České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídicí techniky

BAKALÁŘSKÁ PRÁCE

Zpracování obrazů z více zdrojů pomocí FPGA

Praha, 2007 Autor: Miroslav Dušek

Prohlášení

Prohlašuji, že jsem svou diplomovou (bakalářskou) práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne _____ _____

podpis

Poděkování

Děkuji především vedoucímu bakalářské práce za vedení, podporu a cenné rady v průběhu zpracování této práce.

Abstrakt

Dušek,M. Zpracování obrazů z více zdrojů pomocí FPGA.

Bakalářská práce, Praha 2007

Práce se zabývá rozborem a návrhem systému pro současné zpracování obrazů z více zdrojů pomocí FPGA. Cílem je vytvořit ukázkové algoritmy demonstrující funkčnost tohoto systému. V úvodní části práce je představen obvod FPGA řady Spartan-3, který je základem implementační platformy RC10 a modul kamery. V hlavní části práce je rozebrána vlastní implementace, rozdělení systému na dvě hlavní části a bloková schemata ilustrující jednotlivé části. V poslední části je zhodnocena realizace celého systému a je navržen směr, kterým by se měla práce dále ubírat.

Abstract

Dušek, M. Video Processing of more sources using FPGA

Bachelor thesis, Prague 2007

This text discusses design and realization of a real time video processing of two sources. The goal are sample algorithms, which show functionality of this system. At the beginning of this bachelor thesis are introduced the Field Programmable Gate Arrays (FPGA) Spartan-3, as the main implementation platform, together with Handel-C which is used for description and definition of designed system and module of camera. The main part is aimed to describe the implementation, partitioning of the whole system into two main parts and their own functionality. There one can find many useful block diagrams and tables, which helps him to understand functionality of each part. The realization itself is reviewed in the ending of this thesis, together with ideas for further development of the designed system

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Seznam použitých zkratek

ABEL	Advanced Boolean Equation Language
ASIC	Aplication Specific Integrated Circuit
IOB	Input Output Block
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
RGB	Red, Green, Blue
FPGA	Field Programmable Gate Array
HW	Hardware
GRB	Green, Red, Blue
YUV	barevný model, kde Y je jasová složka a U a V jsou barevné složky
LB	Logic Block
CLB	Configurable Logic Block
LE	Logic Element
CMOS	Complementary Metal Oxide Semiconductor
LED	light-emitting diode
EDIF	Electronic Design Interchange Format
DSP	Digital Signal Processor
DCM	Digital Clock Manager

Kapitola 2

Úvod

V dnešní době je zpracování obrazů z kamer nezbytnou součástí mnoha technických odvětví, z důvodu využívání kamer jako senzorů. To klade na zpracování obrazů veliké nároky. Kamerou získáme snímek, na kterém mimo chtěných dat získáme i množství dat pro danou aplikaci nadbytečných. Z tohoto důvodu potřebujeme obvody takové, aby zajistili dostatečně rychlé vyhodnocení dané situace zachycené snímkem. To nám přinášejí právě obvody FPGA, které obsahují milióny logických hradel schopných pracovat paralelně. Tato robustnost se využívá všude, kde se očekává velký objem dat určený pro zpracování, jako například automobilový průmysl, kosmické aplikace, digitální audio a video technika.

Algoritmy pro zpracování obrazů a videa se tak mnohonásobně akcelerují a zefektivní a je možné díky nim zpracovávat obraz či video v reálném čase. Současné zpracování obrazů z více zdrojů, o čemž pojednává tato práce, je díky těmto obvodům reálné a dokazuje jejich vyjímečnost. To přináší sebou obrovské uplatnění v průmyslu, především v robotice. Důležitou aplikací v robotice, kde se využívá zpracování obrazů z více zdrojů je stereovidění. To se snaží napodobit lidský zrakový systém, jenž místo očí používají dva snímače.

Tato práce se zaměřuje na ukázání možnosti použít FPGA pro zpracování obrazů z více zdrojů a pro ukázku demonstrovat jednoduché algoritmy.

V úvodní části se čtenář seznámí s vývojovou deskou RC10, která byla pro tuto aplikaci vybrána, následně pak s FPGA, který je základem této desky a nakonec s modulem kamery. V následující kapitole jsou popsány možnosti návrhu aplikací pro FPGA s detailnějším popisem vybraného návrhového jazyka.

V hlavní části je práce zaměřená na popis realizace dané aplikace. Bude objasněn postup realizace, při němž se práce rozdělila do dvou větších celků. U každého celku je nástin implementace a popis jeho chování. Poté jsou tyto bloky spojeny ve

výslednou aplikaci s přehledem algoritmů, demonstrujících její funkčnost.

Na závěr je práce zhodnocena a jsou naznačeny možné směry dalšího vývoje.

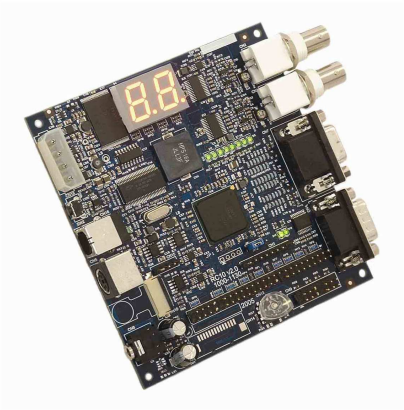
Kapitola 3

Vybranná platforma

3.1 Přípravek

Pro realizaci aplikace byla vybrána vývojová deska RC10 firmy Celoxica s FPGA Spartan3 3S1500L-4 firmy Xilinx. Tato deska svými periferiemi pro danou aplikaci postačuje. Jediný nedostatek, který RC10 bohužel má, je nepřítomnost externí rychlé paměti, což způsobovalo při návrhu aplikace potíže.

Tato deska obsahuje takzvané "LowPower" součástky, a proto ji můžeme napájet pouze z portu USB. Mezi ně patří 48MHz krystal, 16MB flash paměť, konektor pro CMOS kameru, 18bit VGA výstup, 33 I/O pinů, 3 vyvedená napětí (+12V, +5V, +3.3V), jenž první dvě jsou k dispozici pouze s externím zdrojem, 2 hodinové vstupy, LED diody,2 sedmisegmentové displeje USB 2.0, sběrnici CAN, RS232 a PS/2 port, 5-cestný joystik, JTAG connector, 2 ADC kanály, sériový Audio výstup (1 bit DAC) a Piezo sounder. Jak lze vidět na následujícím obrázku 3.1.

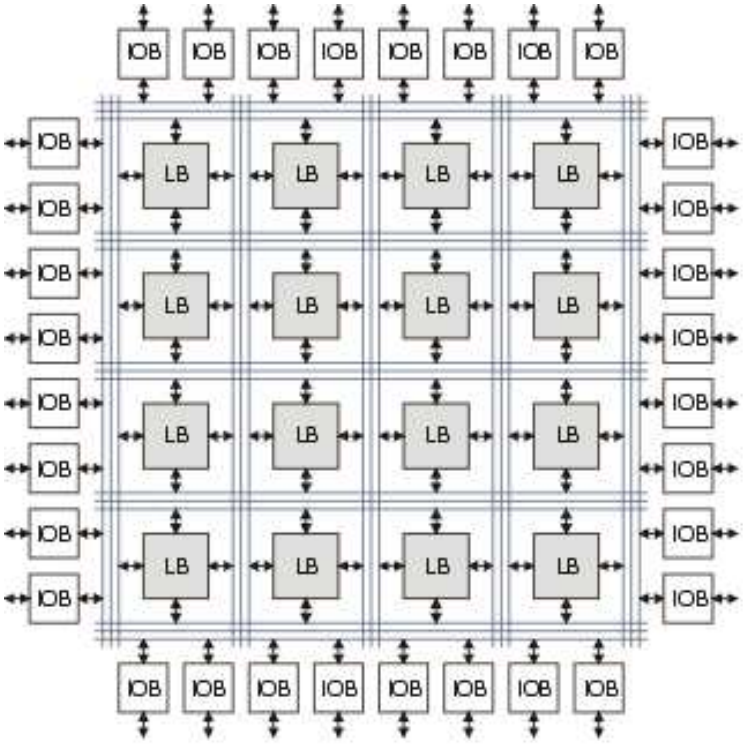


Obrázek 3.1: RC10

Pro splnění zadání budou potřeba dvě desky RC10, znichž každá je vybavena modulem kamery, který bude detailněji popsán dále.

3.2 FPGA - Spartan 3S1500L-4

Field Programmable Gate Array - samostatná třída aplikačně specifických IO, která umožňuje realizovat digitální systém v jediném IO nastavením propojení mezi jednotlivými logickými bloky. Nastavení propojení je možné mnohonásobně krát měnit a je provedeno řádově ve stovkách milisekund až sekundách.



Obrázek 3.2: Struktura FPGA

Základem každého FPGA je

- Matice Logických Bloků - Tyto bloky provádí jednoduché logické funkce. Složitější funkce se realizují ve více propojených LB(Logický Blok). Označení u těchto elementů se liší podle výrobce. Například Xilinx je označuje CLB(Configurable Logic Block), Altera jako LE(Logic Element).
- Vstupně/výstupní bloky(IOB) – komunikace s okolím

- Propojovací matice – Slouží k propojení LB mezi sebou a k propojení mezi LB a IOB

Dnešní FPGA nabízejí více než tento základ. Ať už se jedná o prvky zrychlující činnost obvodu nebo rozšiřující jeho funkčnost. Mohou to být například :

- Bloky interní paměti
- Bloky provádějící DSP operace
- Optimalizované propojení LB pro často používané funkce
- IOB podporujícím různé standardy.

V našem případě je použito FPGA 3S1500L-4 od firmy Xilinx ze série Spartan-3. Tato řada obsahuje 8 členů lišící se počtem hradel a kapacitou paměti. Základní údaje o 3S1500L-4 lze vyčíst v následující tabulce. Podrobnosti lze nalézt v [5], nebo obecné informace o FPGA v [10].

System Gates	Equivalent Logic Cells	Distributed RAM	Block RAM
1.5M	29,952	208K	576K
Dedicated Multipliers	DCMs	Max. User IO	Max. IO Pairs
32	4	487	221

Tabulka 3.1: Parametry FPGA Spartan 3S1500L-4

3.3 Modul kamery

Obrázek 3.3: Modul kamery 20J6

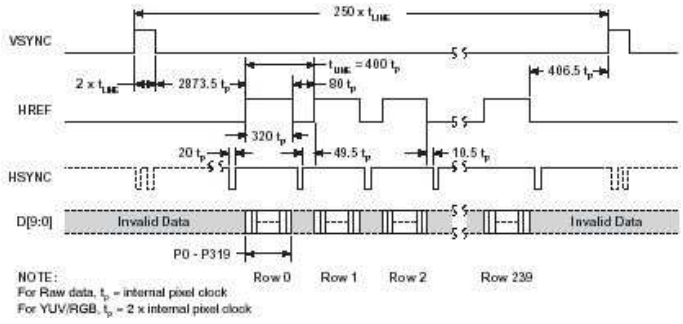
Aby bylo možné zpracovávat obraz, je nutné připojit k RC10 modul kamery. Byl použit modul 20J6 se CMOS chipem s označením OV9650 firmy Omnivision. Výstupní obraz má 1.3 MegaPixelů, jejichž velikost je dána rozlišením kamery. Typy rozlišení, které tento modul zvládá, jsou uvedeny v tabulce spolu s frekvencí, s jakou je dokáže zachytit. Tato frekvence se nazývá snímková frekvence (frame rate) a je udávána v počtu snímků za sekundu (fps).

rozlišení	snímková frekvence
SXGA	15fps
VGA	30fps
QVGA,QQVGA,CIF	60fps
QCIF,QQCIF	120fps

Tabulka 3.2: Přehled rozlišení modulu kamery

Výstupní formát těchto snímků může být v RGB, GRB či v YUV modelu. Pro naší aplikaci byl vybrán RGB model.RGB (červená, zelená, modrá) je aditivní způsob míchání barev, kde každá barva je udána mohutností tří základních barev. Existují různé varianty RGB modelu, které se liší počtem bitů jednotlivých složek(např. 888, 565, atd.). Pro tuto aplikaci budeme používat model 565, u které má nejvýšší počet bitů zelená složka, jelikož je pro lidské oko nejcitlivější.

Každé rozlišení kamery má rozdílný časový diagram vnitřních signálů. To je důležité znát pro správnou synchronizaci se zobrazovacím zařízením. V naší aplikaci je použito rozlišení QVGA tedy 320x240(0-319 sloupec, 0-239 řádek), proto následující obrázek ukazuje časový diagram vnitřních signálů právě pro toto rozlišení.



Obrázek 3.4: QVGA časový diagram

Pro nás jsou důležité signály HREF a VSYNC. Signál HREF indikuje platná data a je aktivní po dobu jednoho vyčteného řádku z kamery. Signál VSYNC nám zaznamenává začátek snímku a konec snímku. V našem programu sice přímo nevyužíváme tyto signály, avšak pro pochopení činnosti kamery jsou pro nás nezbytné. Dále je vidět z obrázku, že výstupní data z kamery jsou 10bitové, avšak při RGB565 je informace o jednom pixelu uložena v 16bitech, a proto potřebujeme 2 hodinové cykly na jeden pixel. Bližší informace o vlastnostech kamery lze nalézt v [11].

Kapitola 4

Způsob návrhu

Způsob návrhu programovatelných obvodů se postupně vyvíjel. Od propojování jednotlivých propojek u jednoduchých programovatelných součástek se s výkonějšími součástky objevil první jazyk ABEL, dále pak VHDL nebo Verilog a v dnešní době se mezi systémy popisující chování FPGA objevili jazyky na vyšší úrovni jako Handel-C nebo System-C.

Pro popis FPGA přímé programování propojek nepřipadá v úvahu, jelikož tyto obvody poskytují kapacitu v řádu statisíců, milionů hradel a o přenositelnosti návrhu též reč být nemůže. ABEL je jednoduchý programovací jazyk na nižší úrovni, avšak má omezené možnosti, a proto není vhodný pro součástky s velkým počtem systémových hradel. VHDL nebo Verilog jsou programovací a simulační jazyky popisující tok dat mezi jednotlivými registry. Poskytují velmi dobrou kontrolu nad finálním uspořádáním v hardwaru. System-C a Handel-C jsou vyšší jazyky, které jako mezikrok v implementaci používají VHDL. Jejich hlavní vyjímečnost spočívá v možnosti psaní paralelních aplikací pro programovací obvody ve vyšším programovacím jazyku. Handel-C se zdál být nejlepší variantou pro psaní aplikací do FPGA, proto byl pro tuto práci zvolen, avšak znalost VHDL alespoň v základní podobě je nezbytná, a proto mu bude věnována následující kapitola.

4.1 VHDL

VHDL je zkratka z VHSIC Hardware Description Language. Jedná se o programovací jazyk pro popis hardware. Konstrukce (model) v jazyku VHDL má dvě základní části: deklaraci entity (entity declaration) a tělo (popis) architektury (architecture body). Deklarace entity popisuje vstupy a výstupy konstrukce, popis architektury definuje její funkci. Architektura může být popsána různými styly jazyka VHDL.

Pojem stylu není v tomto jazyku přímo definován, ale některé jazykové konstrukty ve VHDL mohou být zařazeny do skupin, které tyto styly představují. Nejčastěji se mluví o stylu behaviorálním, o stylu popisujícím tok dat a o stylu strukturálním. Podrobnosti o programování v jazyce VHDL lze nalézt [1], [7] a [8].

4.1.1 Strukturální popis

Strukturální popis spočívá ve vkládání tzv. komponent (component) do kódového útvaru zvaného netlist. Tento styl, na rozdíl od jiných, obvykle nedává mnoho možností pro optimalizaci při syntéze, a poskytuje tak konstruktérovi větší kontrolu nad výsledkem syntézy. To však nemusí vždy představovat výhodu tohoto stylu, protože se tím nevyužívá inteligence zabudovaná do návrhového systému. Strukturální styl popisu je také obvykle méně přehledný ve srovnání s behaviorálním stylem. Proto se obvykle tam, kde je to možné, používá raději jiných stylů.

4.1.2 Popisy datového toku

Často se tento styl pokládá za zvláštní případ behaviorálního stylu. Vedle prostých přiřazovacích příkazů se zde používají příkazy typu podmíněných přiřazovacích příkazů CASE-WHEN nebo výběrových přiřazovacích příkazů WITH-SELECT-WHEN, tedy tzv. souběžné příkazy, z nichž každý můžeme pokládat za zjednodušený tvar procesu s vynechanými slovy PROCESS a END PROCESS, na rozdíl od sekvenčních příkazů používaných v procesech zapsaných úplnou formou. U souběžných příkazů nezávisí výsledek na pořadí jejich zápisu v textu.

4.1.3 Behaviorální popis

Behaviorálním stylem se zde rozumí popis na vysoké úrovni abstrakce, kde nemusí být uvažovány konkrétní hodnoty šířky datových a adresových sběrnic, kde nejsou specifikovány hodinové signály a podobně. Například při abstraktním modelování počítače se operace ADD chápe zcela obecně, bez uvažování určitého způsobu kódování dat. Tento styl se používá k simulaci. Pro syntézu je nutno úroveň abstrakce snížit. Také se charakterizuje užitím klíčového slova zvaného proces (process). Po slovu PROCESS následuje seznam citlivosti (sensitivity list), lze říci také seznam působících signálů, jejichž změna může vyvolat změnu některého ze signálů, které jsou procesem definovány. Popis procesu pak je ohrazen slovy BEGIN a END PROCESS s příslušným návěštím.

4.1.4 XILINX ISE

Pro programování v jazyce VHDL obvodů firmy Xilinx vyvinula ta samá firma vývojový systém ISE a jeho volně dostupnou variantu ISE WebPACK. U varianty WebPACK se omezení týká pouze velikosti hradlových polí, pro které je možno prostředí použít. Navíc neobsahuje některé rozšířené součásti jako například plnohodnotný editor výsledného propojení. Obě tyto verze obsahují nástroje pro syntézu a pro implementaci na dané FPGA, jenž jsou nutné pro vývoj aplikací pro FPGA. Tyto nástroje budou detailněji popsány u podkapitoly implementace do hardware v kapitole Handel-C. K simulování návrhů pak slouží HDL simulátor ModelSim XE.

4.2 Handel-C

Handel-C byl vyvinut firmou Celoxica za účelem abstrahovat program od hardware, a tím urychlit a ulehčit vývoj aplikací. Je to programovací jazyk, který vychází z jazyka C. Jeho základní syntaxe je podle normy ANSI C. Je určen pro sestavování programů do FPGA nebo ASIC. Aby bylo možné využít potenciál HW v plné míře má Handel-C oproti C řadu rozšíření:

1. Klíčové slovo PAR pro vkládání paralelismu
2. Možnost volit šířku slov po bitech
3. Jednoduchá synchronizace
4. Podpora HW rozhraní
5. Přímý přístup k jednotlivým bitům slova

Pro psaní kódu v Handel-C Celoxica vyvinula vývojový systém DK design suite. Tato aplikace se začala psát ve verzi DK4 a byla dokončena ve verzi DK5. Přechod mezi verzemi nečinil žádný problém s kompatibilitou.

4.2.1 Popis Jazyka

V následující části zde popíši některé části Handel-C, které se vyskytují v realizaci mé práce. Podrobnosti lze najít [6].

4.2.1.1 Paralelismus - PAR

KLíčové slovo *par* specifikuje blok kódu, který se provede paralelně. Každý příkaz uvnitř tohoto bloku se provede ve stejném cyklu. Pokuď máme dvě instrukce trvající různý počet cyklů, pak rychlejší instrukce čeká na pomalejší. Ke komunikaci mezi paralelními částmi kódu slouží kanály, klíčové slovo *chan*.

```
// ukazka kodu
par{
        a=10;
        b=20;
}
```

4.2.1.2 Proměnné

Základní typ proměnné v Handel-C je typ integer, který je buď typu signed(s znaménkem) nebo unsigned(bez znaménka). Může bý libovolného počtu bitů, který se definuje při deklaraci proměnné.

```
// ukazka kodu
signed int 8 promena1;
unsigned int 10 promena2;
```

V Handel-C není typ proměnné k uchování čísla s plovoucí řádovou čárkou jako v klasickém C(float).

4.2.1.3 Hodiny a časování

Každá funkce main v našem návrhu musí mít definovány hodiny, které se nastaví příkazem *set clock*. Hodiny mohou být nastaveny z výrazu (internal) nebo z pinu(external). Po nahrání do FPGA se frekvencí danou hodinami vykonává program.
syntaxe nastavení hodin:

```
set clock = umístění with {Rate, resolutiontime, minperiod}
```

Umístění	Popis
internal	hodiny z výrazu
internal_divide	hodiny z výrazu celočíselně vydělenné konstantou
external	hodiny z pinu HW
external_divide	hodiny z pinu HW celočíselně vydělenné konstantou

Tabulka 4.1: Druhy nastavování hodin

Jako referenci k zjištění stavu hodin nám slouží klíčové slovo __clock typu (unsigned 1).

Správné časování hraje důležitou úlohu v komunikaci mezi paralelními větvemi programu, při přístupu na externí HW, při synchronizaci a proto je nezbytné vědět, kolik taktů bude prováděn příkaz, smyčka nebo samotný program. Každé přiřazení trvá jeden hodinový cyklus. K vyladění časování se používá příkaz delay, který provede zpoždění o délce jednoho taktu hodin.

4.2.1.4 Rozhraní

Rozhraní(interface) slouží ke komunikaci mezi dvěmi domény s různými hodinami, nebo s okolím(přiřazení pinů apod.).V Handel-C jsou rozhraní jednosměrná nebo obousměrná(třístavová).
Základní syntaxe:

```
interface TypRozhrani(Vstupy) JmenoInstance (Výstupy)  with{data}
```

typ	popis
bus_in	přímé vstupní rozhraní
bus_out	výstupní rozhraní
bus_clock_in	vstupní rozhraní, hodnota se vzestupnou hranou hodin
bus_latch_in	vstupní rozhraní závislé na podmínce
bus_ts	třístavová sběrnice(obousměrná)
bus_ts_clock_in	třístavová sběrnice s hodinovým vstupem
bus_ts_latch_in	třístavová sběrnice se vstupem závislém na podmínce

Tabulka 4.2: Typy rozhraní

4.2.1.5 Pamět

FPGA obsahují paměti přímo na chipu, avšak liší se místem uložení. Generátory logických funkcí uvnitř CLB (F-LUT a G-LUT) je možné použít jako RAM. Výhodou je velká rychlost čtení a zápisu ve srovnání s externí RAM. Obvykle může být tato paměť využita jako dualportová nebo klasická jednoportová synchronní RAM nebo jako ROM. Při jejím použití se však připravujeme o využitelné prostředky pro ostatní logiku. Od generace SpartanII lze nalézt i blokovou paměť RAM tvořenou skutečnými bloky synchronní statické RAM. V Handel C je možno použít klasické pole, které jsou shodné jako u ANSI C, avšak v FPGA je realizováno jako skupina registrů. Aby se využilo pamětových prvků FPGA je třeba v Handel-C deklarovat ram,rom ci mpram. Nelze však zapisovat nebo číst hodnotu z více než jedné adresy. Pokuď chceme mít pamět uloženou do blokové RAM je nutné uvést block = "BlockRAM", neboť v Handel-C je pamět standartně ukládaná do distribuované paměti. Užití dualportové paměti nám dává možnost mít nezávislé čtecí a zápisové porty, z kterých lze číst a zapisovat ve stejném hodinovém taktu. syntaxe:

```
ram unsigned 32 data[n] with {block = \512\};

// dualportová ram
mpram dualportram{
wom unsigned 8 write[16];
rom unsigned 8 read[16];
}

mpram dualportram pamet with {block="BlockRam"}
```

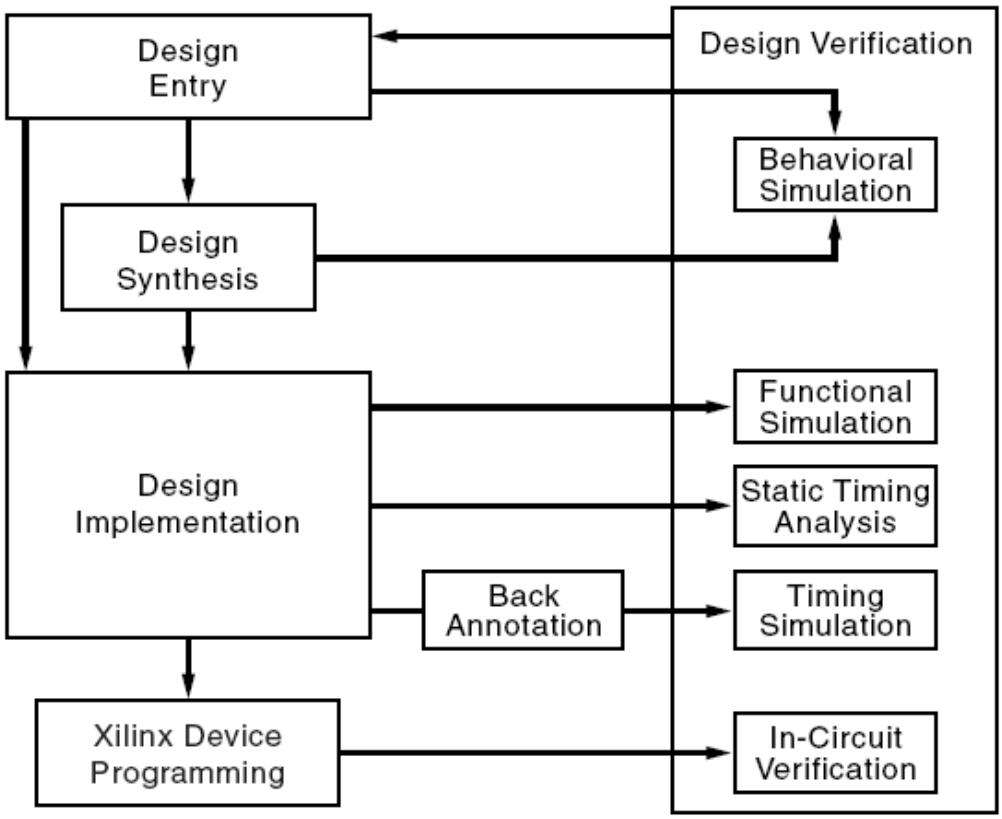
4.2.2 PDK - Platform Developer‘s Kit

PDK je knihovna nabízející tři vrstvy funkcionality, z nichž každá je složena ze čtyř částí. Snahou je soustředit se čistě na implementaci algoritmu a neztrácet čas s detaily hardware.

- DSM je podpora integrace procesoru a FPGA
- PAL je API rozhraní nezávislé na platformě
- PSL je Handel-C knihovna obsahující funkce komunikující s vnějšími zařízeními na FPGA platformě. PSL knihovna je vždy psaná pro jednu platformu

Tato práce se opírá o PAL a PSL knihovny. Jejich makra usnadňují přístup k potřebným periferiím, avšak přinášejí i značná omezení v jejich použití. Jak tato omezení zasahují do této práce, budou pojednávat kapitoly, k nímž se dané omezení vztahuje.

4.3 Implementace do Hardware



Obrázek 4.1: Schéma implementace do hardware

Na obrázku 4.1 je vidět schéma, které ukazuje postup implementace kódu napsaného v Handel-C do FPGA. Samotné DK se stará jen o blok Design Entry, jehož výsledkem je buď vhdl, pak další dva bloky syntéza a implementace už obstarávají nástroje Xilinx ISE, nebo EDIF blok, kde DK vytvoří i netlist a nástroje Xilinx ISE obstarávají pouze implementaci. Pro úplnost se v schematu objevuje verifikace. Této skupiny bloků jsme v naší aplikaci mohli využít jen částečně, protože v Handel-C neni podpora pro simulaci práce s kamerou.

4.3.1 Syntéza

Syntéza je vytvoření "netlistu", tj. zapojení obvodových prvků (logické členy, klopné obvody, registry atd.), tedy vlastně vytvoření schématu zapojení s obvodovými prvky, které jsou obsaženy v předpokládaných cílových obvodech a které vykonávájí požadovanou funkci.

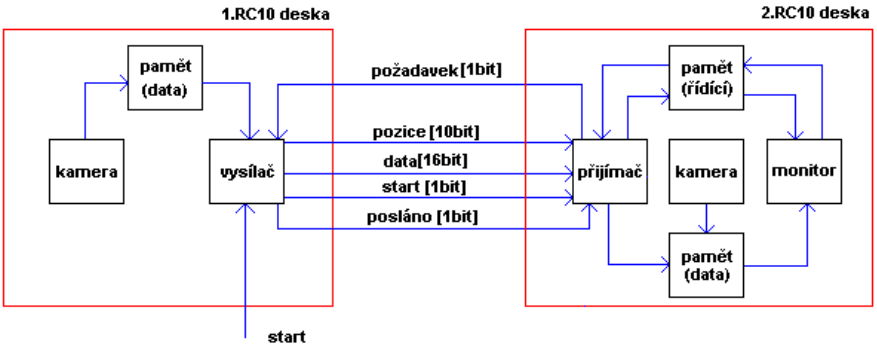
4.3.2 Implementace

Implementace zahrnuje několik postupných kroků, které vyústí do vytvoření popisu propojení cílového obvodu, který je podkladem pro vytvoření tzv. bitstreamu používaného pro konfiguraci FPGA. Důležité jsou zejména kroky Mapping a Place-and-Route. Zhruba řečeno, mapování spočívá v přiřazení obvodových prvků použitých ve výsledku syntézy konkrétním prvkům obsažených v cílovém obvodu - je analogií výběru konkrétních součástek u návrhu desky plošných spojů. Pak následuje rozmístění (placement) těchto prvků, tedy jejich přiřazení odpovídajícím obvodovým strukturám, které jsou v nenaprogramovaném cílovém obvodu vytvořeny. Zdařilá volba rozmístění má velký význam pro provedení následujícího kroku - propojení (routing), opět je vhodné představit si analogii s návrhem desky plošného spoje. Propojením se vytvoří plán výsledné struktury včetně potřebného nakonfigurování programovatelných propojek. Požadovaný stav jednotlivých propojek je pak obsažen v souboru, který je výsledkem implementace (bitstream).

Kapitola 5

Popis realizace

5.1 Úvod



Obrázek 5.1: Schéma aplikace

Aplikace je složena ze dvou hlavních částí, které v následujících podkapitolách budou popsány. První část se zabývá zobrazováním dat kamery na obrazovku monitoru, možnostmi výsledného obrazu s ohledem na vlastnosti desky a kapacitu použitého FPGA a způsoby implementace v Handel-C. V druhé části je popsána realizovaná komunikace mezi dvěma deskama, použitý protokol a možnosti synchronizace. Nakonec sestavení závěrečné aplikace z těchto modulů tak, aby bylo možno zpracování dvou zdrojů obrazů současně a implementace ukázkových algoritmů.

5.2 Zobrazení dat z kamery na VGA

5.2.1 Snímání kamerou pomocí Handel-C a PDK

PDK v sobě obsahuje všechny potřebné makra k ovládání kamery, proto jich využijeme. Tím odpadá konfigurace registrů kamery a máme zajištěnou velmi snadno její správnou funkčnost. K rozběhnutí kamery je třeba makro RC10CameraRun(Mód, hodiny), jenž musí běžet paralelně s každým přístupem k ní. Jako parametry má toto makro mód kamery a vstupní hodiny. Ke správnému chodu kamery a vyčítání pixelů za sebou musí být vstupní frekvence v rozmezí od 10MHz do 48 MHz. Mód kamery se zvolí dle vybraného barevného modelu. Jak už bylo řečeno v kapitole Modul kamery bude použit RGB model, tím pádem mód kamery nastavíme jako OV9650_RGB565_QVGA. Při použití tohoto makra nastaly problémy s rozlišením. Kamera nezobrazovala obraz v rozlišení QVGA, ale zobrazovala ho v jejím největším rozlišením SXGA (1280 x 1024). Tento nedostatek se dal odstranit použitím makra RC10CameraSetMode(mód). Pro vyčtení pixelů z kamery do paměti má v sobě PDK makro RC10CameraReadRGB565 (&X, &Y, &Pixel). To uloží na adresu proměnných X,Y a Pixel informace o pozici a barvě každého pixelu v čase dle časového diagramu pro dané rozlišení, viz obr. 3.4. Po sestavení těchto tří maker do programu, vypadá jednoduché vyčítání pixelů z kamery následovně:

```
#define PAL\_TARGET\_CLOCK\_RATE 12000000
#include "pal\_master.hch"
#include "stdlib.hch"

void main(void){

unsigned X,Y,Pixel;
macro expr ClockRate = PAL\_ACTUAL\_CLOCK\_RATE;

par{

        RC10CameraRun(OV9650\_RGB565\_QVGA, ClockRate);

        seq{

                RC10CameraSetMode(OV9650\_RGB565\_QVGA);

                while(1){

                        RC10CameraReadRGB565(\&X,\&Y,\&Pixel);

                }
        }
}
}
```

5.2.2 Zobrazeni na VGA pomocí Handel-C a PDK

K zobrazovaní dat na VGA byla využita část PDK PAL, jenž svými makry stačila k potřebám práce. K rozběhnutí video výstupu slouží dvě makra PalVideoOutRun (VideoOut, Hodiny), jenž musí běžet paralelně po celou dobu práce s video výstupem a makro PalVideoOutEnable (VideoOut), které zaktivuje zobrazování. Parametr VideoOut, které májí společné, se nastaví dalším makrem z knihovny PAL a to PalVideoOutOptimalCT (Hodiny). To provede nastavení výstupu, který odpovídá zvoleným hodinám. V následující tabulce jsou uvedena, některá rozlišení obrazovky, dosažené opakovací frekvence a k nim příslušné potřebné hodiny.

Rozlišení	Opakovací frekvence [Hz]	Hodinová frekvence [MHz]
640x480	60	25.175
640x480	75	31.500
800x600	60	40.000
800x600	75	49.500
1024x768	60	65.000
1024x768	75	78.750

Tabulka 5.1: Přehled rozlišení Video výstupu

Po spuštění a nastavení VGA se použije PalVideoOutWrite (VideoOut, data) k zobrazení jednotlivých pixelů. Toto makro zapíše pixel na místo, které má souřadnice dány PalVideoOutGetX (VideoOut) pro sloupce a PalVideoOutGetY (VideoOut) pro řádky. Ty procházejí obrazovku od pozice [0,0] do souřadnic daných makry PalVideoOutGetTotalX () a PalVideoOutGetTotalY (). V případě rozlišení 640x480 při 25.175MHz udávají tyto makra hodnoty 800x525. Obraz se pak doplňuje černými pixely a slouží pro správnou synchronizaci obrazu monitoru. Maximální hodnoty souřadnic, na kterých lze ještě pozorovat obraz, zjištují makra PalVideoOutGetVisibleY (), PalVideoOutGetVisibleX (). Procházení souřadnic monitoru nelze zastavit, a proto se musí zajistit, aby PalVideoOutWrite() zapisovalo pixel každý hodinový takt.

5.2.3 Návrh řešení

Abychom správně zobrazili snímky z kamery na monitor, je nutné vědět jakou frekvencí kamera dává jednotlivé pixely a monitor je zobrazuje. V případě, že nemáme k dispozici paměť, do které by se vešel minimálně jeden snímek z kamery, je nutné pro správné zobrazení, aby tyto dvě frekvence byli shodné.

Na RC10 žádná dostatečně rychlá a velká pamět není, a proto se musí vystačit s kapacitou blokové paměti v FPGA. Do nichž lze při správném využití uložit jeden snímek při rozlišení 160x120. Pak je možné snímek zobrazit bez větších problémů na monitor. V Handel-C použijeme dualportovou paměť nebo využijeme PDK PAL, kde existuje už hotový framebuffer, který se sám stará o zobrazování na monitor a na programátorovi zbyde jen postarat se o načítání do něj. PAL má k dispozici mnoho druhů framebufferů avšak na RC10 je možno použít jen block ram framebuffer, který umí jen ukládaná data zobrazit na monitoru, ale nedovolí je získat pro jiné využití. To je zásadní nevýhoda pro využití v naší aplikaci, jelikož chceme posílat data z paměti jedné desky do druhé. Proto v tomto případě využijeme dualportové paměti v Handel-C a vytvoříme si vlastní framebuffer.Toho lze dosáhnout pomocí klíčového slova mpram, které bylo vysvětleno v kapitole Handel-C. Zde jen připomenu, že pokuď má být paměť uložena v FPGA v blokové paměti, musí se použít klíčové slovo block=”BlockRAM”, jinak se pamět uloží do vnitřní logiky FPGA. Výsledná syntaxe použitá v aplikaci pak vypadá takto:

```
#define DELKA 640
#define POCET\_RADKU 30

mpram dualportram{
rom unsigned 16 cti[DELKA];
wom unsigned 16 zapis[DELKA];
}

mpram dualportram pamet[POCET\_RADKU] with {block = "BlockRAM"};
```

Aby se do paměti vešel snímek 160x120 musí být ”DELKA” rovna 640 a ”POCET _RADKU 30”. Daný kód vytváří pole pamětí, z nichž každá je uložena v jedné blokové paměti a obsáhne 4 řádky snímku o velikosti 640 x 16bitů, což je 10240 bitů z dostupných 18432 bitů. Využije se tedy 30 blokových pamětí z 32 možných na použitém FPGA. Při uchování celého snímku už pak jen stačí zapisovat do paměti data z kamery a po načtení snímku je zobrazovat na monitor a to nezávisle na sobě.

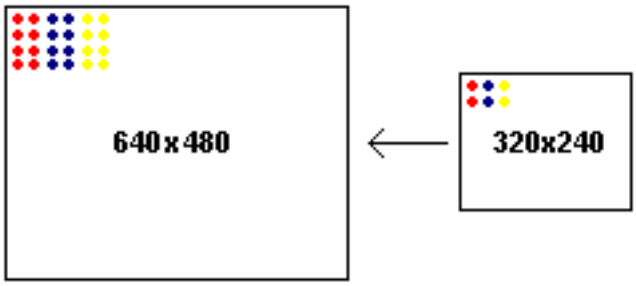
Při větším rozlišení kamery však nastává problém. Do blokové paměti FPGA už se celý snímek nevejde, a tak nastává problém synchronizace mezi monitorem a kamerou. Můj pokus o správnou synchronizaci spočíval ve sladění rychlostí náčtení

jednoho snímku. Toho se dosáhlo při rozlišení kamery 320x240 a budící frekvence 12.5875MHz zobrazované na monitoru o rozlišení 640X480 s frekvencí 25.175MHz, jak dokládají následující výpočty.

pro kameru
$T_{pixel} = 1/12587500Hz = 7.944389x10^{-8}s$
$T_{radka} = 400 * 2 * 7.944389x10^{-8} = 6.3555x10^{-5}s$
$T_{viditframe} = 240 * 6.3555x10^{-5} = 0.01525s$
$T_{frame} = 250 * 6.3555x10^{-5} = 0.015888778s$

pro monitor
$T_{pixel} = 1/25175000Hz = 3.971x10^{-8}s$
$T_{radka} = 800 * 3.971x10^{-8}s = 3.177x10^{-5}s$
$T_{viditframe} = 480 * 3.177x10^{-5} = 0.01525s$
$T_{frame} = 525 * 3.177x10^{-5}s = 0.01668s$

Po srovnání rychlostí zbývá srovnat počet pixelů. Každý pixel, který se načte z kamery, se zobrazí na monitoru čtyřikrát a to způsobem uvedeným na následujícím obrázku.



Obrázek 5.2: Zobrazování pixelu kamery na monitoru

Pixel se zobrazí na výstupu v jedné řádce dvakrát a poté i řádka. Dosáhne se tak obrazu o rozlišení 640x480 načítaný a zobrazovaný stejnou frekvencí. To však platí jen ve viditelné části. Ve skutečnosti při porovnání Tframe kamery a monitoru je kamera rychlejší a při jednom snímku monitoru načte snímek a 12 řádek následujícího snímku. To způsobuje přetečení paměti, jelikož vzdálenost ukazatelů čtení a zápisu v paměti je každým snímkem o 12 řádek vyšší. Tento problém nevyřeší

ani zvýšení paměti, pokuď se do ní nevejde celý snímek. Jediným řešením je možnost zastavení kamery. To se však používáním maker PSL dá těžko realizovat, jelikož se nedá zjistit, jak se kamera chová po opětovném spuštění z časového hlediska. Proto tato realizace při použití PDK maker není možná.

Použitím maker PAL pro zobrazování pixelů není jasné zpoždění náčtu prvního pixelu na pozici [0,0], proto necháme vykreslit první snímek černě. S předstihem prvního pixelu o 21 řádků spustíme kameru. Ta má tak dostatek času, aby načetla do paměti 1.řádek snímku. To dokládá časový diagram kamery pro rozlišení QVGA 3.4.

Ať už je realizováno zobrazení snímku z kamery framebufferem nebo bez něj, je nutné pro správné zobrazení na monitor upravit každý pixel z RGB565 do RGB888. To provede následující kód:

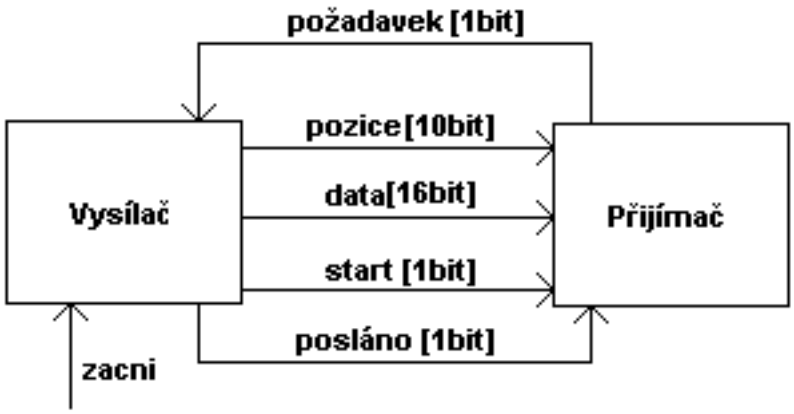
```
R = m[15:11]@0;
G = m[10:5]@0;
B = m[4:0]@0;

PalVideoOutWrite(VideoOut, R@G@B);
```

Ten rozdělí každou složku RGB modelu zvlášť a doplní jí zleva nulami na 8 bitů a poté je ve správném pořadí spojí a zobrazí na monitor.

5.3 Propojení desek

5.3.1 Blokové schéma



Obrázek 5.3: Propojení desek

Zjednodušené schéma aplikace zdůrazňuje jen chování protokolu komunikace desek. Desce, která vysílá data, náleží blok *Vysílač* a druhé desce, která data přijímá, náleží blok *Přijímač*. Přiřazení jednotlivých pinů představuje následující tabulka.

Data								Požadavek	Start
15	14	13	12	11	10	9	8	0	0
F11	E11	C11	D11	E13	E12	C12	D12	F17	G18
7	6	5	4	3	2	1	0	0	0
D14	D13	A14	B14	C14	C15	A15	B15	F17	G18

Tabulka 5.2: Přiřazení pinů FPGA pro jednotlivé signály I

pozice									
9	8	7	6	5	4	3	2	1	0
C18	B18	E16	D18	D17	D16	E18	E17	E15	F15

Tabulka 5.3: Přiřazení pinů FPGA pro jednotlivé signály II

5.3.2 Komunikační Protokol, Časování, Synchronizace

Aby bylo možné vůbec realizovat správnou komunikaci obou desek je nutné, aby se spustili ve stejnou dobu. Na straně *Vysílače* se aktivuje pomocí stisku joystiku, který je na RC10. Po jeho stisku se vyšle *Přijímači* signál, a tím *Přijímač* aktivuje.

Protokol je založen na požadavku a jeho splnění. Jelikož přenášet jedno 16 bitové slovo v rámci plnění jednoho požadavku by bylo značně neefektivní vzhledem k obsluze a pro vyčítání dat na monitor nerealizovatelné, je potřeba najít základní element vhodný k přenosu. Jako ideální se jeví řádek paměti. Je snadno implementovatelný, a když bude zajištěn včasný požadavek, budou data na straně *Přijímače* včas připravena k dalšímu využití.

Přijímač jako ostatně i *Vysílač* se ocitají v čekací smyčce. Jakmile *Přijímač* potřebuje data, ukončí čekací smyčku a vyšle signál *Vysílači*. Ten jakmile přijme signál o požadavku provede totéž a v následujícím taktu hodin začne vysílat data a pozici dat v řádku paměti. *Přijímač* pak současně přijímá data s pozicí, kterou využívá jako ukazatel do vyrovnávací paměti, do které zapisuje data. Po odeslání všech dat *Vysílač* pošle signál o ukončení vysílání. Během přenosu je potřeba resetovat řídící signál, aby po odeslání byl *Vysílač* opět v čekací smyčce. Jakmile je

přenos dokončen oba inkrementují ukazal na řádek v paměti a znovu skočí do čekací smyčky. V dalším požadavku se proces opakuje.

5.3.3 Implementace v Handel-C

Pro propojení desek postačí Handel-C, i když v PSL existují makra zvaná Expansion header. V této aplikaci byl použit k propojení interface a to sběrnicového typu, jehož druhy lze naleznout v tabulce 4.2. Jako vstupní interface byl použit bus_clock_in, jehož hodnota se mění se vzestupnou hranou hodinového signálu. To je užitečné jelikož při přenosu dochází ke zpoždění dat a tak může dojít při použití bus_in k přepisu hodnoty uprostřed taktu, což je nežádoucí. Bus_clock_in tedy dává možnost jednoduché synchronizace s hodinovým signálem. Jeho deklarace, přiřazení pinů FPGA a získání dat použitých v aplikaci ukazuje následující kód:

```
interface bus_clock_in(unsigned 16 read) Prijimac() with
    {data = {"F11", "E11", "C11", "D11", "E13",
        "E12", "C12", "D12", "D14", "D13", "A14",
        "B14", "C14", "C15", "A15", "B15"}};

interface bus_clock_in(unsigned 1 spust) Spust() with
    {data = {"F17"}};

interface bus_clock_in (unsigned 1 je) Pozadavek () with
    {data = {"G18"}};

void main(void){

        unsigned 16 cti;
        cti = Prijimac.read;

}
```

Jako výstupní interface byl použit bus_out, což je jediná varianta v Handel-C. Výstup se vztahuje k interní proměnné, jejíž hodnota ho po všechen čas definuje. Deklarace, přiřazení pinů FPGA a hodnoty je ukázána dále.

```
unsigned 16 vystup = 0;
interface bus_out() Vysilac(unsigned 16 read = vystup) with
    {data = {"F11", "E11", "C11", "D11", "E13",
        "E12", "C12", "D12", "D14", "D13", "A14",
        "B14", "C14", "C15", "A15", "B15"}};

void main(){
        vystup = 7;
}
```

5.4 Současné zpracování obrazů

Současného zpracování obrazů dosáhneme nakombinováním dvou předcházejících částí 5.2, 5.3 a dosáhne se schematu uvedeného v úvodu této kapitoly 5.1, což je cílem této práce.

První deska se bude skládat ze dvou modulů a to z *Vysílače* a modulu pracující s kamerou navzájem spojené přes dualportovou pamět. Jelikož jeden modul do ní zapisuje a druhý z ní jen čte, je toto spojení dostačující a dá se provést tímto kódem:

```
mpram dualportram{
rom unsigned 16 cti[DELKA];
wom unsigned 16 zapis[DELKA];
}

extern mpram dualportram pamet;

void main(void){
        pixel=pamet.cti[DELKA];
}
```

```
mpram dualportram{
rom unsigned 16 cti[DELKA];
wom unsigned 16 zapis[DELKA];
}

mpram dualportram pamet;

void main(void){
        pamet.zapis[DELKA] = pixel;
}
```

Dualportová pamět plní funkci framebufferu a její organizace je provedena tak, jak je popsaná v kapitole Zobrazení dat z kamery na VGA. Data z kamery v rozlišení QQVGA(160x120) jsou tak zapisována do paměti, jejíž jeden řádek se rovná čtyřem řádkům snímku kamery. Nezávisle na tom jak se zapisují data do paměti pracuje *Vysílač*. Ten je řízen *Přijímačem* druhé desky a při požadavku mu posílá 4 řádky snímku.

Druhá deska obsahuje 3 moduly. První modul *Výstup* představuje centrální jednotku, která řídí druhý modul *Přijímač* a realizuje vyčítání dat na monitor. *Přijímač* má za úkol obstarávat data z kamery první desky. Poslední modul stejně tak jako u první desky tvoří modul pracující s kamerou. Aby mohl *Výstup* řídit *Přijímač* je zapotřebí obousměrná komunikace. Využije se proto dvou dualportových pamětí, jelikož každá umožňuje pouze jednosměrnou komunikaci. Jak už bylo zmíněno ke správnému chodu je potřeba, aby obě desky začaly ve stejnou dobu, a tak *Vysílač* po stisku joystiku pošle signál do druhé desky a aktivuje *Přijímač*. Stejně tomu tak je i v případě *Výstupu*. Tomu poslouží již zmiňovaná "řídící" pamět a aktivuje ho. Po zapnutí *Výstupu* přebírá řídící funkci tento blok a v případě potřeby přes druhou "řídící" pamět vysílá požadavky k *Přijímači*, které jsou signalizovány aktivní úrovní '1'. Obě řídící paměti se pak musí navzájem resetovat, aby se jejich hodnota dostala

zpátky na neaktivní úroveň '0'.

Mezi *Výstupem* a *Přijímačem* je ještě jedna dualportová pamět. Ta má funkci vyrovnávací paměti při načítání dat *Přijímačem*. Ke správném chodu je potřeba, aby měla velikost dva řádky, z nichž každý pojme data z jednoho požadavku, 4 řádky snímku. Pak je možno realizovat, že *Výstup* zpracovává jednu řádku a mezitím se načítá *Přijímačem* druhá.
Nezávisle na tom pracuje modul zpracovávající data z lokální kamery. Stejně tak jako tomu bylo u první desky je modul spojen s *Výstupem* další dualportovou pamětí, která má kapacitu na načtení jednoho snímku o velikosti 160x120. Tím se *Výstupu* dává možnost využít dva zdroje dat a současně je zpracovávat.

5.5 Použité ukázkové algoritmy

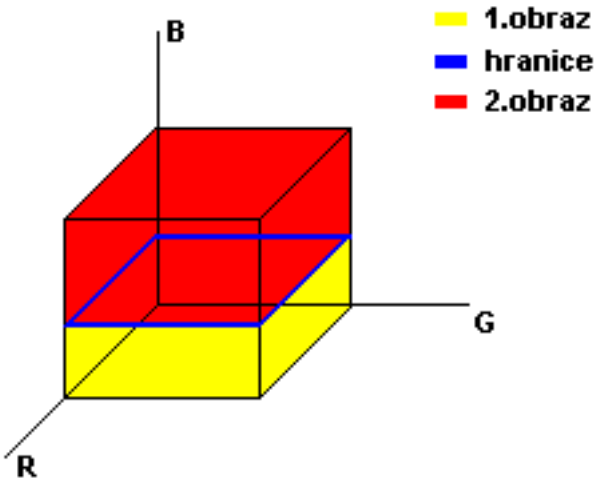
5.5.1 Obraz v obraze

Jako první byl vyzkoušen algoritmus obraz v obraze. Tento algoritmus vkládá do jednoho obrazu druhý, který je v menším rozlišení. V našem případě mají oba zdroje dat shodné rozlišení. Zmenšováním obrazu by bylo nutné makrem změnit rozlišení kamery a navíc při základním rozlišení 160x120 by se stal obraz pro uživatele pomalu "neviditelným", a proto je potřeba zvolit jeden snímek jako základní a upravit ho na vyšší rozlišení. Pak je možné do něj vložit druhý snímek bez úprav. Zvýšení rozlišení se dosáhne zčtveřením každého pixelu snímku, tak že každý pixel se bude vyskytovat v jednom řádku dvakrát a řádek v snímku také dvakrát. Tím dosáhneme u jednoho snímku rozlišení 320x240 a v jedné jeho čtvrtině bude vložen druhý snímek. Algoritmus zčtveření se musel vypořádát s problémem, že jeden řádek paměti jsou čtyři řádky snímku. Výsledný ukazatel proto musel procházet daný řádek paměti elegantněji. Pro druhý snímek se zvolila horní čtvrtina snímku, a tak když scanující makra pro pozici zobrazení pixelu na monitoru se ocitla v tomto místě, prohodil se zdroj dat na druhý snímek.

5.5.2 Klíčování

Dalším algoritmem, který byl vyzkoušen je takzvané klíčování. To je založeno na principu podmínky(klíče), dle které se rozhoduje zda se pixel obrazu nahradí pixelem z jiného obrazu či ne. V našem případě se náhodně zvolila jedna barevná složka z RGB modelu, a to modrá. Podle její hodnoty se pak rozhodovalo z jakého obrazu

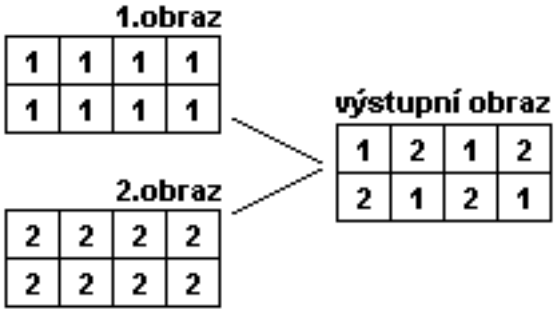
se pixel zobrazí na výstupu. Pokuď byla hodnota modré složky větší než zvolená hodnota, výstupní pixel byl ze snímku přijatého Přijímačem. V opačném případě byl pixel z lokální kamery. To dokladuje následující obrázek 5.4.



Obrázek 5.4: Klíčování modrou složkou v RGB modelu

5.5.3 Šachovnice

Posledním algoritmem, který byl vyzkoušen je Šachovnice. Ta vznikne tak, když za výstupní pixel střídavě volíme ze dvou nám dostupných zdrojů obrazu a to tak, že v lichých řádkách bude první pixel z prvního obrazu a v sudých z druhého obrazu. Výsledný obraz pak bude vypadat jako sachovnice, viz 5.5.



Obrázek 5.5: Postup skládání obrazu Šachovnicí

5.6 Problémy při realizaci

Zásadním problémem při realizaci byl fakt, že RC10 nemá externí rychlou pamět, a tak musela postačit kapacita blokové paměti v FPGA. Ta ale není zdaleka tak veliká, aby obsáhla snímek o větším rozlišení než je 160x120. Pokus obejít nedostatečné množství paměti sesynchronizováním kamery a monitoru při použití maker PDK selhal. To dokládá nevýhodu při používání těchto maker. Na jednu stranu nám usnadňují přístup k periferiím desky, ale na druhou znemožňují její přenastavení. Návrhář je pak odkázán používat danou periferii jen způsobem, jak dovolují makra. Tento problém se vyřešil redukcí rozlišení kamery na rozlišení 160x120, kde se snímek vešel do blokové paměti FPGA.

Další, už ale vyřešitelný problém, byl při incializaci kamery. Ta se vždy spustila v jejím největším možném rozlišení. Naštěstí PSL obsahovalo makro na změnu rozlišení, na které už kamera reagovala, a začala se chovat dle požadavku.

Posledním nedostatkem této realizace je, že výstupní obrázek, který lze pozorovat na monitoru, nezačíná přesně od kraje. To je způbeno nepoužitím maker procházející souřadnice monitoru jako ukazatel do paměti. Při velikosti paměti jí nelze vhodně upravit tak, aby se makra dala použít.

Kapitola 6

Závěr

Seznámil jsem se s vývojovou deskou RC10 s FPGA XC3S1500 a modulem kamery na této desce. Dále jsem prostudoval metody návrhu a implementací pro obvody FPGA pomocí nástrojů Xilinx ISE a Celoxica DK.

Celá aplikace je napsána jazykem Handel-C a je navržena v návrhovém systému Celoxica DK. Část aplikace byla napsána ve verzi DK4 a dokončena byla ve verzi DK5.

Zrealizoval jsem propojení dvou desek RC10 s moduly kamer tak, aby bylo možné snímání a základní zpracování obrazů z obou kamer současně. Toto propojení je funkční pouze při rozlišení kamery 160x120, z důvodu nedostatku paměti na implementační platformě.

Současné zpracování obrazů bylo vyzkoušeno na algoritmech obraz v obraze, klíčování a šachovnice obrazů. Pro lepší vizuálnost byl výstupní obraz převeden do rozlišení 320x240.

Celý systém je zdokumentován v elektronické podobě zdrojovými kódy a projektovými soubory návrhového systému na přiloženém CD.

6.1 Doporučení pro další vývoj

Pro další vývoj této práce bych doporučoval změnit implementační platformu na desku s externí dostatečně rychlou pamětí, aby nedocházelo k potížím při používání maker PDK. Pak by mohlo snímání i zobrazování proběhnout v daleko lepším rozlišení. Po tematické stránce by se další vývoj této práce mohl zaměřit na praktické využití současného zpracování více zdrojů a to v podobě stereovidění, jak už bylo zmíněno v úvodu.

Literatura

[1] Peter J.Ashenden (1995). *The Designer's Guide to VHDL*. San Francisco, California.

[2] Matthew Aubury (2005). *RC 10 Platform Development Manual*. Celoxica, UK.

[3] Al Bovik (2005). *Handbook of Image and Video Processing*. London, UK.

[4] Roger Gook (2005). *PDK Manual*. Celoxica, UK.

[5] Obvody FPGA a vývojový systém firmy XILINX [ONLINE]. [cit. 2007-08-19] Dostupné z: `http://www.xilinx.com`.

[6] *Handel-C Language Reference Manual*. Celoxica, UK. (2005)

[7] Základní struktura modelu v jazyku VHDL [ONLINE]. [cit. 2007-08-19]. Dostupné z:
`http://www.urel.feec.vutbr.cz/k̃olouch/pld/2_cviceni/kapitola03_02.html`.

[8] Postup při vývoji aplikací obvodů PLD a FPGA [ONLINE]. [cit. 2007-08-19]. Dostupné z:
`http://www.urel.feec.vutbr.cz/k̃olouch/pld/1_prednasky/kapitola02_01.html`.

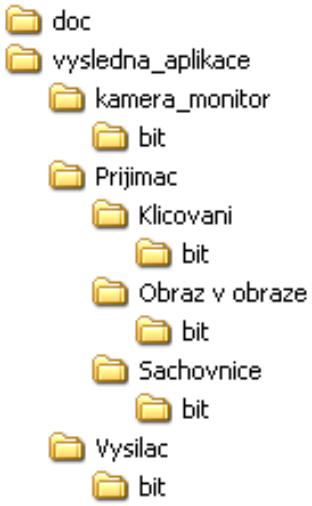
[9] Jan Kořenek *Úvod do Handel-C*. Vysoké učení technické v Brně, Fakulta informačních technologií [ONLINE]. [cit.2007-08-19]. Dostupné z:
`www.fit.vutbr.cz/study/courses/NCS/public/prednasky/pdf/handelC_1.pdf`

[10] Pavel Typl (2005). *Zpracování videosignálu s pomocí FPGA*. Diplomová práce ČVUT, Praha.

[11] *Product Data Sheet 2OJ6* MicroJet Technology Co., Ltd.

Přílohy

CD obsahuje všechny algoritmy, které byly vyzkoušeny. A to v následující adresářové struktuře.



Obrázek 6.1: Adresářová struktura přiloženého CD

Z adresáře Prijimac si lze vybrat algoritmus, který chceme vyzkoušet. Po výběru se dostaneme do adresáře daného algoritmu, v němž se nalézají použité zdrojové kódy a podadresář bit. V něm je konfigurační soubor pro FPGA, ten se pak nahraje do desky s připojeným VGA výstupem. Z adresáře Vysilac a následně podadresáře bit se nakonfiguruje FPGA druhé desky. Mimo to jsou také v adresáři Vysilac použité zdrojové kódy.

Po nahrání programu do obou desek se pak na desce, která slouží jako vysílač, zmáčkne joystik směrem dolů a aplikace se začne vykonávat.

Dále na CD lze najít adresář kamera_monitor, kde jsou zdrojové soubory a konfigurační soubor pro FPGA, pokusu synchronizace mezi kamerou a monitorem.

Posledním adresářem na CD je doc, kde se nachází text bakalářské práce v pdf formátu.