnitřní struktura PLD – programovatelná matice logických součinů (vlevo). Každá vodorovná čára je jedno součinové hradlo, jehož zapojení lze vidět vpravo. Vlnovky značí programovatelné přepínače. Převzato z [17].

.

[1]Programmable ROM – programovatelná paměť ROM.
[2]Ultraviolet-Erasable PROM – PROM paměť vymazatelná ultrafialovým zářením.
[3]Electrically Erasable PROM – elektricky vymazatelná paměť PROM.

První programovatelná hradlová pole se začala vyrábět počátkem sedmdesátých let. Průkopníkem byla firma Texas Instruments, která vyráběla tzv. PLA[4]zařízení, která měla až 17 vstupů, 18 výstupů a 8 registrů z klopných obvodů typu JK. Vstupy se propojovaly s výstupy pomocí dvou matic z AND a OR hradel. O několik let později vyvinula firma General Electric zařízení programovatelné pomocí EPROM paměti, které se stalo prvním rekonfigurovatelným PLD na světě. Umožňovalo také využít více úrovní kombinační logiky. Za několik dalších let stejná firma vyvinula vyspělejší zařízení, které navíc disponovalo sekvenční logikou a mělo přes 100 hradel. Pro toto PLD bylo také vytvořeno vývojové prostředí umožňující konverzi booleovských výrazů na konfiguraci zařízení. Další generace PLD již byla komplexnější a používanější, proto jsou dále v této kapitole popsány podrobněji.

PLD se obvykle skládá z logických obvodů a paměti k uložení konfigurace zařízení, která je nahrána během programování čipu. Tato paměť může být volatilní[5](SRAM[6]) i nevolatilní[7] (EPROM, EEPROM). Zařízení s volatilními paměťmi musí být po zapnutí naprogramována, aby se stala funkčními, oproti tomu zařízení s nevolatilními paměťmi si pamatují svou konfiguraci i po vypnutí.

Konfigurace se do PLD nahrává pomocí tzv. programátoru, což je další zařízení pro transfer konfiguračních dat do čipu. Dříve poskytoval každý výrobce speciální programátor, který podporoval pouze danou rodinu zařízení. Později se začaly objevovat programátory pracující s různými rodinami čipů od několika výrobců. Dnešní programátory jsou univerzální a obvykle umí pracovat s PLD od všech existujících výrobců.

Pro programování hradlových polí se používají tzv. jazyky pro popis hardwaru (častěji HDL – z anglického Hardware Description Language). Dříve se pro jednodušší zařízení používaly jazyky jako PALASM nebo ABEL, ale s rostoucí složitostí PLD vznikly jazyky VHDL a Verilog. Tyto jsou dnes velmi oblíbené a používají se proto i pro popis méně komplexních zařízení.

Následuje podrobnější popis nejpoužívanějších typů programovatelných hradlových polí.

- **PAL** – programovatelné logické pole (z anglického Programmalbe Array Logic). Tato zařízení byla do jisté míry průlomem v oblasti programovatelného hardwaru, protože postrádala programovatelné OR pole, což je činilo rychlejšími, menšími a hlavně levnějšími. Jejich nevýhodou je nulová znovupoužitelnost, protože pro uložení konfigurace používají paměť PROM.

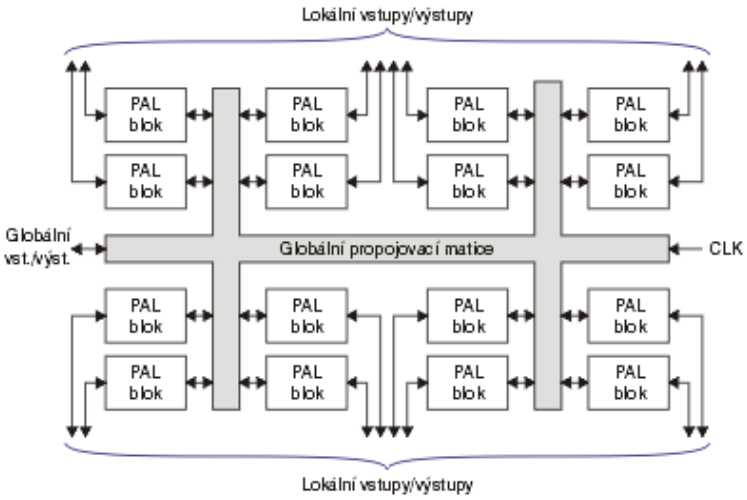
  Zařízení PAL se skládají ze dvou hlavních částí: programovatelného AND pole a výstupních buněk. Programovatelné pole je tvořeno PROM pamětí, jejíž obsah určuje, které vstupní signály povedou do kterých výstupních buněk. Jak už bylo zmíněno, tranzistory uvnitř PAL jsou uspořádány do konfigurovatelného AND pole a fixního OR pole, což umožňuje implementovat Booleovské rovnice ve formě součtu součinů (angl. Sum Of Products – SOP). Jako proměnné mohou být použity nejen vstupní signály, ale i výstupní signály a jejich negace (princip zpětné vazby).

  Výstupy mohly být kombinační nebo registrované, ale všechny stejného typu, protože jejich funkce byla dána konfigurací výstupních buněk už při výrobě. Tato skutečnost často znepříjemňovala vývojářům život, protože v řadě aplikací bylo třeba obou typů výstupů. Teprve po uvedení takzvaných variabilních sérií PAL bylo umožněno uživatelsky konfigurovat výstupy.

[4]Programmable Logic Array (nepřekládá se) – nezaměňovat s PAL.
[5]Paměť závislá na zdroji el. energie; při vypnutí napájení se její obsah ztrácí.
[6]Static RAM – statická paměť. Oproti dynamické paměti nemusí být periodicky obnovována.
[7]Paměť nezávislá na zdroji el. energie; při vypnutí napájení její obsah zůstává nezměněn.

- **GAL** – generické logické pole (z anglického Generic Array Logic). Jedná se o inovaci PAL vyvinutou firmou Lattice Semiconductor v roce 1985. Nově bylo možno zařízení vymazat a přeprogramovat, což bylo velkým přínosem hlavně pro fázi prototypování a ladění aplikace. Přirozeně bylo také přítomno několikanásobně více hradel než v předchozí generaci programovatelných hradlových polí.

- **CPLD** – komplexní programovatelné hradlové pole (z anglického Complex Programmable Logic Device). CPLD obsahují několik ekvivalentů PLA nebo PAL spojených navzájem programovatelnými propojkami. V [10] se uvádí, že tato zařízení stojí úrovní složitosti mezi PAL a FPGA, přičemž od každé třídy zařízení přejímají určité vlastnosti. Stejně jako PAL obsahují nevolatilní paměť, takže jsou schopna se sama nakonfigurovat po zapnutí napájení. Trpí však omezenými možnostmi propojení vnitřních bloků, což lehce degraduje jejich možnosti. Novější rodiny CPLD se ale tohoto neduhu postupně zbavují. K rodině čipů FPGA se blíží počtem hradel (desítky až stovky tisíc) a některými dalšími architektonickými prvky, které umožňují využívat složité zpětné vazby nebo implementovat celočíselné aritmetické operace. Základní architekturu CPLD je možno vidět na obrázku 3.2.



Obrázek 3.2: Typická struktura obvodu CPLD, převzato z [17].

.

Přítomnost nevolatilní paměti činí v současnosti z CPLD často používané zařízení, zejména pro funkci tzv. boot loaderu (po startu systému přebírá kontrolu a stará se o konfiguraci ostatních zařízení bez volatilních pamětí, jako např. FPGA).

Komplexní programovatelná hradlová pole se konfigurují pomocí klasického programátoru. Tento způsob je ale pro čipy s mnoha vývody nepraktický, proto se dnes více používá jiná metoda: Čip je pevně připájen na desku plošných spojů a konfigurace je nahrána ve formě bitstreamu přímo z počítače. CPLD disponuje vestavěným obvodem, který se stará o dekódování serializované informace a nastavení celého zařízení. Každý výrobce vyvinul vlastní rozhraní, v poslední době se však dává přednost standardizovanému rozhraní JTAG[8].

[8] Joint Test Action Group (nepřekládá se) – obvyklý název pro normu IEEE 1149.1 specifikující sériové rozhraní pro testování a konfiguraci integrovaných obvodů.

- **FPGA** – programovatelné hradlové pole (z anglického Field Programmable Gate Array). Zatímco zařízení typu PAL se postupně vyvíjela v GAL a CPLD, zařízení typu FPGA představují poměrně samostatnou vývojovou větev. První exempláře se objevily začátkem osmdesátých let a obsahovaly hradlové pole z členů AND. Moderní kusy obsahují spíše NAND a NOR hradla a další složitější prvky.

  FPGA jsou konfigurovatelná uživatelem (odtud jejich název – „on field programmable“ jako „na místě programovatelné“). To se provádní většinou až po umístění čipu na desku pomocí sériového rozhraní (např. JTAG). Programovatelná hradlová pole neobsahují nevolatilní paměť pro uložení konfigurace, proto musí být naprogramována po každém odpojení napájení. CPLD a FPGA jsou často pro některou úlohu stejně vhodná a tak se rozhodujícím faktorem stává vývojářova osobní preference či zkušenost s daným zařízením.

  Podrobnější popis programovatelných hradlových polí může čtenář najít v následující části 3.3.

- **Ostatní** – velký výzkum nyní probíhá v oblasti samorekonfigurovatelných zařízení. Jedná se o mikroprocesory, které mají část funkčnosti pevně danou z výroby, ale zároveň disponují programovatelnými obvody, které mohou rekonfigurovat na základě běhu programu. Návrh takových systémů vyžaduje od vývojářů užívání nových postupů a metod.

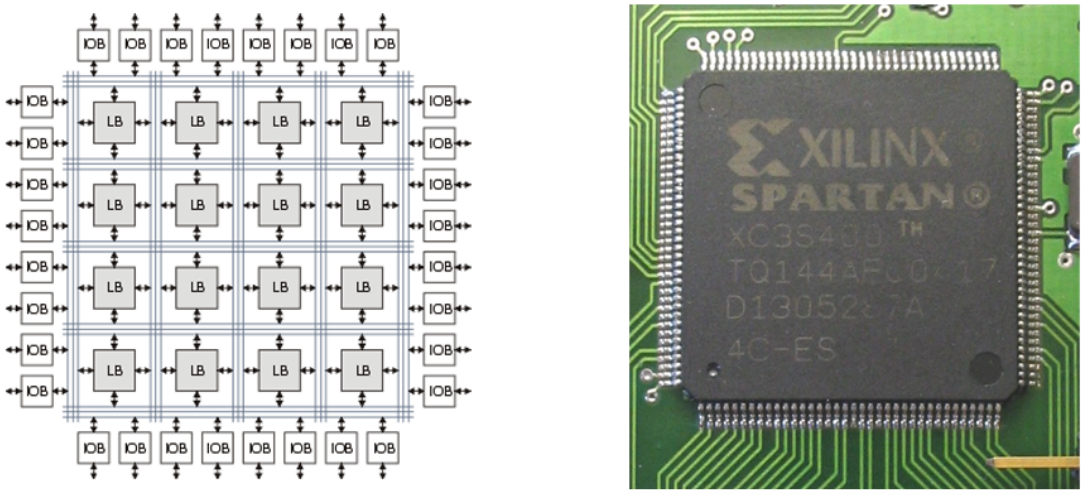
Vzhledem k tématu práce následuje podrobné seznámení s technologií a architekturou čipů FPGA.

3.3 FPGA

Programovatelné hradlové pole je konfigurovatelný čip, jehož základními prvky jsou tzv. logické bloky, vstup-výstupní bloky a propojovací systém (viz obr. 3.3). Logické bloky mohou být naprogramovány pro vykonávání jednoduchých funkcí (například mohou pouze zastupovat jednotlivá hradla jako AND, XOR, atd.) nebo složitějších kombinačních funkcí (jako třeba dekodéry nebo různé matematické funkce). Tyto bloky také zpravidla obsahují paměťové prvky – jak jednotlivé klopné obvody, tak i souvislejší bloky paměti. Vstup-výstupní bloky jsou připojeny ke každému pinu FPGA a obsahují registr, budič, multiplexor a ochranné obvody. Systém programovatelných propojek umožňuje vývojáři téměř libovolně spojovat logické bloky do celků, což je důvod vysoké flexibility těchto zařízení.

FPGA jsou obvykle pomalejší než ekvivalentní aplikačně specifické obvody, také nejsou schopny pojmout tak komplexní design a mají vyšší příkon. Na druhou stranu podporují rychlejší vývoj aplikací a jejich nasazení na trh, snižují náklady na vývoj a umožňují opravování chyb pouhým přeprogramováním čipu.

První čipy podobné programovatelným hradlovým polím se začaly objevovat na začátku osmdesátých let a konstrukčně vycházely z CPLD. Podle [28] spatřilo první skutečné FPGA světlo světa až v roce 1984, kdy jej vynalezl Ross Freeman, zakladatel společnosti Xilinx. CPLD i FPGA obsahují poměrně velké množství programovatelné logiky – CPLD asi desítky až stovky tisíc ekvivalentních logických hradel, zatímco FPGA miliony. Hlavní rozdíl mezi těmito zařízeními je ale v jejich konstrukci. CPLD obsahuje více omezující strukturu sestávající z několika programovatelných PAL polí napojených na malý počet registrů. Výsledkem je omezená flexibilita výměnou za dobře předvídatelné časování obvodů. V FPGA oproti tomu dominuje propojovací systém, který je vysoce flexibilní, ale způsobuje málo

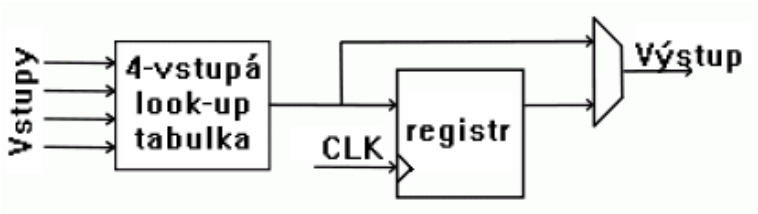


Obrázek 3.3: Vlevo typická struktura FPGA (převzato z [17]), vpravo FPGA Spartan XC3S400 od firmy Xilinx osazené na desce plošných spojů (převzato z [34]).

předvídatelné časování a klade vyšší požadavky na vývojáře. V programovatelných hradlových polích také nalezneme logické bloky vyšší úrovně jako sčítačky či násobičky a vestavěné paměťové bloky. Novější FPGA disponují schopností částečné rekonfigurace, což umožňuje jednu část čipu přeprogramovat zatímco ostatní části stále pracují.

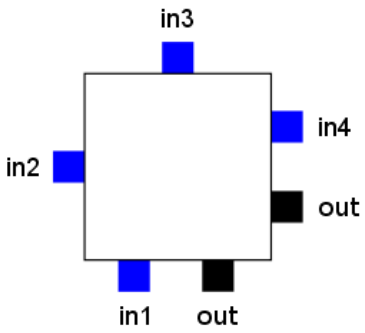
Současný trend ve vývoji programovatelných hradlových polí je přidávat mezi logické bloky a propojovací síť také mikroprocesorová jádra a různé periferie a tím se přiblížit myšlence kompletního systému na čipu. Takovou snahu lze pozorovat například v čipech Virtex-II PRO [29] a Virtex-4 [30] od firmy Xilinx, které obsahují procesor PowerPC nebo v čipu FPSLIC [27] od firmy Atmel, který obsahuje AVR procesor. Alternativní přístup k této myšlence je implementace procesorového jádra pomocí logiky FPGA (pak se mluví o soft procesorech) – takové řešení je možno najít v některých čipech od firmy Xilinx v podobě procesorů MicroBlaze a PicoBlaze.

Standardní logický blok se skládá ze čtyřvstupé look-up tabulky (LUT) a registru. V současné době výrobci nahrazují čtyřvstupé look-up tabulky šestivstupými, což umožňuje do jednoho logického bloku namapovat složitější kombinační funkce. Jak je vidět na obrázku 3.4, blok je vybaven pouze jediným výstupem, který může být volitelně registrovaný. Vstupů je opoti tomu pět (čtyři vstupy do LUT a hodinový signál, který je do bloku veden speciálními rozvody pro omezení zpoždění).



Obrázek 3.4: Typický logický blok FPGA. Převzato z [31].

Ve většině FPGA architektur má logický blok každý vstup připojen z jiné strany, aby se dosáhlo maximální univerzálnosti. Jediný výstup pak bývá duplicitně připojen rovněž z více stran (viz obr. 3.5). Každý ze vstupních nebo výstupních pinů může být připojen pouze k přilehlému segmentu propojovací sítě.



Obrázek 3.5: Typické umístění vstupních (in) a výstupních (out) pinů logického bloku. Převzato z [31].

Nejčastěji bývá jeden segment propojovací sítě dlouhý stejně jako výška či šířka logického bloku - to znamená v místech, kde by se měly propojovací vodiče křížit, se nekříží, ale končí v tzv. přepínači. Přepínač obsahuje programovatelné propojky a umožňuje vodiče ze sousedních segmentů propojovat[9].

Na obrázku 3.6 je znázorněn modelový přepínač, do kterého vstupují z každé strany tři vodiče. Každý vodič je uvnitř přepínače připojen ke třem programovatelným propojkám, které jej umožňují spojit se třemi vodiči ze sousedních segmentů. Topologie tohoto přepínače je tzv. rovinná, což znamená, že vodič č. 1 z jednoho segmentu lze propojit pouze s vodiči č. 1 z ostatních segmentů. Analogicky to platí i pro vodiče č. 2 a 3.

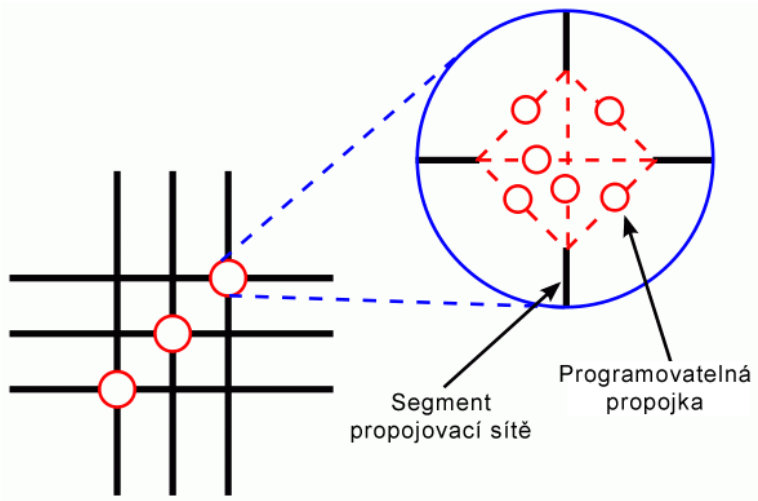
Pro popis designu FPGA se většinou používá některý z jazyků pro popis hardwaru (anglicky Hardware Description Language – HDL), zřídka také schéma. Nejpoužívanějšími HDL jazyky jsou v současnosti VHDL[10] a Verilog. Z popisu ve formě zdrojového kódu se pomocí speciálních nástrojů, které jsou většinou dodávány přímo výrobci FPGA čipů, vygeneruje tzv. netlist. Netlist je seznam funkčních primitiv a jejich propojení v elektrickém obvodu. Tento seznam pak musí být pomocí fáze syntézy zvané „place and route“ namapován na skutečné prvky čipu. Následuje poslední fáze – generování binárního konfiguračního souboru pro FPGA. Všechny tyto fáze jsou doprovázeny validací, časovou analýzou, simulacemi a verifikací ze strany vývojáře, aby se odstranily případné problémy s časováním, chyby v implementaci apod.

Jelikož jsou HDL jazyky poměrně složité (a často bývají srovnávány s assemblery), objevují se snahy o zvýšení úrovně abstrakce během vývoje designu. Jako příklad se často uvádí jazyky SystemVerilog, SystemVHDL nebo Handel-C. Zejména Handel-C se díky své příbuznosti s jazykem C snaží zpřístupnit design elektronických obvodů klasickým programátorům.

Další snahou o zjednodušení návrhu obvodů je tvorba knihoven s často používanými

[9] Za účelem snížení transportního zpoždění propojovacích vodičů obsahují některé architektury propojovací segmenty delší než jeden logický blok. To snižuje počet přepínačů, kterými musí signál na své cestě projít.

[10] Very high speed integrated circuit Hardware Description Language – nepřekládá se.



Obrázek 3.6: Topologie modelového přepínače. Převzato z [31].

funkcemi či celými obvody. Tyto obvody se většinou nazývají „IP-cores“[11] a jsou k dostání přímo u výrobců FPGA nebo u třetích stran. Obvykle bývají poskytovány za úplatu.

FPGA čipy jsou hojně používány v oblasti zpracování digitálních signálů, lékařského obrazu, v oblasti počítačového vidění, rozpoznávání řeči, bioinformatice, kryptografii, v leteckých a bojových systémech, při prototypování aplikačně specifického hardwaru i v jiných oblastech. Většinou se jedná o obory, které dokáží využít masivní paralelismus programovatelných hradlových polí. Tato zařízení se v poslední době dostávají také do superpočítačů, kde místo procesorů vykonávají výpočetně nejnáročnější části algoritmů jako je FFT[12] nebo konvoluce. Překážkou ještě širšího nasazení FPGA čipů je hlavně vysoká složitost procesu tvorby designu ve srovnání se softwarem a také dlouhá doba překladu zdrojových kódů i při jejich nepatrné změně.

[11] Intelectual Propery core – IP jádro.
[12] Fast Fourier Transform (rychlá Fourierova transformace).

Kapitola 4

Akcelerace převzorkování v FPGA

O převzorkování se mluví jako o datově intenzivní úloze, protože každý jednotlivý pixel výstupního obrazu musí být vypočítán obecně z několika pixelů vstupního obrazu a tyto obrazy mohou dosahovat rozměrů až několika megapixelů. Pokud se navíc nepoužije některá z jednoduchých metod interpolace (jako např. metoda nejbližšího souseda), je každý výstupní pixel výsledkem konvoluce určité oblasti vstupního obrazu s převzorkovacím filtrem. Pak se převzorkování stává i výpočetně intenzivní úlohou.

Jak již bylo řečeno, jádrem pokročilého převzorkování bývá konvoluce. Jedná se o operaci typu MAC[1], která není elementární a skrývá v sobě několik násobení a sčítání. Pokud takový výpočet běží na klasickém procesoru, je zpracování většího množství dat neúnosně dlouhé. Jako příklad poslouží klasické osobní PC s procesorem pracujícím na frekvenci 2 GHz. Tento stroj je schopen převzorkovat řádově jednotky megapixelů obrazových dat za sekundu. Takový výkon může stačit pro sporadické zpracování menších obrazů, ale pokud je třeba převzorkovat větší množství snímků v rozlišení desítek megapixelů, je třeba mnohem vyššího výkonu. Pokud navíc zvážíme možnost převzorkování 3D dat, či geometrickou korekci snímků videa v reálném čase, stává se možnost použití klasického procesoru téměř nereálnou.

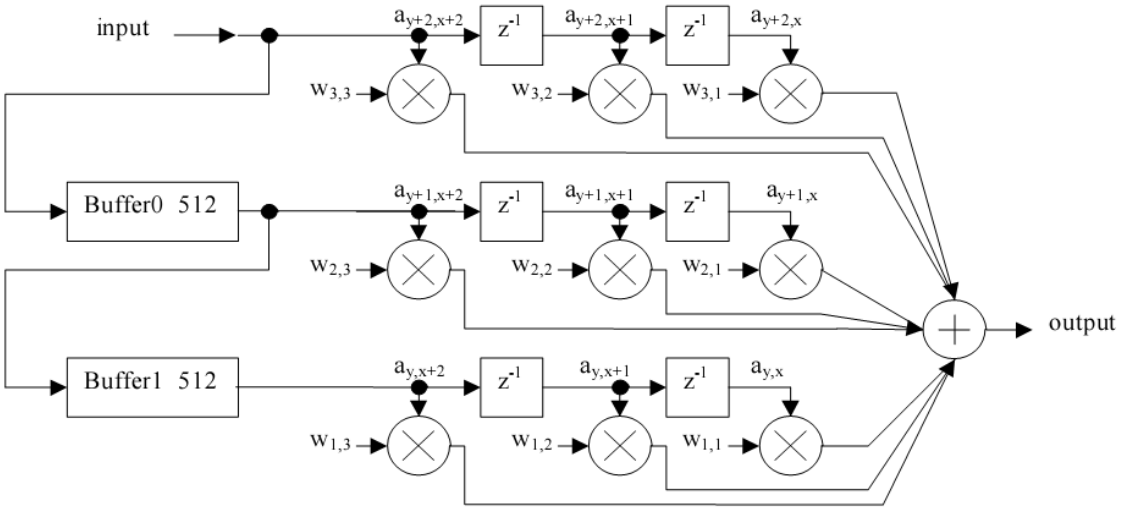
Řešením tohoto problému může být rozdělení zátěže mezi více počítačů nebo procesorů. Výhodou takové volby je možnost tyto počítače využít i pro jiný účel, avšak spatřuji zde dvě základní nevýhody: Nutnost obětovat určitý výkon na rozdělení zátěže mezi jednotlivé výpočetní jednotky a pak také nezanedbatelná spotřeba energie. Rovněž je nutno brát v potaz velikost celé soustavy počítačů.

Jiným řešením je vývoj a výroba specializovaného hardwaru. Tato varianta skýtá možnost paralelizace a využití dedikovaných obvodů pro některé podúlohy. Otázkou zůstává, jaký druh hardwaru využít. Aplikačně specifický obvod by byl určitě nejrychlejší alternativou, ovšem náklady na jeho výrobu jsou nezanedbatelné a není zde ponechán jakýkoliv prostor pro opravu případných chyb. Těmito dvěma nedostatky ovšem netrpí programovatelný hardware, ukazuje se tedy jako optimální volba. Ze skupiny programovatelných logických zařízení se pak jako nejlepší kandidát jeví rodina čipů FPGA kvůli vestavěné RAM paměti pro uložení obrazových dat.

Při dokonalém využití paralelismu hardwaru je teoreticky možné dosáhnout propustnosti až 1 pixel za takt. To při použití konvoluce ovšem znamená mít v jednom taktu přístup k tolika pixelům, kolik má konvoluční filtr koeficientů. Například při použití filtru o rozměru 4×4 pixely je třeba v jednom taktu mít přístup k 16 pixelům. Pokud jsou vstupní data uložena v RAM paměti, vyžaduje to mít k dispozici 16 paměťových portů pouze na

[1] Multiply And Accumulate – volně se dá přeložit jako „násobení a přičítání“.

čtení, nemluvě o dalších portech potřebných pro zápis nových vstupních dat. Možným řešením tohoto problému je využívat některé z portů jako čtecí i zápisové, to ale znamená dvojnásobné prodloužení výpočtu kvůli střídání zápisových a čtecích cyklů. Mnohem častěji je však možno vidět jiné řešení (viz [2] nebo [26]): Data jsou nahrávána do soustavy zřetězených registrů (tzv. shift-registrů), odkud mohou být vyčítána pro konvoluci. Zároveň jsou nahrávána také do fronty FIFO o takové kapacitě, jaký je horizontální rozměr obrazu. Za frontu jsou opět zapojeny shift registry korespondující s následujícím řádkem obrazu atd. Blíže viz obrázek 4.1.



Obrázek 4.1: Konvoluční jednotka pro jádro o rozměru 3×3 a obrázek o rozměru 512×512 pixelů. Převzato z [26].

Řešení popsané na obrázku 4.1 sice odstraňuje potřebu současného vyčítání mnoha pixelů ze vstupního bufferu, ale tentýž problém ve skutečnosti přetrvává u koeficientů konvolučního filtru. Koeficienty je totiž třeba načítat stejnou rychlostí a ve stejném počtu jako vstupní obrazová data. V [21] se navrhuje koeficienty generovat pro každý výstupní pixel pomocí hardwarového generátoru. Taková jednotka by ovšem výrazně zkomplikovala celý systém, protože by musela aproximovat funkci charakterizující konvoluční filtr. Také je třeba zohlednit zdroje, které by její implementace zabrala. Pokud není třeba koeficienty generovat dynamicky v každém taktu, je mnohem lepším řešením uložit je jako vypočtené konstanty do RAM paměti spolu se vstupními daty. Na první pohled by se mohlo zdát, že kapacita paměti obsazené konvolučními filtry bude malá, avšak jak se dále ukáže, je třeba mít filtry k dispozici v různě rotovaných pozicích. Toho se dá dosáhnout dvojím způsobem: Buď použitím barrel shifteru[2] (skládá se z velkých multiplexorů snižujících maximální frekvenci systému) nebo uložením vhodně rotovaných filtrů přímo do paměti. Tento přístup ke konvoluci tedy vyžaduje stále velké množství paměti a paměťových portů.

Redukce počtu portů je možno dosáhnout například snížením propustnosti celého zařízení, kdy by se určitá část koeficientů a vstupních dat předvyčítala do sady registrů. Výpočet konvoluce by proběhl až po načtení všech potřebných dat. Snížení propustnosti je ovšem nežádoucí jev. Další možností je zmenšení velikosti konvolučního filtru. To by mělo za následek zmenšení objemu dat a potažmo počtu portů nutných k výpočtu jednoho výstupního

[2]Obvod realizující posuny a rotace dat.

pixelu. Na druhou stranu je ale nutno počítat také se snížením kvality převzorkování. Východiskem je modifikace převzorkovacího algoritmu do separabilní podoby, což bude mít za následek rozklad dvourozměrného konvolučního filtru do dvou filtrů jednorozměrných (řádkového a sloupcového). Obraz pak bude převzorkován zvlášť v horizontálním a vertikálním směru. Algoritmus bude po této úpravě méně obecný, což však pro účely této práce nevadí.

I když se podaří modifikovat algoritmus do separabilní podoby, stále bude nutné přistupovat k několika pixelům vstupního obrazu současně. Metoda postupného vyčítání již byla zavrhnuta z důvodu snížení propustnosti systému, nezbývá tedy než se vrátit k původní myšlence víceportové paměti. V tomto případě bude nutno navrhnout kruhový buffer s ohledem na vlastnosti převzorkovacího algoritmu tak, aby byly vždy všechny potřebné pixely dostupné v jednom taktu naráz.

Již bylo naznačeno, že je výhodné ukládat vstupní obraz do vestavěných RAM pamětí z důvodu jejich nízké latence. Ačkoliv je kapacita těchto pamětí relativně velká, stále nedostačuje k uložení celého vstupního obrazu o velikosti několika megabajtů. Z tohoto důvodu je nutné obraz zpracovávat po částech, které se pravděpodobně kvůli velikosti konvolučního jádra budou muset z části překrývat. K rozřezání obrazu lze s výhodou použít DSP procesor, který je pro takový druh operací dimenzován.

Dosud neexistuje zařízení, které by bylo při zachování malých rozměrů a nízké spotřeby elektrické energie schopno v reálném čase převzorkovat obrázky ve vysokém rozlišení. Proto je v následující kapitole popsán návrh a implementace takového zařízení založeného na spolupráci DSP procesoru a FPGA čipu.

Kapitola 5

Implementace

5.1 Algoritmus separabilního převzorkování

V předchozí kapitole bylo řečeno, že je třeba sestavit algoritmus separabilního převzorkování. Takový algoritmus může fungovat jen tehdy, pokud i použitý konvoluční filtr bude separabilní a zároveň se bude předpokládat, že prováděná rotace bude zanedbatelná (tzn. například vzniklá pouze poduškovitým zkreslením objektivu).

Matematický popis poduškovitého zkreslení se v hardwaru implementuje poměrně složitě, proto je žádoucí nalézt jednodušší popis zkreslení obrazu. Pokud se obraz rozdělí na dostatečný počet pravoúhlých oblastí, je možné zkreslení v rámci každé takové oblasti považovat za lineární a převzorkovat ji samostatně. Navíc zkreslení každé oblasti je v tomto případě možné úplně charakterizovat pouze pomocí čtyř koeficientů (názorněji viz obrázek 5.1):

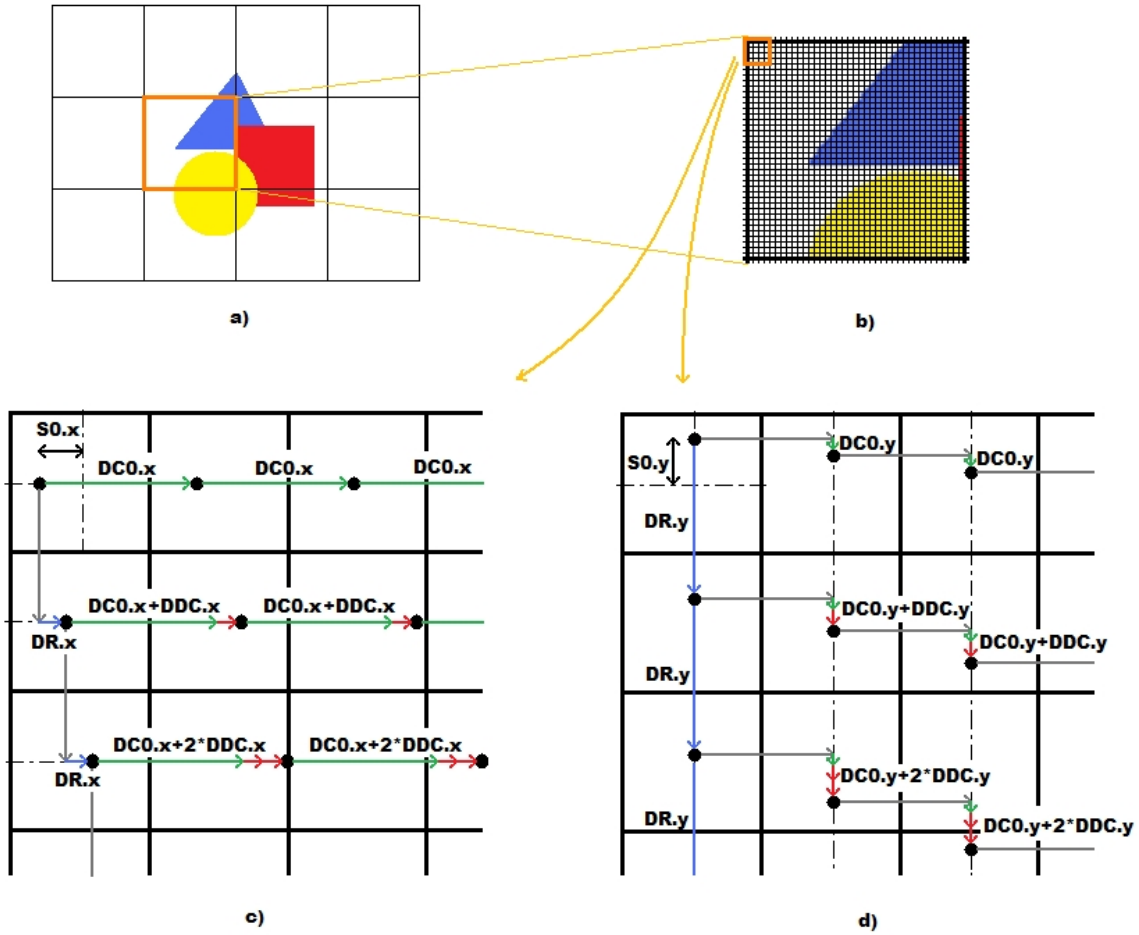
- $\mathbf{S_0}$[1] – pozice levého horního pixelu pravoúhlé oblasti
- $\mathbf{DC_0}$[2] – rozdíl souřadnic mezi sousedními pixely 1. řádku oblasti
- **DR**[3] – rozdíl souřadnic mezi pixely sousedních řádků
- **DDC**[4] – změna rozdílu souřadnic mezi pixely sousedních řádků

Kromě uvedených koeficientů, které jsou vstupními parametry algoritmu, jsou k jeho realizaci třeba ještě tyto proměnné:

- **SoR**[5] – pozice prvního pixelu v aktuálním řádku
- **DC** – rozdíl souřadnic mezi sousedními pixely v aktuálním řádku
- **S** – pozice aktuálního pixelu

Výstupem algoritmu je proměnná **S** určující pozici konvolučního filtru v aktuálně zpracovávané oblasti zdrojového obrazu. Algoritmus je vykonáván takto: Nejprve proběhne inicializace, kdy je pozice prvního pixelu v řádku **SoR** nastavena na pozici levého horního

[1] Podle anglického „Standing“ – pozice.
[2] Podle anglického „Difference – Cell“ – rozdíl buněk.
[3] Podle anglického „Difference – Row“ – rozdíl řádků.
[4] Podle anglického „Difference^2 – Cell“ – změna rozdílu buněk.
[5] Podle anglického „Start of Row“ – začátek řádku.



Obrázek 5.1: Koeficienty algoritmu separabilního převzorkování. **a)** představuje celý obraz rozdělený na pravoúhlé oblasti. Jedna z těchto oblastí je zvětšena v **b)**, kde jsou již patrné jednotlivé pixely. Levá horní část oblasti je dále zvětšena do **c)**, kde je naznačen princip horizontálního převzorkování, a **d)**, kde je znázorněno vertikální převzorkování. Vektor $\mathbf{DC_0}$ je vykreslen zeleně, vektor **DR** modře a vektor **DDC** červeně.

pixelu oblasti $\mathbf{S_0}$ a velikost vektoru posunu v aktuálním řádku **DC** je upravena na velikost vektoru posunu v prvním řádku $\mathbf{DC_0}$. Následuje průchod prvním řádkem, který se skládá z nastavení výstupní proměnné **S** na pozici prvního pixelu v řádku **SoR** a vlastního průchodu, kdy je výstupní proměnná **S** po každé konvoluci inkrementována o vektor posunu v aktuálním řádku **DC**. Na konci řádku je inkrementována pozice prvního pixelu v řádku **SoR** přičtením vektoru posunu mezi řádky **DR** a vektor posunu v aktuálním řádku **DC** je inkrementován o „meziřádkový přírůstek“ **DDC**. Tím je dokončen průchod prvním řádkem a následuje obodobný průchod dalšími řádky.

Nutno podotknout, že veškeré koeficienty i proměnné se vyskytují dvojmo – pro horizontální a vertikální převzorkování. Proto je možné chápat je i jako uspořádanou dvojici $[x, y]$, jak je naznačeno v obrázku 5.1. Komponenty x a y jsou na sebe vždy kolmé.

Celý algoritmus se dá zapsat následujícím pseudokódem:

```
SoR = S0;
DC = DC0;
(foreach row in region)
{
    S = SoR;
    (foreach cell in row)
    {
        Konvoluce(S);
        S += DC;
    }
    SoR += DR;
    DC += DDC;
}
```

Jako konvoluční filtr se používá Lanczosův filtr, který je dobrou aproximací sinc filtru. Podle [14] je sinc filtr separabilní, lze tedy totéž prohlásit o Lanczosově filtru. Tento filtr však nebyl zvolen jen pro jeho separabilitu, ale také proto, že produkuje dobré výsledky při převzorkování se subpixelovými posuny, jak se píše v [18].

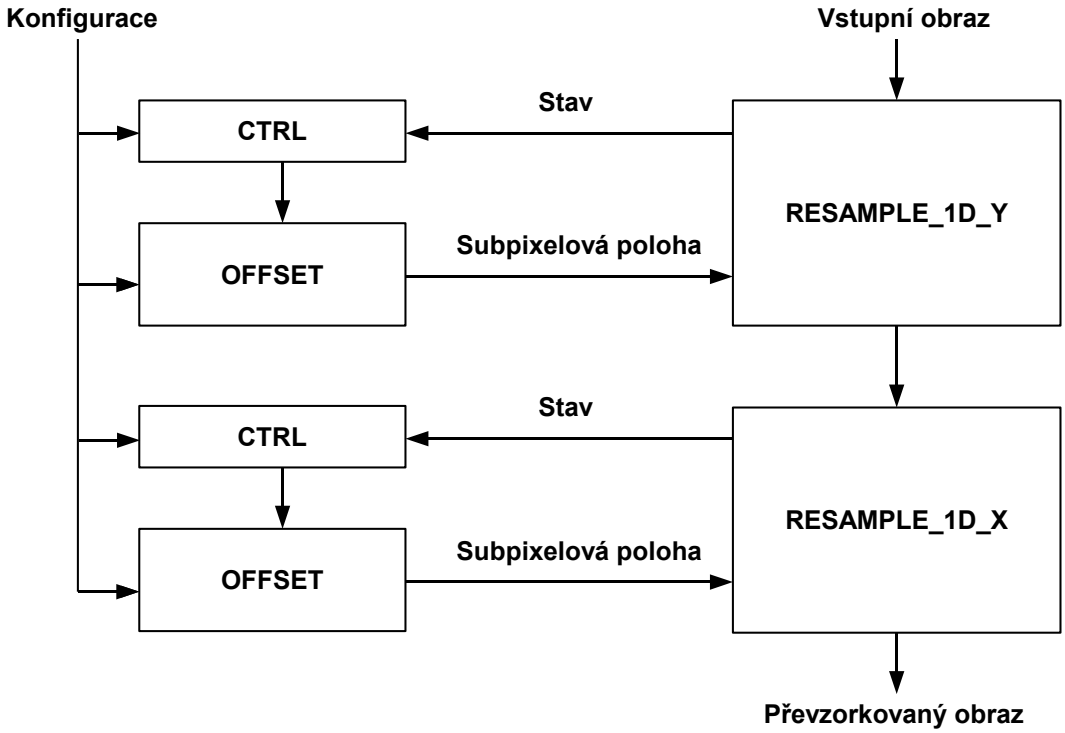
Filtr použitý v této práci má rozměry 1×7 pixelů. Pro horizontální převzorkování se používá „vodorovný“ filtr, pro vertikální převzorkování „svislý“ filtr. Pozice 4. (prostředního) pixelu odpovídá souřadnici **S** produkované algoritmem uvedeným výše. Celočíselná část souřadnice udává skutečnou pozici filtru ve zdrojovém obraze, desetinná část určuje subpixelovou polohu. Tato poloha slouží jako index do tabulky variant filtru, kdy každá varianta odpovídá určitému subpixelovému posunu.

Uvedený algoritmus dosahuje asymptotické složitosti sedmi operací násobení a sedmi operací sčítaní na pixel – konvoluce se skládá ze sedmi násobení koeficientů filtru s pixely obrazu a šesti sčítání takto vzniklých součinů, sedmá operace sčítání je generována inkrementací souřadnice **S**. V porovnání s klasickými algoritmy převzorkování se dosahuje úspory strojového času hlavně při výpočtu souřadnice filtru. Ve srovnání s algoritmy používajícími neseparabilní filtry pak dochází k významné úspoře při výpočtu konvoluce, kde se vyskytuje mnohem méně drahých operací násobení.

Algoritmus popsaný v této části práce byl pro testovací účely a ověření jeho vlastností implementován v jazyce C.

5.2 Hardwarová architektura

Blokové schéma převzorkovací jednotky je vidět na obrázku 5.2.

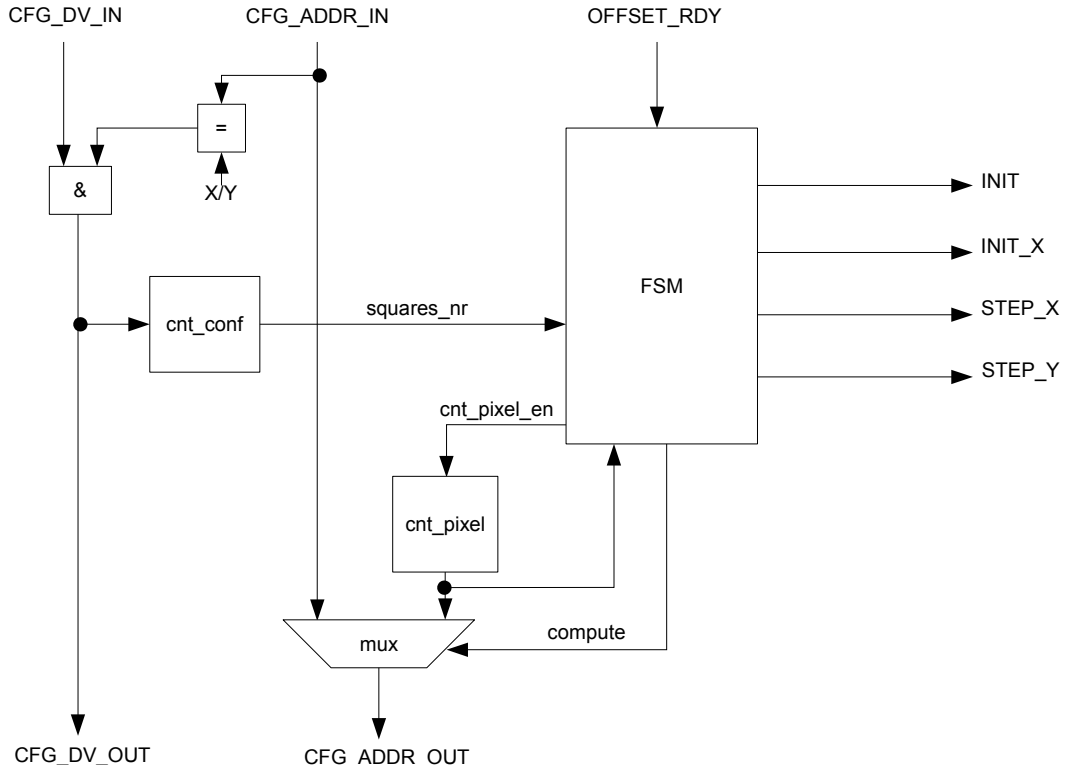


Obrázek 5.2: Bloková struktura převzorkovací jednotky.

Zařízení se skládá z šesti samostatných podjednotek, které jsou uspořádány do dvou trojic – jedna z nich se stará o vertikální převzorkování, druhá o horizontální. Každá trojice je složena z převzorkovacího modulu (dále jen moduly *RESAMPLE_1D_X* a *RESAMPLE_1D_Y*), kterým procházejí obrazová data a který provádí konvoluci, dále z modulu, který počítá aktuální subpixelovou polohu (dále jen modul *OFFSET*) a nakonec z řídícího modulu (dále jen *CTRL*), který na základě stavu převzorkovacího modulu řídí modul pro výpočet subpixelové polohy. Trojice jsou zapojeny do série, takže obě fáze převzorkování probíhají odděleně, avšak díky zřetězenému zpracování současně. Moduly *CTRL* a *OFFSET* sdílejí jedinou konfigurační sběrnici.

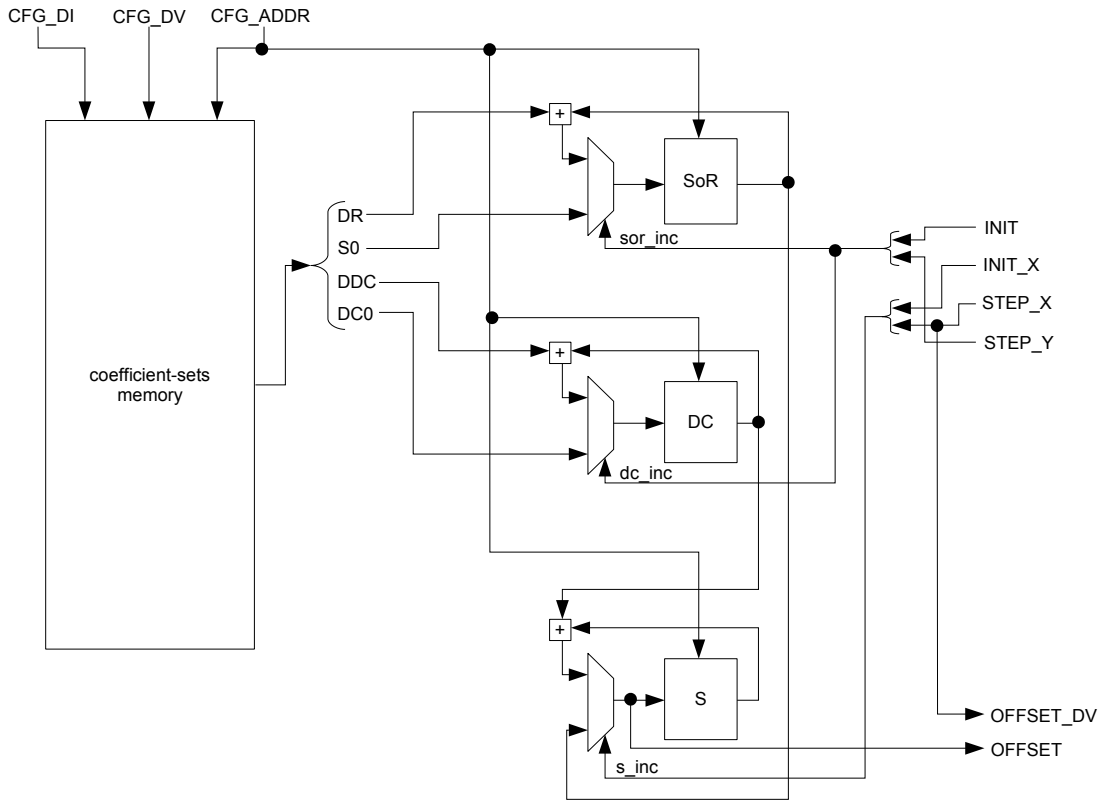
Dříve, než je jednotka připravena k použití, je nutno do ní nahrát konfigurační data. Tím se rozumí uložení čtveřic koeficientů uvedených v části 5.1 do modulů *OFFSET*. Konfigurační sběrnice obsahuje datové a adresové vodiče a moduly *CTRL* rozhodují o tom, zda se příslušná data uloží do přidruženého modulu *OFFSET*. Zároveň kontrolují počet nahraných sad koeficientů a po dokončení konfigurace spustí výpočet v modulech *OFFSET*. Jakmile je konfigurace dokončena, jednotka si zažádá o vstupní data. Po dostatečném naplnění vnitřního bufferu dá modul *RESAMPLE_1D_Y* modulu *CTRL* signál, že je připraven pracovat a tím začne vlastní převzorkování. Pokud je buffer stále dostatečně plný, generuje modul *OFFSET* v každém taktu jednu subpixelovou polohu konvolučního filtru a modul *RESAMPLE_1D_Y* produkuje v každém taktu jeden pixel vertikálně převzorkovaného ob-

razu. Tyto pixely putují do modulu horizontálního převzorkování *RESAMPLE_1D_X*, kde probíhá obdobná procedura včetně komunikace s příslušnými *CTRL* a *OFFSET* moduly. Výstup modulu *RESAMPLE_1D_X* je zároveň výstupem celé převzorkovací jednotky a objevují se na něm každý takt pixely výsledného zkorigovaného obrazu.



Obrázek 5.3: Blokové schéma modulu CTRL.

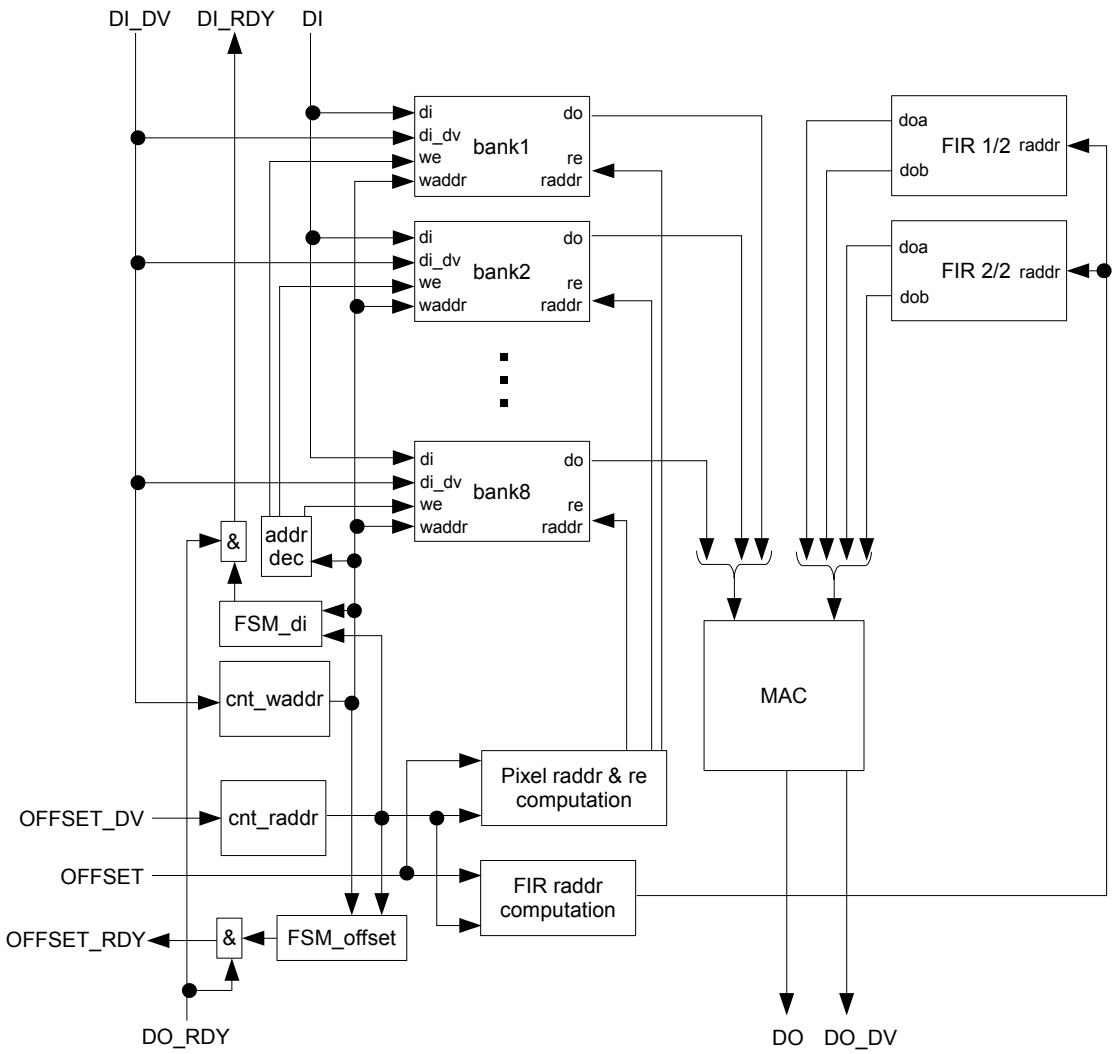
Na obrázku 5.3 je znázorněno blokové schéma kontrolního modulu *CTRL*. Jeho úkolem je řídit přidružený modul *OFFSET* (viz dále) podle aktuálního stavu převzorkování. Do modulu vstupuje adresová část konfigurační sběrnice *CFG_ADDR_IN*, jejíž nejvýznamnější bit určuje, zda jsou konfigurační data určena pro tu část jednotky, která provádí vertikální, respektive horizontální převzorkování. Pokud tento bit vyhovuje a zároveň je nastaven signál *CFG_DV_IN*, dojde k propagaci signálu platnosti dat na výstup *CFG_DV_OUT* a inrementaci čítače *cnt_conf*. Tento čítač uchovává počet nahraných koeficientových sad. Jádrem modulu *CTRL* je konečný stavový automat *FSM*, jehož hlavním úkolem je ovládat signály *INIT*, *INIT_X*, *STEP_X* a *STEP_Y*. Množina těchto signálů je do značné míry podobná množině stavů zmíněného automatu, takže udává, v jaké fázi se nachází interpolační algoritmus. Automat je ovládán signálem *OFFSET_RDY*, který je nastaven, pokud je přidružená převzorkovací jednotka připravena k výpočtu. Druhým vstupem automatu je počet nahraných koeficientových sad reprezentovaný signálem *squares_nr*. Automat pomocí signálu *cnt_pixel_en* řídí čítač *cnt_pixel*, který po skončení konfigurační fáze také slouží jako vstup automatu a generuje mj. číslo právě zpracovávané oblasti obrazu. O multiplexování tohoto čísla a konfigurační adresy se pomocí signálu *compute* rovněž stará automat.



Obrázek 5.4: Blokové schéma modulu OFFSET.

Další obrázek 5.4 obsahuje blokové schéma modulu *OFFSET*. Tento modul sám o sobě není příliš „inteligentní“, protože jeho logika je řízena spřaženým modulem *CTRL*. Přímo do něj vstupuje datová část konfigurační sběrnice *CFG_DI*, naopak její adresová část *CFG_ADDR* a signál platnosti dat *CFG_DV* jsou vyvedeny z řídícího modulu *CTRL*, který tyto signály generuje. Vyjmenované vodiče sběrnice jsou zapojeny do paměti pro uložení čtveřic koeficientů (viz 5.1) *coefficient-sets memory*. Jedna buňka této paměti obsahuje všechny čtyři koeficienty; ty jsou po vyčtení přivedeny na stejnojmenné vodiče *DR*, *S0*, *DDC* a *DC0*.

Druhou částí modulu *OFFSET* je sada pamětí a multiplexorů, kde se realizuje interpolační algoritmus. Průběh algoritmu je řízen z modulu *CTRL* pomocí signálů *INIT*, *INIT_X*, *STEP_X* a *STEP_Y*. Každá z pamětí *SoR*, *DC* a *S* reprezentuje jednu z proměnných algoritmu, jak je popsáno v části 5.1. V tomto případě bylo vhodné použít paměti místo registrů, protože v průběhu převzorkování dochází k přepínání mezi různými oblastmi obrazu a tím i mezi různými sadami koeficientů. Toho je dosaženo adresováním všech pamětí modulu adresovou částí konfigurační sběrnice *CFG_ADDR*. S proměnnými jsou prováděny operace přiřazení a inkrementace, což vede k zapojení multiplexoru před každou paměť proměnné. Příslušné multiplexory jsou rovněž řízeny čtveřicí výše zmíněných signálů z modulu *CTRL*. Výstupem modulu je stejnojmenný signál *OFFSET*, což je vlastně hodnota proměnné *S* interpolačního algoritmu udávající polohu aktuálně zpracovávaného pixelu.



Obrázek 5.5: Blokové schéma modulu RESAMPLE_1D_Y.

Na obrázku 5.5 může čtenář vidět blokové schéma vertikálního převzorkovacího modulu *RESAMPLE_1D_Y*. V něm se nachází největší RAM paměť jednotky – buffer pro uložení vstupních dat. Tento buffer pojme část obrazu o velikosti 256×64 pixelů, což odpovídá čtyřem čtvercovým oblastem obrazu sousedícím horizontálně. Protože konvoluční filtr má sedm koeficientů, je nutno mít možnost naráz přistoupit alespoň k sedmi pixelům obrazu uspořádaným do sloupce. Kvůli snadnější adresaci paměťových bloků[6] bylo toto číslo zvýšeno na nejbližší vyšší mocninu dvou, tedy osm. V bloku č. 1 je uložen 1., 9., 17., 25., 33., 41., 49. a 57. řádek obrazu, v bloku č. 2 je uložen 2., 10. až 58. řádek obrazu, až konečně v bloku č. 8 je uložen 8., 16. až 64. řádek. Toto rozptýlení sousedních řádků obrazu mezi jednotlivé bloky zaručuje možnost vyčtení vždy osmi sousedních pixelů v jednom sloupci za jeden takt kdykoliv během převzorkování.

Buffer potřebuje pro svou práci poměrně velké množství řídící logiky. Ta se na zápisové

[6] Anglicky „memory bank“, v češtině se někdy hovorově používá výraz „paměťový bank“.

straně skládá z čítače zápisové adresy *cnt_waddr* a adresového dekodéru *addr_dec*, který pomocí vektoru zápisových signálů *we* určuje, do kterého bloku se budou vstupní data zapisovat. Na čtecí straně bufferu je logika složitější; opět nalezneme čítač (čtecí) pseudoadresy *cnt_raddr*, ale místo adresového dekodéru je přítomen obvod *Pixel raddr & re computation*. Ten ze zmíněné pseudoadresy počítá skutečnou čtecí adresu *raddr* a čtecí signál *re* pro každý paměťový blok bufferu.

Pro uložení koeficientů FIR filtru[7] jsou použity dva bloky RAM paměti, jejichž obsah je určen už při nahrání designu FPGA. Šířka slova těchto pamětí je dvojnásobná oproti šířce slova vstupního bufferu a protože není potřeba do pamětí nic zapisovat, mohou být oba porty každého bloku využity pro čtení. Tímto způsobem je rovněž možné vyčíst všech sedm koeficientů FIR filtru v jednom taktu. Paměťové bloky jsou adresovány signálem generovaným v obvodu *FIR raddr computation*, který jej počítá ze čtecí pseudoadresy vstupního bufferu a polohy aktuálně zpracovávaného pixelu *OFFSET*.

Pixely vyčtené ze vstupního bufferu je nutno zkonvoluovat s koeficienty filtru. To probíhá v *MAC* jednotce, která se skládá z pole násobiček a sčítačkového stromu. Tato jednotka je navržena s ohledem na maximální propustnost, proto jsou násobičky implementovány pomocí specializovaných primitiv FPGA čipu, která jsou schopna provést násobení v jednom kroku. Sčítačkový strom je binární, to znamená, že výsledek je při osmi sčítancích k dispozici se zpožděním tři takty. Zřetězené zpracování ovšem umožňuje každý takt produkovat jeden výsledek MAC operace.

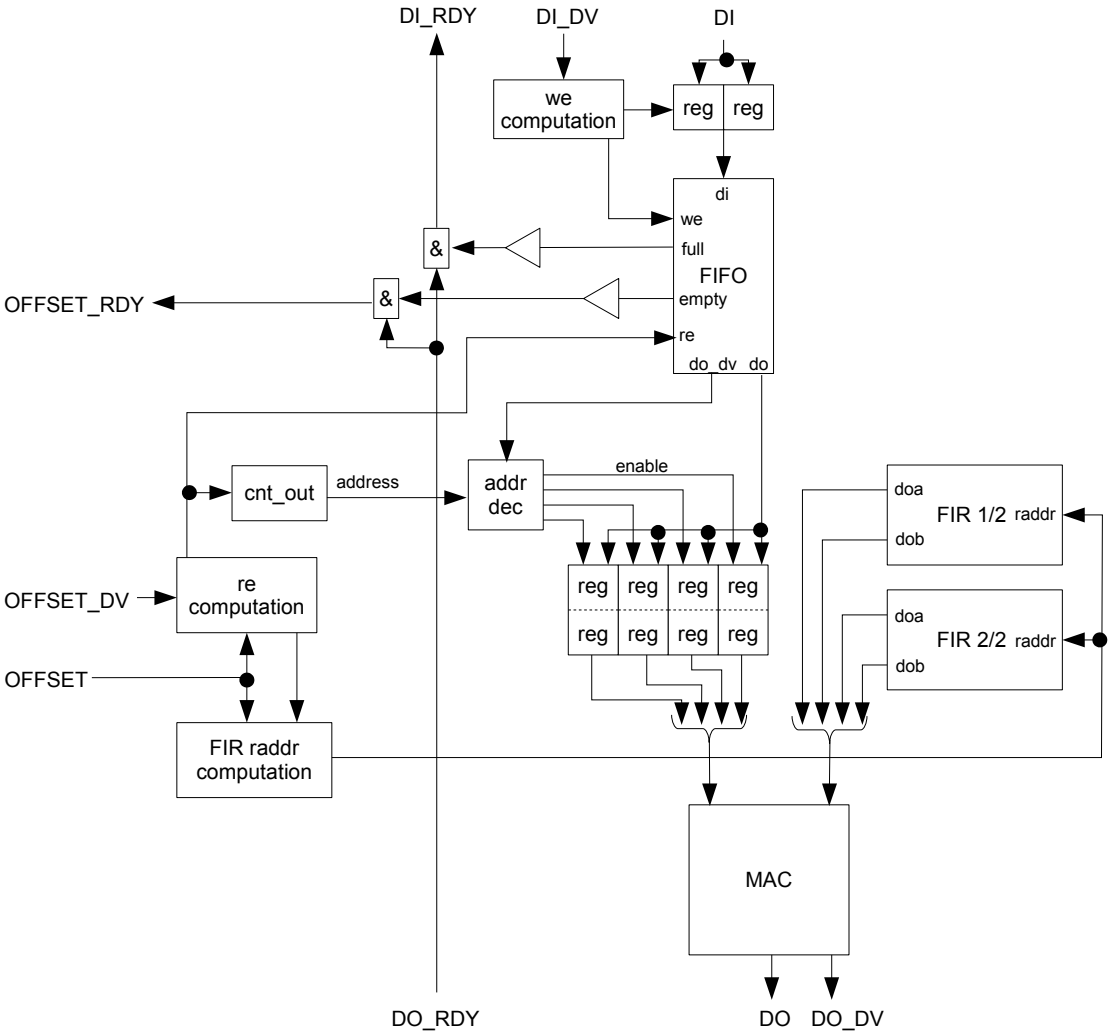
Modul *REAMPLE_1D_Y* je vybaven dvěma konečnými stavovými automaty pro generování RDY[8] signálů. Oba automaty mají tři vstupy: Signál *DO_RDY*, který je nastaven, pokud je vnější prostředí modulu připraveno přebírat výsledky převzorkování a dále čtecí a zápisovou adresu vstupního bufferu. Z těchto tří vstupů generuje automat *FSM_di* výstupní signál *DI_RDY* (aktivní, pokud je modul schopen přijímat data) a automat *FSM_offset* signál *OFFSET_RDY* (aktivní, pokud je modul schopen přijmout polohu aktuálně zpracovávaného pixelu *OFFSET*).

Zbývající modul *RESAMPLE_1D_X* je schematicky znázorněn na obrázku 5.6. Jak je možno si všimnout, jeho rozhraní je identické s modulem *RESAMPLE_1D_Y*, shodné jsou navíc i některé prvky vnitřní struktury. Je to způsobeno tím, že oba moduly vykonávají velmi podobnou činnost, dá se tedy říci, že odlišný je jen způsob uložení vstupních dat.

Jelikož tento modul provádí horizontální převzorkování, tj. používá horizontálně orientovaný konvoluční filtr, stačí mu, aby měl přístup jen k aktuálně zpracovávanému řádku obrazu. Navíc pro výpočet nepotřebuje celý řádek, ale jen těch sedm pixelů, které jsou ke konvoluci potřebné. Z tohoto důvodu je zbytečné implementovat objemné buffery, místo toho stačí příchozí data ukládat do fronty *FIFO*. Problém je však v tom, že při výpočtu dvou po sobě jdoucích pixelů výstupního obrazu se může FIR filtr ve zdrojovém obraze posunout až o dva pixely. To znamená, že v některých případech je nutné z fronty vyčíst dva pixely v jednom taktu. Klasická fronta typu FIFO umožňuje vyčítat právě jednu položku, řešení tedy spočívá v ukládání dvou pixelů do jedné položky fronty. Je tedy nutné implementovat logiku, která zabezpečí, aby se do fronty zapsaly vždy dva příchozí pixely naráz. Proto je na vstupní signál *DI* napojena dvojice registrů, za jejichž střídavé plnění je zodpovědný obvod *we computation*. Jakmile je dvojice registrů naplněna platnými daty, obvod generuje *we* signál pro zápis dvojice pixelů do fronty.

Logika pro obsluhu vyčítání z fronty je o poznání složitější. Je schematicky znázorněna obvodem *re computation*, který na základě aktuální a předešlé hodnoty signálu *OFFSET*

[7] Z anglického „Finite Impulse Response“ filter – filtr s konečnou impulsní odezvou.
[8] Často používaná zkratka anglického „ready“ – připraven.



Obrázek 5.6: Blokové schéma modulu RESAMPLE_1D_X.

(tzv. relativního offsetu) určí, zda nastává posun o žádný, jeden či dva pixely. Pokud nedochází k žádnému posunu, z fronty se nevyčítá. Pokud dochází k posunu o dva pixely, z fronty se vyčte právě jedna položka (dva pixely). Konečně, pokud dochází k posunu jen o jeden pixel, vyčítá se z fronty jen v tom případě, pokud při poslední takové situaci k vyčtení nedošlo. Obvod *re computation* také řídí činnost čítače *cnt_ out*, který adresuje soustavu registrů, do nichž jsou ukládána data vyčítaná z fronty. Soustava registrů funguje jako kruhový buffer, jehož zápisové signály jsou kontrolovány adresovým dekodérem *addr dec*.

Zbylé části modulu *RESAMPLE_ 1D_ X* jsou, jak již bylo řečeno, víceméně shodné s modulem *RESAMPLE_ 1D_ Y*. Jedná se o dva bloky paměti s uloženými koeficienty FIR filtru a *MAC* jednotku, která provádí jejich konvoluci s pixely obrazu. O adresování paměti koeficientů se stará obvod *FIR raddr computation*. Ten adresu počítá ze sledu hodnot vstupního signálu *OFFSET*.

5.3 Dosažené výsledky

Navržená architektura převzorkovací jednotky byla implementována v jazyce VHDL. Dále byla provedena syntéza do programovatelného hradlového pole Xilinx Virtex-II XC2V250, kterým je osazena akcelerační karta DX64 firmy Camea. Tak byly získány důležité parametry, na základě kterých se dají posoudit možnosti převzorkovací jednotky. Jedním z nich je maximální pracovní frekvence, která byla syntetizačním nástrojem Xilinx XST J.30 určena na 147,384 MHz, což odpovídá periodě 6,785 ns[9]. To je dobrý výsledek vzhledem k tomu, že se uvažuje o skutečné pracovní frekvenci 100 MHz. Důležitým parametrem je také zabraná plocha na čipu, která se určuje z počtu zabraných slice[10] bloků. Podle výsledků syntézy jednotka zabírá 62 % plochy FPGA, což je relativně mnoho, ale je nutno si uvědomit, že použitý čip je již poměrně starý a hlavně malý. Dnes vyráběné čipy disponují násobně vyšším počtem obvodových elementů. Tabulka 5.1 přesněji udává využití jednotlivých zdrojů na čipu.

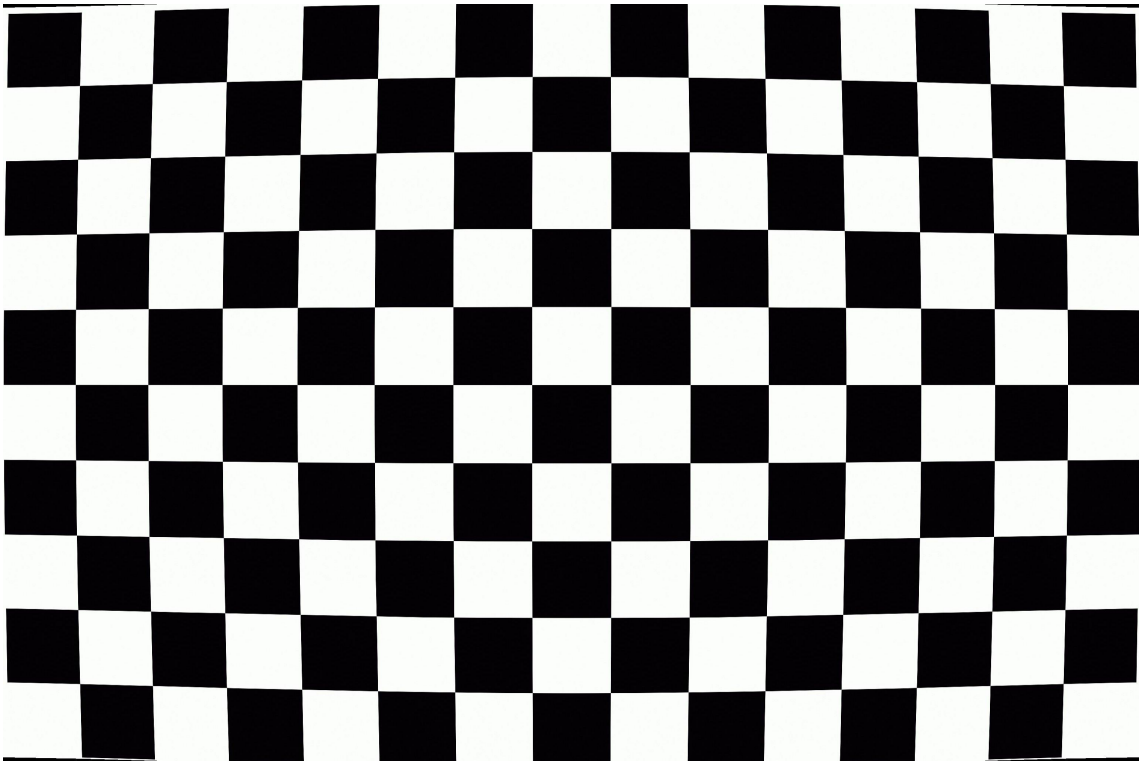
Prostředky na čipu	Zabráno	Celková kapacita	% celkové kapacity
Slice bloky	954	1536	62 %
Klopné obvody	990	3072	32 %
4-vstupé LUT	1410	3072	45 %
Block RAM	21	24	87 %
18-bitové násobičky	16	24	66 %

Tabulka 5.1: Výsledky syntézy převzorkovací jednotky pro hradlové pole Virtex-II XC2V250, rychlostní stupeň -6.

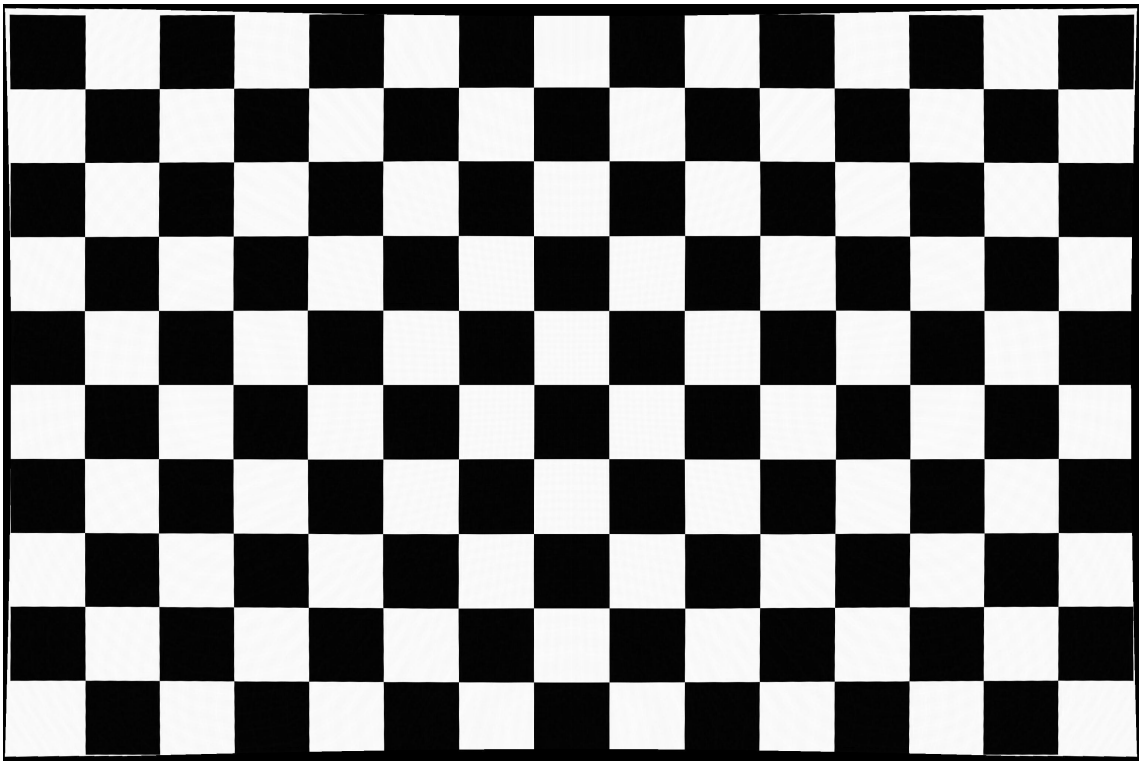
Fáze odladění jednotky v hardwaru ještě neskončila, avšak jsou k dispozici výsledky z referenčního softwaru. Tento software byl implementován v jazyce C za účelem simulace a ověření vlastností navrženého interpolačního algoritmu. Na obrázku 5.7 může čtenář vidět testovací obrazec s uměle vytvořeným poduškovitým zkreslením, které odpovídá zkreslení objektivu fotoaparátu Canon EOS 300D s ohniskovou vzdáleností 18-55 mm. Na dalším obrázku 5.8 je tentýž obrazec po převzorkování a korekci. Oba obrazce mají skutečnou velikost 3072×2048 pixelů, rohové body originálu jsou vychýleny o 16 px v horizontálním a o 26 px ve vertikálním směru, dochází tedy až k 2,5% posunu pixelů. Testovací obrazec záměrně obsahuje ostré hrany a jednobarevné plochy, aby se projevily případné nedostatky převzorkovacího algoritmu či konvolučního filtru. Jak je patrné, všechny hrany zůstaly nerozmazány a v plochách se nevyskytují žádné artefakty.

[9] Při použití čipu XC5VLX30 z nejnovější rodiny Xilinx Virtex 5 byla maximální frekvence stanovena na 239,549 MHz, tomu odpovídá perioda 4,175 ns.

[10] Tzv. „slice“ je elementárním prvkem FPGA čipu skládajícím se z LUT, registru a podpůrné logiky.



Obrázek 5.7: Testovací obrázek s poduškovitým zkreslením.



Obrázek 5.8: Zkorigovaný testovací obrázek.

Kapitola 6

Závěr

Cílem této práce bylo vybrat vhodný algoritmus pro zrpacování v FPGA čipu, navrhnout způsob jeho implementace a implementovat jej. Byl vybrán algoritmus separabilního převzorkování obrazu, který byl drobně optimalizován pro účely implementace v FPGA. Byla navržena architektura hardwarové převzorkovací jednotky a tato jednotka byla v jazyce VHDL implementována.

Klíčem k proniknutí do problematiky programovatelného hardwaru je znalost cílové FPGA platformy. Pro seznámení s její architekturou, vlastnostmi a komponentami dostupnými na čipu bylo použito [29]. Rovněž je důležitá schopnost tvorby efektivně syntetizovatelného VHDL kódu, a proto byl nastudován [1] a další materiály zmíněné v kapitole 3. Základní techniky zpracování obrazu, zejména převzorkování, byly nastudovány v průběhu implementace referenčního softwaru, návrhu architektury převzorkovací jednotky a během tvorby kapitoly 2. Možné alternativy v návrhu jednotky, které jsou diskutovány v kapitole 4, vycházejí z [2], [21] a [26]. Finální architektura převzorkovací jednotky je detailně popsána v kapitole 5. Jednotka byla implementována v jazyce VHDL, její zdrojové kódy jsou k dispozici v dodatku A a výsledky syntézy jsou diskutovány v kapitole 5, části 5.3. Nad rámec zadání práce byl ve spolupráci s Ing. Michalem Seemanem sepsán a publikován článek o algoritmu přesného separabilního převzorkování [18] a byla vytvořena referenční softwarová implementace převzorkovacího algoritmu v jazyce C.

Účelem práce bylo akcelerovat algoritmus převzorkování, což se podařilo. Jednotka pracující na frekvenci 100 MHz je schopna obrázek o rozlišení 2 Mpx zpracovat zhruba za 20 ms. To je pro člověka téměř nepostřehnutelná doba, takže by jednotka v reálném čase mohla zpracovávat HD video signál se snímkovou frekvencí až do 50 Hz. Při srovnání s převzorkováním běžícím na dnes běžném 2 GHz procesoru, kde se dosahuje výkonu řádově několik Mpx za sekundu, je řešení popisované v této práci až 100× rychlejší.

Jednotka bude po odladění v hardwaru prakticky nasazena v rámci projektu Biomarker na převzorkování biomedicínského obrazu. Její výkon bude pro tento účel dostatečný, avšak v budoucnu by bylo možné jednoduše zvýšit její pracovní frekvenci použitím modernějšího FPGA čipu z rodiny Xilinx Virtex 5. Při použití čipu s větším množstvím logiky a vestavěné paměti by mohlo být rentabilní umístit do něj více paralelně pracujících jednotek. Z hlediska zpracování biomedicínského obrazu se také nabízí možnost rozšíření jednotky pro účely převzorkování 3D obrazů.

Literatura

[1] ASHENDEN, P. J.: *The VHDL Cookbook: First Edition*. Dept. Computer Science, University of Adelaide, South Australia, červenec 1990.
URL <http://tams-www.informatik.uni-hamburg.de/vhdl/doc/cookbook/VHDL-Cookbook.pdf>

[2] BENEDETTI, A.; PRATI, A.; SCARABOTTOLO, N.: Image Convolution on FPGAs: the Implementation of a Multi-FPGA FIFO Structure. In *24th. EUROMICRO Conference (EUROMICRO'98)*, August 1998.

[3] BOCKAERT, V.: Interpolation. 2008, [online], [cit. 2008-04-27].
URL <http://www.dpreview.com/learn/?/key=interpolation>

[4] BURNETT, C. M.: Aliasing. [online], [cit. 2008-04-17].
URL <http://en.wikipedia.org/wiki/Aliasing>

[5] van DAM, A.; GOULD, D. L.: Nyquist Limit Guide. [online], [cit. 2008-04-27].
URL <http://www.cs.brown.edu/exploratories/freeSoftware/repository/edu/brown/cs/exploratories/applets/nyquist/nyquist_limit_guide.html>

[6] DERSCH, H.: Testing Interpolator Quality. Červen 1999, [online], [cit. 2008-04-20].
URL <http://www.all-in-one.ee/~dersch/interpolator/interpolator.html>

[7] FORSYTH, D. A.; PONCE, J.: *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002, ISBN 0130851981.

[8] HRÁDEK, J.: Aliasing & Antialiasing. [online], [cit. 2008-04-27].
URL <http://herakles.zcu.cz/education/apg_2002_2003/hradek/html/Aliasing.html>

[9] KABÁT, Z.: Anti-aliasing – Supersampling a Multisampling. *Svět hardware*, prosinec 2003, [online], [cit. 2008-04-27].
URL <http://www.svethardware.cz/art_doc-D68616962C4A3879C1256DD70075C74C.html>

[10] KATZ, R.: Programmable Logic Device Definitions. Leden 2002, [online], [cit. 2008-04-27].
URL <http://klabs.org/richcontent/Tutorial/PLD_Definitions.htm>

[11] KOZEL, L.: Pravda o čočkách a objektivech. Říjen 2006, [online], [cit. 2008-04-16].
URL <http://digiarena.zive.cz/default.aspx?section=31&server=1&article=3722&chapter=49325>

[12] KRŠEK, P.: Základy počítačové grafiky: Antialiasing. 2007, [online], [cit. 2008-04-18].
URL <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IZG-IT/lectures/izg_antialias.pdf>

[13] KURC, T.; KUŽELKA, J.: Windowed-sinc filtr. [online], [cit. 2008-04-19].
URL <http://tvorbawebu.wz.cz/faiirfe/help/windowed-sinc.htm>

[14] LI, A.; Mueller, K.; Ernst, T.: Methods for Efficient, High Quality Volume Resampling in the Frequency Domain. In *IEEE Visualisation 2004*, Austin, Texas, USA, říjen 2004, ISBN 0-7803-8788-0.
URL <http://vis.computer.org/vis2004/DVD/vis/papers/li.pdf>

[15] MCHUGH, S.: Understanding Digital Image Interpolation. [online], [cit. 2008-04-19].
URL <http://www.cambridgeincolour.com/tutorials/image-interpolation.htm>

[16] MUDROVÁ, M.: Geometrické transformace obrazu a související témata. 2004, [online], [cit. 2008-04-19].
URL <http://uprt.vscht.cz/ucebnice/ZOB/prednasky/09-TRANSFORMACE/Transformace.pdf>

[17] PECH, J.: Programovatelné logické obvody. [online], [cit. 2008-04-21].
URL <http://sweb.cz/fpga/>

[18] PŘIBYL, B.; SEEMAN, M.: Precise Image Resampling for Optics Geometry Correction. In *Digital Technologies 2007*, Faculty of Electrical Engineering of Zilina Univerzity, 2007.
URL <http://www.fit.vutbr.cz/research/view_pub.php?id=8534>

[19] QUINNELL, R.: Designing Digital Filters. Leden 2003, [online], [cit. 2008-04-27].
URL <http://www.dspdesignline.com/showArticle.jhtml?articleID=192200504>

[20] SMEČKA, Z.: Oční vady – krátkozrakost, dalekozrakost, astigmatismus, presbyopie. [online], [cit. 2008-04-16].
URL <http://www.klinikazlin.cz/index.php?ACT=ocni-vady>

[21] SRIRAM, V.; KEARNEY, D.: A FPGA implementation of variable kernel convolution. In *Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2007, s. 105–109.

[22] THEVENAZ, P.; BLU, T.; UNSER, M.: Image interpolation and resampling. *Handbook of Medical Imaging, Processing and Analysis*, 2000: s. 393–420.
URL <http://bigwww.epfl.ch/publications/thevenaz9901.pdf>

[23] VINA, J.: Aliasing Artifacts. 2007, [online], [cit. 2008-04-18].
URL <http://support.svi.nl/wiki/AliasingArtifacts>

[24] VINA, J.: Spherical aberration. 2007, [online], [cit. 2008-04-16].
URL <http://support.svi.nl/wiki/SphericalAberration>

[25] WEISSTEIN, E. W.: Sampling Theorem. [online], [cit. 2008-04-27].
URL <http://mathworld.wolfram.com/SamplingTheorem.html>

[26] WIATR, K.; JAMRO, E.: Implementation Image Data Convolutions Operations in FPGA Reconfigurable Structures for Real-Time Vision Systems. In *The International Conference on Information Technology: Coding and Computing (ITCC'00)*, březen 2000.

[27] *AT94KAL Series FPSLIC*. Leden 2008, [online], [cit. 2008-04-28].
URL <http://www.atmel.com/dyn/resources/prod_documents/doc1138.pdf>

[28] Xilinx: Our History. [online], [cit. 2008-04-28].
URL <http://www.xilinx.com/company/history.htm>

[29] *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet*. Listopad 2007, [online], [cit. 2008-04-28].
URL <http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf>

[30] *Virtex-4 FPGA Data Sheet: DC and Switching Characteristics*. Duben 2008, [online], [cit. 2008-04-28].
URL <http://www.xilinx.com/support/documentation/data_sheets/ds302.pdf>

[31] Field-programmable gate array. [online], [cit. 2008-04-28].
URL <http://en.wikipedia.org/wiki/Field-programmable_gate_array>

[32] Sinc function. [online], [cit. 2008-04-19].
URL <http://en.wikipedia.org/wiki/Sinc_function>

[33] Supersampling. [online], [cit. 2008-04-18].
URL <http://en.wikipedia.org/wiki/Supersampling>

[34] Image:Fpga xilinx spartan.jpg. Únor 2005, [online], [cit. 2008-04-28].
URL <http://commons.wikimedia.org/wiki/Image:Fpga_xilinx_spartan.jpg>

[35] ČISTOTOVÁ, T.: Optické vady druhá část. Srpen 2007, [online], [cit. 2008-04-16].
URL <http://clanky.katalogfotoaparatu.cz/technologie-digitalni-fotografie/1632-opticke-vady-druha-cast/>

[36] ČISTOTOVÁ, T.: Optické vady první část. Červenec 2007, [online], [cit. 2008-04-16].
URL <http://clanky.katalogfotoaparatu.cz/technologie-digitalni-fotografie/1525-opticke-vady-prvni-cast/>

Dodatek A

CD nosič