Česká zemědělská univerzita v Praze
Technická fakulta
Katedra elektrotechniky a automatizace

Snímání a zpracování vizuální informace blízké scény pro orientaci slepých osob

Diplomová práce

Autor práce: Ondřej Loučka

Vedoucí diplomové práce: prof. Ing. Jaromír Volf, DrSc.

Studijní obor: Informační a řídící technika

2013

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra elektrotechniky a automatizace

Technická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Loučka Ondřej

Informační a řídící technika v agropotravinářském komplexu

Název práce

Snímání a zpracování vizuální informace blízké scény pro orientaci slepých osob

Anglický název

Scanning and Processing of Visual Information of Near Scene for Blind Persons Orientation

---

Cíle práce

Cílem práce je navrhnout algoritmus, zpracovatelný mikroprocesorem v reálném čase, který je schopen analyzovat blízkou scénu za účelem rozpoznámí blízkých objektů, které mohou slepého člověka ohrozit.

Metodika

Seznámení se současnými systémy, řešícími obdobnou tématiku. Na základě získaných zkušeností navrhnout algoritmus, který bude realizovatelný v reálném čase mikroprocesorem.

Osnova práce

1. Úvod
2. Přehled současných systémů
3. Návrh algoritmu rozpoznávání blízké scény
4. Realizace algoritmu rozpoznávání blízké scény
5. Ověření algoritmu na reálných datech
6. Závěr

Rozsah textové části

cca 60 str.

Klíčová slova

vizuální informace, zpracování vizuální informace, slepec, scéna

---

Doporučené zdroje informací

Anderson, J., Lee, D. J., Archibald, J.: Hardware implementation of feature density distribution algorithm [online]. 2005. Dostupné na http://www.ee.byu.edu/faculty/djlee/Publications/IECON_Hardware_11205.pdf

Digital Image Processing – lecture materiles [online] Dostupné na: http//www.icaen.uiowa.edu/~dip/LECTURE/contents.html

Hlaváč, V., Šonka, M.: počítačové vidění. Grada, Praha 1992

Vedoucí práce

Volf Jaromír, prof. Ing., DrSc.

Termín zadání

listopad 2012

Termín odevzdání

duben 2013

prof. Ing. Jaromír Volf, DrSc.
Vedoucí katedry

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
-3-
TECHNICKÁ FAKULTA

prof. Ing. Vladimír Jurča, CSc.
Děkan fakulty

V Praze dne 4.3.2013

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady uvedené v seznamu použité literatury.

V Praze dne _____ _____
podpis

Poděkování

Děkuji především vedoucímu diplomové práce prof. Jaromíru Volfovi za připomínky a cenné rady při psaní této práce.

Abstrakt

Cílem práce je rozpoznávání objektů blízké scény pro orientaci slepých osob. V první části je přehled existujících technických prostředků pro náhradu zraku. Tyto prostředky zrakové protetiky nahrazují buď část sítnice, nebo celé oko. Čidlo zraku nahrazuje kamera a obrazový signál musí být zpracován v mikroprocesorovém systému. Koncovým prvkem systému je implantát, který slepému předává stimuly a vzniká zrakový vjem. Pro takový systém je ve druhé části navržen algoritmus pro detekci hran, který musí být zpracovatelný v reálném čase. Využívá jednu ze základních metod počítačového vidění – konvoluce. Detekce hran je prováděna na základě konvoluce obrazu s gradientními operátory v kombinaci s prahováním. Algoritmus je implementován v jazyce VHDL, který je určen pro programování logických obvodů – programovatelných hradlových polí FPGA. Realizaci jsem provedl s FPGA výrobce Xilinx a s modulem průmyslové kamery. V poslední části práce jsem algoritmus a ověřil na reálných datech. Systém je přenosný s malými nároky na napájecí soustavu.

Klíčová slova

implantát, detekce hran, konvoluce, FPGA, VHDL, kamera.

Abstract

Goal of this work is use of the recognition of near objects for blind persons orientation. First part of the work summarizes the state of the art in technical means for the sight prosthetic. These involves prosthetic replacing either part of the retina or the whole eye. The eye is replaced by the camera and the resulting video signal is processed by the microprocessor. End part of the system is an implant passing stimuli to the blind person and thus creating sight perception. For such a system is in the second part of the work designed edge detection algorithm, computable in real time. The algorithm utilizes convolution – one of basic methods of computer vision. Edge detection is done by convolving image data with gradient operators and then thresholding. The whole algorithm is implemented in VHDL language. VHDL is the hardware description language for programming of programmable logic devices – field-programmable gate arrays (FPGA). Realization is done with the FPGA from manufacturer Xilinx and with the industrial camera. In the last part of the work is the algorithm checked on real data. Developed system is portable and has only small power supply demands.

Key words

implant, edge detection, convolution, FPGA, VHDL, camera.

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

ASIC – Aplication Specific Inegrated Circuit
BRAM – Block RAM
CPLD – Complex Programmable Logic Device
CPU – Central Procesor Unit
CRT – Cathode Ray Tube
D/A – Digital/Analog
DSP – Digital Signal Processing,
FIFO – First In, First Out memory
FPGA – Field Programmable Gate Array
HS – Horizontální synchronizace
HSTL – High-speed Transceiver Logic
I2C – Inter-Integrated Circuit
JTAG – Joint Test Action Group
LCD – Liquid Crystal Display
LVCMOS – Low Voltage Complementary Metal Oxide Semiconductor
LVTTL – Low Voltage Transistor-Transistor Logic
LUT – LookUp Table
MSB – Most significant bit
PLD – Programmable logic device
RAM – Random Access Memory
RGB – Barevný model Red, Blue, Green
RTL – Register Transfer Level
SCCB – Serial Camera Control Bus
SSTL – Stub Series Terminated Logic
USB – Universal Serial Bus
VGA – Video Graphics Array
VHSIC –Very High Speed Integrated Circuits.
VHDL – VHSIC Hardware Description Language
VS – Vertikální synchrnizace
YUV – Barevný model, Y jas, UV barva

Kapitola 1

Úvod

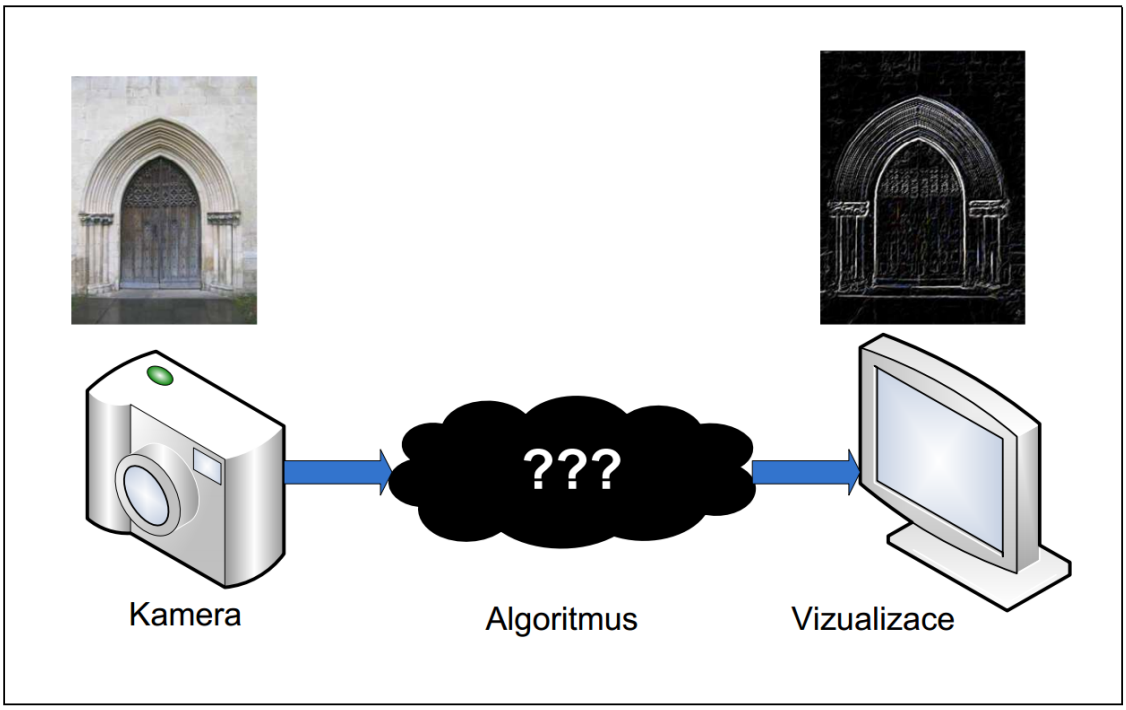
Cílem práce je navrhnout algoritmus, který je schopen analyzovat blízkou scénu za účelem rozpoznání objektů, které mohou slepého člověka ohrozit. Vychází ze základního požadavku na náhradní systém zraku pro nevidomého. Tím je pohyb a orientace slepého v prostoru a schopnost se vyhýbat případným překážkám.

Slepotou z různých důvodů trpí na světě několik milionů lidí. Příčinou částečné nebo úplné ztráty zraku jsou nejčastěji degenerativní nemoci zrakového systému. Navrácení nebo zlepšení zraku technickými prostředky je pro nevidomé velkým přínosem a pomůckou. Běžně jsou zrakově postižení lidé odkázáni na slepeckého psa a bílou hůl. S vývojem techniky se v posledních letech stále více objevují zprávy o bionickém oku a umělé sítnici. Přestože se stále jedná o experimentální formy náhrady zraku, představuje pro nevidomé velkou naději. Na vývoji zrakových protetik se podílí celá řada vývojových a lékařských týmů po celém světě. Technické prostředky náhrady zraku nemohou nahradit oko ve smyslu funkčního orgánu a plně navrátit postiženému zrak. V sítnici nahrazují jen její malou část a obrazová informace je velmi zjednodušená. Zatím neumíme nahradit oko žádným technickým prostředkem z důvodu jeho rozlišovací schopnosti a také ohromné komprimace. Přehledem zrakových protetik se zabývá druhá kapitola.

Systémy náhrady zraku jsou si podobné, nejčastěji realizované kamerovým snímačem, výpočetní jednotkou a implantátem. Výpočetní jednotka zpracovává obrazovou informaci, která se přenáší do implantátu ve formě elektrických impulsů. Pomocí implantátu dochází ke stimulaci funkční části zrakového systému a evokuje v mozku nevidomého vizuální vjem. Umístění implantátu závisí na druhu postižení zraku. Je umístěn buď v oku, kde nahrazuje nefunkční část sítnice, nebo přímo do zrakového centra v mozku. V uvedeném systému je vývoj zaměřen jak na implantáty, tak na jejich biokompatibilitu, počet stimulačních elektrod a aplikaci výpočetních metod zpracování obrazového

signálu. Zpracování obrazu z kamery využívá metod počítačového vidění. Metody řeší *detekci hran* blízké scény. Popsal jsem je ve třetí kapitole spolu s vlastním návrhem algoritmu.

Rozsah návrhu algoritmu je zřejmý z obrázku 1.1.



Obrázek 1.1: Rozsah návrhu algoritmu

Důležitým požadavkem na algoritmus je jeho zpracování mikroprocesorem v reálném čase. Kamerový snímač a výpočetní jednotka musí kompaktní přenosný celek. Výsledný obraz je analýze interpretován na monitoru. Dílčí úlohou je navrhnout vhodnou platformu pro implementaci algoritmu, schopnou zpracovávat velký objem obrazových dat. Dále je třeba zvolit vhodný kamerový snímač s dostatečným rozlišením. Posledním požadavkem vyplývající ze zadání je energetická nenáročnost navrženého systému. Návrh je zrealizován na platformě programovatelné logiky hradlových polí. Výběr vývojového kitu s obvodem FPGA a popis modulů navrženého programu v jazyce VHDL se zabývá čtvrtá kapitola.

Ověření funkce algoritmu na reálných datech jsem zdokumentoval fotografiemi a popsal v páté kapitole. Pro zobrazení snímků jsem použil běžný monitor.

Do přílohy A jsem doplnil další fotografie, na kterých je zachycen obraz po detekci hran běžných předmětů. V příloze B je zdrojový kód programu v jazyce VHDL rozčleněný do jednotlivých modulů.

Z uvedených skutečností vyplývá, že obor umělého zraku je obecně multidisciplinární obor s odbornostmi z elektroniky, programování, matematiky, biologie. Výběru tématu diplomové práce ovlivnilo nejen to, že jsem se touto problematikou zabýval v bakalářské práci, ale také zájem o obvody programovatelné logiky a jejich programování, které jsem se chtěl naučit.

Kapitola 2

Přehled současných systémů

Zrakový systém člověka je tvořen optickým systémem oka, sítnicí a zrakovou dráhou vedoucí do mozkové kůry. Zrakový vjem je zpracován v centru vidění v mozku označovanou *visual cortex*. Prostředky náhrady zraku suplují nefunkční část zrakové dráhy podle druhu postižení nevidomého. Cílem vývojových týmů ve spolupráci s oftalmology [1] je umožnit postiženým smysluplné vizuální stimuly odpovídající snímaným objektům. Nejedná se o plnohodnotnou náhradu zraku, která v současnosti neexistuje. Léčba a testování probíhá na experimentální úrovni pro jednotlivce, dobrovolníky. Přesto některé týmy uvádějí již desítky úspěšných operací na „komerční“ úrovni s uspokojivým výsledkem [3].

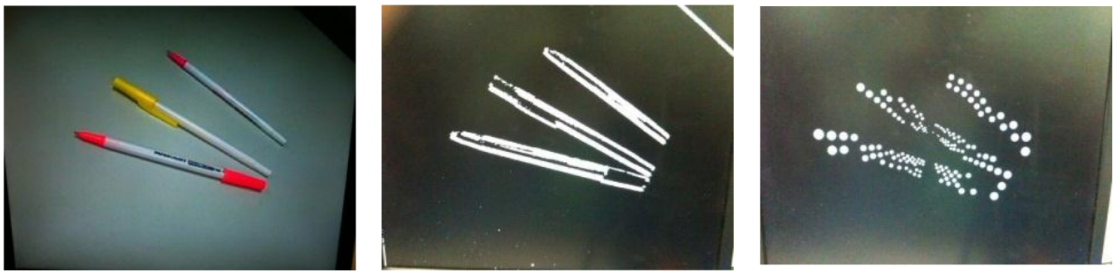
Důvodem slepoty může být více. Ve velkém měřítku jsou to degenerativní choroby sítnice oka, dále pak ztráta zraku vlivem úrazu nebo vrozená vada. Předpokladem pro nahrazení zraku technickými prostředky je nepoškozené centrum vidění v mozku.

Systémy se dají rozdělit podle toho, zda nahrazují sítnici oka nebo celou zrakovou dráhu – oko a zrakový nerv *papilu*. Při poškození sítnice je důvodem slepoty absence stimulačních signálů z oka do mozku a náhradou je retinální implantát. Pokud nefunguje celá zraková dráha, je třeba stimulovat přímo korovou oblast mozku *visual cortex*. Systém je komplikovanější jednak z technického hlediska, ale hlavně z hlediska invazivního zásahu do mozku pacienta. Přesto je výzkum kortikálních implantátů starší a započal již v 70. letech minulého století.

Čidlo zraku oko je nahrazeno kamerou, ze které je signál upraven a přiveden na stimulační implantát, tvořený elektrodovým polem. Implantát stimuluje funkční zrakové buňky a vzniká zrakový vjem. Vjem je postiženým vnímán jako světlé body na černém

[1] Oftalmologie se zabývá onemocněním a chirurgií zrakového systému včetně mozku

pozadí a pacient se proto musí naučit obrazy správně interpretovat, příklad je na obrázku 2.1.



Obrázek 2.1: Vlevo nezpracovaný obraz, uprostřed po zpracování a vpravo interpretace pro nevidomého [1]

2.1 Retinální implantáty

V případě stimulace zrakového nervu v oku se jedná o retinální implantáty a nahrazují sítnici. Implantát je umístěn na povrchu sítnice a je označován jako *epiretální*. Nahrazuje sítnici ve smyslu všech jejích vrstev a je nutné zpracovat obraz „ekvivalentně“ k tomu, co by vykonávaly jednotlivé nefunkční sítnicové vrstvy [2]. Obraz chápeme v intuitivním slova smyslu jako obraz na sítnici lidského oka nebo jako obraz sejmutý kamerou.

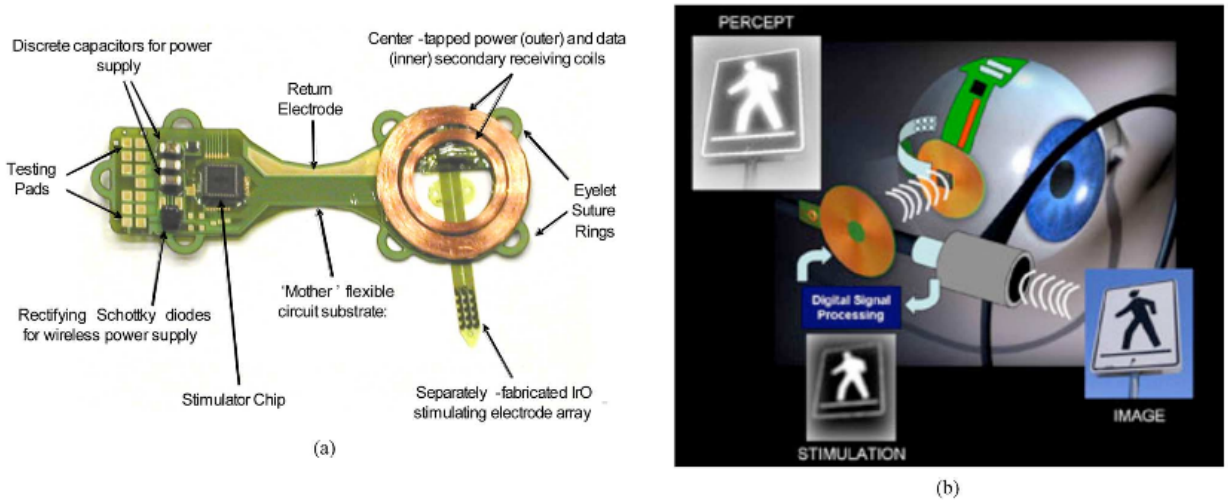
Epiretální náhrada má kameru, signál z kamery je zpracován mikroprocesorem a nejčastěji bezdrátově přenášen na čip. Ten je součástí stimulačního implantátu voperovaného do oka. Mikroprocesor provádí detekci *hran* objektů, stimulační čip přizpůsobuje signál na danou „biologickou“ napěťovou a proudovou úroveň pro buzení elektrod.

Příkladem systému je náhrada z univerzity MIT [2], na obrázku 2.2

Systém má externí část, která snímá a zpracovává obraz, a interní část, která je implantovaná v těle pacienta. Přenos dat mezi externí a interní částí je bezdrátově pomocí dvou cívek umístěných proti sobě a vzdálených od sebe asi 1,5 cm. Každá cívka má dvě vinutí – jedním se přenáší data, druhým se indukuje potřebné napájecí napětí pro implantát. Celý systém se skládá z pěti částí:

- Digitální kamera umístěná na brýlích, zachycuje obraz a v reálném čase ho posílá ke zpracování do mikroprocesoru.
- Signálový mikroprocesor, zpracovává obraz z kamery a převádí na obrazy předmětů – detekuje hrany. Z procesoru je signál přenesen do vysílací cívky v blízkosti oka.

[2] Sítnice má asi 130 milionů světlocitlivých buněk a zrakový vjem je přenášen jedním milionem vláken



Obrázek 2.2: Epiretinální implantát vlevo a princip systému vpravo.[2]

- Vysílací cívka vysílá zpracovaný signál do implantátu.
- Oční implantát s elektrodovým polem. Jádrem implantátu je čip typu ASIC s D/A převodníkem, který převádí binární signál na stimulační pulsy odpovídající elektrické úrovně. Flexibilním kabelem je čip napojen na samostatné stimulační pole elektrod (na obrázku 2.2 není zobrazen). V jádru stimulačního čipu je 25000 tranzistorů, má spotřebu asi 2,5 mW a budící proud je 800 μA na jednu elektrodu. Elektrody jsou platinové.

Firma Boston Retinal Implanct Project [3] má podobný systém. Byla založena v roce 1980 a je jednou z prvních firem, která se zabývala pokusy s epiretinálními implantáty. Další výrobce podobného systému je firma Second Sight [4]. Její výrobky Argus a Argus II jsou úspěšně aplikovány u několika desítek pacientů. Prakticky se neliší v technickém řešení externích částí, ale liší se v provedení implantátu.

Pro úplnost uvedu, že kromě epiretinálních implantátů existují i subretinální, které jsou umístěny na spodní vrstvě sítnice a zastupují tuto vrstvu. Implantátem je matice fotodiod podobná kamerovému čipu. Snímaný obraz po zesílení přiveden na stimulační elektrody. Zpracování obrazu probíhá v buňkách sítnice nad implantátem. Vrstvy sítnice musí být tedy funkční. Příkladem je implantánt firmy Retina Implantat AG [5], obrázek 2.3. Aktivní čip má velikost 3x3 mm a je 70 mikronů tenký. Obsahuje 1500 fotodiod s tranzistorovým zesilovačem a elektrody. Každá elektroda generuje jeden pixel. Čip je napájen z externího zdroje bezdrátově. Čip má dvě části, první je implementována pod sítnici, zatímco druhá je umístěna na oční bulvu.

Obrázek 2.3: Subretinální implantát. [5]

2.2 Kortikální protézy

Kortikální implantát je přímo v centru vidění a obraz předává stimulací mozkových buněk. Tento přístup je jednoznačně komplikovanější. O zpracování zrakového signálu mozkem toho příliš nevíme. Je však jediným řešením pro pacienty s celkově poškozeným okem a zrakovým nervem.

Prvními průkopníky mozkových implantátů byli doktoři Dobelle a Mladejovsky v roce 1974. Elektrody implantátu byly buzeny vysílacími cívkami z oscilátoru v těsné blízkosti hlavy a evokovali zrakový vjem. Další dvě skupiny vědců vznikly v 90. letech – National Institutes of Health a skupina doktora Normanna založená v laboratořích aplikovaného vidění na univerzitě v Utahu. Obě se zabývaly technologií elektrod implantátů s větší hustotou a menšími budícími proudy [3]. Doktor Ricahrd Normann na základě těchto testů vyvinul implantát se 100 mikroelektrodami.

Vývoj kortikálních protéz byl zahájen již před čtyřiceti lety, ale uspokojivá technologie pro vytvoření takového zařízení je známa až v posledních letech. Přestože byl znám způsob zpracování obrazové informace, mohl vzniknout přenosný systém až s vývojem a miniaturizací elektroniky. V laboratořích Williama Dobelleho vznikl v roce 2000 přenosný systém pro pacienta známého jako Jerry, který měl mozkový čip implantovaný již od roku 1978. Původní elektronika zpracovávající obraz byla v přístrojové skříni o rozměrech 3 x 2 x 1.6 metrů a vážící 800 kilogramů. Čip s elektronikou byl propojen svazkem kabelů. Poslední generace korových implantátů mají až 300 mikroelektrod

[3] Během prvních pokusů dosahoval potřebný stimulační proud 3-5 mA!

a další snahou je zvětšovat jejich hustotu. Důležité je dlouhodobá stabilita mikroelektrod bez rizika poškození v lidském těle. Inovací je bezdrátové přenášení dat z kamery do implantátu, což redukuje možnost infekce a dalších pooperačních problémů.[6]

2.3 Systém The vOICe

Systém *The vOICe* [7] uvádím z důvodu, že je založen na snímání obrazu kamerou a jeho následné zpracování. Zpracování je ovšem rozdílné oproti výše uvedeným systémům. Obrazový signál se převádí na zvukové frekvence. Ty představují vizuální vzory snímaných předmětů a prezentují se postiženému pomocí sluchátek. Změna jasu obrazu odpovídá změně hlasitosti. Aby mohl pacient vOICe systém efektivně používat, je třeba určitý tréning. Spoléhá se na adaptabilitu mozku, respektive sluchového centra, které převádí zvuk na obrazové vize. Uvedená vlastnost je i předmětem vědeckých výzkumů. Oproti retinálním a korovým implantátům je nespornou *výhodou* neinvazivního přístupu, kdy nehrozí pooperační komplikace a nestabilita zrakových implantátů v těle člověka. Další výhodou je větší rozlišení 1000 až 10000 pixelů ve srovnání s retinálními a kortikálními implantáty. Nevýhodou je smyslová interference, protože pacient se systémem vOICe má omezené okolní zvuky. Na stránkách projektu vOICe [4] je návod, jak se systémem pracovat a software do osobního počítače nebo mobilního telefonu, který transformaci obrazu na zvuk provádí. Celý systém je otevřený a volně dostupný.

2.4 Shrnutí

Systémy umělého zraku jsou často vybaveny dalšími podpůrnými obvody a senzory pro ultrazvukové nebo laserové měření vzdálenosti. Tím se eliminuje nebezpečí případné chybné detekce a následně interpretace překážek. Chybné vjemy představuje například horní hrana schodů, neukončená hrana, stíny nebo odlesky objektů. Jsou důsledtem toho, že pacientovi chybí jednak prostorové vidění a rozpoznání objektů ve scéně je ovlivněno mnoha aspekty jako je osvětlení scény nebo relativní pohyb objektů vůči kameře.

Všechny systémy, které jsem uvedl, používají kameru a potřebují zpracování obrazového signálu, které je předmětem práce. Výsledkem zpracování je detekce hran. K tomu se

[4] http://www.seeingwithsound.com

využívají metody počítačového vidění probrané v následující kapitole a která se zabývá i návrhem algoritmu a rozborem použitých metod.

Kapitola 3

Návrh algoritmu rozpoznávání blízké scény

Pro návrh algoritmu jsem využil znalostí z počítačového vidění z bakalářské práce [6]. Podle zadání je úlohou algoritmu analýza blízké scény, jejímž výsledkem je detekce hran objektů v obraze. Sítnice oka je uspořádána tak, že především reaguje právě na *hrany*, kontrasty a pohyb. Metody počítačového vidění, které se obecně zabývají zpracováním obrazu, jsou označované jako DSP (digital signal processing). Snaží se napodobit lidské vidění pomocí technických prostředků. Jednou z mocných metod DSP je **konvoluce**, kterou jsem pro detekci hran využil.

Návrh algoritmu pro detekci blízké scény má v tři roviny:

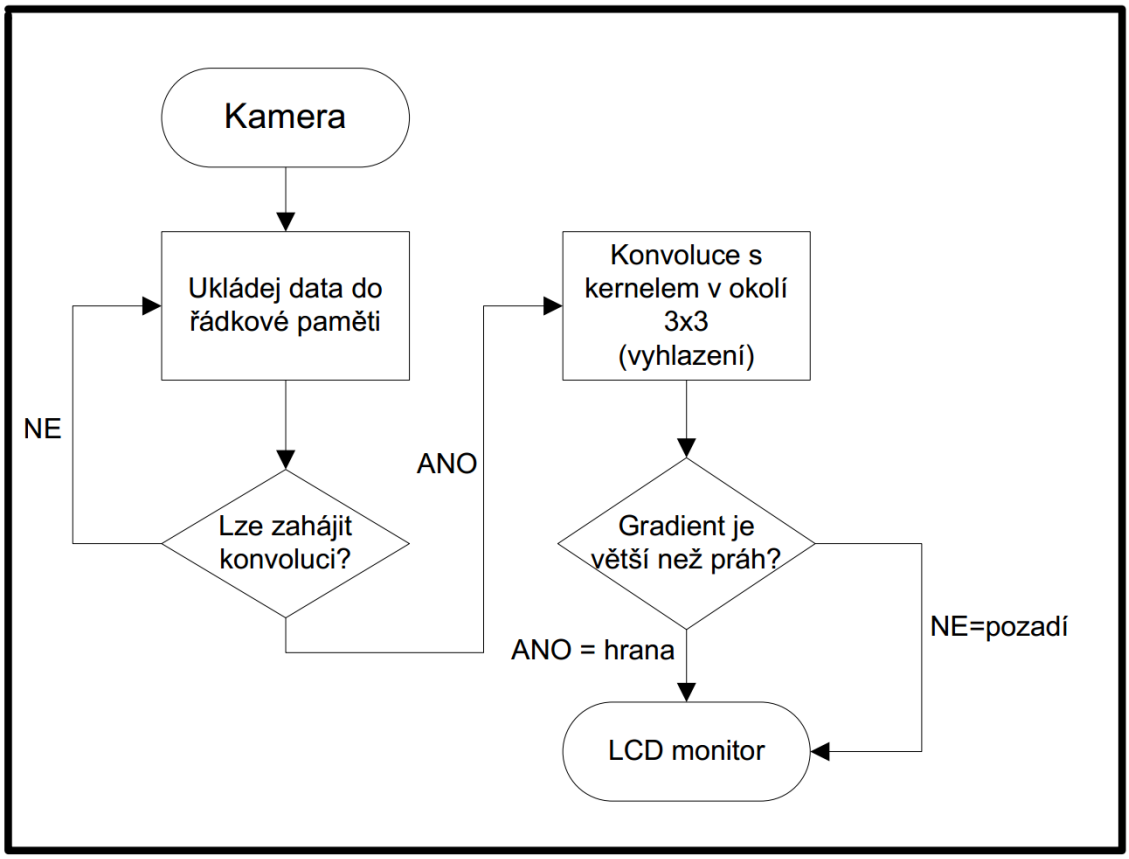
1. Vlastní návrh algoritmu, který zpracuje data z kamery a detekuje hrany. Výstupní data lze využít pro zraková protetika uvedená ve druhé kapitole.
2. Nalezení vhodné platformy mikroprocesorového obvodu pro implementaci algoritmu. Vzhledem k tomu, že detekce musí probíhat v reálném čase, musí být vybraná platforma rychlá a robustní s minimálním zpožděním signálu mezi vstupem a výstupem.
3. Volba kamerového snímače s vhodným formátem a parametry, umožňující přímé zpracování dat.

Celek tvoří komplexní *přenosný* systém. Mobilní systém musí mít malou spotřebu energie a velký výpočetní výkon. Funkci algoritmu musím ověřit a pro výstupní vizualizaci

zpracovaných dat použiji obyčejný LCD monitor, jako ekvivalent obrazu, který by sloužil jako vstup pro bionické oko [1].

3.1 Algoritmus pro detekci hran

Celý algoritmus je na vývojovém diagramu 3.1 a má dva hlavní procesy. Prvním procesem je **konvoluce**. Aplikací konvoluce je získán gradient obrazové funkce pro každý pixel v obraze. Podle velikosti gradientu lze pak určit, kde je nebo není hrana. Gradient je porovnáván se zvolenou prahovou hodnotou a rozhodne se, zda pixel je hranou nebo pozadím. Druhým procesem je **ukládání pixelů** do řádkové paměti. Vstupem algoritmu jsou diskrétní data z kamery, výstupem data pro LCD monitor.



Obrázek 3.1: Vývojový diagram algoritmu detekce hran

[1] Termín bionické oko se často používá v odborné literatuře i odborných článcích a znamená obecně náhradu oka jako orgánu

3.1.1 Konvoluce

V následujícím odstavci jsem provedl teoretický rozbor, potřebný pro hlavní proces algoritmu – konvoluci pixelu v okolí 3x3 pixely s konvoluční maskou – jádrem.

Pro objasnění co je to vlastně hrana, uvedu několik definic z fyzikálního a matematického pohledu [6]:

- Hrana popisuje rychlost změny a směr největšího růstu obrazové funkce *f(x,y)*.
- Hrana odpovídá gradientu obrazové funkce.
- Hrana je místo v obraze, kde se prudce mění intenzita jasu.
- Hrana je přechod z nízkofrekvenční do vysokofrekvenční oblasti signálu.
- První derivace hrany předpokládá v jejím místě lokální maximum.
- Druhá derivace hrany předpokládá v jejím místě průchod nulou.

Hrana odpovídá gradientu, proto je k detekci hran potřeba získat gradient. Gradient dvourozměrné obrazové funkce *f(x,y)* je vektor parciálních derivací

$$|\nabla f(x,y)| = \left(\frac{\partial f}{\partial x}\right) + \left(\frac{\partial f}{\partial y}\right) \tag{3.1}$$

a jeho velikost lze matematicky vyjádřit obvyklým způsobem jako je velikost vektoru:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{3.2}$$

Pro úlohu je vhodná aproximace pro diskrétní signál, parciální derivace v rovnicích 4.1 a 3.2 jsou pak aproximovány diferencemi:

$$\Delta_x f(x,y) = f(x,y) - f(x-n,y) \tag{3.3}$$

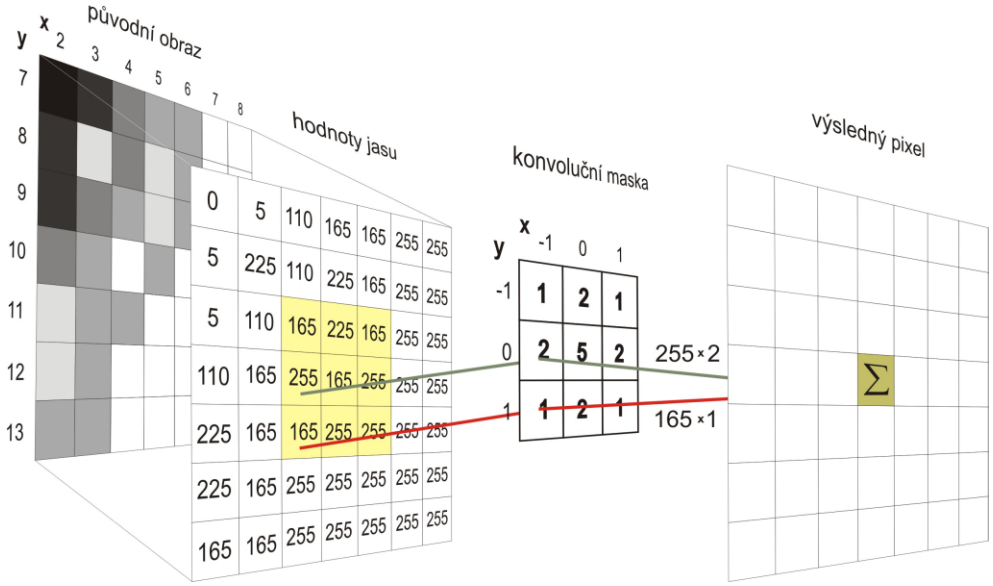
$$\Delta_y f(x,y) = f(x,y) - f(x,y-n) \tag{3.4}$$

Prakticky se k aproximaci parciálních derivací používají gradientní (hranové) operátory, které aproximují první derivaci. Jsou to masky, představující konvoluční jádra pro vertikální a horizontální směr. Protože algoritmus zpracovává diskrétní signál, pro nalezení

hrany se využívá **diskrétní konvoluce**, což je operace s jasem v bodě *(x,y)* v okolí O, velikost M x N obrazové funkce f s jádrem h:

$$g(x,y) = \sum_{m=x-M/2}^{y+M/2} \sum_{n=x-N/2}^{y+N/2} h(x-m, y-n) f(m,n) \tag{3.5}$$

Konvoluci lze chápat operace s jejím konvolučním jádrem, které „položíme“ na aktuálně zpracovávaný pixel v obraze. Každý pixel překrytý jádrem vynásobíme koeficientem v příslušné buňce a provedeme součet těchto hodnot, obrázek 3.2. Výsledkem je aproximace první derivace ve směru x nebo y (konvoluční jádra se pro každý směr liší).



Obrázek 3.2: Princip diskrétní dvourozměrné konvoluce [6]

Protože výpočet velikosti gradientu je podle vztahu 3.2 díky odmocnině komplikovaný, je vztah aproximován součtem absolutních hodnot aproximací derivací obou směrů:

$$\nabla = |g_x| + |g_y| \tag{3.6}$$

Pro výsledek celého algoritmu je důležitý vztah 3.6 označovaný jako modul gradientu. S výslednou hodnotou se provede prahování.

Pro nalezení první derivace g_x a g_y jsem využil konvoluční jádra označované v literatuře jako *gradientní operátory* o velikosti 3x3 pixelů.

Operátory jsou pojmenovány podle jejich autorů a využívají se v počítačovém vidění:

Sobelův operátor

$$h_x = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, h_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{3.7}$$

Operátor Prewittové

$$h_x = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}, h_y = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} \tag{3.8}$$

Robinsonův operátor

$$h_x = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{pmatrix}, h_y = \begin{pmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{pmatrix} \tag{3.9}$$

Konvoluce obrazu s gradientními operátory 3.7, 3.8, 3.9 dává různé výsledky detekce hran. Všechny tři operátory jsem implementoval a provedl jejich srovnání v páté kapitole, kde jsem algoritmus ověřil.

Pro úplnost uvedu, že existují i operátory, které aproximují druhou derivaci. Takovým je například Laplaceův operátor. Operátory druhé derivace indikují hranu tam, kde signál projde nulou. Nevýhodou je, že jsou velmi citlivé na šum v obraze. Pro jednoduché aplikace bez dalšího předzpracování jsou proto nevhodné a pro návrh jsem je předem vyloučil [6].

Prahování

Podle velikosti výsledného gradientu, získaného konvolucí, je třeba rozhodnout, zda je vyšetřovaný pixel hranou nebo pozadím – viz poslední rozhodovací proces vývojového diagramu na obrázku 3.1. V obraze totiž vystupují i nevýznamné hrany a šum. Podle světelných podmínek snímané scény mohou navíc jako hrany vystupovat i stíny a odlesky objektů. Dalším zdrojem falešných hran je šum v obraze. Rozhodnutí závisí na

nastaveném prahu, který je porovnán s výslednou velikostí gradientu. Pokud je gradient větší, než nastavený práh jedná se o hranu. Naopak je-li menší, je pixel nastaven jako pozadí. Nastavení prahu není samozřejmě jednoznačné a závisí na konkrétních podmínkách snímané scény. V návrhu jsem otázku prahování vyřešil možností ručního nastavení prahové hodnoty.

Vyhlazování

Derivace je velmi citlivá na šum v obraze. Vyhlazování obrazu (filtrace) způsobí právě potlačení nežádoucích vyšších frekvencí – šumu. Ten je přítomný v signálu z kamery v důsledku několika vlivů, z nichž zásadní je kvantizační šum kamerového snímače, který vzniká během digitalizace v A/D převodníku. Základní metodou vyhlazování je *obyčejné průměrování*. Ke každému pixelu se přiřadí jas, který je aritmetickým průměrem původních jasů ve zvoleném okolí. Pro vyhlazování jsem použil konvoluční masku pro okolí 3x3:

$$h_1 = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.10}$$

Nevýhodou je, že při vyhlazení dojde na ostrých přechodech k mírnému rozmazání a kvůli tomu utrpí i tenké hrany a další detaily.

Řádková paměť

Ve vývojovém diagramu na obrázeku 3.1 se jedná o první proces. Vstupem dat pro zpracování je diskrétní signál z kamerového snímače a ten je po jednotlivých pixelech ukládán do řádkové paměti. Potřeba paměti vychází z nutnosti mít k dispozici pixel a jeho okolí o velikosti 3x3, pak lze konvoluci zahájit.

3.2 Výběr platformy pro implementaci

Výběr vhodného mikroprocesoru pro zpracování videosignálu z kamery jsem uvažoval z několika možných platforem [8]:

- **ASIC** - zákaznický integrovaný obvod (Aplication Specific Integrated Circulit), podporuje synchronní i asynchronní návrh, levné řešení pro velké série, řádově desetitisíce kusů. Návrh na úrovni hradel je náročný a drahý a možný jen na profesionálním pracovišti. Obvody mají vysoký výpočetní výkon s velmi vysokou možností paralelizace výpočtů.
- **FPGA** - programovatelná hradlová pole FPGA (Field Programmable Gate Array), podporují synchronní návrh, levné řešení pro kusový návrh. Návrh je středně náročný na úrovni hradel a možný na běžném pracovišti. Stejně jako ASIC mají obvody vysoký výpočetní výkon s velmi vysokou možností paralelizace výpočtů.
- **CPLD** - programovatelná logická pole (Complex Programmable Logic Fields), návrh je jednoduchý a možný na běžném pracovišti, cena je nízká, vhodné řešení pro kusový návrh. Nelze však tento typ obvodů uvažovat pro rozsáhlejší systém z důvodu malého množství logických bloků.
- **DSP CPU** - procesor pro zpracování signálů (Digital Signal Processor CPU), návrh je „neomezený“ daný čistě programem a případně speciálními bloky obvodu. Obvody jsou vhodné pro malé série, návrh je nejlevnější, jednoduché programování, ovšem výpočetní výkon je minimálně o řád nižší, než pro obvody programovatelné logiky.

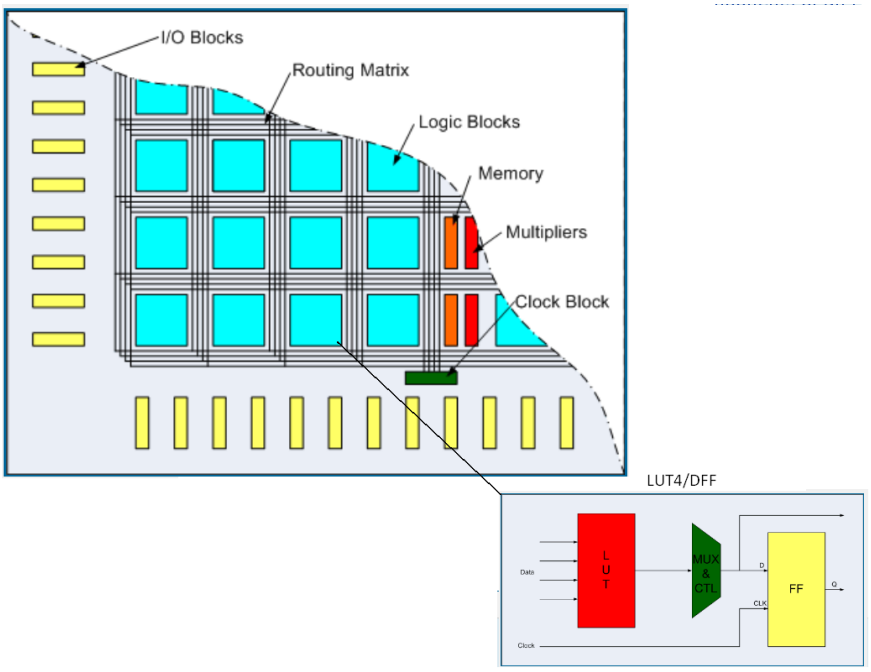
Na základě porovnání uvedených vlastností obvodů vyplývá pro úlohu vhodné využít obvod programovatelné logiky. Obvody jsou vhodné pro zpracování velkých datových toků. Protože navrhuji kusový prototyp, je vhodný konkrétně obvod typu FPGA. Oproti obvodům ASIC jsou obvody FPGA flexibilní z důvodu jejich rekonfigurace.

3.2.1 Obvody FPGA

Důležitou uvedenou vlastností hradlových polí je paralelizace výpočtů. To znamená, že v jednom hodinovém cyklu je provedeno velké množství výpočetních operací a přesunů dat mezi registry. Zpracování videosignálu je typickým příkladem takového přístupu, kdy pracuji s velkým množstvím dat a pro jejich rychlé zpracování je výhodné provést výpočty paralelně. V jednom hodinovém cyklu je tak možné zpracovat n bitů video-

signálu a provést desítky početních operací. Návrh obvodu má být synchronní. To znamená, že všechny stavy registrů se v obvodu mění současně. Představuje to použití synchronních sekvenčních obvodů a nejlépe jednoho hodinového signálu. Oba požadavky nejsou zavazující, nicméně nedodržení synchronního návrhu vede ke vzniku hazardů a nepředvídatelnému chování obvodu. Takové chování lze obtížně ve fázi návrhu zachytit nebo odladit a vede k nedefinovaným chybám. Návrh programu pro konfiguraci obvodu FPGA je vlastně návrhem hardware obvodu. Program popisuje chování obvodu a to vede k vytvoření aplikace přesně na míru specifikovaných požadavků a je v tomto ohledu ekvivalentem ASIC obvodů. Výhodou je malá spotřeba, která je srovnatelná s CMOS obvody.

Historie obvodů sahá do roku 1984, kdy započala jejich výroba firmou Xilinx a má v současnosti nejširší sortiment obvodů. Jejich struktura vychází z obvodů programovatelné logiky PLD. Z obvodů PLD vychází jak FPGA, tak obvody CPLD, které jsou prakticky funkčně shodné, ovšem s významným rozdílem: obvody CPLD mají podstatně méně logických programovatelných bloků a navíc nemají specifické bloky, jako je interní paměť RAM nebo hardwarové násobičky. Mezi další světové výrobce programovatelných hradlových polí patří Altera, Atmel, Cypress Semiconductor, Achromix, Lattice Semiconductor [9]. Typická struktura obvodu FPGA je na obrázku 3.3.



Obrázek 3.3: Vnitřní architektura obvodu FPGA, Zdroj [9]

Architekturu FPGA tvoří programovatelné bloky propojené konfigurovatelnou maticí spojů [10]:

- Konfigurovatelné logické bloky (Logic Blocks) - buňka s podporou implementace logické funkce čtyř proměnných pomocí LUT4 bloků „Look UP Table“ a D klopných obvodů pro realizaci sekvenční funkce – na obrázku 3.3 vpravo dole.
- Vstupní a výstupní bloky (I/O Blocks) - jsou vyvedeny na vlastní piny integrovaného obvodu s podporou třístavové logiky (H, L, Z [2]) a lze je programově konfigurovat. Důležité je, že vstupně / výstupní bloky podporují celou řadu signálových standardů (LVTTL, LVCMOS, HSTL, SSTL) a lze je tak nakonfigurovat podle logických úrovní připojených periferií.
- Blok paměti RAM (Memory) - velikost je 18Kbit, celkový počet dán typem obvodu, každý blok je synchronní dvouportová paměť RAM s nezávislými řídícími signály.
- Multiplexer - hardwarové násobičky, počet je dán typem obvodu.
- Hodinové obvody (Clock Block)- správa a distribuce globálních hodinových signálů a rozvodů z externích zdrojů, mohou být až čtyři.
- Propojovací matice (Rountig Matrix) - fyzické propojení logických bloků.

Způsob konfigurace FPGA je dána technologií. Existují dva typy – FPGA s volatilní a nevolatilní konfigurací [8].

- Volatilní konfigurace vyžaduje pro funkci obvodu externí paměť ROM, ze které je po připojení k napájení obvod nakonfigurován. Tím se programově nastaví konfigurační propojky ve vnitřní paměti obvodu typu SRAM. Vývojové kity mají osazeno FPGA s nevolatilní konfigurací a lze ji proto snadno měnit dokonce i za běhu systému. Nastavení obvodu po startu systému je provedeno řádově ve stovkách milisekund. Výhodou tohoto řešení je téměř „nekonečná“ reprogramovatelnost FPGA.
- Nevolatilní princip využívá obvodovou paměť typu EEPROM nebo FLASH a konfigurace je uložena napevno. Nevýhodou je u tohoto principu nemožnost snadné rekonfigurace, což pro je pro vývojáře nevhodné. Výhodami je ochrana intelektuálního vlastnictví znemožňující vyčtení programu, nezávislost na napájení obvodu a také nižší spotřeba. Navíc je systém plně funkční ihned po zapnutí systému.

[2] Logicá nula L, logická jednička H a stav vysoké impedance Z

3.3 Kamerový snímač

Zdrojem obrazových dat pro algoritmus je kamerový snímač. Pro uvedený postup zpracování obrazových dat je nutné splnit tři základní vlastnosti, které kamera musí mít:

- Progresivní skenování - zachycuje obraz řádek po řádku a je nutné vzhledem ke stejnému režimu ukládání dat. Zpracování dat s progresivním skenováním je obecně jednodušší vzhledem ke spojitosti datového toku v čase.
- Digitální snímač - kamera může být digitální nebo analogová, respektive kamera s integrovaným A/D převodníkem. Je výhodné použít digitální kameru a nutnost A/D převodu (externího převodníku) odpadá.
- Paralelní osmibitový výstup pixelů standardního formátu VGA.

Volba technologie kamerového snímače je běžně možná z CMOS nebo CCD technologie. Pro obecnou aplikaci je lepší CCD snímač hlavně pro kvalitní parametry světelné citlivosti a menšímu šumu. Kamera se snímačem CMOS je naproti tomu levnější a pro ověření algoritmu detekce hran postačující. Nevýhodou je o řád větší šum než u CCD snímačů. Proto je nutné zajistit dobré světelné podmínky snímané scény, což v obecném případě nelze zajistit. Pro práci jsem zvolil CMOS snímač pro jeho cenu, která je i o řád nižší než u CCD snímačů a také kvůli provedení. Snímače CMOS jsou dostupné i jako průmyslový modul určený k zabudování *(embedded module)* a mají k dispozici výstup „syrových“ dat na úrovni bitů.

Základním prvkem CMOS snímače je dvourozměrné pole fotocitlivých elementů – fotodiod. Počet elementů udává celkové rozlišení kamery. Výhodou je malá spotřeba, protože v CMOS technologii jsou polovodičové prvky řízené elektrickým polem s jednou hladinou napájecího napětí. Snímače se dále dělí na pasivní a aktivní. Pasivní snímač přímo generuje elektrický náboj, odpovídající intenzitě světelného záření a následuje zesílení signálu, protože má velmi malou hodnotu. Aktivní snímače mají u každé fotodiody zesilovací tranzistor a zesílený signál je dále zpracován. Dochází k eliminaci šumu, který je u pasivních snímačů větší. Uvedený princip funguje pro snímání monochromatického signálu, protože fotodiody jsou schopné zaznamenat jen informaci, s jakou intenzitou světlo dopadá. Není možné zaznamenat barvu dopadajícího světla. Barevné snímače mají před každou fotodiodou barevný RGB filtr. Dopadající světlo prochází nejdříve filtrem a pak dopadá na fotodiodu. Filtr dané barvy zajišťuje průchod světla o dané frekvenci a tím je možné získat novou informaci o jasu patřičné barvy. Následné digitální zpracování signálů ze snímače musí mít informaci, k jakému snímacímu elementu

přísluší daná barva. Navržený algoritmus jsem se rozhodl ověřit s **monochromatickým i baravným** CMOS snímačem.[11]

3.4 Jazyk a vývojové prostředí

Po nalezení vhodné platformy pro implementaci algoritmu, je třeba zvolit vhodný programovací jazyk a vývojové prostředí.

Pro programování obvodů hradlových polí FPGA existují speciálně navržené jazyky. Umožňují popis chování i simulaci rozsáhlých číslicových systémů a jsou primárně určeny pro popis hardware. Návrh programového kódu je značně odlišný od jiných programovacích jazyků. Vytvořený kód, který popisuje chování systému je syntetizován na fyzické propojení hradel v hradlovém poli. To znamená, že přeložený kód parametruje fyzickou strukturu obvodu. Pro popis algoritmu aplikovaného do FPGA jsem mohl zvolit jazyk *nižší* nebo *vyšší* úrovně [12].

Srovnatelné a v praxi používané jazyky *nižší úrovně* jsou VHDL a Verilog. Standard jazyka VHDL byl poprvé publikován v roce 1987. Verilog je mladší systém poprvé uvedený v roce 1995. Lze říci, že jazyky mají stejné vyjadřovací schopnosti. Oba jsou založené na překladu do RTL úrovně (Register Ttransfer Level) – jazyky pro návrh založený na přesunech mezi registry. Návrh je rozdělen na moduly, datové cesty mezi moduly a řídící obvody, operující nad daty. V návrhu je tak proveden popis funkčních modulů a jejich propojení datovými cestami – signály.

Jazyky *vyšší úrovně* jsou jazyky pro popis číslicových systémů na bázi C a C++ jako je SystemC nebo Handel-C. Tyto jazyky dosahují vyššího stupně abstrakce, jsou však vzdálené od návrhu na úrovni hradel. To nemusí být s ohledem na optimální kód výhodné. Je třeba se spolehnout na syntézu – překlad, což neodpovídá kódu, který navrhne vývojář na nižší úrovni. Uvedené jazyky kladou důraz na popis algoritmů a nejsou koncipovány pro přímý popis hardware.

Pro návrh jsem použil jazyk VHDL ze dvou důvodů. Výrobce zvoleného obvodu FPGA Xilinx dává k dispozici i návrhový systém pro jazyk VHDL, který je zdarma. Druhým důvodem byla má dřívější praktická zkušenost s jazykem a základní znalost jeho syntaxe.

3.4.1 Jazyk VHDL

Historicky jazyk VHDL vznikl v rámci výzkumného projektu VHSIC - Very High Speed Integrated Circuits- ministerstva obrany USA. Z toho vychází i jeho zkratka, totiž VHDL je akronym VHSIC Hardware Description Language. Důvodem vzniku byl efektivní popis rozsáhlých integrovaných obvodů, který byl do té doby řešen pomocí různých a navzájem nekompatibilních jazyků V současnosti se používá pro návrh a simulaci v jazyce VHDL standard IEEE Std 1076-1993 označován jako VHDL93.

Základní konstrukce v jazyce VHDL má dvě části: deklarace entity a tělo architektury. *Entita* je primární návrhová jednotka popisující modul a definuje jeho vstupy a výstupy. Úroveň abstrakce může být od jednoduchého logického hradla až po celý systém [3]. Hlavní kód kompletního programu může obsahovat mnoho bloků propojených mezi sebou *signály*.

Příklad definice ENTITY:

```
entity AND is
port (a,b: in bit; q: out bit)
end entity AND;
```

Architektura definuje vlastní chování a algoritmy entity – vztahy mezi vstupy a výstupy entity. Každá entita musí mít alespoň jednu architekturu, takže je možné ke každé entitě definovat více architektur. Pak architektura stejné entity musí mít různé jméno.

Příklad architektury pro entitu AND:

```
architecture RTL of AND is
begin
q <= a and b;
end architecture RTL;
```

Jednám ze základních struktur definované uvnitř architektury je *procedura*. Z vnitřního pohledu entity je velmi důležité, že příkazy uvnitř procedury jsou vykonávány sekvenčně. Z vnějšího pohledu má celý program charakter paralelního chování. Procedura má v hlavičce citlivostní seznam, na základě kterého se v proceduře spustí proces. Pro moduly sekvenční logiky to je často změna hodinového signálu. Propojení modulů je provedeno v programu „vodiči“ *signal*. Jsou definovány v architektuře a jsou datovými cestami mezi entitami.[12]

[3] Pouze teoreticky, pro rozsáhlý systém se nepoužívá

Specifickou deklarací je soubor **.ucf*, kde jsou nadefinovány parametry fyzických pinů FPGA pro jeho propojení s vnějším světem. Proto je nutná specifikace připojovaných externích periferií. V souboru jsou definovány například logické napěťové úrovně, systém hodinových signálů z krystalu, podrobně dostupné v *Constraints Guide* [13]. V návrhu jsou periferiemi monitor a kamera. Podrobnou specifikaci parametrů kamery jsem provedl ve čtvrté kapitole.

3.4.2 Vývojové prostředí

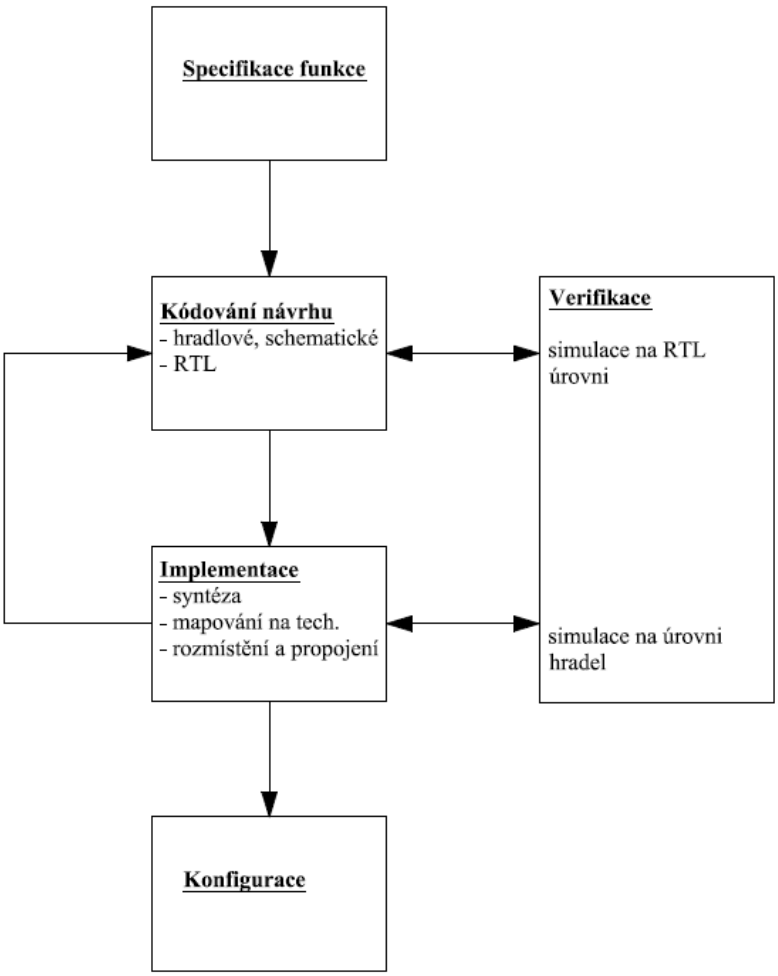
Návrh algoritmu jsem provedl ve vývojovém prostředí ISE Web Pack verze 14.1. 2012 firmy Xilinx. S instalací prostředí je k dispozici systém s podporou celé řady obvodů firmy Xilinx, nástroj pro syntézu i simulační prostředí ISim. Vše je volně dostupné na stránkách výrobce [14], je nutné se pouze zaregistrovat. Prostředí je omezenou plnou verzí, omezení se vztahuje na rozsah návrhu a zpomalení překladu, což pro úlohu nevadí.

Vývojové prostředí musí umět převod textového popisu v jazyce VHDL na logické bloky, což je označováno jako obecná syntéza. Obecná syntéza je pak aplikována na konkrétní vybraný obvod FPGA, kdy dojde k rozmístění logických bloků a jejich propojení. Proto je důležitá podpora výrobce obvodu a je výhodné použít kompaktní nástroj obsahující obě části - syntézu i propojení (mapování) obvodu. Jinak by bylo potřeba nástrojů dvou. Výhodou je, pokud má vývojové prostředí i nástroj, kterým lze syntéza simulovat a odhalit chyby v návrhu. Všechny požadavky vybrané prostředí ISE WebPack splňuje.

Typický průběh návrhu systému ve vývojovém prostředí je na obrázku 3.4. Etapy návrhu jsou: specifikace požadavků, zachycení návrhu, simulace, verifikace a implementace a všechny se můžou opakovat v důsledku změny požadavků a ladění chyb návrhu. Proto se jedná o návrhový cyklus [15].

Specifikace funkce

Specifikace znamená popis chování obvodu, definice hodinových signálů, časové chování obvodu, reakce na náběžnou nebo sestupnou hranu, definice vstupů a výstupů, komunikačních rozhraní, atd. V prvním kroku etapy je potřeba vytvořit si teoreticky rozbor dané úlohy, aby bylo možné zadanou úlohu dále naprogramovat.



Obrázek 3.4: Návrhový cyklus vývoje a jeho jednotlivé etapy [15]

Kódování návrhu

Kódování lze ve vývojovém prostředí provést buď ve formě popisu v jazyce VHDL, nebo na hradlové/schématické úrovni. Jsou to dvě odlišné úrovně abstrakce návrhu:

1. Hradlové/schematické

   Navrhuje se přímo kreslením schématu budoucího obvodu. Výhodou je jeho srozumitelnost a zachycení skutečné podoby návrhu. Realizace odpovídá přímo tomu, co je ve schématu. Kreslené schéma je ale obvykle specifické pro zvolený obvod, protože často jsou použity struktury, které můžou být k dispozici jen na zvoleném obvodu FPGA. Konverze návrhu do jiného obvodu FPGA, nebo do zákaznického integrovaného obvodu, znamená překreslení schématu. Vlastní proces kreslení je pomalý v důsledku nízké úrovně abstrakce. U schématického návrhu

převažují nevýhody nad výhodami. Zbytečně komplikované a časově náročné jsou i opravy chyb v navrženém obvodu. Může tak nastat situace, která vyžaduje kompletní překreslení celého obvodu. Uložené schéma je navíc obvykle nepřenosné mezi různými vývojovými prostředími, protože je uložené v proprietárním formátu. Schematický návrh obvodů FPGA se pro jeho nevýhody téměř nepoužívá. Je výhodný pro malé a jednoduché návrhy nebo pro didaktické účely pro svoji názornost.

2. Meziregistrové přenosy, tzv. RTL.

   Digitální synchronní obvody se skládají ze dvou základních typů logiky. Jsou jimi registry a kombinační funkce. Návrh systému na RTL úrovni probíhá tak, že jednotlivé struktury funkčních bloků jsou popsány v textové formě pomocí těchto dvou typů logiky a stejně tak je pomocí popisu realizováno jejich propojení. Textový popis je ve specializovaném programovacím jazyce VHDL. Hlavními výhodami jsou technologicky nezávislý popis obvodu na relativně vysoké úrovni abstrakce. Jazyky HDL mají běžně podporu pro vytváření vysoce konfigurovatelných (generických) struktur. To umožňuje navržené bloky použít znovu.

   Popis RTL je dnes standardním prostředkem pro zachycení návrhu. Jedinou nevýhodou je, že není nevhodný pro ryze asynchronní návrh. To však není při práci s hradlovými poli omezující, protože hradlová pole nejsou pro takové návrhy vhodné. Jazyk VHDL není ovšem omezen jen na práci na úrovni RTL. Můžeme stejně dobře popisovat chování obvodu na behaviorální (vhodné pro různé pomocné bloky pro simulaci, obvykle nelze užít pro návrh), strukturní (hierarchie bloků) i hradlové úrovni (pak vlastně získáme textovou podobu schématu - tzv. netlist). Ne všechny konstrukce, které jazyk VHDL umožňuje, jsou použitelné pro návrh logiky. Množina konstrukcí, které můžeme použít pro překlad, je označována jako „syntetizovatelná podmnožina jazyka“.

Verifikace

V etapě ověření probíhá kontrola, zda je daný kód správně naprogramovaný a zda lze po jeho úspěšném zkompilování provést simulaci. Verifikace je významným krokem v procesu návrhu obvodu. V rámci verifikačního procesu je ověřena prostřednictvím simulací funkční správnost navrhovaného obvodu a dodržení časových parametrů použitých prvků. Pro složitější systém je potřeba na základě požadavků ve specifikaci sestavit seznam simulací a pro každou simulaci navrhnout stimuly, které vybudí obvod do předpokládaného stavu. Proto je třeba specifikovat i očekávané odezvy obvodu pro kontrolu správné funkce. Dalším krokem je návrh verifikačního prostředí (testbench). Prostředí je

systém napsaný v jazyce VHDL, který instancuje verifikovaný obvod, aplikuje požadované stimuly a v ideálním případě i sám ověří správnost odezvy obvodu. Simulaci lze provést na dvou různých úrovních abstrakce:

1. Simulace RTL

   Simuluje se návrh zapsaný v kódu VHDL na úrovni abstrakce RTL. Prováděná simulace nerespektuje reálná zpoždění na jednotlivých prvcích logiky, ale je velmi rychlá. Simulace RTL je vhodná pro ověření správné funkce popisu. Pomocí ní jde ověřit, zda je kód funkční a obvod správně reaguje. Není ovšem možné ověřit korektnost interního časování (zpoždění na kombinačních prvcích apod.).

2. Simulace na hradlové úrovni s reálnými zpožděními

   Simulace chování obvodu probíhá až po rozmístění a propojení hradel, tedy finální schéma obvodu implementovaného ve zvolené technologii. Simulátor navíc potřebuje informace o reálném zpoždění a požadovaném časování na jednotlivých spojích a prvcích obvodu. Modely skutečných obvodových prvků se načítají z technologické knihovny poskytnuté výrobcem obvodu FPGA. Ty zajišťují modelování reálného funkčního chování bloků na FPGA, simulují zpoždění šíření signálu a nakonec kontrolují dodržování správných časových parametrů jednotlivých bloků. Při porušení předepsaného časování je během simulace generováno automaticky chybové hlášení.

Implemantace

Posledním krokem etapy návrhu, který je podmíněný úspěšnou kompilací, simulací a syntézou, je převod naprogramovaného kódu na zvolený typ hradlového pole – implementace. Proces implementace se skládá ze tří kroků:

1. Syntéza – mapují se virtuální logické prvky popsané jazykem pro hardwarový návrh na základní logické prvky, které obsahuje konkrétní programovatelné logické pole. Obecně se jedná o proces konverze popisu systému na vyšší úrovni abstrakce na nižší úroveň abstrakce. V případě syntézy RTL se jedná o konverzi kódu RTL VHDL zapsaného pomocí syntetizovatelné podmnožiny jazyka do seznamu logických prvků a jejich vzájemného propojení. Kromě kódu HDL je proces syntézy řízen ještě technologickou knihovnou. Technologická knihovna obsahuje popisy funkcí, časování a další důležité parametry všech prvků dostupných ve zvoleném typu obvodu FPGA. Až v této fázi návrhu je důležité, pro jaký obvod je návrh vytvářen a pokud zvolíme nevhodný obvod, překladač nahlásí chybu – typicky, že překročil dostupný počet konfigurovatelných prvků.

2. Mapování na technologii (mapping)

   Obecně se jedná o konverzi technologicky nezávislého netlistu [4] do buněk dané technologie. Další funkcí mapování je seskupování LUT, registrů a dalších logických prvků do řezů obvodu a doplnění LUT použitých jen pro propojení bloků kdekoliv, kde je to nezbytné. Současně jsou prováděny další jednodušší optimalizace obvodu, odstranění případné zbývající nepoužité logiky, replikace registrů apod.

3. Rozmístění a propojení (place and route)

   Během rozmístění jsou jednotlivé logické prvky rozvrženy do matice obvodu a jejich pozice je zafixována. V druhém kroku propojení jsou postupně zpracovávány jednotlivé spoje mezi prvky v obvodu a jsou identifikovány vhodné propojovací struktury v matici hradlového pole a postupně zafixovány i jednotlivé spoje. Nástroje pro rozmístění a propojení jsou dodávány výrobci obvodů FPGA, protože pro svou správnou funkci potřebují detailní znalost struktury obvodu.

Konfigurace

Posledním krokem návrhového procesu je konfigurace programovatelného hradlového pole. Konfigurace znamená propojit počítač s vývojovou deskou a nakonfigurovat FPGA souborem **.bit*. To lze provést přímo z vývojového prostředí nebo pomocí samostatného software. To záleží na podpoře výrobce případného vývojového kitu.

3.5 Shrnutí

V navrženém algoritmu jsem aplikoval konvoluci pro tři různé konvoluční kernely – gradientní operátor Perwitové, Sobelův a Robinsonův. Spolu s konvolucí jsem v jednom korku provedl vyhlazování, které je vhodné nebo dokonce nutné pro aproximaci první derivace. Vyhlazování rozmazává hrany a je tak protichůdným požadavkem při hranové detekci. Výhodou je, že odstraňuje šum v obraze.

Vybraná platforma mikroprocesoru pro implementaci algoritmu je obvod FPGA s volatilní konfigurací od výrobce Xilinx. Obvody hradlových polí mají konfigurovatelnou strukturu, která není myslitelná u konvenčních programovatelných CPU, kde je dána výrobcem a chování procesoru určuje firmware. Právě pro tuto hlavní výhodu jsem zvolil FPGA obvod od renomovaného výrobce Xilinx.

[4] Textový popis schématu

Návrh programu jsem provedl v jazyce VHDL na RTL úrovni ve vývojovém prostředí ISE Web Pack 14.1 od firmy Xilinx.

Navržený kamerový snímač má technologii aktivní CMOS s progresivním skenováním, integrovaným A/D převodníkem a VGA formátem. Pro ověření algoritmu jsem předpokládal použít kameru s monochromatickým i barevným CMOS snímačem.

Kapitola 4

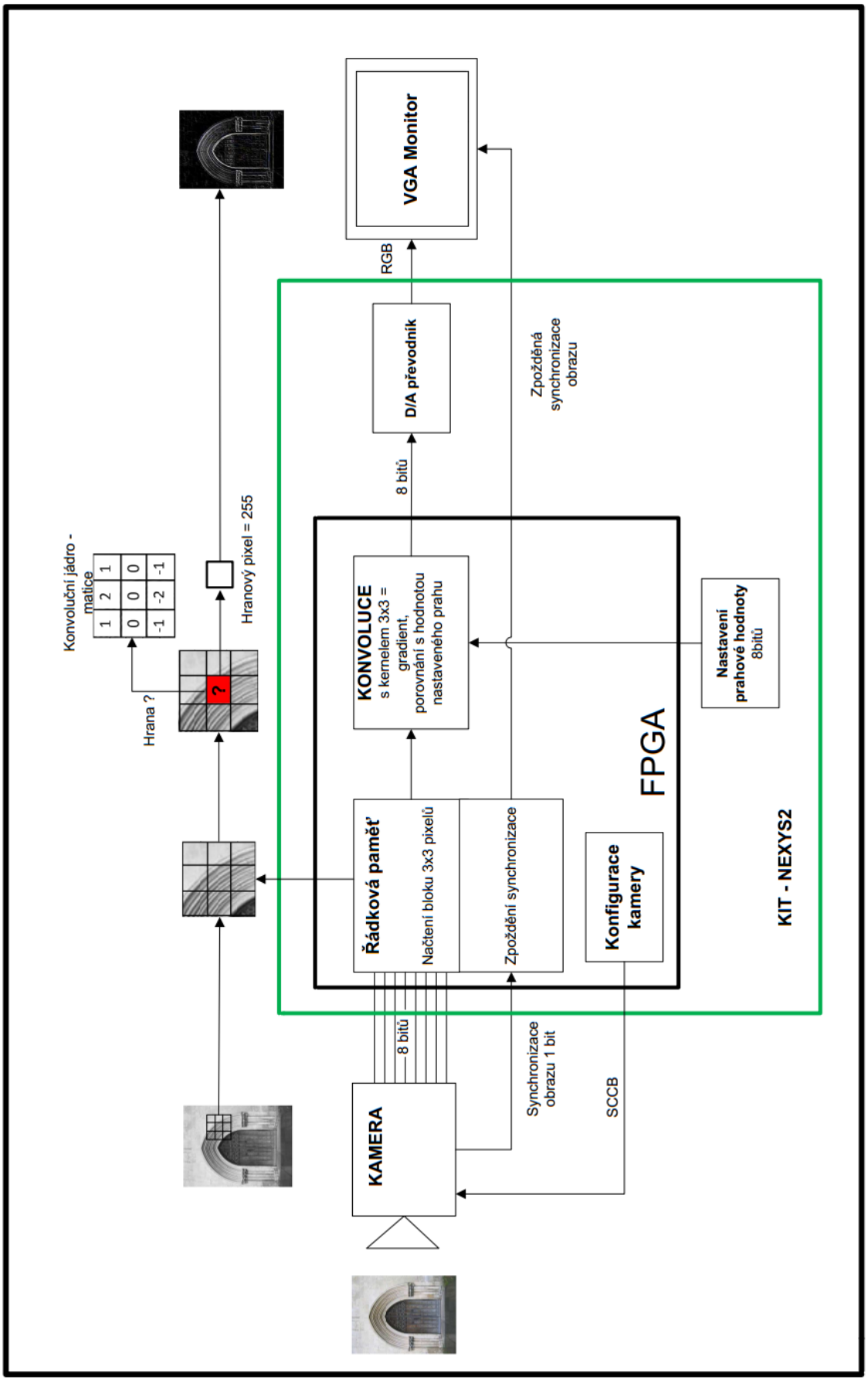
Realizace algoritmu rozpoznávání blízké scény

Následující kapitola se zabývá realizací algoritmu, pro kterou je nutné vybrat konkrétní technické prostředky – kameru, vývojový kit s FPGA a monitor. Algoritmus je navržený v jazyce VHDL a implementoval jsem ho obvodu FPGA na vývojovém kitu Nexys2 od firmy Digilent [16]. Výrobcem obvodu FPGA je Xilinx z rodiny typů Spartan a řady III. Důvodem volby byla jednak dřívější praktická zkušenost s tímto obvodem na vývojovém kitu firmy PK Design [17] a jednak volně dostupné vývojové prostředí Xilinx ISE Web Pack. Celý zdrojový kód členěný na moduly je v příloze [A].

Blokové schéma navrženého systému je na obrázku 4.1.

Celý systém je rozdělen na bloky:

- Vývojový kit
- Periferie na kitu
- Kamera
- Monitor
- Programové bloky realizované v obvodu FPGA
  - Konfigurace kamery – nastaví registry kamery komunikačním protokolem SCCB.
  - Konvoluce pixelu – v okolí 3x3 s filtrem o stejné velikosti nalezne hrany objektů

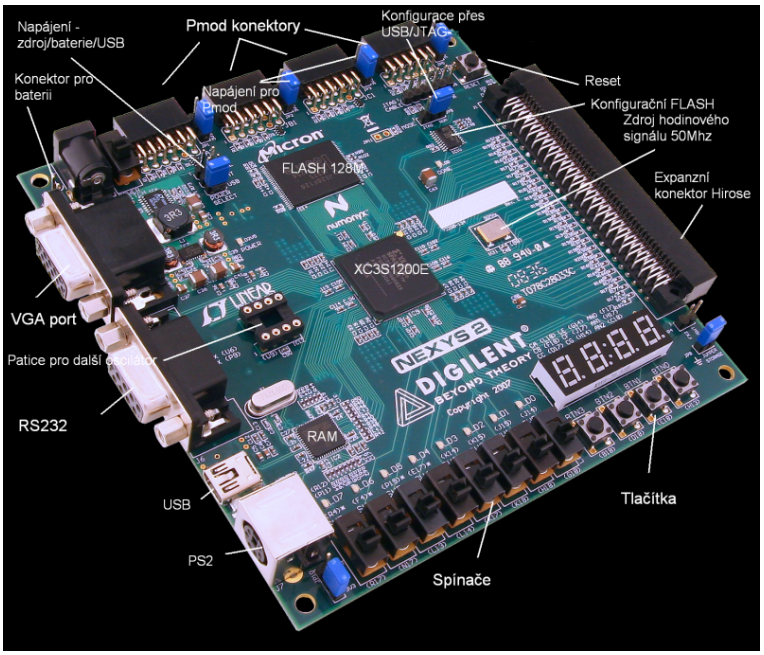


Obrázek 4.1: Blokové schéma celého systému

– Řádková paměť – uchovává osmibitová data z kamery, výstupem z paměti je matice pixelů 3x3 a odpovídající synchronizační signály s definovaným zpožděním

4.1 Vývojový kit

Vývojový kit Nexys2 od firmy Digilent je na obrázku 4.2. Kit je osazen obvodem Spartan XC3S1200E. Firma Digilent nabízí i kit s menším obvodem XC3S500 (500 ekvivalnetních hradel), ovšem cenový rozdíl je minimální. Pro konfiguraci FPGA nutný volně dostupný software *Dilgilent Adept* na stránkách výrobce [16]. Parametrizace obvodu je možná přes rozhraní USB případně JTAG.



Obrázek 4.2: Vývojový kit NEXYS2 [16]

Uvedu několik základních parametrů obvodu XCS1200E osazeného na vývojovém kitu [10]:

- Obvod XC3S1200E má kapacitu 1200 ekvivalentních logických hradel. Kapacita obvodu se udává právě počtem hradel.
- Paměť RAM 504 kbit blokové paměti RAM (označovaná v katalogových listech zkratkou BRAM).
- Krystalový oscilátor pro generování hodinového signálu s kmitočtem 50MHz

- 28 hardwarových násobiček 18x18 bitů
- 304 I/O pinů dostupných uživateli z celkových 320 pinů

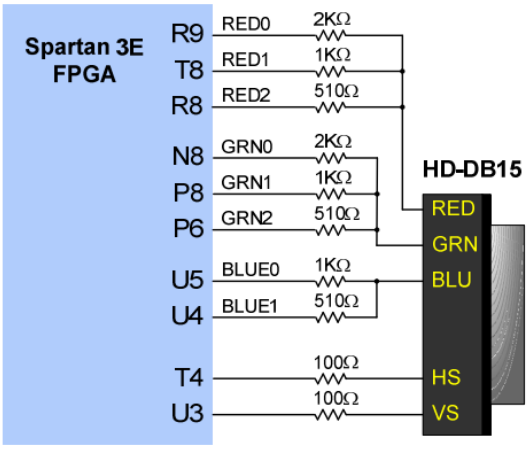
4.2 Periferie kitu

Následující rozhraní kitu jsem využil při realizaci[18]:

- Osm indikačních LED diod, tři stavové LED pro napájení, program a rekonfigurace.
- Tlačítko využité pro restart kamery.
- Osm vypínačů pro nastavení osmibitového prahu pro detekci hran.
- Konfigurační paměť FLASH pro konfiguraci hradlového pole.
- Napájecí konektor – přepínači na kitu lze zvolit napájení ze síťového zdroje nebo pro mobilní využití napájení z externí baterie. Rozsah napájecího napětí je nutné dodržet v rozmezí 5 až 15 V.
- Desetipinové konektory Pmod – piny konektoru jsou napojeny přímo na piny FPGA s ochranou ESD [1] a ochranu proti zkratu. Konektory jsou využity pro připojení kamery.
- Port grafického rozhraní VGA pro LCD monitor – standardní 15 pinový konektor s vyvedenými pěti základními signály: horizontální synchronizace, vertikální synchronizace a tři signály základních barev (červená R, zelená G, modrá B).
- Konektor mini USB – rozhraní nutné pro konfiguraci obvodu a lze využít i pro napájení kitu.
- D/A převodník pro VGA rozhraní – signály FPGA jsou napojeny na příslušné piny VGA konektoru přes rezistorovou síť, která tvoří jednoduchý D/A převodník, obrázek 4.3. Hodnoty rezistorů odpovídají vahám RGB signálu a výstupní napěťové úrovně jednotlivých složek se pohybuje mezi 0 až 0,7 V. Pro červenou barvu jsou vyvedeny tři bity, pro zelenou barvu také tři a pro modrou dva bity [2]. Na kitu je k dispozici osmibitový výstup reprezentující 256 barev. Při vývoji aplikace pro VGA rozhraní je nutné respektovat časování VGA standardu. Synchronizační signály

[1] Electostatic discharge - elektrostatický výboj
[2] Lidské oko je na modrou barvu nejméně citlivé

monitoru pracují s TTL úrovněmi. Je ověřeno, že horizontální i vertikální synchronizace pracuje spolehlivě již od úrovně 2,4 V [19]. Je proto běžné, že synchronizace generovaná grafickým adaptérem VGA má úroveň 3,3 V odpovídající LVTTL a stejné napěťové úrovně využívá i FPGA na kitu. Ostatní piny VGA konektoru nejsou zapojeny.

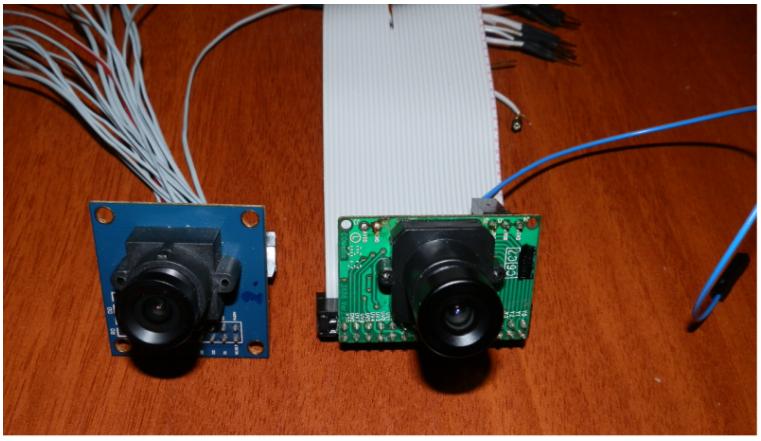


Obrázek 4.3: Vnitřní propojení FPGA a konektoru VGA přes rezistorovou síť [18]

4.3 Kamera

Základní parametry kamerového snímače vychází z návrhu algoritmu:

- CMOS technologie
- Progresivní skenování obrazu - pro navržený algoritmus nelze se snímkem pracovat v režimu prokládaného řádkování
- Osmibitový barevný RGB formát
- Osmibitový monochromatický formát
- Režim VGA s rozlišením 640 x 480, počet snímků 60 za sekundu pro zobrazení na běžném monitoru. Větší rozlišení nemá smysl pro velký objem dat. Pro detekci hran v obraze je velké rozlišení zbytečné, naopak by stačilo i menší rozlišení (320x240). To však nepodporuje běžný monitor.



Obrázek 4.4: Kamerové snímače CMOS firmy OmniVision – vlevo barevná OV7670, vpravo monochromatická OV7620

Kamerové snímače jsem vybíral z nabídky firmy OmniVision [19]. Na fotografii 4.4 jsou kamery, které jsem pro realizaci zakoupil a které splňují výše uvedené parametry. Oba snímače jsou v průmyslovém provedení a jejich základní parametry jsou v tabulce 4.1. Oba moduly mají na snímači jednoduchý objektiv s ručním zaostřováním.

Parametr	OV7620	**OV7670**
Technologie	CMOS	CMOS
Výstup	8 bit / 16 bit	8 bit
Formát	YUV, RGB Bayer	RGB, YUV, RGB Bayer
Počet snímků	60 fps	30 fps
Formát snímače	WVGA 752x480	VGA 640x480
Napájecí napětí	5V	3,3V
Konfigurační rozhraní	I2C	SCCB
Režim snímání	Progresivní/prokládaný	Progresivní
Rozhraní	PFL– 30pin	PFL 16
Cena	1500 Kč	400 Kč

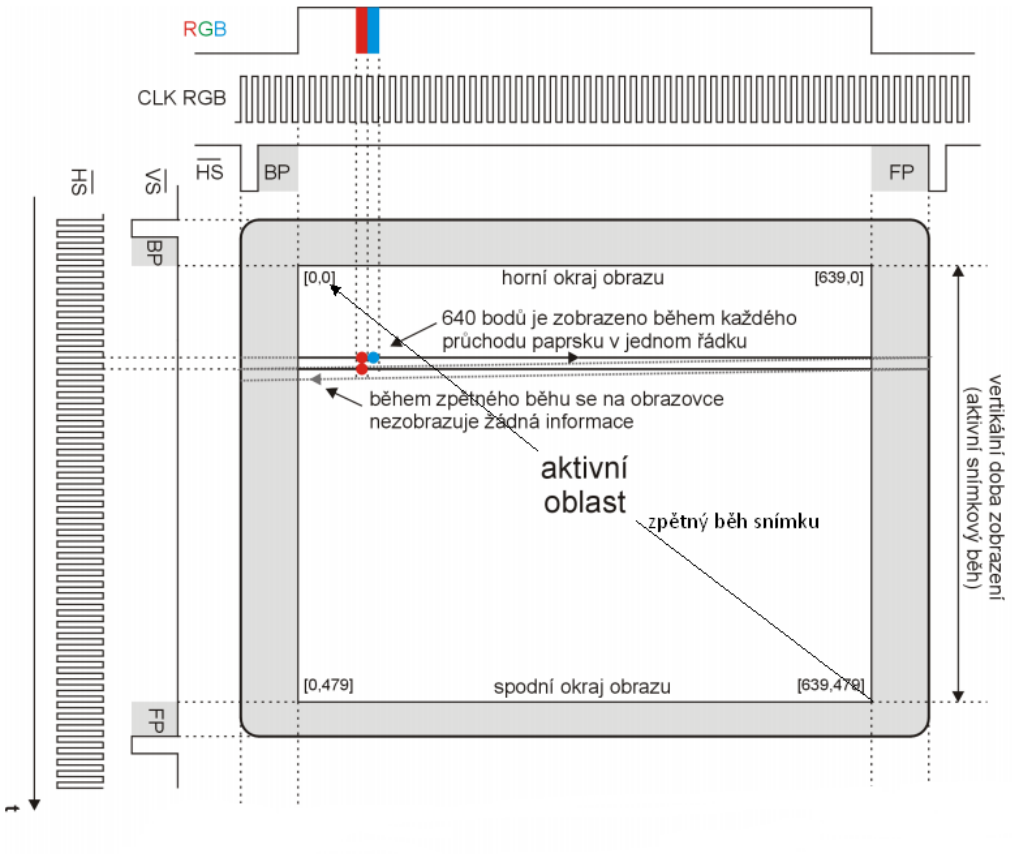
Tabulka 4.1: Srovnání parametrů snímače OV7670 a OV7620

Pixely načítané kamerou jsou ihned zpracovávány v FPGA a následně zobrazeny na obrazovku monitoru bez dalšího uložení. Během zpracování jsou ukládány jen pixely potřebné pro výpočet. Do vlastního obvodu FPGA by se data celého snímku nevešla. Z uvedeného přístupu vychází potřeba stejného časování synchronizačních signálů kamery, jako má standard VGA, protože kamera synchronizuje přímo monitor. Musí být

dodrženy tři parametry: frekvence vzorkovacího signálu 25MHz, řádkový kmitočet 31,25 KHz a snímkový kmitočet 60 Hz.

4.3.1 Časování VGA

Časování souvisí s časovým průběhem synchronizačních signálů. Horizontální synchronizace (dále HS) určuje, kde skončil řádek a vertikální synchronizace (dále VS) určuje, kde skončil snímek. Při aktivním HS se vykresluje řádek zleva doprava po jednotlivých pixelech v horizontálním směru. Posun na další řádek je podmíněn aktivní VS, které posouvá vykreslování po řádcích a je tak aktivní po dobu vykreslení všech 480 řádků, což odpovídá celému snímku. Pro vykreslení dalšího řádku nebo snímku nastává určité prodlení, vycházející z konstrukce CTR monitoru. Vykreslování paprsku a jeho pohyb po obrazovce probíhá zleva doprava pro řádek a z pravého dolního rohu do levého horního rohu pro snímek. Princip je na obrázku 4.5.



Obrázek 4.5: Princip horizontální a vertikální synchronizace monitoru [20]

Časové úseky HS a FS lze rozdělit do čtyř fází [20]:

Horizontální synchronizace udávaná počtem pixelů

- Synchronizační puls – slouží jako informace o tom, že bude další **řádek**, trvá 96 pixelů, během této doby nejsou pixely zobrazovány, jedná se o tzv. zatemňovací pulsy (aktivní v logické nule), paprsek se vrací šikmo z leva doprava na následující řádek v obraze.
- Aktivní oblast – během této doby jsou pixely vykreslovány na displej monitoru a odpovídá 640 pixelům.
- FP (Front porch) – prodleva, kdy se čeká na další synchronizační zatemňovací puls. Technicky je to doba potřebná pro ukončení paprsku a trvá 16 pixelů, šedá oblast pravého kraje obrazu, pixely se nevykreslují.
- BP (Back porch) – prodleva pro návrat na začátek následujícího řádku, než se začnou pixely skutečně vykreslovat, trvá 48 pixelů. Technicky je to doba potřebná pro obnovení paprsku, šedá oblast levého kraje obrazu, pixely se nevykreslují.

Vertikální synchronizace udávaná počtem řádků

- Synchronizační puls – slouží jako informace o tom, že bude další **snímek**, parsek se vrací z dolního pravého rohu zpět nahoru do levého horního roku. Trvá 96 pixelů, během této doby nejsou pixely zobrazovány.
- Aktivní oblast – během této doby jsou vykreslovány řádky na displej monitoru a odpovídá 480 řádkům.
- FP front porch – prodleva, kdy se čeká na další synchronizační zatemňovací puls. Technicky je to doba potřebná pro ukončení paprsku a trvá 10 řádků, šedá oblast dolního kraje obrazovky, pixely se nevykreslují.
- BP back porch - prodleva pro návrat na začátek obrazovky následujícího řádku, než se začnou pixely skutečně vykreslovat. Trvá celkem 33 řádků, technicky je to doba potřebná pro obnovení paprsku, šedá oblast horního kraje obrazu, pixely se nevykreslují.

4.3.2 Časování a nastavení kamery

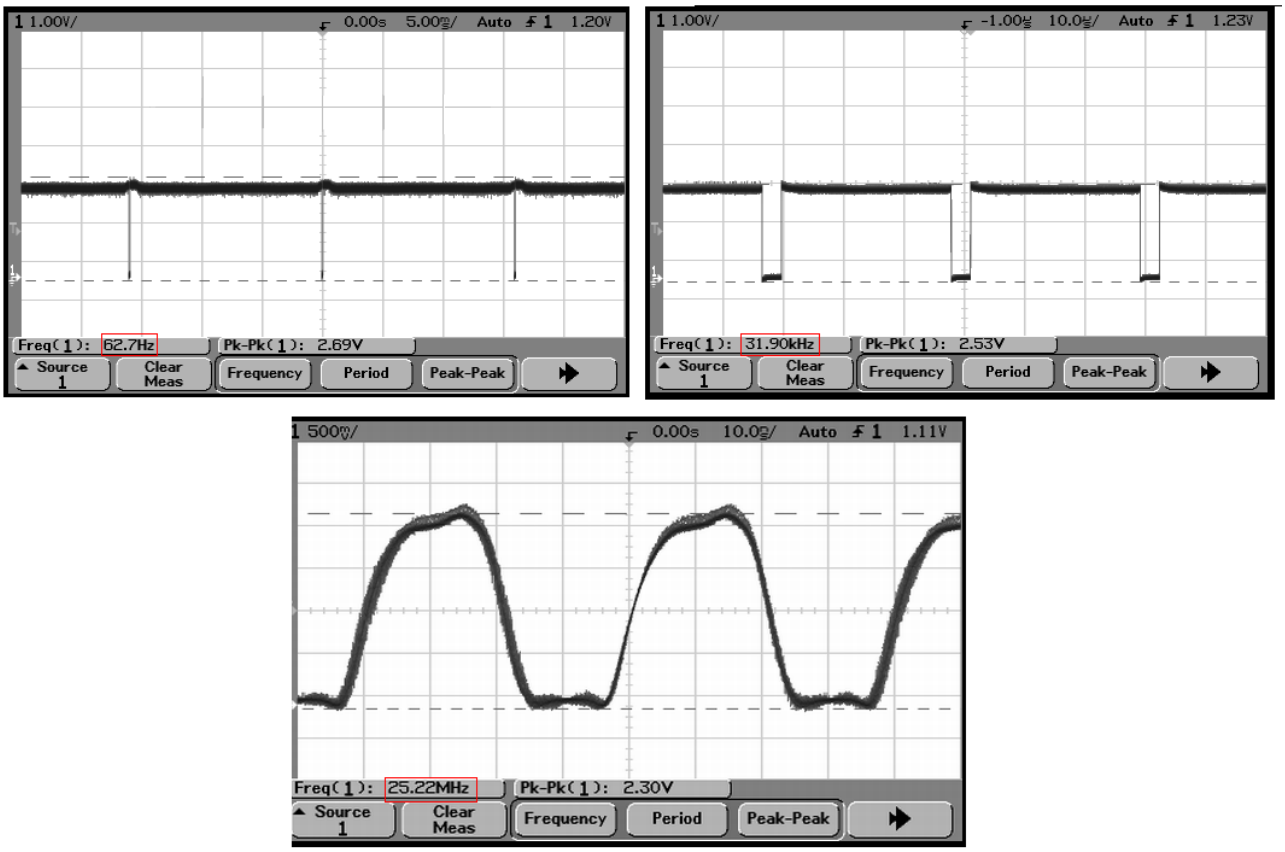
Pro správnou funkci musí být kamera správně nakonfigurována, což platí pro monochromatickou i barevnou. To umožňuje nastavení vnitřních registrů kamery přes rozhraní SCCB, které je obdobou I^2C. Nejdůležitější je nastavení výstupního formátu a jeho definováním je dáno časování horizontální a vertikální synchronizace. V podstatě je rozdíl v tom, zda kamera pracuje v šestnáctibitovém nebo osmibitovém formátu. Nastavení registrů jsem provedl podle *Preliminary Datasheed OmniVision* [21]. Nastavení není jednoduché pro mnoho kombinací a velkého množství registrů a dokonce rozpory v dokumentaci. Proto jsem navrhl jednoduchý testovací modul ve VHDL, který nastavení usnadnil a umožnil snímaný obraz kamerou pozorovat na monitoru. Smyslem bylo dosáhnout parametrů horizontální synchronizace (HS) 31,25 KHz a vertikální synchronizace (VS) 60 Hz.

```
architecture Behavioral of vga is
signal VS : std_logic;
begin
   process(pclk)     --citlivy na vzorkovaci frekvenci kamery
   begin
      if pclk'event and pclk ='1'  then
      if href = '1' then               -- je-li aktivni radek
            vga_red   <= d(7 downto 5);    -- prirad data na port VGA   R
            vga_green <= d(4 downto 2);    --                           G
            vga_blue  <= d (1 downto 0);   --                           B
                       else
            vga_red   <= (others => '0');  -- jinak tma
            vga_green <= (others => '0');
            vga_blue  <= (others => '0');
                      end if;
      end if;
   end process;
end Behavioral;
```

Pomocí testovacího modulu jsem ověřil nastavení registrů jednotlivých typů kamer. Došel jsem k zjištění, že typ **OV7620 nepodporuje VGA časování** a proto jsem musel modul z dalšího návrhu vyloučit. Signály VS a HS totiž odpovídají šestnáctibitovému režimu, přestože kamera má samostatně vyvedených osm pinů pro jasovou složku – osmibitový výstup monochromatického signálu. Doba řádkového a snímkového kmitočtu

je tak dvojnásobná, protože je třeba číst 16 bitů na jeden pixel. Z katalogového listu [22] a parametrů modulu nejsou tyto zásadní omezení zřejmá.

Standardu VGA odpovídají signály kamery druhého typu OV7670. Skutečný průběh signálů HS a VS jsem ověřil osciloskopem a je na obrázku 4.6

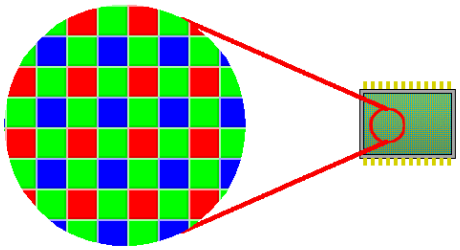


Obrázek 4.6: Časování kamery, zleva vertikální synchronizace, horizontální synchronizace, vzorkovací frekvence

Z naměřených průběhů je zřejmé, že signály skutečně odpovídají VGA časování s malými odchylkami, která nemají zásadní vliv. Kamera má základní výstup dat formátu *RAW Bayer*. Jsou to „syrová“ data bez jakéhokoli zpracování odpovídající přímo snímaným datům CMOS snímače přes Bayerův filtr [3]. Ten má mozaikovité uspořádání, viz obrázek 4.7.

Obrazový signál má vlastně formát RGGB, protože na jeden pixel červené a modré barvy připadají dva pixely zelené barvy [4]. Pro získání skutečné barvy pixelu se používá bilineární interpolace vypočítaná z jeho okolí. Tento proces se označuje také jako demo-

[3] Filtr vymyslel Bryce Bayer z Kodaku a v roce 1975 byl patentován
[4] Oko je na zelenou barvu citlivější



Obrázek 4.7: Mozaika barevného Bayer filtru u CMOS snímačů

zaikování a provádí ho DSP procesor snímače. Výstupem je formát označený výrobcem jako *RAW Bayer Processed*.

Kromě Bayer formátu umožňuje kamera další formáty: RGB422, RGB565, RGB555, YUV, YCbCr. Data jsou pro uvedené formáty také zpracována DSP procesorem snímače a mají šestnáctibitový formát, což je pro návrh nepoužitelné.

Pro ilustraci je na obrázku 4.8 logo ČZU, snímané kamerou pomocí testovacího modulu.

Obrázek 4.8: Obraz kamery na monitoru formátu RAW Bayer

Naměřené a požadované parametry pro rozlišení 640x480 jsem porovnal v tabulce 4.2. Požadované parametru VGA signálů vychází ze specifikace asociace VESA a jsou volně přístupné na internetových stránkách [23].

	VGA rozhraní		Kamera OV7670	
	HS	FS	HREF	VSYNC
	pixelů	řádků	pixelů	Řádků
Aktivní oblast	640	480	640	480
Front porch	16	10	17	17
Synch. puls	96	2	144	3
Back porch	48	33		10
Celý řádek	800	525	801	510

Tabulka 4.2: Srovnání časování kamery a VGA standardu

Horizontální synchronizace kamery odpovídá VGA standardu. Výhodou je, že ekvivalentně k signálu HS kamera generuje signál HREF a ten je aktivní pouze v době platných dat z kamery. To znamená, že zatemňovací puls má délku 144 pixelů. Pro vertikální synchronizaci je potřeba nastavit signálu VSYNC negativní polarizaci, která je jinak opačná viz obrázek 4.9. Jeden snímek trvá kameře jen 510 řádků a zatemňovací puls 3 řádky. Rozdíl není kritický a bylo by nutné parametry dodržet pravděpodobně v případě monitoru CRT. Monitoru typu LCD rozdíl nevadí pro vstupní toleranci signálů. Ostatně standard VGA časování není výrobci monitorů striktně dodržován a parametry časování monitorů se často liší.

Důležitým parametrem je vzorkovací frekvence 25 MHz, která vychází ze tří proměnných:

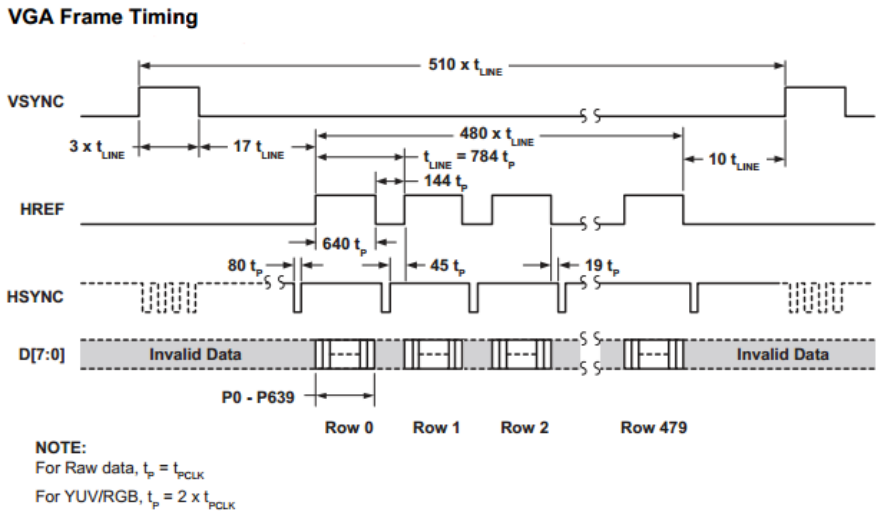
$p = 800$ celkový počet pixelů v řádku

$l = 525$ celkový počet řádků v obraze

$s = 60$ počet snímků za sekundu

Výsledná vzorkovací frekvence pixelů :

$$pls = 800 \cdot 525 \cdot 60 = 25,2MHz \tag{4.1}$$



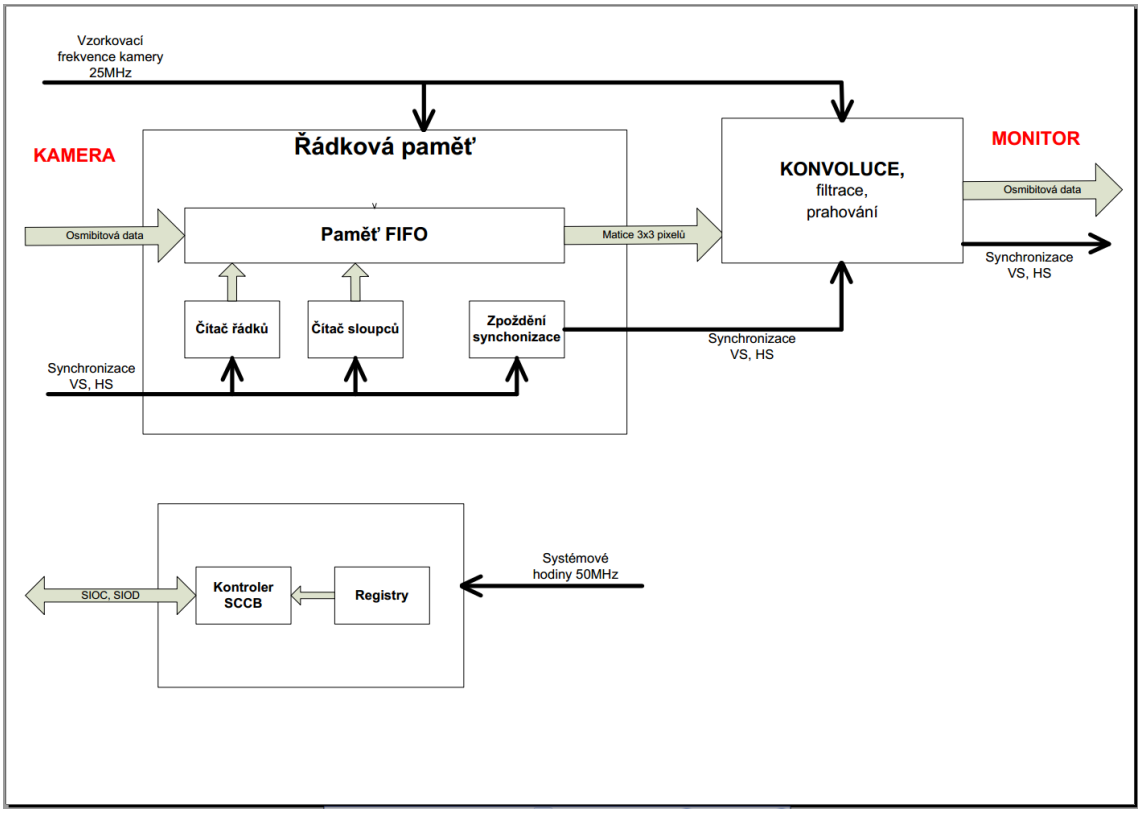
Obrázek 4.9: Časovací diagram kamery OV7670 [21]

4.3.3 Shrnutí

Hodinový signál kamery musí být nastaven na 50 MHz, pak je výstupní vzorkovací frekvence 25 MHz. Kamerový modul OV7670 vyhovuje zásadním parametrům časování pro režim VGA s rozlišením 640 x 480 ověřených měřením: **vzorkování 25,2 MHz, HS = 31,90 kHz a VS = 62,7 Hz s negativní polarizací VS a HS.**

4.4 Bloky realizované v FPGA

Celý systém je rozdělen do tří funkčních bloků na obrázku 4.10. Bloky přímo odpovídají modulům naprogramovaným v jazyce VHDL. Zdrojové kódy bloků jsou v příloze [A]. Části výpisu kódu v následujícím textu jsem uvedl, pokud souvisí a vysvětlují funkci modulu. Pro návrh typické logické jednotky ve VHDL (čítače, paměti, děličky atd.) lze využít knihovny *Language Templates* ve vývojovém prostředí.



Obrázek 4.10: Blokové schéma modulů realizovaných v FPGA

4.4.1 Řádková paměť

Vstupem modulu jsou osmibitová data z kamery formátu *RAW Bayer Processed* a signály vertikální a horizontální synchronizace. Modul ukládá pixely pro následné zpracování. Jedná se uložení řádků do paměti typu FIFO. Výstupní data jsou organizována do matice 3x3. Modul zpožďuje časování synchronizačních signálů z kamery pro monitor.

Pro uložení řádku jsem mohl použít dvě úložiště, jejichž volba má zásadní dopad na návrh:

1. Celý snímek po řádcích načíst do *externí paměti* (SRAM, DRAM) a provést konvoluci. Je zřejmé, že vznikne zpoždění o jeden snímek mezi vstupem a výstupem. Hrají zde roli požadavky na datovou propustnost a velikost externí paměti, danou rozlišením. Výhodou je znalost celého snímku a možnost několika průchodů snímkem v několika hodinových cyklech pro potřeby zpracování, ovšem na úkor zpoždění mezi vstupem dat a výstupem.

2. Konvoluci zahájit, jakmile je to možné – když jsou potřebná data k dispozici. Jedná se o podstatně menší objem dat a je možné je ukládat *přímo do obvodu FPGA*. Nevzniká prakticky žádné zpoždění mezi vstupem a výstupem. Je však nutné všechny výpočetní operace s aktuálním pixelem provést v jednom hodinovém cyklu.

Pro algoritmus jsem využil druhý způsob ukládání dat přímo do obvodu FPGA. Výhodou je, že není potřeba externí paměť, pro kterou by navíc bylo potřeba vytvořit řadič [5]. Navíc zpracování dat v reálném čase nemá znatelné zpoždění. Další výhoda vnitřní paměti je v rychlém přístupu a přímé začlenění do programového kódu.

Vnitřní paměť FPGA je RAM realizována buď jako distribuovaná nebo bloková paměť.

- Bloková paměť – po syntéze vytvořená z jader blokových pamětí dostupných v obvodu. Paměť má synchronní zápis a čtení.
- Distribuovaná paměť – je vyskládána z LUT tabulek a lze ji využít, pokud už není k dispozici paměť bloková. To je na úkor zdrojů použitelných pro implementaci logických funkcí.

Modul **řádková paměť** na obrázku 4.10 má čtyři bloky:

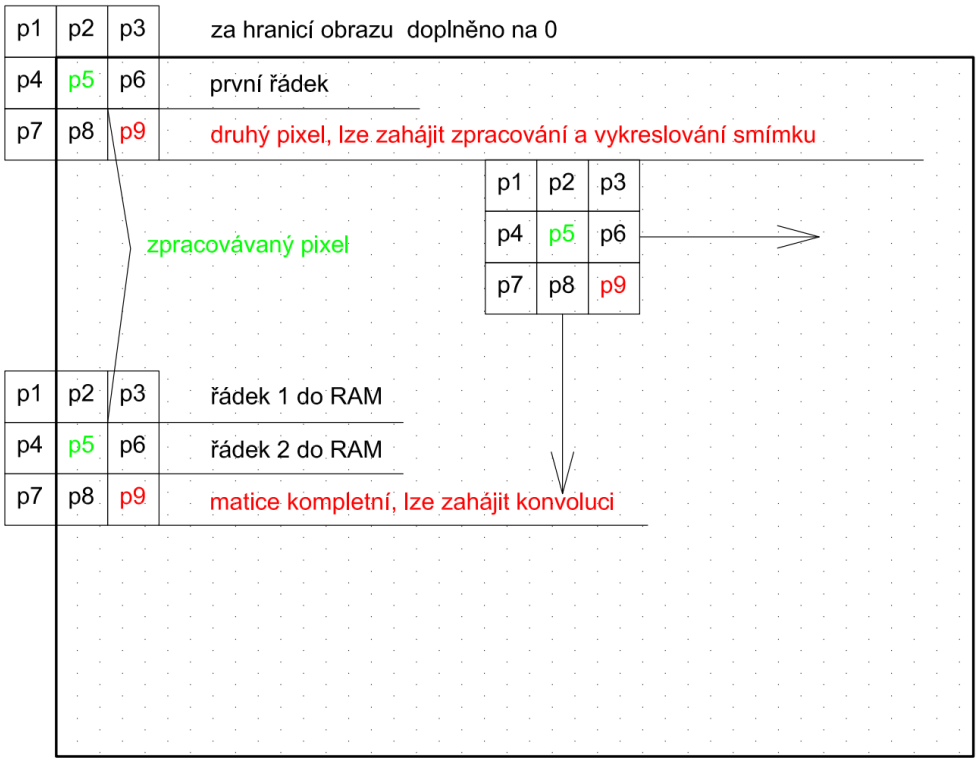
- Paměť FIFO
- Čítač řádků
- Čítač sloupců
- Zpoždění synchronizace

4.4.1.1 Paměť FIFO

Pro výpočet konvoluce s hranovým filtrem potřebuji znát pixel a jeho okolí o velikosti 3x3. Kolik je skutečně potřeba průběžně ukládat dat pro konvoluci je zřejmé ze zna-

[5] Poslední generace obvodů FPGA z rodiny Virtex již mají velkou interní paměť RAM a snímek formátu VGA lze uložit

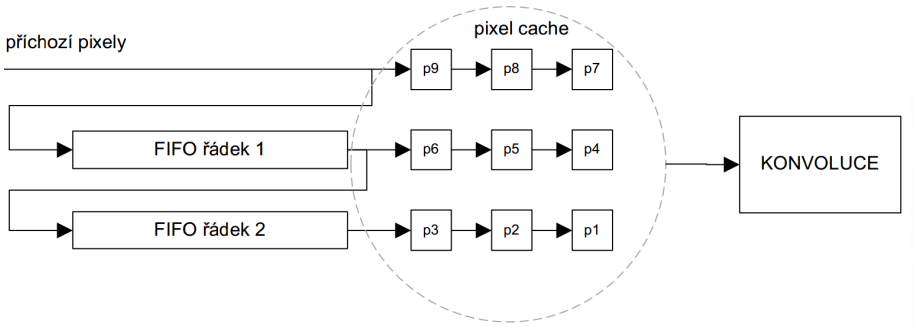
losti velikosti konvolučního jádra. Po příchodu prvního pixelu nemůžu pixel zpracovat, protože jeho okolí neznám. Výpočet lze zahájit v okamžiku, kdy jsou v řádkové paměti uloženy dva řádky a tři pixely z následujícího řádku, viz obrázek 4.11. To platí, pokud je maska uvnitř obrazu. Pak je matice pixelů kompletní. Je-li maska na hranici obrazu monitoru, to znamená na okraji nebo v rohu, je část mimo masku nahrazena hodnotami pixelů pro černou barvu (0). Z toho vyplývá, že úplně první výpočet lze zahájit po načtení a uložení jednoho řádku a dvou pixelů. Zbytek hodnot je doplněn hodnotami 0, na obrázku 4.11 první matice v levém horním rohu.



Obrázek 4.11: Pohyb matice 3x3 aktivní plochou monitoru

Paměťová architektura má organizaci FIFO [6]. a je na obrázku 4.12. Pro návrhu architektury jsem čerpal z článku [24]. Matice pixelů *pixel cache* označených jako p1 až p9, je využita modulem konvoluce, *FIFO řádek 1* ukládá první řádek a *FIFO řádek 2* ukládá druhý řádek. Plnění pixelů do FIFO paměti se řídí řídícím vzorkovacím signálem pixelů z kamery a stejným inverzním signálem je řízen zápis. Tím je zajištěno, že po přečtení paměťové buňky je možné ji přepsat novou hodnotou.

[6] První pixel, který vstoupil, jde první ven po naplnění celého zásobníku



Obrázek 4.12: Návrh paměťové architektury řádkové paměti [24]

Celková velikost paměti je $(640 \cdot 2) + 2 = 1282$ bitů. Paměť pro jeden řádek je ve VHDL realizována jako datové pole osmibitových vektorů:

```
ARCHITECTURE rtl OF RAM_memory IS TYPE ram_type is
ARRAY (639 downto 0) of std_logic_vector ( 7 downto 0);
```

4.4.1.2 Čítač řádků a sloupců

Čítač řádků udává souřadnici pixelu ve směru y, čítač sloupců souřadnici ve směru x. Souřadnice sledují polohu masky v obraze a podle nich lze určit, jestli je maska na okraji aktivní plochy a tento stav ošetřit. V rozích je třeba doplnit masku pěti hodnotami pixelů, na okrajích stačí tři hodnoty. Čítač pro řádky je řízen vzestupnou hranou horizontální synchronizace a nulován sestupnou hranou vertikální synchronizace, to znamená nulován po ukončení snímku . Analogicky je čítač pro sloupce řízen vzestupnou hranou vzorkovacího signálu za podmínky, že je aktivní řádek. Nulován je sestupnou hranou signálu horizontální synchronizace, kdy je řádek ukončen.

4.4.1.3 Zpoždění synchronizace

Modul zajišťuje správné časování synchronizačních signálů. Ty musí být zpožděny o dobu naplnění FIFO řádkové paměti a jsou aktivní v momentě, kdy je uložen jeden řádek a dva pixely. Podmínka spuštění synchronizačních signálů je vyřešena v modulu pomocí čítače řádků a čítače pixelů.

4.4.2 Konvoluce

Modul **konvoluce** zpracovává matici pixelů v okolí 3x3 a provádí konvoluci s jádrem – gradientním operátorem. Operátory jsem vybral Sobelův, Prewittové a Robinsonův a jejich masky jsou uvedené ve třetí kapitole.

Pro příklad je uveden Sobelův filtr v kombinaci s vyhlazením průměrováním (filtrací) pro horizontální a vertikální směr:

$$h_x = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & 0 \\ -\frac{1}{4} & -\frac{1}{2} & -\frac{1}{4} \end{pmatrix}, h_y = \begin{pmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & 0 & \frac{1}{4} \end{pmatrix} \tag{4.2}$$

Konvolucí s obrazovou maticí je získán gradient G_x a G_y, velikost je součet absolutních hodnot obou směrů:

$$Df = |G_x| + |G_y| \tag{4.3}$$

Konvoluce s prahováním je realizována v kódu VHDL následujícím kódem:

```
sumaX:=
("0000000" & p3)+("0000" & p6 & '0')+  --matice pro smer X
("0000000" & p9)-("0000000" & p1)-
("0000" & p4 & '0')-("0000000" & p7);
sumaY:=
("0000000" & p7)+("0000" & p8 & '0')+  --matice pro smer Y
("0000000" & p9)-("0000000" & p1)-
("0000" & p2 & '0')-("0000000" & p3);

if sumaY(10)='1' then -- je zaporne?
suma1:= not sumax+1;
else
suma1:= sumaX;
end if;
if sumaY(10)='1' then  -- je zaporne?
suma2:= not sumay+1;
else
suma2:= sumaY;
end if;
suma:=(suma1+suma2);  --vysledny gradient
```

```
if suma> treshold   then         --prahovani
pdata_o<=(others => '1'); --hrana
else
pdata_o<=(others => '0');   --pozadi
```

Násobení v matici je řešeno rotací bitů doleva, dělení je převedeno na násobení (hodnota $\frac{1}{4}$ odpovídá 0,01b). Na nejvyšším bitu MSB ve výsledné sumě X a Y je otestováno, zda není hodnota pixelu po konvoluci záporná. Pokud ano, pak je dvojkovým doplňkem převedena na kladnou hodnotu pomocí negace a přičtením jedničky. Pro další filtry je výpočet analogický.

Po provedení konvoluce a získání gradientu funkce je důležité rozhodnout, zda pixel je nebo není hrana. To je provedeno na základě porovnání s prahovou hodnotou, která lze nastavit v rozmezí 0 až 256. Nastavení hodnoty prahu lze pomocí osmi spínačů na vývojovém kitu, na které jsou jednotlivé bity proměnné *tresh* namapovány. Je-li hodnota větší než zvolený práh, je přiřazena pixelu hodnota 255 odpovídající bílé barvě, v opačném případě je nechána barva pozadí. Po prahování je hodnota pixelu poslána na rozhraní VGA.

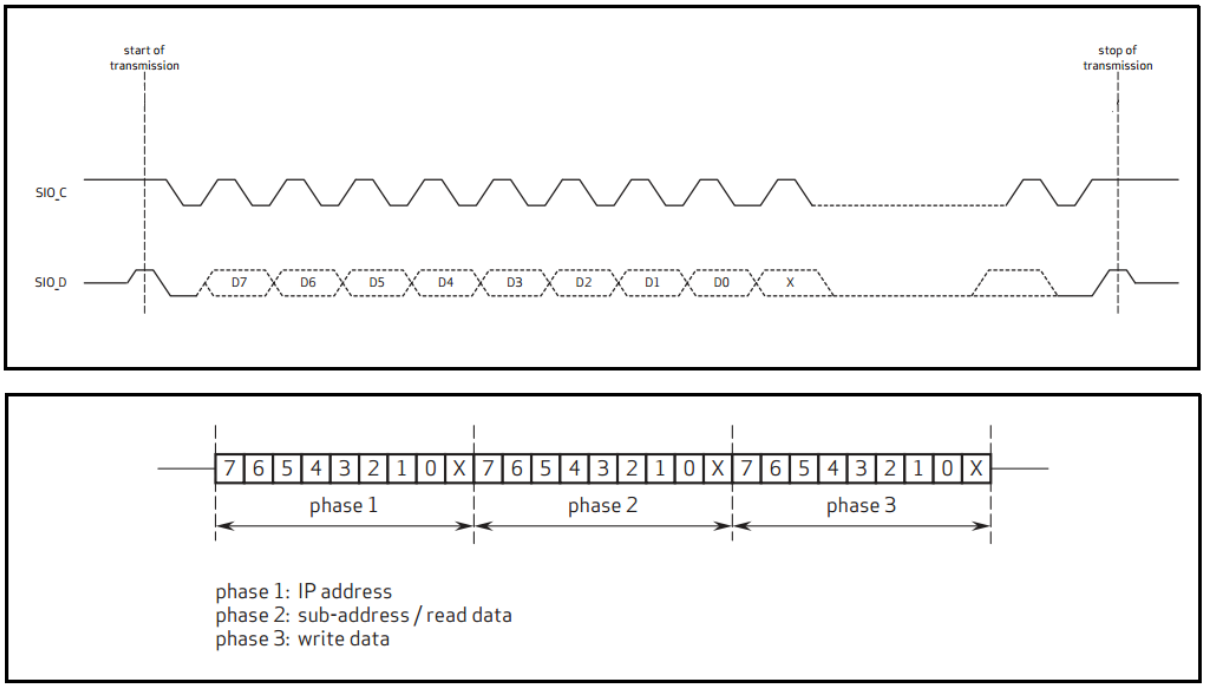
4.4.3 Konfigurace kamery

Kamera nemá žádnou konfigurační paměť a po odpojení od napájení se její nastavení ztratí. Proto je nutné při každém zapnutí kameru znovu nastavit zápisem do registrů kamery. Podrobné nastavení je k dispozici v aplikační příručce od výrobce[25]. Pro konfiguraci má kamera dvoudrátové komunikační rozhraní s protokolem SCCB – Serial Camera Contorl Bus. Jeden vodič je řídící hodinový signál pro data (SIOC) a druhý vodič je datový pro obousměrnou komunikaci (SIOD). Tento signál musí mít nastavenou klidovou úroveň pullup rezistorem na výstupním portu FPGA. Propojení je realizováno s definovanými piny obvodu FPGA, obrázek 4.13. Další dva signály slouží pro restart kamery a pro nastavení napájecího režimu.

Obrázek 4.13: Propojení komunikace SCCB pro konfiguraci kamery [26]

Časový průběh signálů SIOC a SIOD je na obrázku 4.14 nahoře. Signál SIOD je obousměrný – pro čtení i zápis dat do kamery. Jedna datová sekvence má celkem 9 bitů tj. jeden byte. Osm bitů jsou data a devátý znak je příznak konce sekvence. Režim komunikace je master – slave, kdy řízení dat udává master a tím je zaručen bezkolizní přenos dat. Před zahájením komunikace jsou oba signály SIOD i SIOC v logické jedničce. Master zahájí komunikaci nastavením SIOD na logickou nulu a po definované době měřené od sestupné hrany SIOD se nastaví i SIOC na nulu. Přechodový stav odpovídá *start bitu*. Ukončení přenosu dat probíhá opačným způsobem. Tedy přechod z nuly do logické jedničky s definovaným posunem měřeným od náběžné hrany signálu SIOD a odpovídá *stop bitu*.

Průběh zápisu dat do registrů kamery je na obrázku 4.14 dole. Celý datový rámec má tři byte. Byte jsou organizována od nejvyššího bitu. První byte udává ID adresu slave (42h pro zápis dat, 43h pro čtení dat), druhý adresu vybraného registru a poslední byte jsou data pro nastavení registru.



Obrázek 4.14: Časový průběh signálů SIOD a SIOC nahoře, třífázový zápis dole [26]

Každá fáze je ukončena devátým bitem, jeho smyslem je potvrzení úspěšného přenosu rámce dat od mastera, slave připojí SIOD na logickou nulu. Signál PWDN je nastaven na logickou nulu, RESET na logickou jedničku. Inicializace registrů je nastavena na adrese 12h a musí mít hodnotu 80h. Vzorkovací kmitočet SIOC je 400kHz, další parametry protokolu dostupné v dokumentaci OmniVision [26].

4.4.3.1 Kontrolér SSCB

Modul provádí parametrizaci registrů kamery, tedy jejich inicializaci a funkčního nastavení po startu/restartu kamery. Hodnoty registrů jsou nastaveny napevno v modulu *Registry*. Modul komunikace s protokolem SCCB ve VHDL je převzatý z otevřené knihovny CodeForge [27].

4.4.3.2 Registry

Kamera má celkem 166 registrů, z nich že podstatné nastavit některé základní pro funkci, tabulka 4.3.

adresa	hodnota	Význam
12	00	COM7 Reset
11	00	CLKRC prescaler 25Mhz (50Mhz/2)
12	05	COM7 Raw Bayer Processed format
0C	00	COM3 Disable scaling
3E	00	COM14 PCLK diveded 1
40	00	COM15 Output range 0-FF , Raw Bayer Processed format
3A	04	TSLB Do not auto-reset window
8C	00	RGB 444 disable
17	14	HSTART HREF start (high 8 bits)
18	02	HSTOP HREF stop (high 8 bits)
32	A4	HREF Edge offset and low 3 bits of HSTART and HSTOP
19	03	VSTART VSYNC start (high 8 bits)
1A	7B	VSTOP VSYNC stop (high 8 bits)
03	0A	VREF VSYNC low two bits
70	3A	SCALING_XSC
71	35	SCALING_YSC
72	00	SCALING_DCWCTR
73	00	SCALING_PCLK_DIV Clock pixel divider
A2	00	SCALING_PCLK_DELAY PCLK scaling
15	42	COM 10 set HREF, VSYNC negative

Tabulka 4.3: Nastavení registrů kamery OV7670 [25]

4.5 Shrnutí realizace

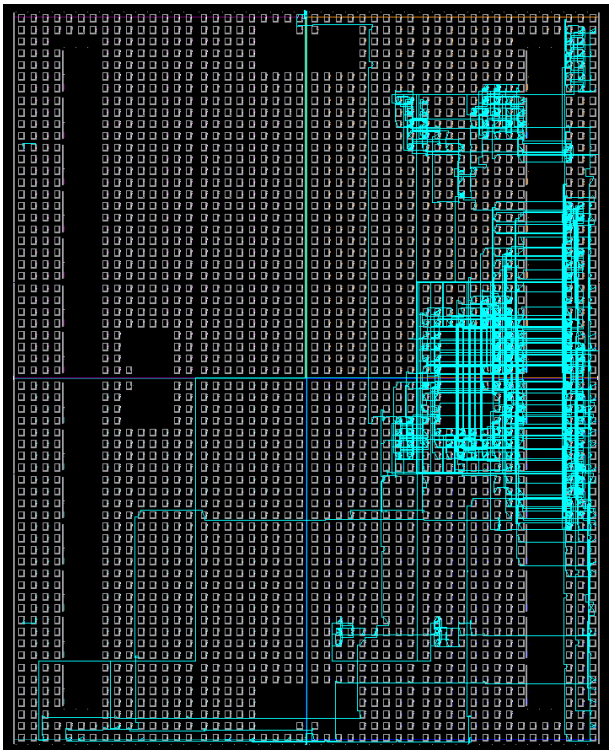
Všechny moduly v jazyce VHDL byly po syntéze odladěny a verifikovány v simulačním prostředí ISim. V tabulce 4.4 je shrnuto celkové vyžití jednotlivých prvků v obvodu FPGA, které se zobrazuje v reportu generovaným vývojovým prostředím po úspěšném překladu.

Prvek	Použito	K dispozici	Využití
Piny FPGA	43	250	17,2%
LUTs4	526	17 344	3,0%
BRAM	3	28	10,7%
Multiplexer	4	24	16,7%

Tabulka 4.4: Konečný výpis z vývojového prostředí o využití obvodu

Z tabulky je patrné, že obvod má po implementaci algoritmu dostatečné rezervy, což byl i záměr při výběru obvodu a kitu pro možnost dalšího využití.

Na obrázku 4.15 je vidět návrh tak, jak je zobrazen v programu *Xilinx FPG Editor*, který je součástí vývojového prostředí a odpovídá skutečnému propojení hradel.



Obrázek 4.15: Implementovaný návrh v obvodu Xilinx Spartan III XC3S1200E

Celkové náklady na realizaci jsou rozepsány v tabulce 4.5 podle komponent.

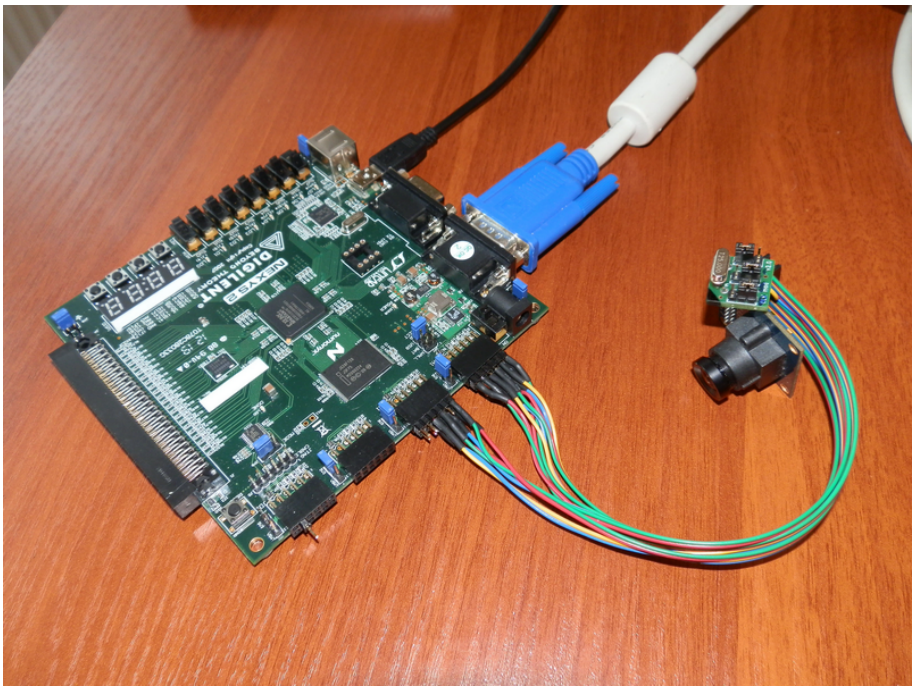
Položka	cena (Kč)
Nexys2 kit FPGA	4400
OV7670 kamerový modul	400
OV7620 kamerový modul	1500
Krystalový oscilátor 10-50 MHz	170
Propojovací vodiče	120
celkem	6590

Tabulka 4.5: Cena komponent pro realizaci

Kapitola 5

Ověření algoritmu na reálných datech

Celý přípravek je zapojený do jednoho celku propojovacími vodiči, obrázek 5.

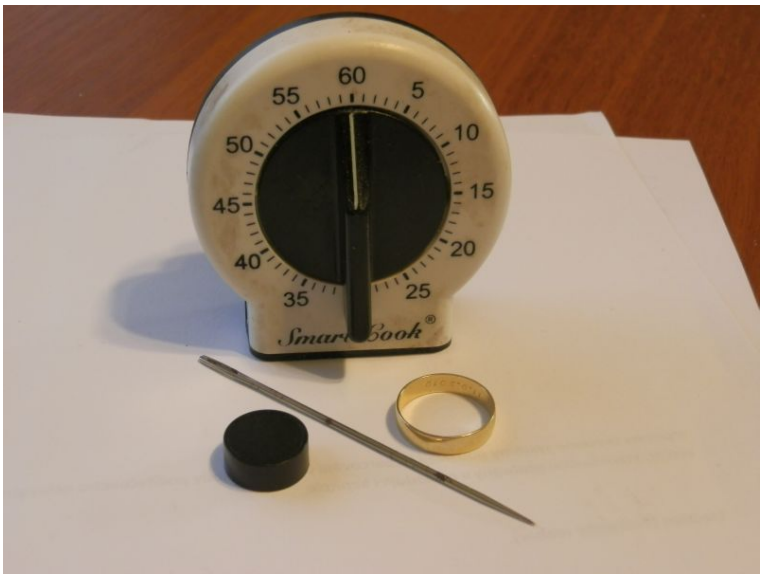


Obrázek 5.1: Kompletní sestava a propojení kamery s vývojovým kitem

Kamera je s kitem propojena vodiči průřezu 0,15 mm^2. První zapojení jsem provedl s plochým vodičem o průřezu 0,08 mm^2, vzorkovací a synchronizační signály byly na osciloskopu silně zašuměné. Pro konkrétní snímaný obraz se musí objektiv kamery ručně doostřit a lze snímat objekty od několika centimetrů. Modul kamery má doplněný externí

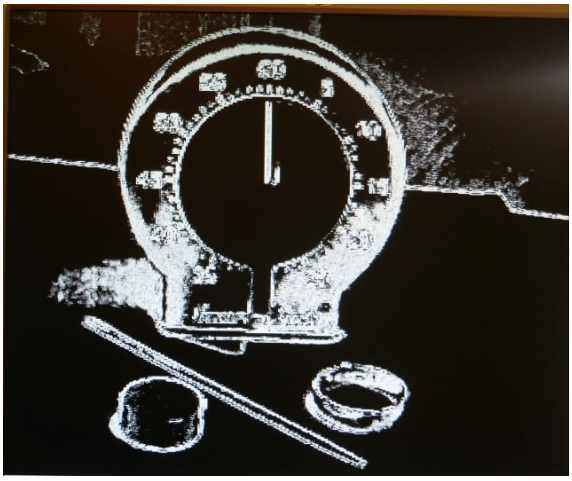
krystalový oscilátor 50 MHz napájený přímo na piny hodinového signálu. Důvodem je zamezení šumu a rušení. Napájení kitu Nexys2 je ze síťového zdroje 5 V a bez problémů pracuje i s napájením přes datové rozhraní mini USB z PC. Kit má vlastní DC-DC měnič s napětím 3,3 V a to je využito i pro napájení kamery. Monitor je připojen standardním VGA kabelem. Monitor má analogový vstup VGA a formát 4:3.

Pro ověření algoritmu jsem snímal běžné předměty na fotografii 5.2.



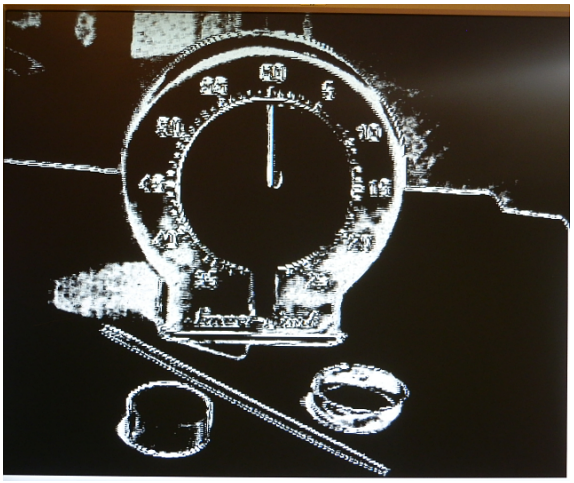
Obrázek 5.2: Originální obraz

Obrázky zpracované pomocí Sobelova, Prewittové a Robinsonova operátoru jsou na fotografiích 5.3, 5.4, 5.5.

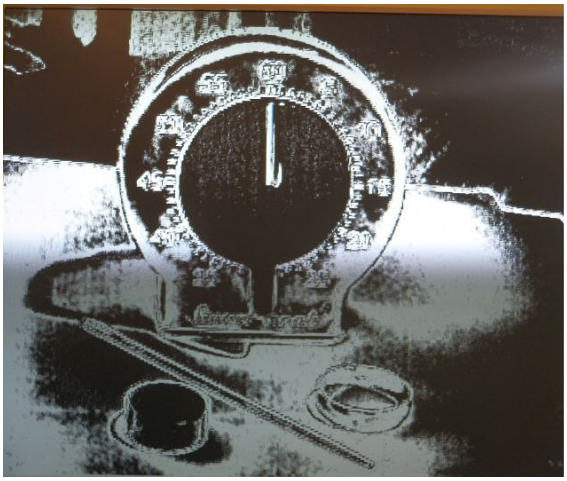


Obrázek 5.3: Hranová detekce operátorem Sobelovým, práh 128

Protože operátory (matice) jsou součástí zdrojového kódu, musel jsem pro každý z nich provést překlad a konfiguraci FPGA. Výhodným zlepšením pro rychlé porovnání detekce



Obrázek 5.4: Hranová detekce operátorem Prewittové, práh 128



Obrázek 5.5: Hranová detekce operátorem Robinsonovým, práh 255

jednotlivými operátory, by bylo přepínání provádět softwarově například na základě stisku tlačítka. Přepínání jsem nerealizoval a je námětem pro další vývoj.

Z porovnání zpracování stejné scény různými filtry vyplývá:

- Sobelův operátor detekuje spolehlivě kontrastní hrany.
- Prewittové operátor detekuje spolehlivě kontrastní hrany a je méně citlivý na šum než Sobelův. Při stejném prahu 128 je výsledek detekce hran prakticky stejný, jako pro Sobelův operátor.
- Robinsonův operátor je i po nastavení maximální prahové hodnoty 255 zašuměný, detekuje i detailní hrany. Pro danou scénu by musel mít nastavenou větší prahovou hodnotu (přímo v kódu programu).

5.1 Shrnutí

Hodnocení je subjektivní a nelze rozhodnout, který z filtrů dosahuje nejlepších výsledků, protože ty se budou lišit podle snímané scény, osvětlení, kontrastu a nastaveném prahu. Nepozoroval jsem zpoždění snímků s obrazovou frekvencí 62 Hz ani při dynamickém snímaní objektů – pohybu kamery. Celý systém odebírá konstantní proud 118 mA a pro přenosný systém napájený z baterie je vhodný.

Další fotografie ukazující detekci hran různých běžných předmětů jsou v příloze [B].

Kapitola 6

Závěr

Práce se zabývala problematikou umělého zraku s přehledem existujících systémů, které využívají kameru a následné zpracování obrazu vhodným algoritmem. Na základě toho jsem navrhl algoritmus, který je pro takovou náhradu použitelný.

Obrazová data představují obecně velký datový tok (černobílý signál standardu VGA s 25 snímky za sekundu je roven 64,4 Mbit/s). V každém snímku jsou algoritmem rozpoznány objekty obecné scény na úrovni detekce jejich hran. Detektory hran se používají i pro pohyb autonomních strojů v prostoru. Přestože člověk není stroj, šlo v této úloze o stejný cíl a tím je detekovat ze scény, kterou snímá kamera, to důležité. Proto jsem využil metod počítačového vidění používaných ve strojovém vidění a robotice. Současně s detekcí hran dojde ke značné komprimaci obrazového snímku, což je výhodou proto, aby bylo možné zpracovaná data dále využít a vhodně interpretovat nevidomému. Přes velkou komprimaci si obraz zachovává důležité informace pro orientaci a navigaci nevidomého v prostoru a může být snadno reprezentován a přiveden na danou zrakovou protézu – snímač. Překážky a objekty ve scéně jsou po zpracování představovány v podobě kontur a hran, což je pro orientaci v neznámém prostředí dostačující. Pro zpracování obrazu algoritmem v reálném čase jsem s ohledem na ohromný datový tok zvolil mikroprocesor s velkým výpočetním výkonem. Návrh jsem zrealizoval na vývojovém kitu s obvodem typu FPGA Spartan III. Popsal jsem obecný návrh algoritmu a vlastní realizaci v jazyce VHDL, což je speciálně navržený jazyk určený pro programování logických obvodů hradlových polí. Vývojové prostředí jsem s ohledem na výrobce obvodu Spartan zvolil od firmy Xilinx.

Algoritmus jsem úspěšně ověřil na reálných datech, provedl jsem srovnání detekce hran s využitím třech rozdílných operátorů – Prewittové, Sobelova a Robinsonova. Kvalita

detekce hran je závislá na snímané scéně, osvětlení a nastaveném prahu, udávající od jaké hodnoty je detekován hrana.

Pro realizaci algoritmu rozpoznávání blízké scény mi byly přínosné znalosti číslicové techniky a programování, které jsem si zdokonalil. Detekci hran objektů jsem si ověřil na běžném LCD monitoru a algoritmus je vhodný pro další využití a v oblasti výzkumu zrakových náhrad.

Literatura

[1] Horace Josh, Benedict Yong, Lindsay Kleeman. *A Real time FPGA based Vision System for a Bionic Eye* [online]. [citováno 3.3.2013]. Dostupné na https://ssl.linklings.net/conferences/acra/program/attendee_program_acra2011/includes/files/pap166.pdf

[2] Douglas B. Shire, Shawn Kelly. *Development and Implantation of a Minimally Invasive Wireless Subretinal Neurostimulator* [online]. 2009 [citováno 1.3.2013]. Dostupné na http://mimetic.ece.ucsb.edu/publications/pubs/16_jorn.pdf

[3] *Retinal Implant Project*, [online]. Dostupné na www.bostonretinalimplant.org

[4] *Second Sight*, [online]. Dostupné na http://2-sight.eu/en/argus-ii-rps-pr-en

[5] *Retina Implantat AG*, [online]. Dostupné na http://retina-implant.de/default.aspx

[6] Loučka O. *Snímání vizuální informace za účelem zpracování v biomedicíně*, 2009. 42 s. Bakalářská práce na ČZU. Vedoucí práce Jaromír Volf.

[7] Kreps J. *Problematika umělého zraku*, 2005. 57 s. Bakalářská práce na ČZU. Vedoucí práce Jaromír Volf.

[8] Šťastný J. *FPGA prakticky*. BEN, 2010. 200 s. ISBN 978-80-7300-261-9

[9] *Systems Design Using FPGAs* [online]. [citováno 12.2.2013]. Dostupné na http://www.ami.ac.uk/courses/ami4460_fpga/u02

[10] *Spartan-3E FPGA Family Data Sheet* [online]. [citováno 31.2.2013]. Dostupné na http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[11] *Obrazové senzory* [online]. [citováno 30.2.2013]. Dostupné na http://radio.feld.cvut.cz/courses/D37LBR/materialy.php?

[12] Pikner J. Poupa M. *Číslicové systémy a jazyk VHDL*. BEN, 2006. 345 s. ISBN 80-7300-198-5

[13] *Constraints Guide* [online]. Dostupné na http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/cgd.pdf

[14] Dostupné na http://www.xilinx.com

[15] Šťastný J. *Návrh obvodů založených na programovatelných hradlových polích* [online]. [citováno 15.3.2013]. Dostupné na http://www.automatizace.cz/article.php?a=2223

[16] Dostupné na http://www.digilent.com

[17] Dostupné na http://www.pk-design.net/HtmlCz/Index.html

[18] *Digilent Nexys2 Board reference manual* [online]. [citováno 16.3.2013]. Dostupné na http://www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf

[19] Dostupné na www.ovt.com/

[20] Hradecký D. *VGA řadič na FPGA* [online]. 2005. [citováno 16.2.2013]. Dostupné na http://amber.feld.cvut.cz/fpga/studenti/david_hradecky/vga_fpga.pdf

[21] *Preliminary Datasheed OmniVision OV7670/7671* [online]. [citováno 10.3.2013]. Dostupné na http://www.eleparts.co.kr/data/design/product_file/Board/OV7670_CMOS.pdf

[22] *Advanced Information Preliminary OV7620/OV7120* [online]. 2006. [citováno 21.3.2013]. Dostupné na http://elcodis.com/parts/6200720/ov7620.html

[23] *VGA Signal 640 x 480 @ 60 Hz Industry standard timing* [online]. [citováno 22.3.2013]. Dostupné na http://tinyvga.com/vga-timing/640x480@60Hz)

[24] *Image Filtering in FPGAs* [online]. [citováno 15.3.2013]. Dostupné na http://blog.teledynedalsa.com/2012/05/image-filtering-in-fpgas/

[25] *OV7670/7671 Implementation Guide* [online]. 2005. [citováno 1.3.2013]. Dostupné na http://www.robokits.co.in/datasheets/OV7670.pdf

[26] *OmniVision Serial Camera Control Bus (SCCB)Functional Specification* [online]. 2007. [citováno 1.3.2013]. Dostupné na http://www.ovt.com/download_document.php?type=document&DID=63

[27] Dostupné na http://www.codeforge.com/s/0/SCCB-VHDL