České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídící techniky

Bakalářská práce

Snímání a předzpracování obrazu pomocí FPGA

Vedoucí práce:
Ing. Roman Bartosinski

Autor:
Přemysl Papík

2007

Děkuji Ing. Romanu Bartosinskemu za vedení práce, odbornou pomoc a poskytnutí vývojových prostředků.

Děkuji Ing. Josefu Papíkovi za nabídnuté téma práce, konzultace a morální podporu.

Anotace

Tato bakalářská práce byla motivována potřebou získat zkušenosti s programováním obvodů FPGA (Field programmable gate array), které budou základem pro následný vývoj kamerového senzoru nečistot.

Práce se zabývá vývojem programu na předzpracování a zobrazení video signálu na monitoru. K realizaci byla využita vývojová deska RC10 od firmy Celoxica, která obsahuje obvod FPGA typu Xilinx XC3S1500 a další potřebné komponenty, jako jeho periferie. Veškeré operace s video signálem jsou pak prováděny jen v tomto hlavním obvodu, který umožňuje paralelní zpracování dat. Pro vývoj byly využity programovací jazyky VHDL a assembler pro mikrokontroler PicoBlaze3. Aplikaci je možno řídit pomocí definovaných zpráv přes sériovou linku RS232 z počítače.

Annotation

This project was motivated by requirement of obtain knowledges and experiences with programming of FPGA devices, which will be the base of the camera sensor of impurity consequent developement.

The project is about developement of the software for video preprocessing and presenting video by a monitor. For the realisation was used the developement kit RC10 by Celoxica company, which contains FPGA Xilinx XC3S1500 device and others usefull components as it`s peripheries. All operations with the video signal are executed only in this device, which allows paralel data processing. For the developement were used programming languages VHDL and special assembler for the PicoBlaze3 microcontroler. It is possible to control the application by predefined messages over the serial bus RS232 from a computer.

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídicí techniky

Školní rok: 2006/2007

Zadání bakalářské práce

Student: Přemysl Papík

Obor: Kybernetika a měření

Název tématu: Snímání a předzpracování obrazu pomocí FPGA

Zásady pro vypracování:

1. Seznamte se s návrhem a implementací hardwaru pro obvody FPGA pomocí nástroje Xilinx ISE a s mikro-kontrolérem PicoBlaze.
2. Seznamte se s vývojovou deskou RC10 s FPGA XC3S1500 a připojením modulu kamery k této desce.
3. Implementujte rozhraní pro připojení kamery k mikro-kontroléru PicoBlaze.
4. Demonstrujte funkčnost implementace na aplikaci snímání obrazu modulem kamery.

Seznam odborné literatury:

- Peter J.Ashenden, The Designer's Guide to VHDL, San Francisco, California, 1995
- Matthew Aubury, RC 10 Platform Development Manual, Celoxica, UK, 2005
- Al Bovik, Handbook of Image & Video Processing, London, UK, 2005

Vedoucí bakalářské práce: Ing. Roman Bartosinski

Datum zadání bakalářské práce: zimní semestr 2006/07

Termín odevzdání bakalářské práce: 15. 8. 2007

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

Prof. Ing. Zbyněk Škvor, CSc.
děkan

Prohlášení

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

V Praze, dne ...16.8.2007..... .....

podpis

OBSAH

	Obsah	VII
1.	**Úvod**	1
2.	**Rozbor zadání**	2
2.1	Výkonný mikroprocesor v PC	2
2.2	Signálový procesor	3
2.3	Programovatelné hradlové pole	3
2.4	Vzájemné zhodnocení návrhů	4
3.	**Popis hardware**	5
3.1	Vývojová deska	5
3.2	Kamera	6
3.2.1	Sběrnice SCCB	7
3.2.2	Přenos dat	7
3.3	VGA výstup	9
3.4	RS232 port	10
3.5	Obvod FPGA	10
4.	**Popis vývojových prostředí a software**	11
4.1	Mikrokontroler PicoBlaze3	11
4.2	UART moduly	12
4.3	Prostředí ISE	12
4.4	Prostředí ModelSim	13
5.	**Koncepce aplikace**	14
6.	**Rozbor synchronizace časování kamery a monitoru**	15
7.	**Rozbor jednotlivých bloků aplikace**	18
7.1	Blok TOP_propojeni	18
7.2	Blok HLAVNI	18
7.3	Blok UART	19
7.4	Blok CAMERA	20
7.5	Blok DEKODER	22
7.6	Blok AVG	22
8.	**Ovládání aplikace**	24
9.	**Statistické shrnutí**	26
10.	**Závěr**	27
	Literatura	28
	Seznam příloh	29
	Přílohy	30

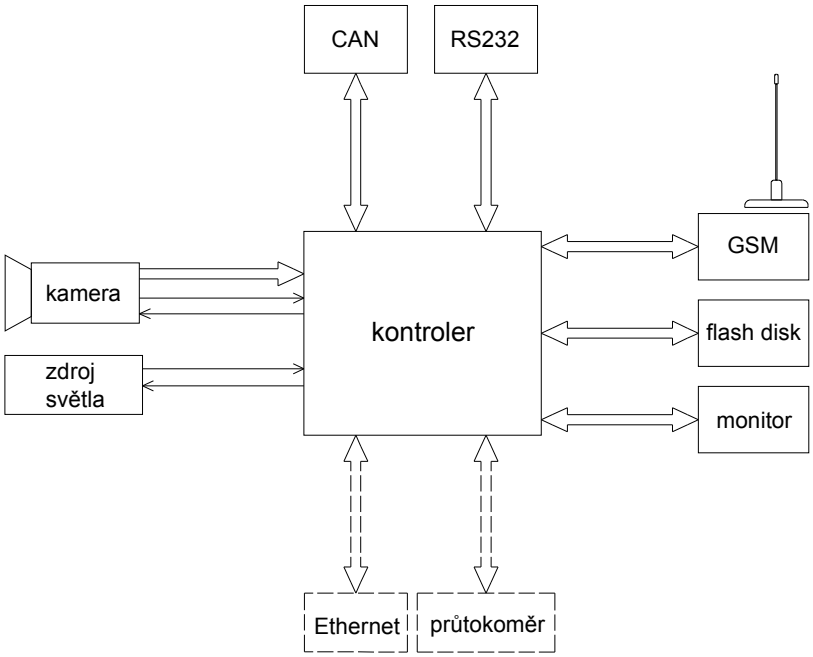
1. Úvod

Rámcové zadání mé práce vzniklo z požadavku firmy, se kterou spolupracuji, na vývoj kamerového senzoru nečistot. Hlavními požadavky bylo vytvořit samostatné zařízení, skládající se ze tří hlavních částí – zdroje světla, snímače obrazu a vyhodnocovací jednotky, měřící pohyblivý vzorek v reálném čase. Ta by měla být schopná zobrazovat na display aktuální hodnotu znečistění vzorku, poskytovat tento i další údaje o měření okolním zařízením pomocí sběrnic CAN a RS232 a zároveň přes tyto sběrnice kalibrovat a ovládat senzor. Nadstavbovými funkcemi by měla být archivace dat na flash disk a ovládání GSM modemu na rozesílání varovných hlášení o překročení povolené hladiny znečištění či poruše senzoru.
Samozřejmostí byl požadavek na kvalitu a spolehlivost naměřených údajů, jednoduchou údržbu senzoru a také nízkou koncovou výrobní cenu.

Přestože toto rámcové zadání je relativně krátké, jedná se o příliš rozsáhlý úkol na to, abych ho mohl celý uskutečnit v rámci bakalářské práce. Proto jsem celý úkol po dohodě s vedoucím práce rozdělil na dvě hlavní etapy. První etapou je seznámení se s problémem měření a sestavení koncepce senzoru, následně vytipovat vhodné komponenty a naučit se s nimi pracovat. Učení se ovládat digitální kameru a zpracovávat obraz pomocí obvodu FPGA jsem zařadil do mé bakalářské práce. Zbytek projektu, který zahrnuje vývoj měřících algoritmů, fyzickou realizaci prototypu, kompletní oživení a odzkoušení spolehlivosti měření senzoru, jsem zařadil do druhé etapy. Druhá etapa by měla být zadáním mé diplomové práce, pro kterou je tato bakalářská práce nezbytným základním kamenem.

2. Rozbor zadání

Senzor, který bude výsledkem celého vývoje, by měl dle požadavků ze zadání zvládnout několik věcí najednou. Graficky jsou požadavky vyjádřeny na Obr. 1. Měl by ovládat celý proces měření znečištění protékající vody, jejíž rychlost bude dle praktických zkušeností okolo 1,5 m/s. Zároveň bude komunikovat s okolními zařízeními po minimálně dvou typech sběrnic, a to CAN a RS232, a při možném dalším rozšíření i po sběrnici Ethernet. Je i velmi pravděpodobné, že bude senzor v budoucnu spojen i s průtokoměrem či jiným obdobným zařízením. Další činností senzoru bude obsluha a řízení GSM modemu a archivace naměřených dat na flash disk. Při předpokladu paralelního běhu všech výše popsaných činností je třeba, aby senzor byl dostatečně výkonný. Zároveň je třeba držet se požadavku nízké ceny senzoru v jeho sériové výrobě. Tyto požadavky vedly k následnému porovnání možných variant výběru kontroleru.



Obr. 1: Zapojení senzoru

2.1 Výkonný mikroprocesor v PC

Zpočátku jsem se při volbě kontroleru klonil ke klasickým procesorům, s jejichž programováním jsem měl předchozí zkušenosti. Prvotní návrh předpokládal, že nejjednodušší vývojová cesta by byla přes počítač. Vyhodnocovací jednotkou by byl osobní počítač a snímačem obrazu webová kamera. Výhodou by bylo používání libovolného programovacího jazyka nejvyšší úrovně, jako je např. JAVA, C# nebo i MATLAB. Další nespornou výhodou by bylo vyřešení celé hardwarové stránky projektu, protože připojení webové kamery k PC je dnes zcela běžná věc.

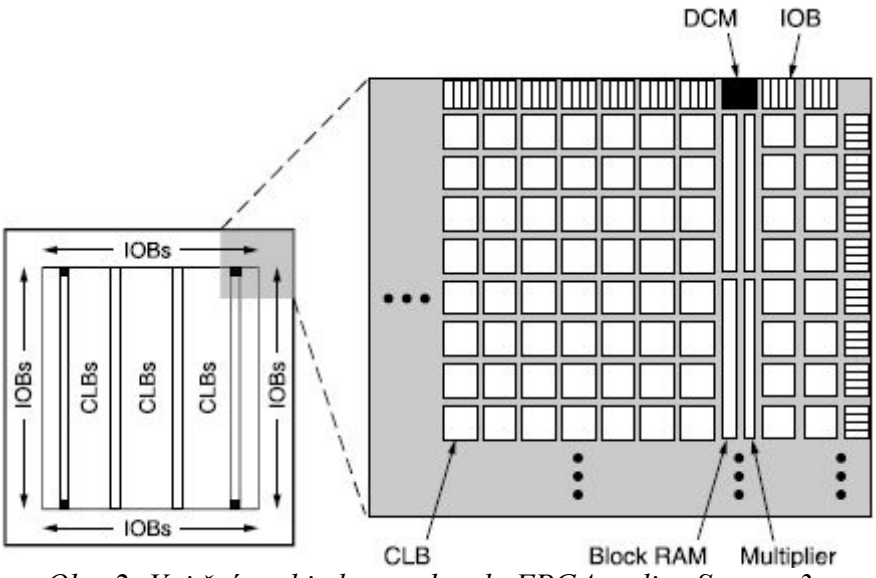
Nevýhody tohoto řešení byly, že PC není příliš vhodné na měření v reálném čase v průmyslovém prostředí, kde by měl senzor pracovat. Důvodem jsou velké rozměry a možnost rušení okolními stroji a zařízeními. Při sériové výrobě senzoru by bylo toto řešení neúnosně drahé.

2.2 *Signálový procesor*

Toto řešení spočívalo v návrhu jak programu na vyhodnocování jako u předešlého řešení, tak i návrhu hardwaru. Protože je dnes běžné využívat ke zpracování videosignálu DSP (Digitální Signálový Procesor), bylo toto řešení možné. Přesto, že je tento procesor velmi rychlý, pracuje sekvenčně, což znamená, že jeden hodinový takt odpovídá zpracování jednoho řádku programu. Pokud by byla využita kamera, která by obrazová data nekomprimovala a byla nastavena do režimu VGA s rychlostí 30 snímků za sekundu, byla by frekvence obrazových bodů vystupujících z kamery okolo 10MHz. Pokud bych chtěl splnit původní zadání senzoru s ohledem na všechny požadované funkce, vedlo by pravděpodobně toto řešení na použití několika DSP, což by učinilo návrh hardwaru i programů složitějším. Složitý hardware by pak bránil flexibilním úpravám aplikace na možné konkrétní požadavky nasazení senzoru v odlišných prostředích.

2.3 *Programovatelné hradlové pole*

Při hledání dalšího řešení jsem narazil na zadání bakalářské práce na téma Zpracování videosignálu pomocí FPGA (programovatelné logické pole - Field Programmable Gate Array). Tato možnost mě přivedla ke studiu vlastností tohoto obvodu a možností jeho využití. Vnitřní architekrura obvodu je znázorněna na Obr. 2. FPGA je programovatelný obvod, který neobsahuje jednu aritmetickologickou jednotku jako klasické či DSP procesory, ale velké množství nezávislých CLB (Configurable Logic Block). Tyto CLB implementují logickou funkci i paměťové elementy. Tento fakt umožňuje používat každý CLB pro jinou funkci, což je extrémní pojetí. Ve skutečnosti je však možné spojit libovolný počet těchto CLB do většího bloku, který má svoji funkci a zpracovává svá data.



Obr. 2: Vnitřní architektura obvodu FPGA rodiny Spartan3

Je to jistá obdoba objektového programování na PC. Zde je však možné, aby jednotlivé bloky vykonávaly svou práci současně, což vede k paralelnímu zpracování dat a tedy k paralelnímu programování.

Tyto obvody se běžně používají v mnoha typech aplikací. Příkladem aplikací mohou být univerzální řadiče sběrnic, řízení autonomních robotů, multimediální systémy v automobilech či jako emulátory starých typů počítačů, jako např. ZX Spectrum, Atari 2600 nebo jednotka televizních her. Nasazení obvodů FPGA je široké jak v oblasti typů aplikací, tak v oblasti typů pracovních prostředí. Obvody mohou pracovat například v domácnostech, mobilních zařízeních s nízkou energetickou náročností či v průmyslovém prostředí.

Výhodou je, že není třeba navrhovat plošný spoj s mnoha obvody, protože tento problém vyřeší použití FPGA, které se dá při tomto úhlu pohledu považovat za univerzální plošný spoj, jehož elektrický obvod se programuje.
Nevýhodou tohoto řešení je nutnost naučit se programovat tento obvod, což zvyšuje pracnost vývoje senzoru.
Podrobný popis obvodu FPGA použitého v této aplikaci je v [11].

2.4 Vzájemné zhodnocení návrhů

Tři výše popsané návrhy jsou navzájem porovnány v tabulce (Tab. 1).

Kontrolér	Nasazení v průmyslovém prostředí	Pracnost návrhu	Koncová cena	Možnost modifikací
Počítač	--	++	--	++
DSP	+	--	-	-
FPGA	+	-	+	+

Tabulka Tab.1: Porovnání navrhovaných kontrolerů

Z tabulky vyplývá, že i když je počítač vhodný v oblasti návrhu a možných modifikací, nelze toto řešení použít pro nevhodnost nasazení v průmyslovém prostředí a kvůli vysokým nákladům, což byly silné faktory vedoucí k zamítnutí tohoto řešení. Proti použití DSP je složitost návrhu hardwaru, jehož každá změna by byla nákladná a pracná.

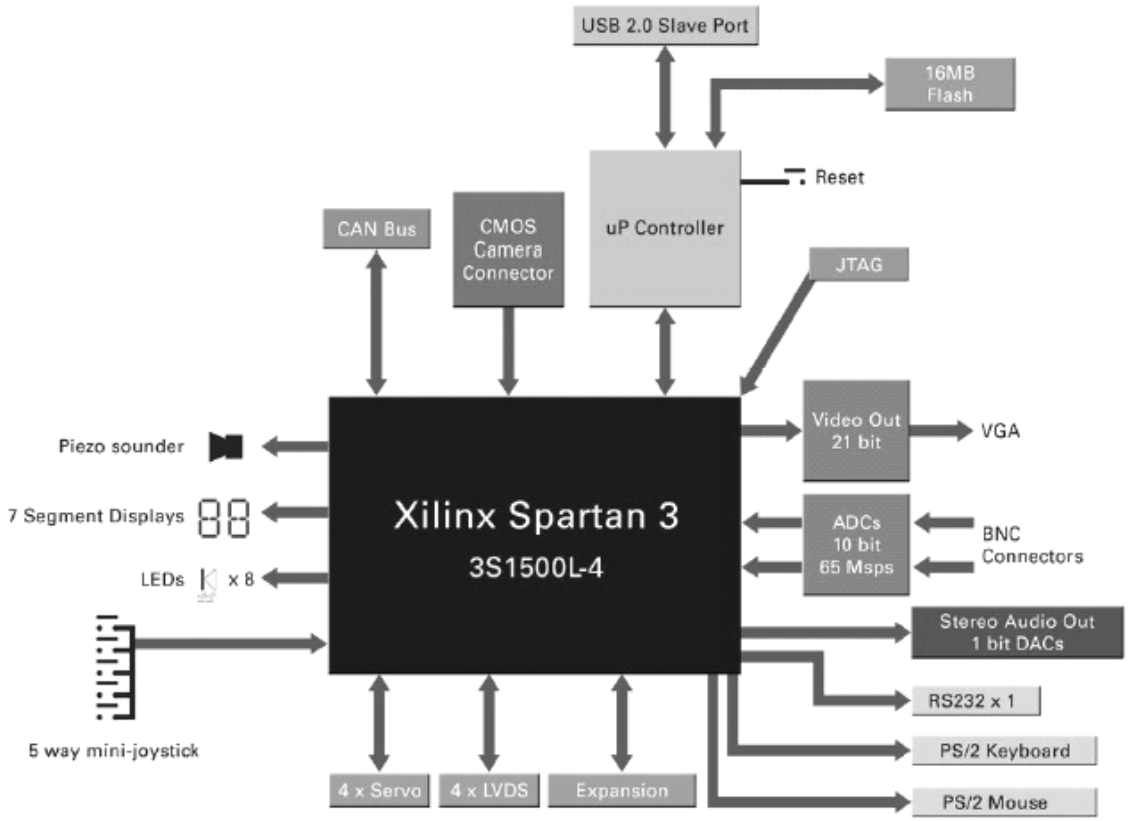
Téměř všechny nevýhody předešlých dvou návrhů řeší použití FPGA. Nasazení v průmyslu se již osvědčilo a cena odvodu je nízká (Xilinx XC3S1500 dnes stojí okolo 80$). Náklady na okolní podpůrné obvody jsou také nižší, protože je možné většinu funkcí senzoru vykonat v jediném obvodu FPGA. Tento aspekt vede i k možnosti rozsáhlejších modifikací, které se provedou jen programově.

Při předložení těchto návrhů zadavatelské firmě byla zvolena právě třetí varianta s obvodem FPGA.

3. Popis hardware

3.1 Vývojová deska

Pro vypracování této bakalářské práce mi vedoucí práce poskytl vývojovou desku RC10 od firmy Celoxica [7]. Tato deska je vhodným prostředkem k seznámení se s programováním FPGA typu Spartan3 a vývoji aplikace popsané v zadání práce. Deska obsahuje všechny potřebné komponenty, jak je vidět na blokovém schématu na Obr. 3. Pro mou aplikaci jsem využil samozřejmě obvod FPGA (kap. 3.5), dále kameru (kap. 3.2), port RS232 (kap. 3.4) a převodník VGA (kap. 3.3).



Obr. 3: Blokové schéma desky RC10

Deska je osazena krystalem o frekvenci 48MHz, což je i základní pracovní frekvence pro obvod FPGA. Dále je osazena tlačítkem Reset, kterým se konfigurace FPGA vymaže, a USB port, který poskytuje desce napájení a přes který se přenáší na desku RC10 konfigurační *.bit soubor, kterým se provádí nastavení hradel v obvodu FPGA.

3.2 Kamera

Vývojová deska je osazena kamerou se snímacím CMOS čipem typu OV9650 od firmy OmniVision [1], [2]. OV9650 je vysoce výkonný čip pro obrázkové či video kamery [5], [6]. Čip obsahuje obrazové pole o velikosti 1280 x 1024 (SVGA) a 10bitový A/D převodník, schopný zpracovat až 15 obrázků v plném rozlišení za sekundu. Dále obsahuje mikročočku a regulaci clony. Patentovaná senzorová technologie používá pokročilé algoritmy k odstranění stálého vzorkovacího šumu (FPN), k potlačení rozmazání obrazu a k významnému snížení chybovosti. Řídící registry dovolují flexibilní kontrolu časování, polarity a operací čipu, který umožňuje uživateli velkou svobodu v návrhu produktu. Základní technické parametry kamery popisuje tabulka Tab.2. Komunikace s kamerou probíhá přes sběrnici SCCB (Serial Camera Control Bus), jejíž fyzická vrstva i protokol jsou velmi podobné známé sběrnici IIC [10].

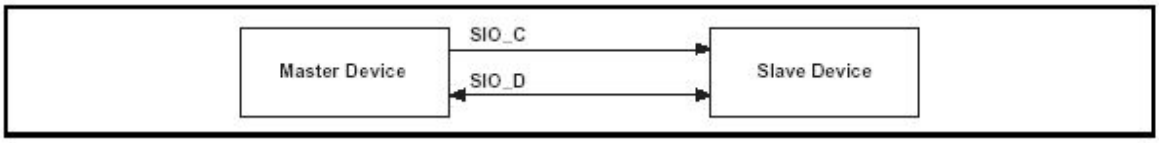
	Active Array Size	1300 x 1028
Power Supply	**Core**	1.8VDC ± 10%
	Analog	2.45 to 2.8 VDC
	I/O	2.5V to (V_{DD-A}+0.3V)
Power Requirements	**Active**	50 mW (15 fps, no I/O power)
	Standby	30 µW
Temperature Range	**Operation**	-20°C to 70°C
	Stable Image	0°C to 50°C
	Output Formats (8-bit)	• YUV/YCbCr 4:2:2 • GRB 4:2:2 • Raw RGB Data
	Lens Size	1/4"
Maximum Image Transfer Rate	**SXGA**	15 fps
	VGA	30 fps
	QVGA, QQVGA, CIF	60 fps
	QCIF, QQCIF	120 fps
	Sensitivity	0.9 v/Lux-sec
	S/N Ratio	40 dB
	Dynamic Range	62 dB
	Scan Mode	Progressive
	Maximum Exposure Interval	1050 x t_{ROW}
	Gamma Correction	Programmable
	Pixel Size	3.18 µm x 3.18 µm
	Dark Current	30 mV/s at 60°C
	Well Capacity	28 K e
	Fixed Pattern Noise	<0.03% of $V_{PEAK-TO-PEAK}$
	Image Area	4.13 mm x 3.28 mm
	Package Dimensions	5095 µm x 5715 µm

Tab.2: Technické parametry kamerového čipu

3.2.1 Sběrnice SCCB

OmniVision má definovanou třídrátovou komunikaci pro řízení většiny svých kamer. V redukovaném stavu pinů na pouzdru se využívá jen dvoudrátové spojení (Obr. 4).

Kamera operuje pouze jako slave. Modifikované dvoudrátové spojení umožňuje masteru ovládat jen jeden slave. Tento režim se provozuje na čipech, kde není zvenku na pouzdru přístupný pin ENABLE, který je vnitřně nastaven.



Obr. 4: Dvoudrátové zapojení sběrnice SCCB

Popis SCCB a řídících signálů použitých v aplikaci:

SIO_C - vstupní hodiny, obsluhuje master
- jednosměrný, aktivní na logické úrovni 1, řídící signál ovládaný z masteru. Indikuje každý přenesený bit. Master musí nastavit hodnotu log1, pokud není sběrnice používána. Přenos dat začne, když SIO_C je nastaven na log0. Log1 na SIO_C během přenosu indikuje přenesený bit. Data na SIO_D se mohou měnit jen pokud je SIO_C na log0. Perioda přenosu se nazývá t_{CYC} a její minimální hodnota je 10us.

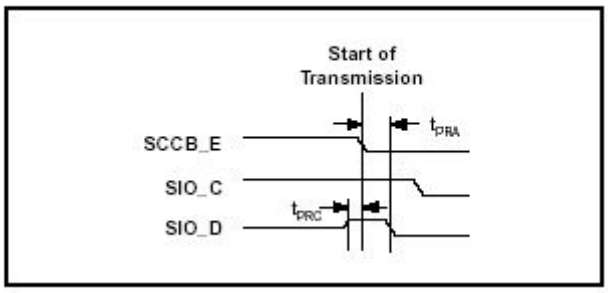
SIO_D - obousměrný datový signál, který může být řízen masterem i slavem. Hodnota na sběrnici je plovoucí nebo ve třetím stavu, pokud není sběrnice využívána. Správa signálu je úkolem masteru i slave tak, aby nedošlo k nedefinovanému stavu.
- plovoucí hodnota a spor jsou povoleny jen během přenosu Don´t-Care bitu nebo NA bitu, viz níže. Master se musí vyhnout neznámým stavům na sběrnici a konfliktům. Požaduje se ochrana proti konfliktu ke snížení statického proudu v podobě ochranného rezistoru.

PWDN – vstup – pokud je na pin přiveden signál s logickou hodnotou 1, pak se kamera přepne do spacího režimu (Power Down Mode)

RESET - vstup - pokud je na pin přiveden signál s logickou hodnotou 1, tak je kamera uvedena do počátečního stavu včetně obsahu řídících registrů

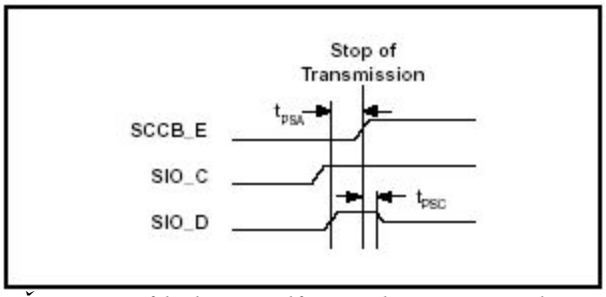
3.2.2 Přenos dat

Začátek vysílání (Obr. 5) – SIO_D přejdou z neutrálního stavu na log1. SIO_C jsou na log1. Tento stav musí vydržet aspoň 1,25us, pak datový signál je nastaven na log0. Po pauze délky opět 1,25us dlouhé je nastaven na log0 i signál SIO_C. Pak už následuje datová komunikace, tedy periodické měnění dat a jejich potvrzování pomocí signálu SIO_C.



Obr. 5: Časový průběh signálů při začátku vysílání zprávy

Konec vysílání (Obr. 6) začíná nastavením signálu SIO_C na log1. Po čekací době t_{PSA} se i SIO_D nastaví na log1. Pak je pauza min 15ns. Potom se SIO_D přepne do neutrálního stavu. SIO_C a Enable zůstávají na log1 až do dalšího přenosu.

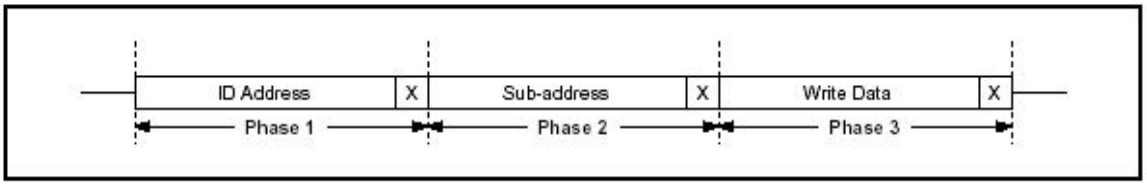


Obr. 6: Časový průběh signálů při ukončení vysílání zprávy

Základním prvkem přenosu je tzv. fáze. Každá fáze obsahuje 9bitů. Prvních 8bitů tvoří datové slovo a poslední 9. bit je Don´t-Care nebo NA bit v závislosti na směru přenosu. Maximální počet fází v přenosu je 3. MSB je vždy přenášen v každé fázi jako první.

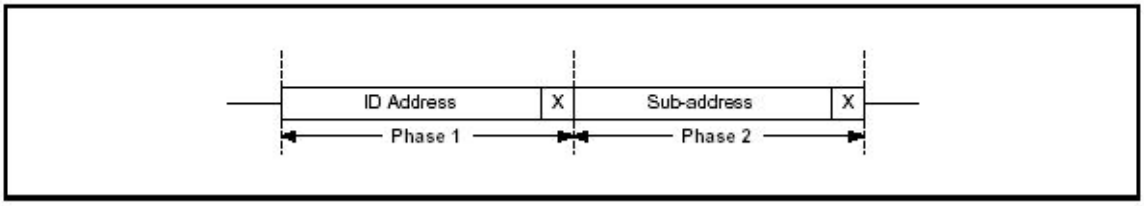
Jsou definovány tři druhy přenosů:

1. **3-fázové zapsání dat do kamery** (Obr. 7)
   Je to plný zapisovací cyklus, kdy master může zapsat jeden byte do vybraného slave. ID adresa určuje slave, do kterého bude master zapisovat. Podadresa určuje registr ve vybraném slavu. V zaadresovaném registru dojde k přepsání hodnoty novým bytem. Devátý bit se při zápisu nazývá Don´t-Care bit.



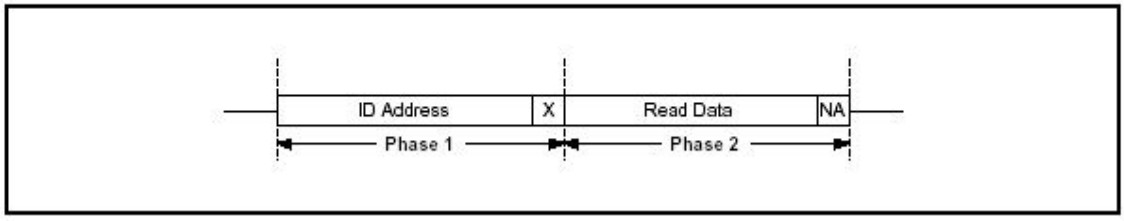
Obr. 7: Diagram třífázové zprávy při zápisu dat do kamery

2. **2-fázový zápis do kamery** (Obr. 8)
   Tento dvoufázový zapisovací přenos je následován 2-fázovým čtecím cyklem. Úkolem tohoto vysílání je identifikace podadresy ve slave, ze kterého bude chtít master v dalším cyklu číst. Devátý bit se zde nazývá Don´t-Care bit.



Obr. 8: Diagram dvoufázové zprávy při zápisu do kamery

3. **2-fázové čtení dat z kamery** (Obr. 9)
   Před každým 2-fázovým čtecím cyklem musí proběhnout buď 3-fázový nebo 2-fázový zápis, aby došlo k zaadresování registru, protože 2-fázové čtení neumí zaadresovat registr. První byte vysílá master do kamery. Následně musí dojít k přepnutí signálu SIO_D do třetího stavu a kamera začne vysílat data, která master čte. Fáze obsahují 8 bitů s informací a devátý bit se nazývá Don´t-Care nebo NA bit. Master musí držet NA bit na log1.



Obr. 9: Diagram dvoufázové zprávy při čtení z kamery

3.3 VGA výstup

Převodník je přímo napojen na obvod FPGA [7]. Obsahuje tři kanály pro tři základní barvy režimu RGB (Červená Zelená Modrá). Každý kanál používá k převodu na analogový signál 7 bitová slova. Číslicově analogový převod je proveden tak, že na každý ze sedmi bitů každého slova je připojen rezistor. Pokud je příslušný bit ve slově na logické úrovni 1, pak je přes rezistor přivedeno do společného uzlu pro každou barvu příslušné napětí. To odpovídá váze bitu a tedy i hodnotě odporu rezistoru. Výstup dále obsahuje dva signály na řádkovou a snímkovou synchronizaci monitoru.

3.4 RS232 port

Tento komunikační kanál je realizován programovým blokem v obvodu FPGA. Na desce je pak pro převod napěťových úrovní mezi LVTTL a RS232 osazen obvod ICL3222ECA od firmy Intersil [9]. Ten je s FPGA spojen signály TransmitData, RecieveData a řídícími signály Clear To Send a Request To Send [8].

3.5 Obvod FPGA

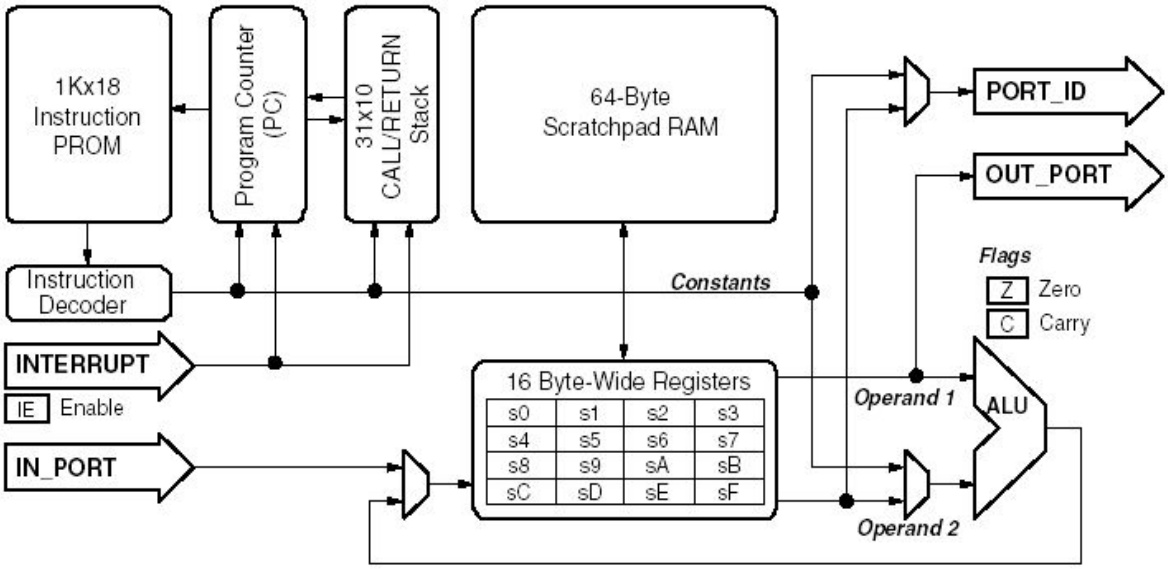
Na desce je osazen obvod FPGA Obr. 2 od firmy Xilinx [11]. Tento obvod patří do rodiny typu Spartan3, která se vyznačuje 90nm osmivrstvou kovovou technologií a velkým výkonem s taktovací frekvencí přes 300MHz. Dle výrobce je určena také pro zpracování video signálů. Obvod osazený na desce patří do základní skupiny rodiny Spartan3 a je typu XC3S1500. Pouzdro obvodu o rozměrech 19x19 mm obsahuje 1,5 milionu systémových hradel. Velká část z nich je spojena do tzv. konfigurovatelných logických bloků (Cofigurable Logic Blocks – CLBs), kterých je celkem 3328. Ty obsahují programovatelné kombinační obvody Look Up Table – LUT, kde každé kombinaci vstupních dat odpovídá logická informace L/H na výstupu podobně jako u pamětí ROM. Dále jsou v CLB obsaženy klopné obvody. Propojení mezi jednotlivými členy CLB zajišťují programovatelné multiplexery. CLB se sdružují do matic – polí. Další část systémových hradel tvoří Block RAM o celkové kapacitě 576Kbit a 487 uživatelských vstupů/výstupů.

Pro programování tohoto obvodu je možné využít několik jazyků. Mezi vysoké programovací jazyky patří například Celoxica HandleC, což je jazyk C upravený pro paralelní programování. Deska RC10 podporuje použití tohoto jazyka vlastní ovládací knihovnou, s jejíž pomocí se pak dají jednotlivé periferie přímo ovládat hotovými makry. Jazyky nižžší jsou například Verilog, používaný převážně v USA, či VHDL, používaný v Evropě. Abych splnil podmínku zadání, týkající se nízkých vývojových nákladů, zvolil jsem pracnější variantu použití jazyku VHDL, protože nástroj a vývojové prostředí je volně dostupné.

4. Popis vývojových prostředí a software

4.1 Mikrokontroler PicoBlaze3

Mikrokontroler PicoBlaze3 [14] je kompaktní osmibitové RISC mikrokontrolerové jádro, optimalizované také pro FPGA rodiny Spartan3. PicoBlaze3 je optimalizován na výkon a velikost tak, že zabírá jen okolo 1% použitého Spartanu XC3S1500. V typickém použití zabírá mikrokontroler jednu blokovou RAM, do níž je možné uložit program o délce 1024 instrukcí, které jsou automaticky načteny během konfigurace FPGA. Rychlost mikrokontroleru je v této aplikaci 24 milionu instrukcí za sekundu (MIPS). Vnitřní uspořádání PicoBlazu3 je vidět na Obr. 10. Mikrokontroler obsahuje jednoduchou aritmetickologickou jednotku pro dva operandy, 16 jednobytových registrů, 64 bytovou paměť a vstupní a výstupní port s až 256 adresami. Mikrokontroler také podporuje funkci přerušení.



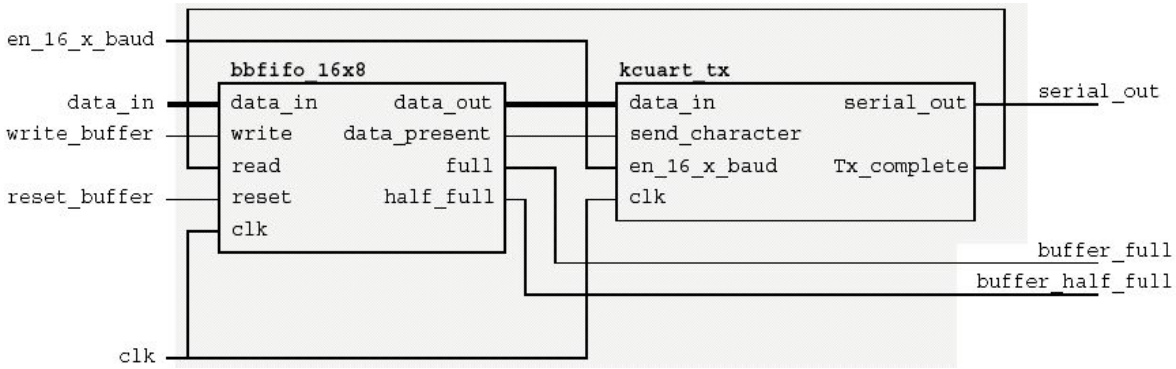
Obr. 10: Blokové schéma vnitřního uspořádání mikrokotroleru Pico Blaze

Výhodou mikrokontroleru je, že je napsán v jazyce VHDL a je tedy možné si jej upravit dle vlastních potřeb.

Psaní programu pro mikrokontroler je možné v jakémkoli textovém editoru. Souboru s kódem je pak třeba změnit příponu na *.psm a tento soubor je následně možné přeložit pomocí překladače KCPSM3.EXE. Výsledkem je několik souborů. Hlavním je soubor, jehož obsahem je konfigurace blokové paměti obvodu FPGA v jazyce VHDL. Ten se následně přidá do projektu v prostředí ISE na přesně definovaném místě. Pro ladění a zkoušení programu pro PicoBlaze3 slouží nástroj pBlaze IDE od společnosti Mediatronix. Tento nástroj umožňuje ladit program klasickým způsobem sekvenčního programování. Program je možné krokovat, sledovat obsahy registrů a paměti a program kdykoli pozastavit. Tyto možnosti u programování ve VHDL nejsou.

4.2 UART moduly

Pro komunikaci přes sériovou linku jsou volně dostupné moduly s UART vysílačem a přijímačem, vytvořené jako periferie pro PicoBlaze3 Kenem Chapmanem [13]. Blokové schéma vysílače s FIFO zásobníkem spolu s řídícími signály je možné vidět na Obr. 11.



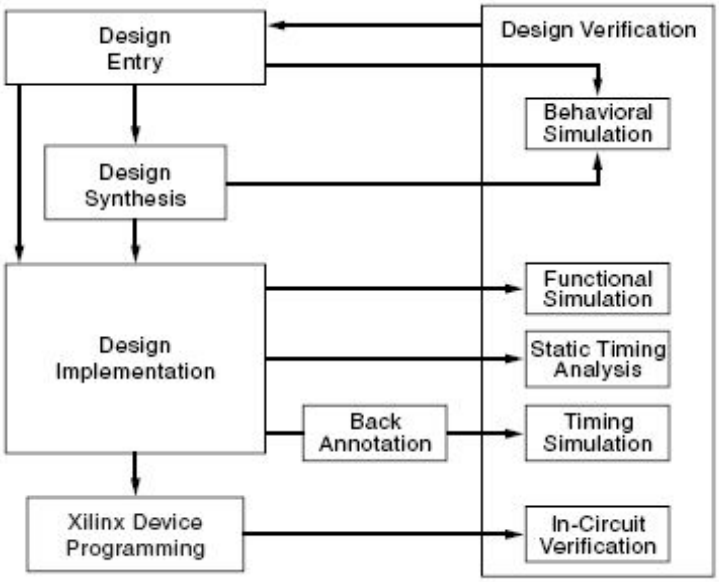
Obr. 11: Blokové schéma makra UART vysílače

Modul je zde ve významu programového bloku ve vlastním souboru, které se ovládá přes několik řídících signálů. V manuálu jsou moduly popisovány jako makra optimalizovaná pro FPGA Xilinx, zejména pro rodiny Spartan3 a Virtex. Makra nepoužívají signály řídící tok dat v kanále (signály RTS a CTS). Každé makro je složeno z vysílače či přijímače a First In First Out (FIFO) zásobníku na zprávy o obsahu 16 byte. Makra potřebují ke své činnosti připojení na signál s pracovními hodinami, zde 48MHz, a na signál s hodinami, které jsou využity ke vzorkování seriové linky. Nastavením frekvence tohoto signálu se určí vysílací či přijímací rychlost maker. Maximální komunikační frekvence může být až 1/16 frekvence pracovních hodin, což v tomto případě je 3Mbit/s. Tento hodinový signál je třeba vytvářet mimo makra. Parametry rámce komunikačního protokolu jsou jeden start bit, slova délky 8 bitů (přenos je seriový a první je posílán nejvyšší bit (MSB)), žádná parita a jeden stop bit. Výhodou je, že není třeba tento problém znovu vymýšlet a navíc tato makra jsou optimalizována a zabírají minimální počet CLB.

4.3 Prostředí ISE

Vývojové prostředí WebPack Xilinx Integrated Software Environment (ISE) 8.2 je volně dostupný nástroj na práci s obvody FPGA v jazyce VHDL [15], [3], [12]. Jazyk VHDL jsem zvolil, protože je standardně využíván v Evropě na rozdíl od jazyku Verilog, používaném především v USA. ISE je složeno z několika částí a mnoha funkcí, jež je možné vidět ve schématickém znázornění na Obr. 12. Editor slouží k psaní kódu a řazení bloků programu do hierarchické struktury. Tyto zdrojové kódy je pak možné syntetizovat dohromady pomocí překladu prvního stupně. Ten zkontroluje syntaxi a převede text programu do podoby fyzických hardwarových bloků a hradel. Tuto fyzickou podobu kódu je pak možné prohlížet v grafickém prohlížeči. Překlad druhého stupně rozmísťuje fyzické

bloky z předchozího překladu na přesně adresovaná hradla obvodu FPGA v závisloti na zadaném typu obvodu. Výsledek tohoto překladu je model, který si lze také prohlédnout v grafickém prohlížeči. Tam můžeme vidět, kde přesně budou bloky uvnitř obvodu umístěny. Poslední třetí překlad pak vytvoří z modelu konfigurační soubor *.bit, kterým se přes speciální program a port USB nakonfigurují hradla v obvodu na požadované funkce.



Obr. 12: Grafické znázornění prostředí ISE

4.4 Prostředí ModelSim

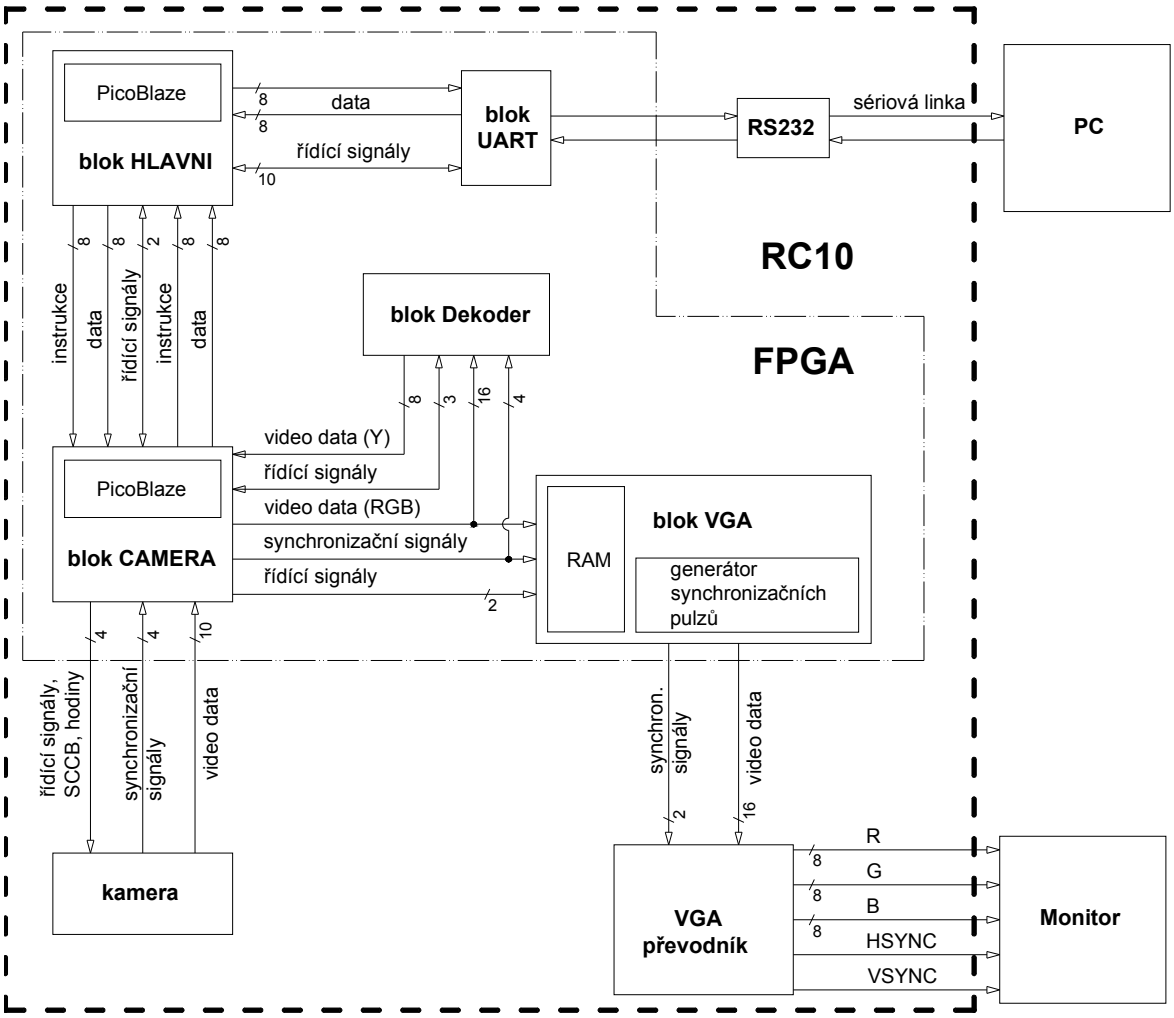
Program ModelSim Xilinx Edition III v6.1e od společnosti Mentor Graphics, který je také volně dostupný, je simulátor chování napsaného kódu. Tento simulační program zpracoval zdrojové kódy VHDL jazyka dohromady se speciálním souborem, vytvořeným také v ISE, kterým si uživatel definuje signály na vstupech. Výsledkem simulace je grafické znázornění chování a reakce modelu FPGA na vstupní signály. Tyto simulace slouží k ladění programů na počítači a není tak třeba zkoušet je přímo na hardwaru, kde někdy není chování aplikace díky její rychlosti pozorovatelné. V mé aplikaci jsem simuloval téměř všechny části kódu.
Tento nástroj je velmi vhodný pro ladění menších celků. Při ladění větších celků jsou často potřeba simulace delší doby běhu kódu a složitější namodelování vstupních signálů.

5. Koncepce aplikace

Zadání bakalářské práce jsem si rozdělil do několika funkčních částí, které se lépe tvoří a ladí samostatně. Jsou to části nazvané dle jejich funkcí:

- připojit kameru k mikrokontroleru PicoBlaze3,
- načítat obrazová data z kamery,
- zobrazit vyčtená data na monitoru,
- zobrazovat změny obrazových dat v PC (jen kontrolní funkce),
- komunikace mezi PC a aplikací pomocí sběrnice RS232.

Po prostudování všech dostupných materiálů a zvážení možných detailů a návrhů jsem došel ke koncepci aplikace, jejíž grafická podoba je na Obr. 13.



Obr. 13: Blokové schéma aplikace

Z pracovního zadání vyplynulo, že budu muset použít jeden mikrokontroler PicoBlaze3, který bude komunikovat přímo s kamerou. Protože zpracovává data

sekvenčním způsobem, což by mohlo později vyvolávat časově kritické problémy, určil jsem tento mikrokontroler jen pro řídící komunikaci s kamerou přes kanál SCCB a obrazová data jsem nechal zpracovávat přímo modulem ve VHDL. Tento blok obsahující PicoBlaze3 spolu s podpůrnými funkcemi psanými ve VHDL jsem nazval jako blok CAMERA.

Jako hlavní řídící prvek aplikace jsem zvolil také PicoBlaze3, protože počítám s dalším vývojem. Při použití tohoto řešení zůstává zapojení mikrokontroleru do aplikace stejné a vyměňuje se jen VHDL soubor s programem. Toto řešení jsem zvolil i z důvodu, že zde není potřeba řešit časově kritické úkoly a tedy 24 MIPS je dostačující, a z důvodu snadnějšího ladění kódu v sekvenčním prostředí. Tento mikrokontroler spolu s podpůrnými funkcemi jsem nazval blok HLAVNI.

Hierarchické uspořádání jsem zvolil tak, že PicoBlaze3 HLAVNI pracuje v režimu master a PicoBlaze3 CAMERA je v režimu slave. Znamená to, že blok CAMERA nemůže začít sám vysílat a musí jen odpovídat na příkazy od HLAVNI.

Problém s komunikací přes sériovou linku RS232 jsem na doporučení vedoucího práce vyřešil použitím hotových volně dostupných bloků. Jedná se UART Transmitter and Reciever Macros (kap.4.2). Je to pár hotových programových bloků, popsaných v samostatných souborech jazykem VHDL. Každé makro je složeno z jednoduchého vysílače či přijímače a FIFO zásobníku.

Obrazová data z kamery předzpracovává ještě blok CAMERA, avšak již ne v mikrokontroleru, ale jen podpůrnou funkcí. Ta skládá obrazová data do jednotného formátu 16 bitového slova, odpovídajícího popisu jednoho obrazového bodu (pixelu), a generuje příslušný hodinový signál, který určuje platnost jednotlivých pixelů.

Pro zobrazení obrazových dat na monitoru slouží blok nazvaný blok VGA. Zde jsou data přijatá z bloku CAMERA uložena do mezipaměti předtím, než jsou odeslána do VGA převodníku a monitoru spolu se zde generovanými synchronizačními signály.

Jako výchozí bod pro další vývoj jsem do aplikace zařadil blok DEKODER, který zpracovává stejná obrazová data jako blok VGA převodník. Zde se převádí informace o pixelech z obrazového modelu RGB na obrazový model YUV, repektive jen na jasovou složku Y. Tento osmibitový signál by měl být zdrojem dat pro bloky, jejichž vývoj je zařazen do druhé etapy. V této aplikaci slouží pro kontrolu existence a změn obrazových dat poskytovaných kamerou. Proto je tento signál zaveden zpět do bloku CAMERA a pomocí příkazu z bloku HLAVNI je možné tato data posílat do řídícího počítače. Protože by byl datový tok při přenosu celého obrazu příliš velký, je počet přenesených dat na jeden snímek zredukován na jeden referenční pixel na snímek.

6. Rozbor synchronizace časování kamery a monitoru

Dle technického popisu kamery z Tab. 2 je možné vyčíst, že je schopná poskytovat data v několika rozměrových režimech. Právě při volbě režimu bylo zapotřebí zvážit několik aspektů. Hlavním aspektem byl rozměr obrazu. Protože bude senzor využit pro měření v reálném čase, není dle mého názoru příliš vhodné používat režim SVGA s rozměry 1280 x 1024 obrazových bodů, který má sice velmi kvalitní rozlišení, ale je možné využít jen 15 snímků za sekundu. Tento počet považuji v této fázi za nedostatečný, ale je možné, že jej bude nutné či možné v senzoru využít. Obraz o rozměrech VGA, tedy

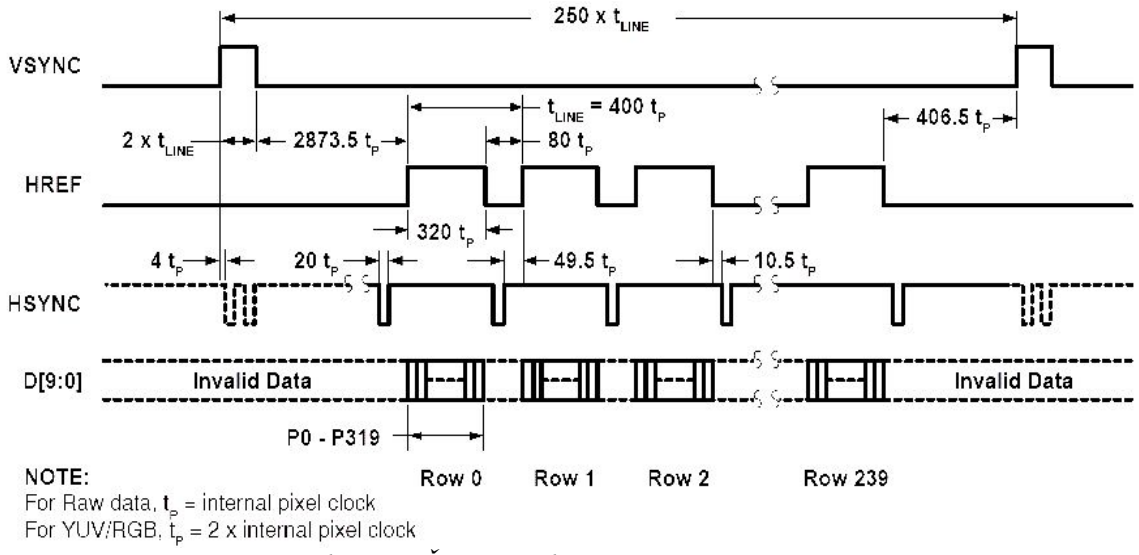
640 x 480 obrazových bodů, má sice horší rozlišení, ale nabízí 30 snímků za sekundu. Protože je tento formát velmi bežně používán a je podporován CRT monitory, zvolil jsem prvotně tento formát [4].

Obrazový formát VGA je definován rozměry 640 x 480 obrazových bodů na snímek. Pokud kamera kóduje obrazové body do formátu RGB 565, tak je každý bod popsán jedním 16bitovým slovem, jak je již popsáno výše. Dohromady má tedy jeden snímek velikost 640 x 480 = 307200 obrazových bodů. Při uložení tohoto snímku by bylo tedy potřeba 4,915 Mbitů. Obvod FPGA má však paměť o celkové kapacitě 576Kbitů. Z toho vyplývá, že nelze uložit ani jeden celý snímek. Proto tedy nelze realizovat mechanismus, který funguje v grafických kartách počítačů. Tam je obraz uložen v paměti a periodicky vyčítán do monitoru. Uložený obraz je pak možné kdykoli změnit.

Uložit celý snímek v obvodu tedy nelze a externí rychlá paměť na desce RC10 není k dispozici. Proto je třeba obrazová data z kamery vyčítat a ihned je bez skladování posílat do monitoru. Tento problém však naráží na časování monitoru.

Základem CRT monitoru je trojité elektronové dělo, které se periodicky zaměřuje na každý bod na stínítku. Proudem elektronů nabije energií odpovídající jasu příslušného bodu a jeho tří barev luminofor v tomto bodě. Luminofor od tohoto okamžiku začne díky dodané energii vyzařovat světlo složené ze tří základních barev, odpovídajících barevnému modelu RGB. Energie v luminoforu se mění na světlo relativně rychle, a proto je třeba na každý bod posvítit znovu a dobít tak jeho energii. Pokud se energie luminoforu dodává s malou frekvencí opakování, dochází ke zhasnutí bodu na stínítku. Tento jev zapříčiňuje nepříjemné blikání obrazu na monitoru. Proto se nedoporučuje obnovovat energii každého bodu na monitoru méně jak 50krát za sekundu.

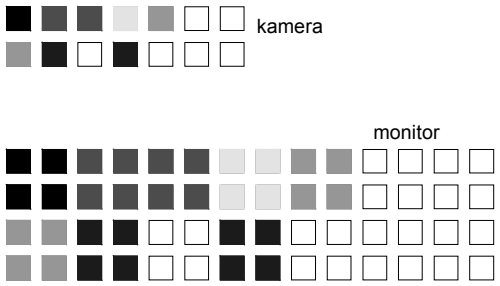
Tento důvod znemožňuje posílat data z kamery přímo do monitoru, protože kamera umí obraz o velikosti VGA poskytnout je 30krát za sekundu. Z této úvahy tedy vyplývá, že je potřeba mít stejnou snímkovou frekvenci jak na kameře, tak na monitoru. Toto řešení nabízí obrazový režim QVGA o rozměrech obrazu 320 x 240 obrazových bodů. Tento obraz umí kamera poskytnout v požadované frekvenci 60 snímků ze sekundu. Grafické znázornění časování kamery při tomto režimu je zobrazeno na Obr. 14.



Obr. 14: Časování kamery v režimu QVGA

Z diagramu na Obr. 14 lze spočítat, že vyčtení jednoho obrazového řádku z kamery trvá po dobu t_{LINE}=66,66us. To je však čas, po který přesně trvá vykreslení dvou řádků obrazu na monitoru viz [4]. Při přímém propojení kamery a monitoru se ukázalo, že je toto řešení nereálné, protože obraz byl složen ze dvou vzájemně proložených snímků. Každý snímek odpovídal geometrické polovině snímku kamery a tak vznikal v obraze tzv. „duch“.

Nakonec jsem došel k funkčnímu řešení. Je třeba zachovat stejnou snímkovou frekvenci jak u kamery, tak u monitoru, a to 60 snímků za sekundu, aby obrazová data obvodem FPGA jen protékala, protože není možné celý snímek uchovávat. Porovnáním obou režimů vyjde najevo, že snímek v režimu VGA má čtyřikrát více obrazových bodů než snímek v režimu QVGA. Proto je třeba snímek z kamery roztáhnout, čehož se dosáhne čtyřnásobným zobrazením jednoho obrazového bodu z kamery. Aby obraz neztratil smysl, nelze jeden bod z kamery zobrazit čtyřikrát za sebou v jednom řádku. Protože je obrazový bod z kamery tvaru čtverce, je třeba tento tvar zachovat a zobrazovat bod z kamery v jednom řádku na monitoru jen dvakrát za sebou a celý řádek pak zobrazit také dvakrát za sebou. Graficky je tento mechanismus znázorněn na Obr. 15.



Obr. 15: Mechanismus roztažení obrazu

Tento předpoklad je třeba ještě zvážit po stránce časování. Jak již bylo napsáno výše, trvá vyčtení jednoho obrazového řádku z kamery stejně dlouho jako vykreslení dvou řádků obrazu na monitoru. To naznačuje, že je možné stíhat zdvojování řádků a tedy i roztažení obrazu ve směru svislém. Aby byl dodržen předpoklad čtyřnásobného zobrazení každého bodu z kamery, jak ukazuje Obr. 15, je třeba roztáhnout obraz i ve směru vodorovném. Toho se dosáhne zobrazením každého bodu v každém řádku dvakrát za sebou. Problém však nastává, pokud si představíme vzájemné časové průběhy a poměry rychlostí. Sice lze stihnout vše dle porovnání řádkové frekvence, ale je třeba mít po celou dobu 66,66 us k dispozici neměnná správná data, která se budou vykreslovat na monitoru. Toho se dá dosáhnout uložením jednoho řádku do paměti v obvodu FPGA. Pokud tedy vezmu jednu BRAM o rozměru 16 x 640 bitů, mohu do ní uložit jeden celý zdvojený řádek obrazových bodů z kamery tak, že každý uložím dvakrát za sebou. Tuto paměť pak dvakrát přečtu a zobrazím na monitoru. Tím je vyřešeno i roztažení obrazu QVGA na obraz VGA.

Řešení pomocí jedné BRAM by však nefungovalo, protože by docházelo k přepisování dat v paměti během jejího vyčítání. To je sice proveditelné, protože je použita dvoubránová BRAM, ale silně by to poškodilo smysl obrazu. Z tohoto důvodu jsem použil paměti dvě. Výhodou je pak paralelní běh procesů, protože je možné do jedné z pamětí zapisovat a zároveň z druhé data číst s dvojnásobnou frekvencí, než se kterou probíhá zápis. Takže po dobu ukládání dat do první paměti, což trvá oněch 66,66us, je druhá paměť dvakrát přečtena a data jsou zobrazena na monitoru. Po uložení dat do první

paměti se funkce pamětí prohodí a data, která byla právě uložena do první paměti, začnou být čtena a zobrazována na monitoru. Do paměti druhé se začnou současně ukládat nová data z kamery. Tento proces probíhá po dobu zobrazování jednoho celého snímku. S každým novým snímkem se tento proces opakuje od začátku. Z popisu je jistě jasné, že první vyčítání paměti je odstartováno až po jejím prvním naplnění daty, takže vzniká skluz mezi skutečným časem přečtení obrazového bodu v kameře a jeho zobrazením na monitoru. Skluz je přesně 66,66us, což je v této demonstrační aplikaci zanedbatelné.

7. Rozbor jednotlivých bloků programu

7.1 Blok TOP_propojeni

Tento blok je na nejvyšším hierarchickém stupni všech bloků ve struktuře kódu. Jeho funkcí je jen propojit níže popsané bloky dohromady a přiřadit vstupům a výstupům aplikace konkrétní piny obvodu FPGA. Grafickou podobu tohoto bloku je možné nalézt v příloze. Tento blok neobsahuje žádný proces, takže není považován za výkonný. Proto není zakreslen jako položka v blokovém schématu na obrázku 13. Nejhlavnějším výkonným blokem je tedy blok HLAVNI.

7.2 Blok HLAVNI

Tento blok je určen k řízení celé aplikace, a proto je na nejvyšším místě hierarchické struktury všech výkonných bloků. Je zde prvotně zpracovávána komunikace s PC, k níž je využíván přímo podřízený a ovládaný blok UART. Druhým přímo podřízeným blokem je blok CAMERA, který slouží ke komunikaci s kamerou a k řízení bloků zpracovávajících obrazová data. Komunikace mezi bloky HLAVNI a CAMERA je realizována pomocí paralelního přenosu řídícího znaku a dat. V každém směru komunikace je použito sedmnácti signálů. Prvních osm je určeno pro paralení přenos řídícího znaku ve formátu ASCII, který přesně definuje postup zpracování dat zasílaných pomocí druhých osmi signálů. Poslední signál slouží k přenosu potvrzovacích pulzů. Zpráva mezi těmito bloky se tvoří tak, že mikrokontroler nejprve pošle na sběrnici řídící znak a v dalším kroku pošle na druhou polovinu sběrnice data. V tomto okamžiku je i vygenerován potvrzovací pulz. Tento způsob komunikace jsem zvolil ze dvou důvodů. Prvním je, že zpráva je zaslána a potvrzena v jednom hodinovém taktu, což zjednodušuje zpracování zprávy. Druhým důvodem je možnost v budoucnu zaměnit mikrokontroler PicoBlaze3 za jiný typ kontroleru, například soft-core procesor MicroBlaze. Tento mikrokontroler má 32-bitovou architekturu, takže bude moci bez problémů odeslat jednu zprávu najednou a jeho případné budoucí připojení bude značně zjednodušeno. Protože blok HLAVNI je v pozici Master, začíná komunikaci právě on zasláním instrukce a dat. Většinou se jedná o zprávu, kterou sestavil na základě příkazu přijatého z počítače. Blok CAMERA přijetí zprávy nepotvrzuje. Protože pro tento blok není tato zpráva nejvíce prioritní, doba akceptování zprávy je závislá na aktuálním stavu programu. Po zpracování zprávy a vykonání případných úkolů je případně odeslána do bloku hlavní zpráva s výsledkem úkolu a následně vždy i potvrzovací zpráva, že úkol byl proveden správně a je možné příjmout novou zprávu. Blok HLAVNI tyto zprávy předá bloku UART, čímž je zašle do počítače. V době mezi odesláním zprávy

do přijetí potvrzovací zprávy je v bloku HLAVNI zablokované zasílání zpráv do bloku CAMERA. Při tomto řešení odpadá problém mezipaměti na zprávy mezi bloky či průběžné potvrzování přijetí zpráv. Toto řešení je z hlediska rychlostí nekonfliktní, protože přijetí zprávy po sériové lince i při nejvyšší rychlosti trvá déle, než zpracování a vykonání řídící zprávy v bloku CAMERA.

Blok HLAVNI je sestaven z několika částí. Každá část odpovídá procesu popsaném ve zdrojovém kódu. Hlavní částí je tzv. Procesor, což je objekt sestavený z mikrokontroleru PicoBlaze3 a blokové RAM s jeho programem. K němu jsou přímo připojeny podpůrné části nazvané Demux_out a Demux_in. Procesy se starají o kódování a dekódování adres portů, kam chce Procesor data zapsat nebo odkud je chce číst. Po dekódování adresy portu pak Demux_out převezmě data od Procesoru a přiřadí je signálům, které byly určeny adresou, nebo naopak Demux_in přečte data z požadovaných signálů a předá je Procesoru.

Další částí bloku HLAVNI je proces generující komunikační hodinový signál pro blok UART. Tento proces pracuje jako dělička základního hodinového signálu koeficientem k_d. Tento koeficient je možné ovládat z počítače speciální zprávou. Popis komunikačních rychlostí a koeficientů k_d je podrobně uveden v kapitole 7.3 o bloku UART. Popis komunikace a řízení tohoto procesu je uveden v samostané kapitole 8.

7.3 Blok UART

Tento blok je složen jen z UART maker. Makra jsou plně synchronní na základní frekvenci FPGA, tedy na frekvenci 48MHz. Tento blok je řízen výhradně blokem HLAVNI. Ten zpracovává signály o stavu maker a z přijímacího makra čte přijatá data. Dále je ovládá řídícími signály a do vysílacího makra zapisuje data k odeslání. Signály o stavu maker jsou signály o zaplnění a o polovičním zaplnění FIFO zásobníků. Makro přijímající data ze seriové linky navíc poskytuje signál o tom, že jsou v zásobníku ještě nepřečtená přijatá data.

Řídící signály společné pro obě makra jsou reset FIFO zásobníků a hodinový signál určující komunikační rychlost na sériové lince. Tento hodinový signál je generován v bloku HLAVNI dělením základního hodinového signálu FPGA. Dle pokynů z materiálů k UART makrům musí být jeho frekvence 16krát větší než je skutečná rychlost seriové linky. Tento požadavek vyplývá z vnitřního uspořádání maker.

Výběr komunikačních rychlostí seriové linky jsem zvolil tak, abych pokryl případné budoucí potřeby na nasazení aplikace. Celkem jsem zvolil 5 komunikačních rychlostí. Jako nejvyšší rychlost jsem zvolil 115200baud pro případ potřeby rychle vyčítat obrazová data do PC. Jako nejpomalejší jsem zvolil rychlost 9600baud pro případné potřeby komunikovat s aplikací přes linku velké délky či v zarušeném prostředí.

Výpočet frekvence hodinového signálu jsem provedl pomocí vztahu 1, kde f_z je frekvence základního hodinového signálu 48MHz a f_k je komunikační rychlost linky.

$$k_d = \frac{f_z}{16 \cdot f_k} \qquad 1$$

Výsledkem byl přesný poměr (koeficient dělící základní frekvenci k_d), kterého však dělením základního hodinového taktu nelze přesně dosáhnout. Proto bylo třeba poměr zaokrouhlit

na celé číslo a provést zpětný přepočet pomocí upraveného vztahu 1. Výsledkem zpětného přepočtu byla skutečná rychlost, kterou je možné generovat. Dle dokumentace k makrům může být rozdíl mezi požadovanou a skutečnou rychlostí maximálně 0,2%. Všechny použité komunikační rychlosti f_k, dělící koeficienty k_d a odchylky od přesných hodnot komunikačních rychlostí znázorňuje tabulka 3.

Požadovaná rychlost f_k [baud]	Skutečná rychlost f_k [baud]	Dělící koeficient k_d po zaokrouhlení	Rozdíl [%]
115200	115384	26	0,16
57600	57692	52	0,16
38400	38461	78	0,16
19200	19230	156	0,16
9600	9615	312	0,16

Tab. 3: Přehled komunikačních rychlostí seriové linky

Řídícím signálem specifickým pro přijímací makro je signál o přečtení dat z FIFO zásobníku, po kterém makro připraví ke čtení další datové slovo, pokud však již není zásobník prázdny. Posledním řídícím signálem pro vysílací makro je signál indikující zápis datového slova do zásobníku.
Popis řídících zpráv a komunikace je posán v samostané kapitole 8.

7.4 Blok CAMERA

Druhým blokem, který obsahuje mikrokontroler PicoBlaze3, je blok CAMERA. Tento blok je určen k ovládání kamery a řízení celého procesu zpracování obrazových dat. Základem bloku je mikrokontroler PicoBlaze3, který tvoří stejně jako v bloku HLAVNI dvousložkovou část nazvanou Procesor. Tyto dvě složky jsou samotný mikrokontroler a bloková RAM s programem. Způsob komunikace s blokem HLAVNI je popsán v kapitole 8.

Ovládání kamery, jejíž základní popis je uveden v kapitole 3.2.2, probíhá dle firemního standardu výrobce kamery a dále pomocí přímých řídících signálů. Komunikační protokol je nazván SCCB (Seriál Camera Control Bus). Je zde definován způsob fyzického přenosu zpráv a jejich nutný formát. Koncepce komunikace je taková, že Procesor je v pozici Master a kamera v pozici Slave. To znamená, že komunikaci iniciuje vždy Procesor. Ten obsahuje univerzální metodu, která sestaví zprávu a zapne vysílání zprávy do kamery. Zpráva je vždy délky 3 nebo 2 byty plus pomocné bity za každým bytem. Pokud se jedná o zkrácenou dvoubytovou zprávu, je vysílací metoda nastavena tak, že je přeskočeno vysílání prvního bytu. Je to proto, že ukončování vysílání je trochu složitější než začátek. V okamžiku, kdy je zpráva postavena, je program Procesoru uveden do čekací smyčky a je zapnut časovač, který generuje pravidelné pulzy přerušení Procesoru. Při každém přerušení se provede jeden ze tří úkonů. Buď je signál SIO_C nastaven na logickou úroveň 1, kdy jsou data na signálu SIO_D platná, nebo je tento signál nastaven na logickou úroveň 0, kdy je možné měnit hodnotu signálu SIO_D a nebo se mění hodnota signálu SIO_D. Průběh vysílání, realizovaný v aplikaci, přesně odpovídá požadavkům definovaným ve specifikaci komunikačního protokolu. Tento fakt byl ověřen měřením pomocí paměťového logického analyzátoru. Komunikační rychlost je nastavena na 80kHz.

Tento způsob řízení kamery spočívá v zapisování hodnot do osmibitových registrů kamery, kterých je dohromady okolo 170. Kamera pak podle dat zapsaných do těchto registrů upravuje své chování a funkce. Při každé inicializaci kamery jsou do registrů nahrány hodnoty, které nastaví kameru do stavu, kdy poskytuje obrazová data způsobem vhodným pro aplikaci. Právě nastavení registrů byl jeden z největších problémů při vývoji aplikace. Důvodem byl tragický nedostatek informací o kameře. Firma OmniVision, výrobce kamerového čipu, nebyla ochotna mi zpřístupnit dokumentaci s popisem registrů. Podmínkou poskytnutí informací bylo maximální utajení těchto informací stvrzené podepsáním smlouvy s výrobcem. Tímto podpisem bych byl nucen utajovat informace o kameře i v této práci. Nakonec se podařilo dokument s popisem registrů získat neoficiální cestou. Bohužel popis registrů je v získaném dokumentu velmi stručný a nedostatečný. Proto byla data k nastavení registrů získána z obdobné aplikace pomocí odposlechu SCCB sběrnice a následným dešifrováním naměřených dat. V poslední fázi byla data pozměněna postupným laděním.

Přímé signály k řízení kamery jsou signály s hodinovým taktem, který je přejat pro vnitřní chod kamery. Dalším je signál RESET s aktivní úrovní v logické hotnotě 1 a dále signál PWDN umožňující uvést kameru do spícího režimu (Standby mode), kdy její spotřeba klesá na 30uW. Před uvedením kamery do spacího režimu je třeba přepsat data v jednom z registrů.

Kamera je v této aplikaci nastavena do obrazového režimu QVGA (320 x 240 obrazových bodů) s opakovací frekvencí 60 obrazů za sekundu. Barevný model dat je RGB 565, kdy je červená barva popsána slovem o pěti bitech, zelená barva slovem o šesti bitech a modrá barva také slovem o pěti bitech. Dalšími výstupními signály kamery jsou hodinový signál určující platnost každého samostatného slova obrazových dat, dále signál VSYNC, který uvozuje každý nový obraz a jeho opakovací frekvence je tedy 60 Hz, a signál HREF, který je ve stavu logické 1 po dobu, kdy jsou obrazová data na sběrnici platná. Přesná podoba a vzájemné umístění signálů jsou znázorněny na Obr. 14 a podrobně popsány v manuálu ke kameře [6].

Podpůrnými částmi tohoto bloku jsou opět Demux_out a Demux_in. Tyto procesy se starají o předávání dat mezi Procesorem a ostatními signály v bloku CAMERA. Dalším paralelně běžícím procesem v tomto bloku je proces Konvertor a jeho podpůrný proces priprava_Konvertor. Ty slouží ke zpětné rekontrukci obrazových dat z kamery. Jak jsem již popsal výše, obrazová data z kamery jsou v barevném modelu RGB 565. Dohromady je tedy zpráva o jednom obrazovém bodě vyjádřena pomocí 16bitového slova. Protože sběrnice na obrazová data, kterou jsou data z kamery vyčítána, je jen 10 bitů široká, je slovo o každém obrazovém bodě rozděleno do dvou slov o osmi bitech. Každé je přeneseno samostaně a potvrzeno hodinovým signálem. Proces Konvertor pak skládá vždy dva po sobě jdoucí obrazové byty do jednoho 16bitového slova a generuje hodinový signál potvrzující tato data. Proces je aktivní jen pokud jsou data na sběrnici kamery platná, což je signalizováno signálem z kamery zvaným HREF.

Poslední částí je proces nazvaný Odposlech a jeho podpůrný proces Priprava_Odposlech. Ty jsou určeny k odposlouchávání dat na sběrnici SCCB. Pokud jsou vysílána data do kamery, jsou data z mikrokontroleru tímto procesem jen předána na sběrnici SCCB. Hlavní funkci tyto procesy plní při čtení dat z kamery. V tomto případě je proces Odposlech aktivován. Přestane přeposílat data z mikrokontroleru na sběrnici a místo toho ji uvede do třetího stavu vysoké impedance a začne příjímat data vysílaná kamerou. Po přijetí všech dat z kamery začne opět pracovat v předešlém pracovním režimu. Přijatá data jsou pak nabídnuta mikrokontroleru k dispozici ve formě uceleného bytu. Toto řešení je

z důvodu nemožnosti uvést datový port mikrokontroleru do třetího stavu vysoké impedance.

7.5 Blok DEKODER

Blok DEKODER je přímo podřízen bloku CAMERA, který jej ovládá. Tento blok je určen k převodu obrazových dat, přebíraných z bloku CAMERA v podobě 16bitových slov, z barevného modelu RGB na barevný model YUV, respektive jen na jasovou Y složku tohoto modelu. Tento převod je vykonáván z důvodu komprese dat, protože každý obrazový bod je pak popsán slovem o osmi bitech. Tato obrazová data jsou určena v této práci ke kontrole činnosti kamery. Aby byl splněn 4. bod pracovního zadání, tedy zobrazovat změny obrazových dat v PC (jen kontrolní funkce), je třeba tato data posílat do počítače. Dříve popsaná koncepce komunikace mezi bloky a bloku HLAVNI s počítačem dává jasně nejjednodušší řešení v podobě zasílání obrazových dat po osmi bitech. A nejlépe tak, aby jedno slovo odpovídalo jednomu obrazovému bodu. Druhým důvodem převádění dat do jasového vyjádření je potřeba tohoto vyjádření v další etapě vývoje, kdy budou tato data použita pro funkci prahování. Obrazová data z toho bloku jsou tedy vstupní data pro druhou etapu vývoje senzoru.

Převod barevných modelů mezi sebou je dán vztahem popsaným v [14]. V této aplikaci jsou data v modelu RGB převáděna jen na jasovou složku Y, takže vztah pro převod se značně zjednoduší na tvar ve vztahu 2, kde R, G, B jsou jasové hodnoty těchto barev.

$$Y = 0{,}229 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \qquad 2$$

Protože vztah 2 používá velmi přesné hodnoty, které vyžadují operace s desetinnými čísly, aproximoval jsem vztah tak, aby jeho implementace byla jednodušší. Výsledkem aproximace je vztah 3.

$$Y = 0{,}25 \cdot R + 0{,}5 \cdot G + 0{,}125 \cdot B = \frac{R}{4} + \frac{G}{2} + \frac{B}{8} \qquad 3$$

Obrazová data jsou na jasovou složku Y převáděna neustále procesem nazvaným Konvertor. Pokud je přijat požadavek na jejich vyčítání do počítače, je aktivován proces Vycitani. Ten odešle informaci o jasové hodnotě Y čtvrtého obrazového bodu v prvním řádku každého snímku do bloku CAMERA. Pomocí potvrzovacího pulzu potvrdí platnost dat. Zasílat informaci o jednom obrazovém bodě v každém snímku, tedy 60 zpráv za sekundu, je v této aplikaci dostatečné.

7.6 Blok VGA

Blok VGA slouží k zobrazení obrazových dat přijímaných z bloku CAMERA na monitoru. Tento blok neobsahuje mikrokontroler, protože se zde data zpracovávají při frekvencích 24MHz, což by nestíhal. Tento blok je stejně jako blok DEKODER přímo řízen z bloku CAMERA. Řídícími signály jsou signály RESET pro ukončení zobrazení dat na monitoru a signál Zobraz k jejich zobrazení na monitoru. K bloku jsou připojeny další dva pomocné bloky. Každý z nich obsahuje jednu blokovou paměť BRAM. Ty jsou nastaveny do režimu ukládání dat 18 x 1024 bitů, aby se do nich vešla obrazová data, která jsou v již dříve popsaném formátu a o velikosti 16 bitů. Paměti BRAM jsou používány

v dvouportovém režimu, takže je možné data do paměti zapisovat a zároveň číst pomocí druhého portu. Popis pamětí je v [11].

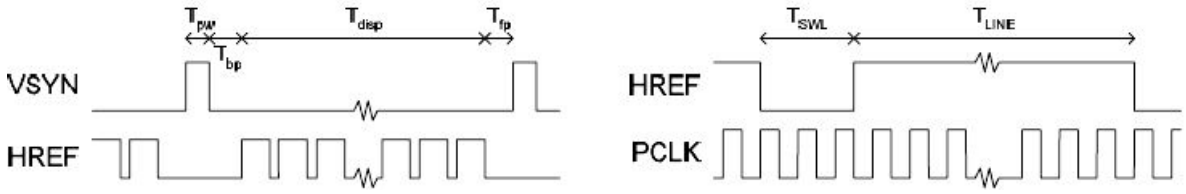
Blok VGA je složen z pěti paralelně běžících procesů. Prvním je proces nazvaný Hodiny. Ten funguje jako dělička základního hodinového signálu o frekvenci 48MHz. Vytváří hodinový signál o frekvenci 24MHz, kterým jsou synchronizovány zbylé procesy v bloku a monitor. Tato frekvence je základním taktem pro posílání obrazových dat do monitoru, což vychází ze specifikace VESA pro VGA režim.

Druhým procesem je proces nazvaný Ukladani_QVGA. Tento proces ukládá obrazová data do mezipaměti, která je tvořena podřízenými pamětmi BRAM, a určuje která z pamětí je právě určena ke čtení dat a která k zápisu. Při této konstrukci nedochází k současnému čtení a zápisu dat do jedné paměti, i když to jejich deklarace dovolují. Proto by šly použít i jednoportové paměti. Z důvodu lepší přehlednosti kódu jsem však zvolil dvouportové paměti.

Třetí proces nazvaný AktivniVideo naopak data z pamětí vyčítá. Po vyčtení dat je pošle pomocí signálu mon_video_data mimo proces, kde jsou asynchronně převedeny do podoby potřebné na jejich zaslání do převodníku. Jak jsem již popsal výše, data jsou uložena v 16bitovém slově se strukturou RGB 565. Protože monitor umí zpracovávat jen analogový signál o barvách, je třeba provést číslicově-analogovou konverzi. Ta je zajištěna VGA převodníkem popsaným v kapitole 3.3. Ten však pracuje se třemi sedmibitovými slovy. Proto je signál mon_video_data rozebrán na barevné složky a doplněn nulami do sedmibitových tvarů. K signálům s barvou červená a modrá jsou tedy přidány vždy dva signály s hodnotou nula a k signálu s informací o barvě zelené jen jeden signál s hodnotou nula.

Kdy přesně se budou vyčítat data z pamětí a zasílat do převodníku určuje proces s názvem Monitor. Zde se generují synchronizační signály pro monitor. Generování signálů řádkové a obrazové synchronizace je prováděno tak, aby odpovídalo standardu VESA pro VGA režim, tedy 640 x 480 obrazových bodů se snímkovou frekvencí 60Hz. Obecné znázornění průběhů synchronizačních signálů je na Obr. 16. Generování je zde odvozeno od hodinového taktu, který je generován procesem Hodiny, pomocí dvou čítačů. Jeden počítá obrazové body v řádcích, na obrázku 16 je signál značen PCLK, a generuje pulz vždy na konci každého řádku znázorněného průběhem HREF. Druhý čítač počítá práve tyto pulzy a počítá celé řádky. V příslušných okamžicích jsou nastaveny vnitřní signály určující na monitoru plochu, ve které bude vykreslen obraz. Tyto signály pak spouští vyčítání pamětí.

Posledním procesem je proces Synchronizace. Tento blok má za úkol zapnout proces Monitor právě v tom okamžiku, aby bylo generování pulzů pro monitor přesně zasynchronizováno s vyčítáním obrazu z kamery. Tento proces zapíná a vypíná proces Monitor na základě zpracovávání řídících příkazů z bloku CAMERA, resp. z počítače.



Obr. 16: Synchronizační pulzy v monitoru

8. Ovládání aplikace

Jak je napsáno u většiny bloků, je možné ovládat aplikaci příkazy z počítače. Přenos řídících zpráv je po sériové lince a jsou prvotně zpracovány v bloku HLAVNI. Pro komunikaci s aplikací stačí mít na počítači jakýkoliv terminál, umožňující nastavit následovné parametry přenosu: rychlost přenosu 38400baud, délka slova 8 bitů, žádná parita a vypnutí řízení komunikace. Každá zpráva v obou směrech je složena ze dvou částí. První část zprávy tvoří řídící ASCII znak určující typ zprávy. Druhou část tvoří data. Ty mohou být také ASCII znak, pokud jde o provedení některé implementované funkce, či to může být hexadecimální vyjádření obsahu řídícího registru kamery. Na každou správně přijatou zprávu aplikace po provedení požadovaného úkolu odpoví potvrzovací zprávou a případně i daty. Jedinou vyjímkou je reset zásobníků zpráv a změna komunikační rychlosti sériové linky, protože dojde k okamžité změně rychlosti, na kterou není program v počítači připraven a tak nemá smysl cokoli vysílat.
Přesný popis řídících zpráv je v tabulce 3.

Řídící zpráva			Odpověď		
Instr.	Data	Popis	Instr.	Data	Popis
C	D	pokyn k zasílání obrazových dat do počítače	D	1 byte v binárním tvaru	obrazová data jasu z referenčního obrazového bodu
C	K	konec vyčítání obrazových dat	C	@	OK
C	S	zjištění režimu kamery	C	Q	kamera je v režimu QVGA
			C	@	OK
C	R	reset kamery	C	@	OK – proveden reset
			C	@	OK – provedena synchronizace kamery s monitorem (jen pokud byl před příkazem monitor zapnutý)
C	P	zapnout uspávací režim	C	@	OK
C	U	obnovení činnosti kamery po spacím režimu	C	@	OK – monitor vypnut
			C	@	OK – provedení restartu kamery a obnovení jejího nastavení
			C	@	OK – spuštěn monitor
			C	@	OK – kamera probuzena
C	jiný znak	pokud dojde k odeslání výše nedefinovaného znaku	C	#	přijata chybná data

			C	@	OK
W	1 byte v hexa	číslo registru kamery, do kterého se bude zapisovat	C	@	OK – zpráva přijata
X	1 byte v hexa	data, která se zapíší do registru kamery určeného příkazem W	C	@	OK – zpráva přijata a odeslána do kamery po SCCB
R	1 byte v hexa	Příkaz k přečtení hodnoty z registru definovaném v Data	C	@	OK – zpráva přijata
			R	data v hexa	obsah přečteného registru
			C	@	OK
		obecně	C	B	kamera je zaneprázdněna
U	1	nastavení komunikační rychlosti sériové linky na 9600 baud			
U	2	nastavení komunikační rychlosti sériové linky na 19200 baud			
U	3	nastavení komunikační rychlosti sériové linky na 38400 baud – defaultní hodnota			
U	4	nastavení komunikační rychlosti sériové linky na 57600 baud			
U	5	nastavení komunikační rychlosti sériové linky na 115200 baud			
U	S	zjištění komunikační rychlosti sériové linky	U	1 až 5	stupeň rychlosti – odpovídá příslušnému příkazu

U	R	reset FIFO zásobníků v bloku UART			

Tab. 3: Popis řídících příkazů a jejich odpovědi

Po zapnutí aplikace se její inicializace a rozběhnutí potvrdí vysláním zprávy o rychlosti sériové linky, potvrzení inicializace bloku CAMERA a potvrzení inicializace kamery. Zpráva je ve tvaru U3C@C@. Je to jediný případ, kdy aplikace vysílá zprávu do počítače bez předchozího dotazu.

9. Statistické shrnutí

V Tab. 4 je vidět výsledná zpráva vývojového prostředí ISE o překladu aplikace a počtu použitých částí obvodu FPGA. Statistický popis aplikace udává, že v obvodu FPGA je využito přibližně 19,4% systémových hradel. Ta jsou nakonfigurována pomocí programu čítajícím přes 8000 řádek kódu v jazyce VHDL. Přibližně 30% kódu jsem vytvořil samostatně a zbylá část je převzata nebo vygenerována z programů, které jsem napsal pro mikrokontrolery PicoBlaze3.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	540	26,624	2%	
Number of 4 input LUTs	732	26,624	2%	
Logic Distribution				
Number of occupied Slices	588	13,312	4%	
Number of Slices containing only related logic	588	588	100%	
Number of Slices containing unrelated logic	0	588	0%	
Total Number 4 input LUTs	951	26,624	3%	
Number used as logic	732			
Number used as a route-thru	31			
Number used for Dual Port RAMs	32			
Number used for 32x1 RAMs	104			
Number used as Shift registers	52			
Number of bonded IOBs	69	221	31%	
IOB Flip Flops	32			
Number of Block RAMs	4	32	12%	
Number of GCLKs	4	8	50%	
Total equivalent gate count for design	291,278			
Additional JTAG gate count for IOBs	3,312			

Tab. 4: Shrnutí obsazenosti obvodu

10. Závěr

Aplikace pro snímání a předzpracování obrazu pomocí obvodu FPGA je nyní ve funkčním stavu. Podařilo se mi splnit všechny body zadání bakalářské práce i všechny body upřesněného pracovního zadání. Funkčnost aplikace a demostrace splnění zadání se dá ukázat posláním příkazu CM z počítače do FPGA. Tímto příkazem se zobrazí obrazová data vyčítaná z kamery na monitoru.

Obraz je původně v režimu QVGA. Po úpravě v obvodu FPGA je zobrazen na monitoru jako obraz o velikostním režimu VGA a barevném modelu RGB. V obraze jsou vidět barevné vrstevnice, které oddělují plochy s podobnou barvou. To je dáno nastavením kamery, respektive zapnutím funkce na zvýraznění hran v obraze. Tato funkce bude potřeba pro zpracovávání dat v další etapě vývoje, kde bude obraz, vyjádřený jen pomocí jasu, prahován.

Aplikace je napsána co nejvíce univerzálně a modulárně tak, aby se dala použít jako základní kámen pro další etapu vývoje senzoru jen s případnými minimálními úpravami. Modulární složení aplikace je graficky znázorněno na Obr. 13 a v RTL schematu v příloze C. Každý z bloků je možné ovládat příslušnými řídícími příkazy z počítače.

Přínosem této bakalářské práce je vytvoření základního kamene pro další etapu vývoje kamerového senzoru nečistot, která bude tématem mé diplomové práce. Pro mě osobně je obrovským přínosem získání mnoha vědomostí a praktických zkušeností s návrhem, implementací a laděním aplikace tohoto druhu.

Literatura

[1] Model no: 20J6 - Product Data Sheet. Ver. 1.1
[online] <http://www.microjet.com> [cit. 2007-03-13]

[2] Datasheet OV9625. Ver. 1.3
[online] <http://www.alldatasheet.com> [cit. 2007-04-05]

[3] Programmable Logic Design – Quick Start Handbook. Ver. June 10 2006
[online] <http://www.xilinx.com> [cit. 2006-11-10]

[4] Vesa coordinated video timing generator. Ver. 1.1
[online] <http://www.vesa.org> [cit. 2007-03-19]

[5] Advanced Information Product Brief OV9650
[online] <http://www.ovt.com> [cit. 2007-03-13]

[6] Datasheet OV9653. Ver. 1.5
[online] <http://www.google.com> [cit. 2007-04-17]

[7] Platform Developer's Kit RC10 manual. – dodáno s HW
[online] <www.celoxica.com> [cit. 2006-12-19]

[8] HW server představuje - RS-232.
[online] <http://www.rs232.hw.cz> [cit. 2006-12-19]

[9] Datasheet ICL3222E. Ver. Sept. 2001, File Number 4910.6.
[online] <http://www.intersil.com> [cit. 2007-03-20]

[10] Application Note OmniVision Serial Camera Control Bus (SCCB) – Functional specification. Ver. 2.1
[online] <http://www.ovt.com> [cit. 2007-03-19]

[11] Spartan-3 FPGA Family: Complete Data Sheet. Ver. DS099 – May 25 2007
[online] <http://www.xilinx.com> [cit. 2007-06-12]

[12] Douša, Jiří. *Jazyk VHDL* 1.vyd. Praha : ČVUT, 2003. 76 st. ISBN 80-01-02670-1

[13] UART Transmitter and Reciever Macros. Ver. Jan. 2003
[online] <http://www.xilinx.com> [cit. 2007-03-24]

[14] PicoBlaze3 8-bit Embedded Microcontroller User Guide. Ver. 1.1.1 UG129
[online] <http://www.xilinx.com> [cit. 2007-03-05]

[15] ISE In-Depth Tutorial.
[online] <http://www.xilinx.com> [cit. 2007-05-08]

Seznam příloh

Příloha A: Obsah přiloženého CD

Příloha B: Nastavení registrů kamery

Příloha C: RTL schéma bloku Top_propojeni

Příloha A: Obsah přiloženého CD

Součástí bakalářské práce je i přiložený disk. Jeho obsahem je jak samotná bakalářská práce, tak i zdrojové kódy, zveřejnitelné materiály a programy.

Obsahu disku:

BP P. Papik - bakalarska prace.pdf (*text bakalářské práce*)

- aplikace
  - aplikace.bit (*konfigurační soubor pro FPGA XC3S1500*)
  - RS232 terminal Hercules (*freeware terminál od HW group*)

- materialy (*složka obsahující většinu materiálů ze seznamu literatury, pořadová čísla navzájem odpovídají*)

- nastroje
  - pBlazIDE (*ladící prostředí pro PicoBlaze*)
  - KCPSM3
    - KCPSM3.exe (*překladač pro PicoBlaze*)
    - ROM_form.vhd (*pomocný soubor pro překladač*)
    - ROM_form.v (*pomocný soubor pro překladač*)
    - program.psm (*založen nový ukázkový soubor*)

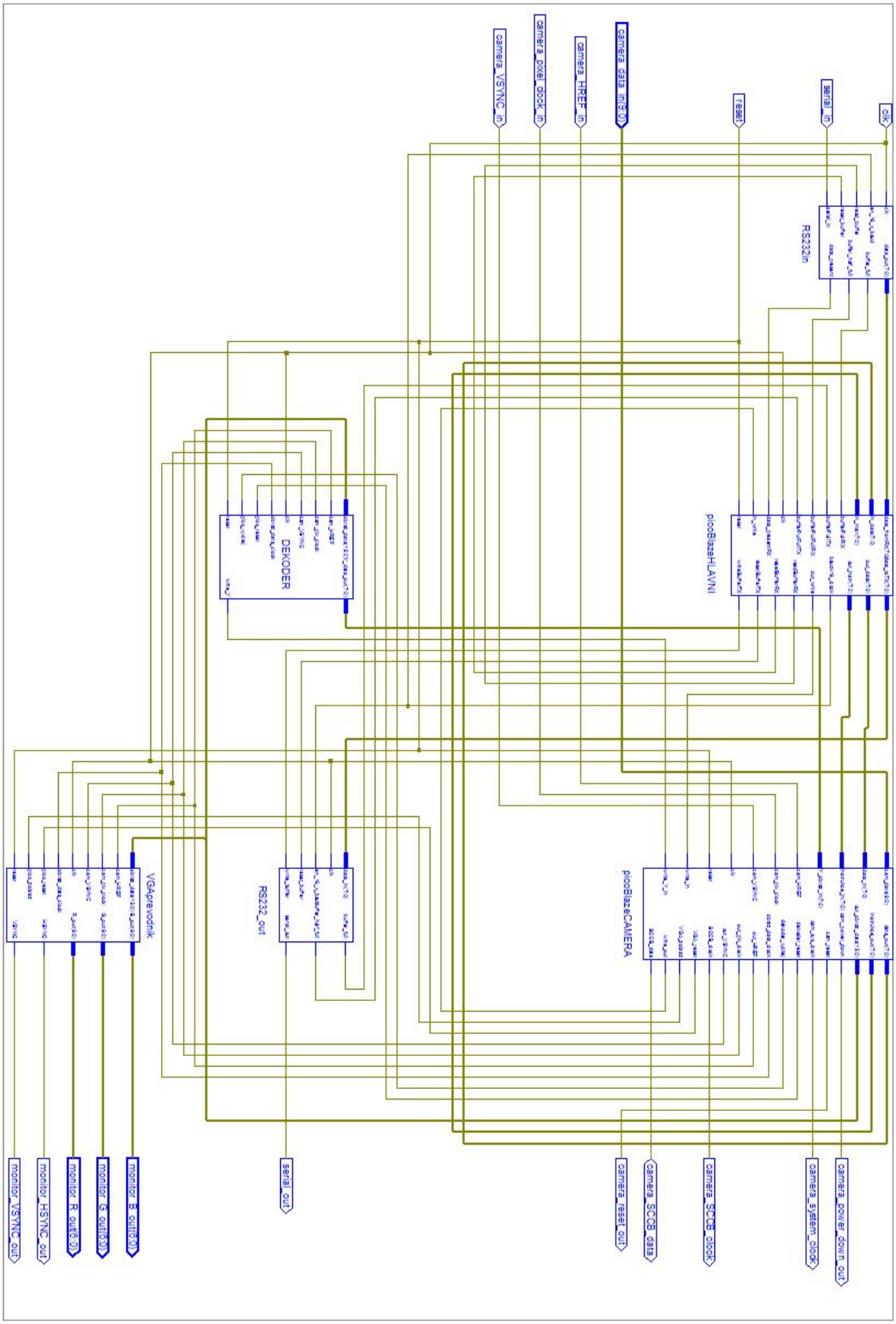
- zdrojove kody
  - projekt ISE (*kompletní projekt BP*)
  - projekt CAMERA (*PicoBlaze pro blok CAMERA*)
  - projekt HLAVNI (*PicoBlaze pro blok HLAVNI*)

Příloha B: Nastavení registrů kamery

Sub-address	Hodnota Hex	Popis
12	80	reset všech SCCB registru
11	00	Data format and internal clock
6B	0A	reserved
6A	3E	Manual banding filter vaule
3B	09	Common control 11
13	E0	Common control 8
01	80	AWB
02	80	AWB
00	00	AGC
10	00	Exposure value
13	E5	Common control 8
39	50	reserved
38	92	reserved
37	00	reserved
35	81	reserved
0E	01	Common control 5
1E	04	Mirror/Vflip enable
A8	80	reserved
12	14	Common control 7
04	00	Common control 1
0C	04	Common control 3
0D	80	Common control 4
18	C6	Output format - horizontal frame
17	26	Output format - horizontal frame
32	AD	HREF control
03	00	Vertical frame control
1A	3D	Output format - vertical frame
19	01	Output format - vertical frame
3F	A6	Edge Enhancement Adjustment
14	2A	Common control 9
15	00	Common control 10
41	02	Common control 16
42	08	Common control 17
1B	00	Data format - pixel delay select
16	06	reserved
33	E2	reserved
34	BF	reserved
96	04	reserved
3A	00	reserved
8E	00	reserved
3C	77	Common control 12
8B	06	reserved
94	88	reserved
95	88	reserved
40	D0	Common control 15
29	3F	Analog BLC and Regulator control
0F	C3	Common control 8
3D	09	Common control 13
69	40	Manual Banding Filter MSB

5C	B9	reserved
5D	96	reserved
5E	10	reserved
59	C0	reserved
5A	AF	reserved
5B	55	reserved
43	F0	reserved
44	10	reserved
45	68	reserved
46	96	reserved
47	60	reserved
48	80	reserved
5F	E0	reserved
60	8C	reserved
61	20	reserved
A5	D9	Common control 26
A4	74	Common control 25
8D	02	Common control 23
13	E7	Common control 8
4F	3A	Matrix coefficient
50	3D	Matrix coefficient
51	03	Matrix coefficient
52	12	Matrix coefficient
53	26	Matrix coefficient
54	38	Matrix coefficient
55	40	Matrix coefficient
56	40	Matrix coefficient
57	40	Matrix coefficient
58	0D	Matrix coefficient signs
8C	23	Common control 22
3E	02	Common control 14
A9	B8	reserved
AA	92	reserved
AB	0A	factory reseved
8F	0F	Digital BLC offset sign
90	00	Digital BLC Blue channel offset value
91	00	Digital BLC Red channel offset value
9F	00	Digital BLC Gb channel offset value
A0	00	Digital BLC Gr channel offset value
3A	01	Line buffer test option
24	70	AGC/AEC – Stable operating region
25	64	AGC/AEC – Stable operating region
26	C3	AGC/AEC – Fast mode operating limit
2A	0	Dummy pixel insert
2B	0	Dummy pixel insert
6C	40	gamma curve
6D	30	gamma curve
6E	4B	gamma curve
6F	60	gamma curve
70	70	gamma curve
71	70	gamma curve
72	70	gamma curve

73	70	gamma curve
74	60	gamma curve
75	60	gamma curve
76	50	gamma curve
77	48	gamma curve
78	3A	gamma curve
79	2E	gamma curve
7A	28	gamma curve
7B	22	gamma curve
7C	04	gamma curve
7D	07	gamma curve
7E	10	gamma curve
7F	28	gamma curve
80	36	gamma curve
81	44	gamma curve
82	52	gamma curve
83	60	gamma curve
84	6C	gamma curve
85	78	gamma curve
86	8C	gamma curve
87	61	gamma curve
88	BB	gamma curve
89	D2	gamma curve
8A	E6	gamma curve

Registry, které nejsou v seznamu uvedeny, jsou automaticky nastaveny do základní hodnoty kamerou. Tyto hodnoty a přesný popis registrů lze nalézt v [6].

Příloha C: RTL schéma bloku Top_propojeni