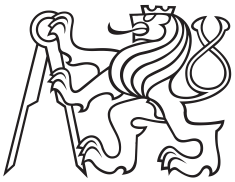
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra mikroelektroniky

Zpracování videosignálu ve světelné technice využívající systém na čipu

Bc. Martin Čurda

Vedoucí: prof. Ing. Pavel Hazdra, CSc.
Obor: Elektronika
Studijní program: Elektronika a komunikace
Květen 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Čurda** Jméno: **Martin** Osobní číslo: **456956**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Elektronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Zpracování videosignálu ve světelné technice využívající systém na čipu.

Název diplomové práce anglicky:

Video signal processing using the system on chip for stage lighting control.

Pokyny pro vypracování:

1) Seznamte se se systémy na čipu (SoC) Zynq-7000 firmy Xilinx, způsobem jejich návrhu, vývojovým systémem Vivado HLS a platformou ZYBO.
2) Prostudujte technické specifikace zařízení, které jsou využívány ve světelné jevištní technice a řízeny protokolem DMX512. Vyberte vhodnou kamery pro snímání světelných efektů.
3) Navrhněte a realizujte systém využívající SoC Zynq, který v reálném čase umožní ve snímaném obraze identifikovat vybrané světelné zařízení. Zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

[1] L. H. Crockett, R. A. Elliot, M. A. Enderwit, D. Stewart, The Zynq Book Tutorials for Zybo and ZedBoard, First Edition, Strathclyde Academic Media, Glasgow, 2015.
[2] J. Eade, The DMX 512-A Handbook, Entertainment Technology Press, Ltd., Cambridge, 2013.

Jméno a pracoviště vedoucí(ho) diplomové práce:

prof. Ing. Pavel Hazdra, CSc., katedra mikroelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **10.02.2020** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2021**

prof. Ing. Pavel Hazdra, CSc. podpis vedoucí(ho) práce	prof. Ing. Pavel Hazdra, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Poděkování

Děkuji prof. Ing. Pavlovi Hazdrovi, CSc. za pomoc při vedení diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2020

.....
Martin Čurda

Abstrakt

Cílem práce je navrhnout a realizovat funkční zařízení založené na systému na čipu (SoC) Zynq-7000 firmy Xilinx za využití vývojového prostředí Vivado HLS, jež bude možné využít při reálných aplikacích světelné jevištní techniky. Systém umožňuje v reálném čase identifikovat vybraná světelná zařízení řízená protokolem DMX512 ve snímaném obrazu.

Teoretická část práce je zaměřena na analýzu daného problému, použité platformy a vývojového prostředí. Část textu je věnována i zpracování obrazu a protokolu DMX512.

Praktická část se zabývá řešením detekce světelných zařízení v obrazu za využití Soc Zynq a popisem funkce realizovaného zařízení.

Klíčová slova: SoC, Systém na čipu, Zynq, Xilinx, Zybo, Vivado, HLS, zpracování obrazu, DMX512, pódiové osvětlení, světelné zařízení

Vedoucí: prof. Ing. Pavel Hazdra, CSc.
Katedra mikroelektroniky FEL ČVUT,
Technická 2,
Praha 6

Abstract

The aim of the thesis is to design and implement a functional device based on a system on chip (SoC) Zynq-7000 by Xilinx company while using the Vivado HLS development environment. This may be used in real applications of stage lighting technology. The system allows to identify selected lighting instruments controlled by the DMX512 protocol in the captured image in real time.

The theoretical part of the paper focuses on the analysis of the problem, the platform used and the development environment. Some part of the text is also devoted to image processing and the DMX512 protocol.

The practical part deals with the solution of detection of lighting devices in the image using Soc Zynq and it also describes the function of the implemented device.

Keywords: SoC, System on chip, Zynq, Xilinx, Zybo, Vivado, HLS, Video signal processing, DMX512, stage lighting

Title translation: Video signal processing using the system on chip for stage lighting control

Obsah

Obrázky

Tabulky

Seznam použitých zkratek a symbolů

Symbol	Význam (jednotka)
f	frekvence (Hz)
I	elektrický proud (A)
P	výkon/příkon (W)
R	elektrický odpor (Ω)
U	elektrické napětí (V)
(-)	snímková frekvence (fps)

Zkratka	Význam
ABLC	Auto Black Level Calibration
ADC	Analog to Digital Converter
AEC	Auto Exposure Control
AGC	Auto Gain Control
APU	Application Processing Unit
ASIC	Application Specific Integrated Circuit
AWB	Auto White Balance
AXI	Advanced eXtensible Interface
BRAM	Block RAM
CAN	Controller Area Network
CLB	Configurable Logic Block
CPU	Central Processing Unit
CSI	Camera Serial Interface
DAC	Digital to Analog Converter
DDR	Doble Data Rate
DMA	Direct Memory Access
DMX	Digital Multiplex
DPS	Deska Plošných Spojů
DSP	Digital Signal Processor
EMIO	Extended MIO
FF	Flip-Flop
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPIO	General-Purpose Input/Output
GPS	Global Positioning System
HDL	Hardware Description Language
HDMI	High-Definition Multimedia Interface
HLS	High-Level Synthesis
I^2C	Inter-Integrated Circuit
IOB	Input/Output Block
IP	Intellectual Property
JTAG	Joint Test Action Group

Zkratka	Význam
LED	Light Emitting Diode
LUT	Lookup Table
MAB	Mark After Break
MIO	Multiplexed Input/Output
MIPI	Mobile Industry Processor Interface
MTBF	Mark Time Between Frames
MTBP	Mark Time Between Packet
OLED	Organic LED
PAR	Parabolic Aluminized Reflector
PL	Programmable Logic
PLA	Polyactic Acid
PS	Processing System
QSXGA	Quad Super eXtended Grafics Array
QVGA	Quarter Video Grafics Array
RAM	Random Access Memory
ROM	Read-Only Memory
RTL	Register Transfer Logic
RX	Receive
SD	Secure Digital
SDK	Software Development Kit
SLA	Stage Light Analyzer
SoC	System on Chip
SPI	Serial Peripheral Interface
TX	Transmit
UART	Universal Asynchronous Receiver and Transmitter
UNC	Unified National Coarse Thread
USB	Universal Serial Bus
VHDL	Very High Speed Integrated Circuit HDL

Kapitola 1

Úvod

Pódiové osvětlení je využíváno v divadlech, operách, tanečních klubech, hudebních klubech a na hudebních festivalech a všeobecně při umělecké produkci. V dnešní době existuje mnoho světelných zařízení, která jsou určena pro tyto aplikace, jako například LED reflektory, otočné hlavice, světelné panely a další. Většina z těchto zařízení je řízena protokolem DMX512.

Cílem práce je návrh a realizace zařízení, jež v reálném čase dokáže automaticky zmapovat jednotlivá zařízení světelné scény, která jsou řízena protokolem DMX512, a umožní tak aplikaci univerzální předprogramované světelné show na různá rozmístění světelných zařízení. Pro návrh tohoto vestavěného systému je využit systém na čipu Xilinx Zynq-7000.

Práce je organizována následujícím způsobem. Na začátku teoretické části je rozebrána koncepce pódiového osvětlení a motivace k realizaci zařízení. Je zde popsána architektura realizovaného zařízení a je vysvětlen princip analýzy světelných zařízení, jenž bude aplikován v rámci této práce. Pozornost je věnována systému na čipu (SoC) Zynq-7000 od firmy Xilinx, který je základem celého zařízení, vývojové desce Digilent Zybo Z7-20 a použitým modulům Pcam 5C a Pmod RS485.

Jsou zde uvedeny výhody syntézy na systémové úrovni a proces návrhu ve vývojovém prostředí Vivado HLS. Dále je věnována část textu protokolu DMX512 a zařízením, která jsou jím ovládána. Speciální kapitola je také věnována zpracování obrazu se zaměřením na postupy, jež jsou využívány v praktické části. K nim se řadí například detekce hran, aplikace konvolučních filtrů a prahování.

Praktická část se věnuje výběru a nastavení kamerového senzoru OV5640 pro snímaní světelných efektů. Je zde popsán realizovaný IP blok pro detekci světelných zdrojů a podrobně je rozebrán proces zpracování obrazových dat v programovatelné logice systému na čipu Zynq-7000, jež jsou získána z kamerového senzoru, přes filtraci obrazu, prahování, detekci hran atd. až po předání souřadnic detekovaných hran do procesorového systému zmíněného SoC.

Dále je naznačen proces zpracování získaných souřadnic v procesorovém systému a identifikace světelných efektů na základě těchto dat. Zvláštní pozornost je věnována

jednotlivým fázím, jimiž program prochází od konfigurace procesorového systému a inicializace IP bloků přes fáze procesu detekce světelných zařízení až po vyhodnocení toho procesu a přiřazení detekovaných zařízení k předprogramované světelné show. Také je představeno uživatelské rozhraní a ovládací prvky. Na závěr práce je prezentována realizace zařízení a jsou shrnuty dosažené výsledky.

Část I

Teoretická část

Kapitola 2

Pódiové osvětlení

2.1 Koncepce

Koncepce pódiového osvětlení je různorodá a závisí jednak na typu jeviště, kde hlavní roli hraje hlavně jeho velikost, z čehož také vyplývá počet zařízení, jež je možné použít. Dále závisí na druhu zvolených přístrojů. To je ovlivněno subjektivním názorem osvětlovacího technika a jeho zvykem používat určitý typ zařízení, ale také přístroji, které má technik k dispozici. Tudíž neexistuje univerzální koncepce jevištního osvětlení.

Jinak bude řešena koncepce na jevišti v malém klubu (viz obrázek 2.1), kde mezi omezení patří hlavně prostor na pódiu, ale také finanční prostředky určené na světelná zařízení a efekty. Jinak bude koncipováno osvětlení na velkém pódiu na festivalu ve vnějších prostorách (viz obrázek 2.2). To bývá přehlídkou dostupných technologií a inovací, kde se jednotliví osvětlovači předhánějí v tom, kdo zařídí lepší světelnou show. A to rovněž kvůli tomu, aby byli angažováni i v dalších ročnících festivalu, neboť část rozpočtu, která je určena pro osvětlení pódia je rozhodně větší než v klubu.

V klubu je často používáno statické osvětlení bez pohyblivých světelných zařízení. To je právě dáno nedostatkem prostoru na jevištích tohoto druhu a snaze vyhnout se tomu, aby světelná zařízení zabírala hodně místa už v tak malém prostoru. Přeci jen

Obrázek 2.1: Klubové jeviště [1].

Obrázek 2.2: Jeviště ve vnějších prostorách [2].

přístroje typu otočná hlavice potřebují pro správnou funkci větší prostor než světelné efekty PAR nebo Striplight. Typy používaných zařízení jsou blíže popsány v kapitole 7.5.

Naopak na velkých pódiích je dostatek prostoru pro využití světelných zařízení všeho druhu, takže je možné používat i zmíněné otočné hlavice, které pohybují se světelnými paprsky a v tomto případě je možné hovořit o dynamickém osvětlení.

Na základě toho, že koncepce pódiového osvětlení a rozmístění přístrojů je pokaždé jiná, je v rámci této práce realizováno zařízení, jež dokáže zmapovat jednotlivé prvky osvětlení na pódiu a umožní tak aplikaci univerzální předprogramované světelné show na vždy jiné rozmístění světelných zařízení.

2.2 Analýza

Světelná zařízení používaná k osvětlení pódií jsou ve většině případů ovládána přes protokol DMX512. Každé z ovládaných zařízení obsazuje několik adres tohoto protokolu, kdy každá adresa slouží maximálně k jednomu úkonu (viz kapitola 7). Tato zařízení jsou ovládána z ovládacího pultu nebo počítače, jenž má adresy zařízení uloženy v paměti.

Ovládací pult nebo počítač je obsluhován osvětlovacím technikem, pomocí tlačítek, tahových potenciometrů, rotačních enkodérů a dalších ovládacích prvků, které jsou přiřazeny k různým úkonům. Také je možné mít předprogramované určité sekvence příkazů, jež jsou z ovládacího pultu odesílány za sebou. Nebo je také možnost mít předprogramovanou celou světelnou show. Nicméně takto předprogramovaná show bude fungovat pouze se stejným ovládacím pultem, se stejnými zařízeními a shodným rozmístěním.

Úkolem realizovaného zařízení je zmapovat polohu světelných zařízení na daném pódiu a přiřadit jim předprogramovanou světelnou show tak, aby nebyla vázaná na vždy stejná světelná zařízení a jejich stejné rozmístění. Tudíž, aby bylo možné předprogramovanou show využít na různých pódiích a na různých světelných zařízeních. A také, aby celý proces analýzy a přiřazení proběhl automaticky. Což vede k tomu, že nebude potřeba práce osvětlovacího technika.

Kapitola 3

Stage Light Analyzer (SLA)

Analyzátor pódiového osvětlení je název zařízení, které je realizováno v rámci této práce. Zařízení slouží k detekci světelných zařízení, která jsou ovládána protokolem DMX512. Detekcí je myšleno nejen nalezení pozice daného zařízení na jevišti, ale také určení barev světla, které dané zařízení emituje.

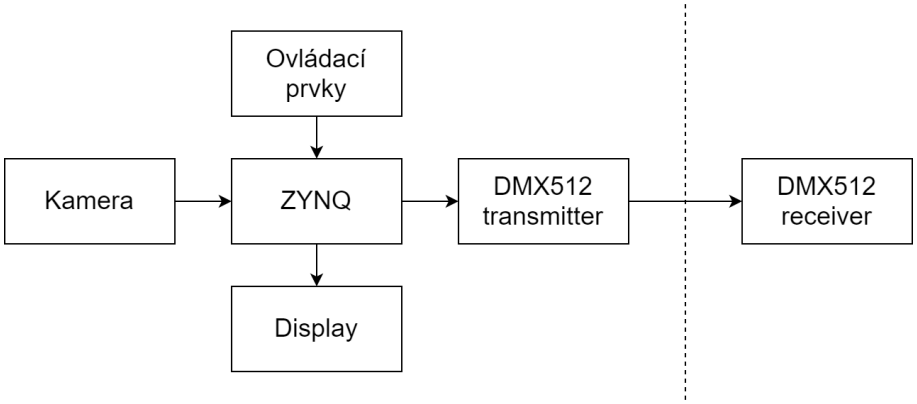
Detekce jednotlivých barevných kanálů daného světelného zařízení je rozšířena o možnost přiřazení detekovaných zdrojů k pozici na jevišti, která je následně ovládána externím zařízením, a to závislosti na předprogramovaných pokynech, jenž jsou specifické pro danou oblast v níž se zařízení nachází.

Mezi další rozšíření se také řadí mód, ve kterém je možné testovat jednotlivé adresy protokolu DMX512 mimo proces detekce světelných zařízení.

3.1 Architektura systému

Architektura systému je znázorněna na obrázku 3.1. Celý systém je možné rozdělit na následující části:

- **ZYNQ** - základem celého systému je SoC Zynq-7000 (viz kapitola 5), který je použit jako součást vývojového kitu Zybo Z7-20. Uvnitř Zynq jsou zpracována data z kamery a následně odesílána do zobrazovací jednotky.
- **Kamera** - k vývojovému kitu je připojen modul Pcam 5C s kamerovým senzorem OV5640, jenž je zmíněn v kapitole 5.6. Hlavním úkolem kamery je získat obrazová data, jež budou následně zpracována.
- **Displej** - sedmipalcový zobrazovač slouží ke snadnému nastavení pozice zařízení před jevištěm a následnému zobrazení výsledků analýzy. Displej je k vývojovému kitu připojen přes rozhraní HDMI.
- **Ovládací prvky** - čtveřice tlačítek, která slouží k ovládání zařízení, jako je například spuštění procesu detekce.



Obrázek 3.1: Architektura systému.

- **DMX512 transmitter** - slouží k odesílání dat po sběrnici DMX512, je realizován pomocí modulu Pmod RS485 (viz kapitola 5.5).
- **DMX512 receiver** - jakékoliv světelné zařízení, které je možné ovládat po sběrnici DMX512.

3.2 Princip analýzy světelných zařízení

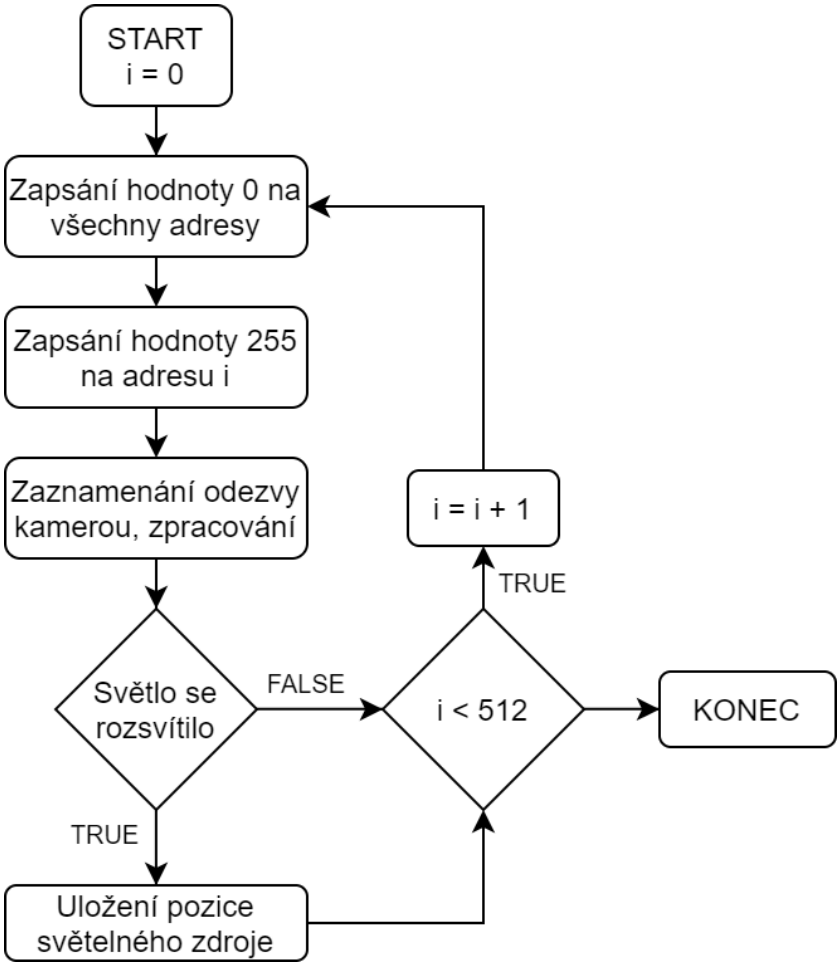
Proces detekce světelných zdrojů je znázorněn v blokovém schématu na obrázku 3.2. Před začátkem analýzy je uživatelem nastavena vhodná pozice zařízení tak, aby kamera zachytila celou oblast jeviště, kde se vyskytují světelná zařízení, jež by měla být detekována. Stiskem příslušného tlačítka je spuštěna analýza světelných zařízení.

Před samotným procesem detekce jsou zmapovány všechny světelné zdroje, které jsou zachyceny kamerou. Poté jsou nastaveny počáteční hodnoty pro detekci. V případě nevhodných okolních podmínek je analýza pozastavena, uživatel je upozorněn na okolnosti výskytu chyby. V opačném případě je spuštěn proces detekce světelných zařízení.

Na začátku procesu je proměnná `i`, jež představuje adresu, na kterou je odesílán povel, rovna nule a také všechny datové bajty jsou rovny nule, paket je vyslán na sběrnici DMX512. Tímto je zajištěno, že všechny světelné zdroje budou zhasnuty. Dále je vyslána na adresu `i` maximální hodnota a kamerou je snímána odezva na vyslaný povel.

Pokud se žádný ze světelných zdrojů nerozsvítil, uvažuje se, že se na adrese `i` nevyskytuje žádný světelný zdroj. Když bude kamerou zaznamenáno rozsvícení některého ze světelných zdrojů, je určena pozice daného světla a uložena do paměti. V dalším kroku v obou případech je zkontrolována podmínka, že proměnná `i` je menší než 512.

Poté je možno inkrementovat proměnnou `i`, smyčka je opakována od události, kdy jsou vyslány na všechny adresy nulové bajty. V opačném případě, je-li proměnná `i`



Obrázek 3.2: Blokové schéma principu detekce světelných zdrojů.

rovna 512, je dosažen maximální počet adres, proces analýzy končí.

Po ukončení procesu detekce je možné zkontrolovat detekovaná zařízení a následně je přiřadit k oblastem, které mají přednastavenou funkci.

Kapitola 4

Systém na čipu (SoC)

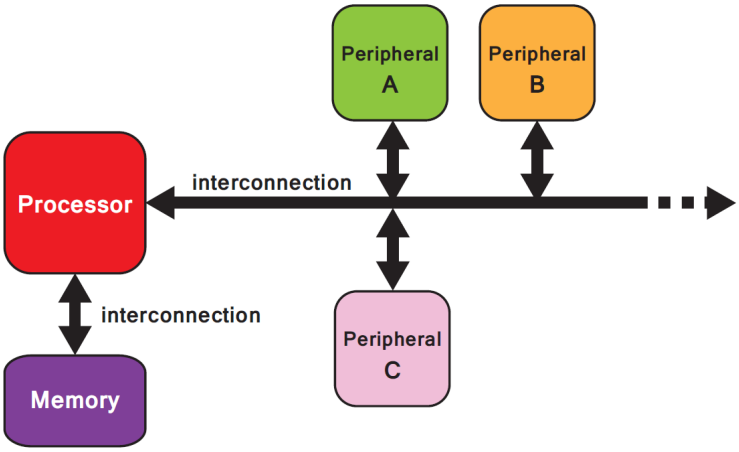
4.1 Úvod

System-on-Chip (SoC) neboli systém na čipu je založen na tom, že sdružuje funkcionalitu celého systému na jeden křemíkový čip místo využití několika různých čipů. Dříve byl termín systém na čipu používán v souvislosti s aplikačně specifickými integrovanými obvody (ASIC), které mohou obsahovat analogové, digitální a radio-frekvenční obvody společně s bloky mixovaných signálů pro implementaci analogově-digitálních a digitálně-analogových převodníků (ADC a DAC).

SoC může kombinovat všechny aspekty digitálního systému jako například: zpracování digitálních signálů, vysokorychlostní logiku, paměť atd. Na druhou stranu mohou být všechny tyto funkce realizovány jako fyzicky oddělená zařízení a spojená v jeden systém na úrovni desky plošných spojů (DPS). Řešení v podobě SoC je levnější, umožňuje rychlejší a bezpečnější přenos dat mezi jednotlivými částmi systému, nabízí větší rychlost celého systému, nižší spotřebu energie, menší fyzikální rozměry a vyšší spolehlivost.

Hlavní nevýhody systémů na čipu na bázi ASIC je doba a cena vývoje a také nedostatek flexibility. Vývoj tohoto typu SoC je vhodný pouze pro velkou sériovou výrobu, kde není potřeba budoucích vylepšení. Mezi zástupce systémů na čipu na bázi ASIC patří integrované procesory v počítačích, tabletech a smartphonech, které ve většině případů obsahují minimálně dvoujádrové procesory, paměť, grafiku, periférie a další funkce. Tyto systémy na čipu jsou vyráběny ve velkých sériích pro produkty s omezenou dobou života. Mezi omezení ASIC systémů na čipu patří nekompatibilita s velkým počtem aplikací zejména tam, kde hrají hlavní roli rychlost uvedení na trh, flexibilita a možnost vylepšení.

Pro střední série, kde není výhodné užití ASIC, je určeno programovatelné hradlové pole (FPGA). Což je flexibilní zařízení, ve kterém může být implementován jakýkoliv systém včetně procesoru. V případě potřeby je možné FPGA překonfigurovat, což nabízí větší flexibilitu proti systému na čipu na bázi ASIC.



Obrázek 4.1: Procesor se sběrnicí a periferiemi [3].

Systém na čipu, který disponuje výhodami obou uvedených řešení je například Zynq. Zynq se skládá ze dvou hlavních částí: výpočetní systém, jenž je tvořen dvoujádrovým procesorem typu ARM Cortex-A9 a programovatelnou logikou, která je shodná s tou, která tvoří FPGA. Také se zde nachází integrovaná paměť, vysokorychlostní komunikační rozhraní a různé periférie, jako například: I^2C, SPI, UART, atd [3].

4.2 Model systému na čipu

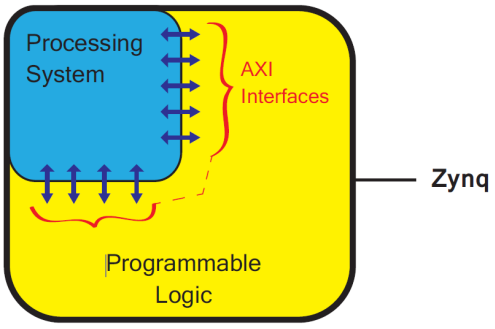
Základní model digitálního systému se skládá z procesoru, pamětí a různých periférií. Všechny tyto dílčí části jsou propojeny sběrnicí a společně tvoří hardwarovou část systému. Procesor může být považován za základní stavební prvek hardwaru celého SoC. Softwarová část je zpracovávána právě v procesoru. Zahrnuje aplikace, obvykle založené na operačním systému, a poté nižší vrstvu softwarových funkcí, které obstarávají spojení s hardwarem. Komunikaci mezi jednotlivými elementy systému zajišťují jednak přímá propojením point-to-point a nebo sběrnice, což se využívá při spojení více komponentů [3]. Příklad propojení procesoru se sběrnicí a periferiemi je na obrázku 4.1.

Kapitola 5

ZYNQ - 7000

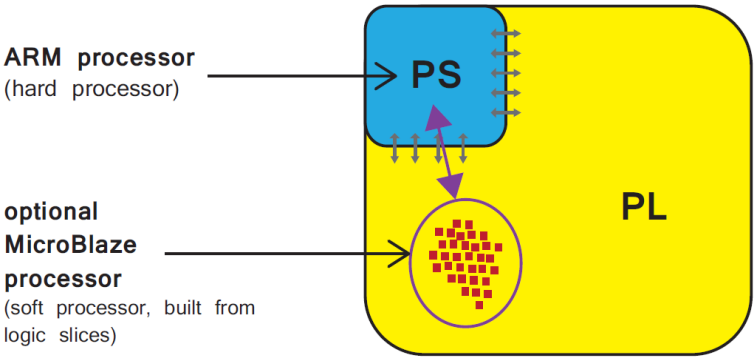
Zynq je programovatelný systém na čipu, jenž kombinuje dvoujádrový procesor ARM Cortex-A9 a programovatelné hradlové pole (FPGA). ARM Cortex-A9 je aplikační procesor schopný obsluhovat plnohodnotný operační systém, jako je například Linux. Programovatelná logika je založena na architektuře Xilinx 7-series FPGA. Architektura celého zařízení využívá standard průmyslového rozhraní AXI, které poskytuje velkou šířku pásma a nízkou latenci mezi oběma hlavními funkčními bloky. Z toho vyplývá, že procesor a programovatelná logika mohou být využity pro to, co umí nejlépe, bez potřeby řízení komunikace mezi dvěma fyzicky oddělenými zařízeními. Mezi výhody minimalizace celého systému na jeden čip patří i zmenšení fyzikálních rozměrů a snížení celkové ceny.

Sekce programovatelné logiky je vhodná pro implementaci vysokorychlostní logiky, aritmetických operací a subsystémů pro řízení toku dat, zatímco procesor obstarává například aplikace nebo operační systém, z čehož vyplývá, že funkcionalitu celého systému jakéhokoliv designu lze patřičně rozložit mezi hardware a software [3].



Obrázek 5.1: Zjednodušený model architektury Zynq [3].

Jak již bylo zmíněno, Zynq je tvořen dvěma částmi: Procesorovým systém a programovatelnou logikou viz obrázek 5.1. Tyto části mohou fungovat jak nezávisle, tak i společně, k čemuž je i přizpůsobeno napájení čipu, kdy je možné vypnout napájení právě nepoužívané části. Nicméně nejčastější způsob využití Zynq je společné využití obou částí.



Obrázek 5.2: Umístění soft a hard procesorů [3].

5.1 Procesorový systém

Zynq obsahuje dvoujádrový procesor ARM Cortex-A9. Ten je je také nazývaný jako „hard“ procesor, protože je tvořen jednoúčelovou a optimalizovanou částí křemíku v celém SoC.

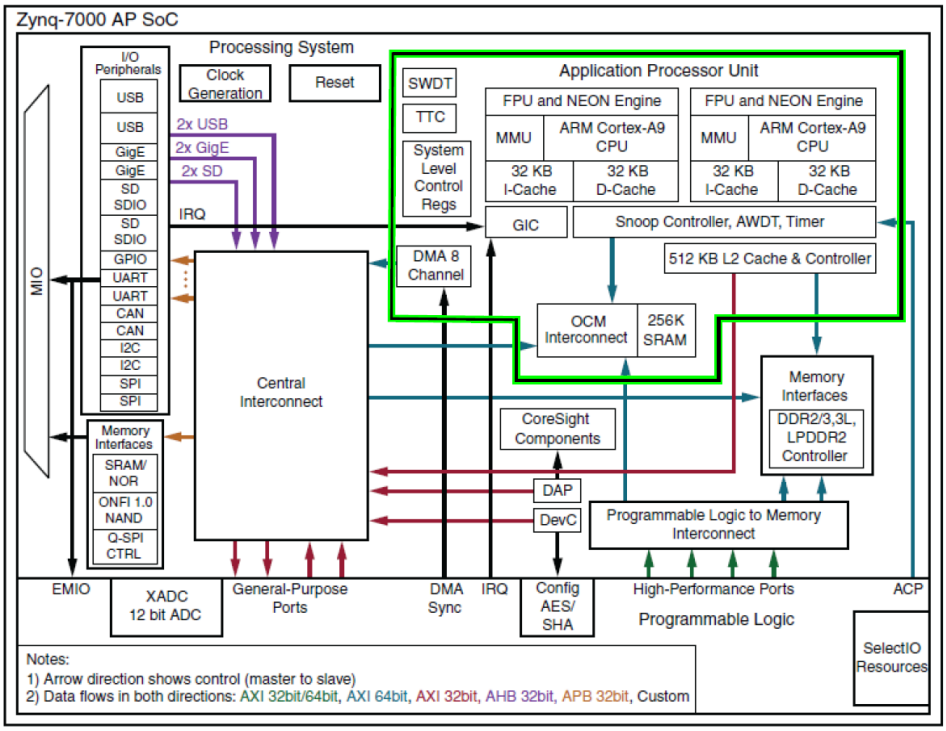
Pro porovnání je zde alternativa k „hard“ procesoru tzv. „soft“ procesor, jako například Xilinx MicroBlaze, který je tvořen kombinací elementů programovatelné logiky. Implementace soft procesoru probíhá stejně jako u kteréhokoliv IP bloku u FPGA. Výhodou soft procesoru je, že počet a preciznost implementace procesorových instancí je flexibilní. Na druhou stranu hard procesor dosahuje vyššího výkonu. Nicméně tím není vyloučeno užití soft procesoru ve funkci koprocesoru. Umístění hard a soft procesorů viz obrázek 5.2.

Procesorový systém Zynq (obrázek 5.3) neobsahuje pouze procesor typu ARM, ale také další části tvořící aplikační procesní jednotku (APU), která je tvořena jednak již zmíněnými procesorovými jádry ARM a dále například jednotkou pro správu paměti nebo jednotkou pro výpočet s pohyblivou řádovou čárkou. Dále se v procesorovém systému nachází vstupně výstupní periférie, cache paměť, obvody generace hodinového signálu a propojení s programovatelnou logikou.

Procesorový systém Zynq obsahuje různé druhy propojení, jednak s programovatelnou logikou, tak i s externími komponenty. Komunikaci procesorových periférií se vstupně výstupními piny zajišťuje multiplexer, jenž zajišťuje flexibilní propojení s 54 piny, což znamená, že piny mohou být konfigurovány tak, jak je potřeba. Dále je zde ještě 30 pinů, které jsou sdílené s programovatelnou logikou.

Vstupně-výstupní piny zahrnují standardní komunikační rozhraní a také možnost konfigurace jako univerzální vstupně-výstupní piny (GPIO). Tento mód může být použit k mnoha účelům, jako je například čtení stavu tlačítek, přepínačů nebo rozsvícení LED.

Mezi periférie, které se nacházejí v procesorovém systému Zynq se řadí [3]:



Obrázek 5.3: Procesorový systém Zynq [3].

SPI - Serial Peripheral Interface je standardní sériová komunikace, která je založena na 4 vodičovém rozhraní. Může být použita buď v master nebo slave módu.

I^{2}C - Inter-Integrated Circuit je nízkorychlostní sběrnice, která je založena na 2 vodičovém rozhraní. Podporuje master a slave mód.

CAN - Controller Area Network je sériová sběrnice dosahující rychlosti až 1Mb/s. Používá se ke komunikaci mezi prvky systému. Uplatňuje se zejména v automobilovém průmyslu.

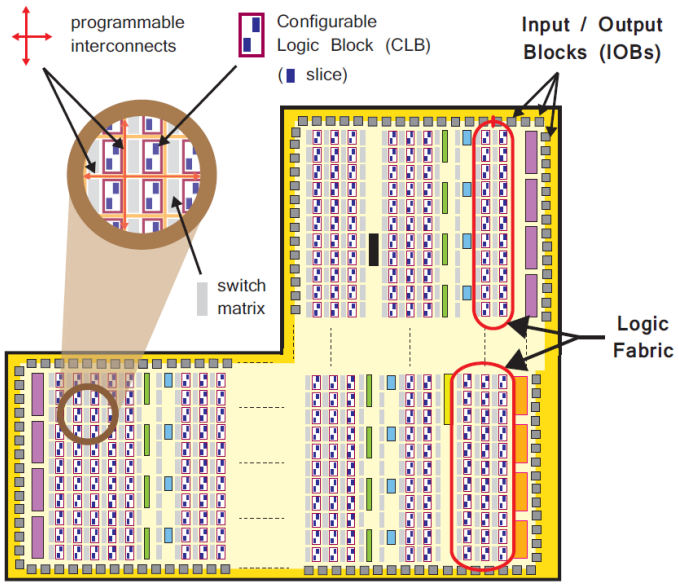
UART - Universal Asynchronous Receiver Transmitter je nízkorychlostní sběrnice, jež je často používaná pro připojení terminálu k PC.

GPIO - General Purpose Input/Output je univerzální vstup a výstup. Zynq-7000 obsahuje 4 banky po 32 bitech.

SD - Rozhraní pro paměťovou kartu typu Secure Digital.

USB - Universal Serial Bus. Zynq-7000 podporuje USB 2.0, které lze používat v módech host, zařízení nebo kombinovaně (OTG mód).

Ethernet - Ethernetové rozhraní podporuje 10 Mb/s, 100 Mb/s a 1 Gb/s.



Obrázek 5.4: Struktura programovatelné logiky [3].

5.2 Programovatelná logika

Druhou hlavní částí SoC Zynq je programovatelná logika zobrazená na obrázku 5.4, která je založena na FPGA struktuře Artix-7 a Kintex-7. Skládá se z velké části z univerzálních FPGA logických struktur, které jsou tvořeny konfigurovatelnými logickými bloky (CLB), vstupně-výstupními bloky (IOB), přepínací logikou a speciálními bloky DSP (Digital Signal Processing) a BRAM komponenty. Mezi jednotlivými bloky se nacházejí programovatelné spoje. Zdroje programovatelné logiky jsou [3]:

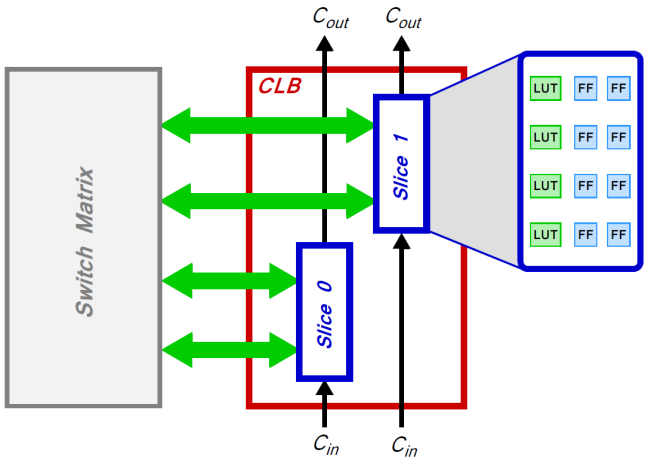
CLB - Configurable Logic Block (viz obrázek 5.5) je tvořen malými skupinkami logických elementů, které jsou uspořádané ve dvoudimenzionálním poli programovatelné logiky a jsou připojené k dalším stejným blokům pomocí programovatelných propojení. Každý CLB je umístěn vedle přepínací matice a obsahuje dva logické plátky.

Slice - (logický plátek) je stavebním prvkem CLB. Obsahuje zdroje pro kombinační a sekvenční logické obvody. Skládá se ze 4 vyhledávacích tabulek (LUT) a 8 klopných obvodů (FF) a další logiky.

LUT - Lookup Table je flexibilní prvek pro implementaci: 1. logické funkce s až 6 vstupy, 2. malé Read Only Memory (ROM), 3. malé Random Access Memory (RAM), 4. posuvného registru (Shift Register LUT - SRL)

FF - Flip-flop je bistabilní klopný obvod, který se používá k implementaci jednobitového registru.

Switch Matrix je přepínací matice, která je umístěna vedle každého CLB. Poskytuje



Obrázek 5.5: Struktura CLB [3].

flexibilní směrování pro tvorbu propojení jednak mezi jednotlivými CLB a také mezi CLB a ostatními zdroji programovatelné logiky.

IOB - Input/Output Block zajišťuje propojení mezi částmi programovatelné logiky a fyzickými piny čipu k propojení s vnějšími obvody.

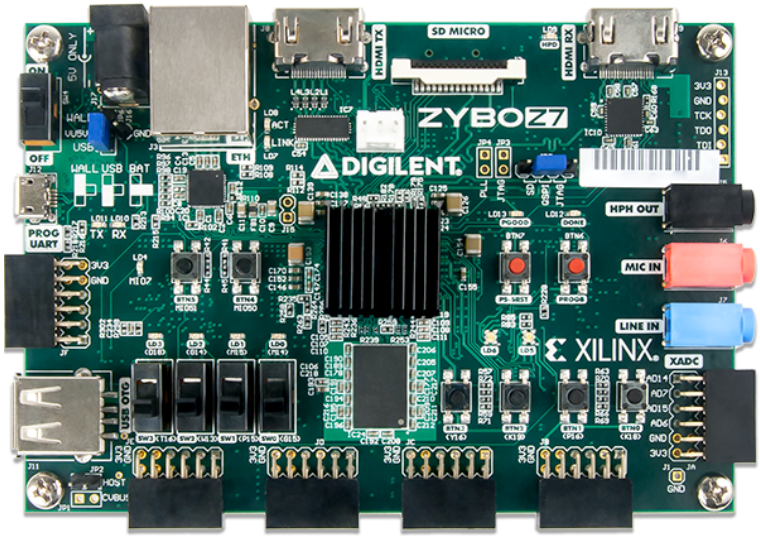
BRAM - Block RAM jsou paměti o velikosti 36Kb, které jsou rozmístěné na čipu mezi CLB. Paměť lze také použít jako 2 nezávislé bloky po 18Kb. BRAM je určena pro ukládání větších dat na čipu. Je vhodná pro implementaci RAM, ROM a FIFO pamětí. Každá obsahuje 2 porty, pro čtení nebo zápis. Například při implementaci FIFO paměti je jeden port pro čtení a druhý pro zápis. Pokud je třeba číst nebo zapisovat více prvků naráz, je nutné využít paměti LUT.

DSP48E1 slouží k implementaci rychlých aritmetických operací, které mají střední až dlouhou velikost slova. Pro krátké bitové operace se používají logické funkce v LUT. DSP obsahuje pre-adder/subtractor, násobičku a post-adder/subtractor. Postadder/substractor obsahuje rovněž logické funkce (NOT, AND, OR, NAND, NOR, XOR, a XNOR). Výsledek je na 48 bitech. DSP může být taktovaný na maximální frekvenci zařízení při malých nárocích na příkon [4].

5.3 Komunikační rozhraní PL-PS

O zprostředkování komunikace mezi procesorovým systémem a programovatelnou logikou se starají propojení dle standardu AXI. AXI (Advanced eXtensible Interface) neboli pokročilé rozšířitelné rozhraní je část standardu ARM AMBA 3.0. V současnosti existuje verze AXI4. AXI propojení jsou hlavním komunikačním rozhraním a dělí se na:

AXI4 - využívané pro paměťově mapované spoje. Zajišťuje nejvyšší výkon: adresa je následována dávkovým přenosem dat s až 256 datovými slovy.



Obrázek 5.6: Vývojová deska Zybo Z7-20 [5].

AXI4-Lite - zjednodušené propojení podporující pouze jeden datový přenos. Je také paměťově mapované. V tomto případě je přenášena adresa a jedno datové slovo.

AXI4-Stream - slouží pro vysokorychlostní přenos dat. Podporuje dávkové přenosy s neomezenou velikostí bez adresního mechanismu. Je využíván k přímému toku dat ze zdroje do cílové destinace.

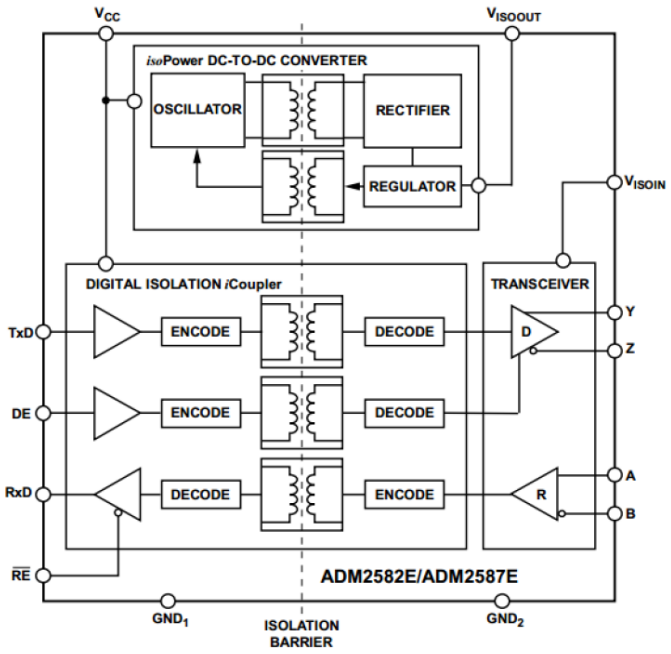
Hlavní rozhraní mezi programovatelnou logikou a procesorovým systémem je zajištěno souborem devíti AXI rozhraní, kdy každé z nich je tvořeno několika kanály.

Dalším typem propojení mezi programovatelnou logikou a procesorovým systémem je například EMIO (Extended multiplexed Input/Output). EMIO je obdobou MIO (Multiplexed Input/Output), které slouží k přímému propojení procesorového systému s vnějším rozhraním. EMIO je však vedené skrz programovatelnou logiku a může být využito k propojení PS s IP blokem nebo s některým vstupně-výstupním blokem.

Mezi další signály, které propojují PS a PL, patří například watchdog timer, reset signály, přerušení a signály přímého přístupu do paměti (DMA) [3].

5.4 Vývojová deska Zybo Z7-20

Vývojová deska Zybo Z7-20 od firmy Digilent (viz obrázek 5.6) je osazena čipem Xilinx Zynq-7020, k němuž je připojena paměť typu DDR3 o velikosti 1GB a Quad-SPI flash paměť o velikosti 16 MB. Nachází se zde různé periférie: 1G Ethernet, USB 2.0, SD karta, SPI, UART, CAN, I^2C, GPIO - programovatelné vstupně-výstupní piny, HDMI. Na desce jsou mimo jiné různé přepínače, tlačítka a LED.



Obrázek 5.7: Funkční blokové schéma modulu Pmod RS485 [6].

SoC Zynq-7020, jenž se nachází na této vývojové desce je možné programovat přes rozhraní JTAG, z Quad-SPI flash paměti a nebo z microSD karty. Deska Zybo Z7-20 je vhodná pro video aplikace, neboť se zde nachází konektor pro kamerové rozhraní MIPI CSI-2, HDMI TX a RX konektor. Pro audio aplikace je určen konektor pro stereo audio výstup, stereo line-in a mikrofon.

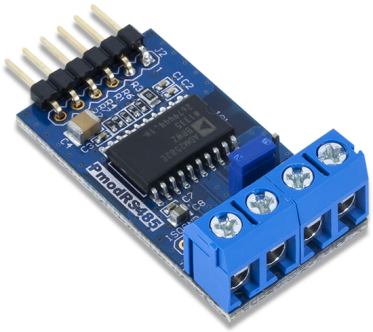
Na vývojové desce se také nachází šest Pmod konektorů, jež slouží k připojení dalších modulů, které firma Digilent nabízí. Mezi Pmod periférie například patří: Pmod RS232, Pmod RS485, Pmod OLED, Pmod GPS atd.

5.5 Pmod RS485

Pro účely této práce byl použit modul od firmy Digilent Pmod RS485, jenž umožňuje použití sběrnice RS-485. Avšak modul byl použit jako fyzická vrstva pro protokol DMX512, jenž je blíže popsán v kapitole 7.

Modul Pmod RS485 (viz obrázek 5.8) je osazen čipem ADM2582E od firmy Analog Devices, jenž převádí elektrické úrovně na výstupu Zynq-7020 na elektrické úrovně sériových komunikačních protokolů RS-485 a RS-422. Dále tento čip poskytuje galvanické oddělení obou komunikačních stran. Funkční blokové schéma čipu je znázorněno na obrázku 5.7.

Pmod RS485 je na primární straně připojen k hostitelskému zařízení pomocí šesti vodičů. Vodič *TxD* (Transmit Data) je určen k odesílání dat. *DE* (Driver Enable) je



Obrázek 5.8: Modul Pmod RS485 [6].

Obrázek 5.9: Modul Pcam 5C [7].

vstup čipu ADM2582E, kterým se povoluje odesílání dat. Vodič *RxD* (Receive Data) je využíván pro příjem dat a $\overline{RE}$ (Receive Enable) slouží k povolení příjmu dat. Zbylé dva vodiče jsou určeny pro napájení čipu.

Na sekundární straně modulu Pmod RS485 se nacházejí dva diferenciální výstupy Y a Z, jež slouží pro odesílání dat a dva diferenciální vstupy A a B, které data přijímají.

5.6 Pcam 5C

Modul Pcam 5C od firmy Digilent (viz obrázek 5.9) je připojován k vývojovému kitu přes rozhraní MIPI CSI-2. A je osazen kamerovým senzorem OV5640, který umí pracovat s rozlišením 5 megapixelů a podporuje rozlišení obrazu QSXGA (2592x1944) při snímkové frekvenci 15 fps.

Se snižujícím se rozlišením obrazu vzrůstá snímková frekvence, kdy při nejnižším podporovaném rozlišení obrazu QVGA (320x240) dosahuje snímková frekvence 120 fps. Pro běžně používaná rozlišení 1080p (1920x1080) klesá na 30 fps a při 720p (1280x720) je 60 fps.

Senzor obsahuje vnitřní zpracování obrazu pro vylepšení kvality obrazu jako je například vyvážení bílé barvy (AWB), automatické nastavení expozičního času (AEC), automatické nastavení zisku senzoru (AGC), automatické nastavení úrovně černé barvy (ABLC), detekce blikání na frekvencích 50 Hz a 60 Hz a mnoho dalších možností úpravy obrazu. Kromě výše uvedených korekcí obrazu je možné ovládat jednotlivé registry kamerového senzoru přes rozhraní I^2C a tím měnit nastavení senzoru. Například tak lze provádět otočení obrazu nebo výřez z obrazu.

Kapitola 6

High-Level Synthesis

6.1 Úvod

Jedním z hlavních trendů v návrhu digitálních systémů v současné době je zrychlování vývojových cyklů. Tato skutečnost je zapříčiněna snahou dostat nové produkty na trh v co nejkratším čase a také snížit výrobní náklady. Mezi způsoby zrychlení patří opětovné využití již hotového designu, jako je například použití IP bloků, a také zvýšení úrovně abstrakce.

Nejnižší úroveň abstrakce se zabývá spojováním jednotlivých hardwarových elementů jako jsou LUT a FF. Vyšší úrovní je RTL (Register Transfer Level), při které návrhář pracuje s operacemi mezi jednotlivými registry. Další úrovní je behaviorální popis, kdy je obvod popsán funkcí a ne operacemi mezi jednotlivými registry. Vysokoúrovňový design již nevyužívá k popisu systému HDL jazyky, ale používá jazyky určené pro popis designu na algoritmické úrovni abstrakce. Mezi tyto jazyky patří C, C++, SystemC.

S vyšší úrovní abstrakce se odkrývá méně nízkoúrovňových detailů, což vede k zjednodušení návrhu obvodu. Rozdělení funkcionality a implementace na systémové úrovni zaručuje, že zdrojový kód není fixován na architekturu. Variace pro různé architektury je specifikována direktivami v HLS (High-Level Synthesis) procesu, místo přepisování zdrojového kódu, jako tomu je u RTL úrovní designu [3]. K hlavním výhodám HLS patří:

Rychlejší návrh - Jazyk na vyšší úrovni přináší snadnější návrh a implementaci, než popis chování obvodu na úrovni RTL.

Efektivnější verifikace a validace obvodu - Obvod popsaný v jazyce C můžeme snadněji simulovat na běžném procesoru a zároveň popis ve vyšším jazyce je méně náchylný na chyby. Nástroje pro HLS umožňují psát Test Bench aplikace, které porovnávají výsledky s referenčním algoritmem, což umožní rychlou verifikaci obvodu.

Efektivnější ladění obvodu - Tyto nástroje poskytují možnost ladění na úrovni vyššího jazyka, což umožní rychlejší nalezení chyb v návrhu či implementaci.

Snadný průzkum možných mikroarchitektur obvodu - Nástroje pro HLS umožňují velmi snadno výsledný obvod upravovat podle žádoucích cílů. Je možné v krátkém čase zjistit například spotřebu zdrojů, latenci nebo propustnost obvodu při různých frekvencích, při rozbalení nebo zřetězení smyček a podobně. Průzkum jednotlivých mikroarchitektur při návrhu na úrovni RTL je o mnoho složitější a časově náročnější [4].

6.2 Metody optimalizace

Pokud je při návrhu obvodu použit zdrojový kód jazyka C, C++ nebo SystemC a nejsou určeny preprocesorové direktivy, syntézou je získán obvod, který je logicky ekvivalentní a dává správné výsledky, avšak nebudou splněny požadavky na propustnost, latenci a plochu na čipu. K optimalizaci těchto parametrů jsou používány následující principy:

- **Optimalizace propustnosti** - zřetězení (pipelining), rozbalování smyček (unrolling), rozdělení polí (arrays partitioning)
- **Optimalizace latence** - slučování smyček (merging), flattening u vnořených smyček
- **Optimalizace plochy na čipu** - použití bitových datových typů, vložení funkcí (inlining), mapování menších polí do velkého pole

Zřetězení (Pipelining) je často používanou technikou při návrhu digitálních obvodů. Používá se k dosažení vyšší propustnosti obvodu díky paralelnímu zpracovávání. Při zřetězení se daná kombinační síť rozdělí na menší bloky a s ohledem na datové závislosti se naplánují jednotlivé bloky, které lze vykonat současně. U HLS je možné zřetězení aplikovat na funkce nebo smyčky. Parametrem zřetězení je tzv. inicializační interval (označovaný II), který definuje propustnost zřetězené linky. Tedy kolik taktů hodinového signálu trvá, než je načten nový vstup do linky. Například při inicializačním intervalu 1 (II=1) je načítána každý takt nová hodnota do linky. Také každým taktem je na výstupu spočtená hodnota. Dalším parametrem je latence, která udává délku linky v počtu hodinových taktů.

Rozbalení smyček (Unrolling) je technika, kdy je rozepsáno tělo cyklu n-krát za sebe. Několik cyklů probíhá paralelně, namísto iterativního zpracovávání ve smyčce. Výhodou této techniky je zvýšení propustnosti, nevýhodou je výrazné zvýšení požadavků na plochu čipu. Úplné rozbalení smyček lze provádět pouze u cyklů, u kterých je dopředu znám počet iterací. Kompromisem mezi požadavky na zdroje a propustností může být částečné rozbalení smyčky. V tomto případě se udává počet iterací, které se rozbalí, ale zároveň bude zachován cyklus.

Rozdělení polí (Arrays Partitioning) je technika, která se používá v případě, pokud je použito pole a je pouze omezený počet portů pro čtení. Typicky se jedná o pole uložené v BRAM. Původní pole lze rozdělit na menší pole a tím dosáhnout většího počtu paralelních přístupů k hodnotám. Pole lze rozdělit několika způsoby. Původní pole se může rozdělit do menších po blocích nebo cyklicky. Další možností je kompletní rozdělení, kdy každý prvek pole je přesunut do registrů.

Slučování smyček (Merging) se používá pro sloučení těl podobných cyklů do jednoho cyklu. Při použití slučování na běžném CPU není možné dosáhnout velkého přínosu. Ovšem u HLS na FPGA má každá smyčka svůj řídící konečný automat (FSM), který bude vytvořen při sloučení pouze jednou. Tím dojde k ušetření plochy na čipu.

Flattening je technika, která se používá u smyček, které jsou vnořené uvnitř jiné smyčky. Typicky u průchodu dvourozměrnou maticí se přidávají takty režie při inicializaci a ukončení vnitřní smyčky. Tímto způsobem je možné odstranit přebytečné takty a sloučit obě smyčky do jedné, podobně jako to dělá technika slučování smyček (merging).

Vkládání funkcí (Inlining) odstraňuje hierarchii volání funkcí. Tělo volané inline funkce je vloženo do místa volání. Vložené tělo funkce pak mùže být sloučeno a optimalizováno s volající funkcí [4].

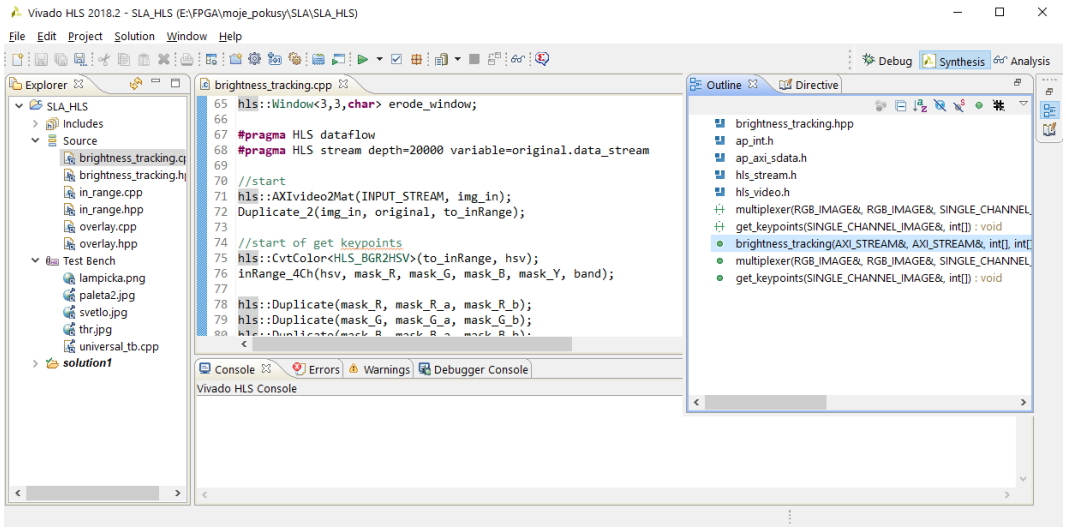
6.3 Xilinx Vivado Design Suite

Xilinx Vivado Design Suite je balík programů od firmy Xilinx pro syntézu a analýzu systémů založených na HDL (Hardware Description Language), neboli jazycích pro popis hardwaru. Součástí tohoto souboru jsou aplikace Vivado, Xilinx SDK a Vivado HLS. Vivado umožňuje návrh FPGA. Provádí se zde návrh obvodu, logická syntéza, implementace a generování bitstreamu. Xilinx SDK je určen k tvorbě kódu například pro procesorový systém zařízení Zynq. Vivado HLS je určeno k tvorbě komponentů (IP bloků) ve vysokoúrovňovém jazyku, jež jsou následně použity v aplikaci Vivado.

6.3.1 Vivado HLS

Vivado HLS převádí design popsaný jazyky C, C++ nebo SystemC do RTL implementace, která může být syntetizována nebo implementována do programovatelného hradlového pole Xilinx nebo do programovatelné logiky zařízení typu Zynq. Je třeba zdůraznit, že design na bázi C jazyků v kontextu s HLS je určen pro implementaci v programovatelné logice. Na rozdíl od softwaru, který běží v procesoru, jako je například ARM procesor v systému na čipu Zynq.

Vývojové prostředí Vivado HLS je rozděleno do tří módů, mezi kterými lze přepínat vpravo nahoře, jak ukazuje obrázek 6.1:



Obrázek 6.1: Vývojové prostředí Vivado HLS.

- **Debug** - slouží ke klasickému ladění kódu. V tomto režimu je možné sledovat proměnné, přidávat breakpointy a ladit algoritmus napsaný ve vysokoúrovňovém jazyce.
- **Synthesis** - režim je určený k optimalizování algoritmu pro vysokoúrovňovou syntézu. Do kódu je možné vkládat direktivy, spustit simulaci pomocí Test Bench souborů a hlavně se zde spouští syntéza a generování RTL popisu obvodu.
- **Analysis** - slouží k analýze výsledného řešení obvodu. Jsou zde v tabulkách rozepsané potřebné zdroje pro řešení. Také tady lze najít tabulku, kde jsou rozepsané operace do jednotlivých hodinových taktů.

6.3.2 Postup návrhu komponenty

Definice počátečních podmínek - Při založení projektu je specifikována frekvence komponenty a také čip, pro který je komponenta tvořena. Zdrojový kód se může skládat z více funkcí, avšak jedna musí být označena jako top funkce. Pro optimalizaci kódu a specifikaci chování se využívají direktivy pro syntézu na systémové úrovni.

C simulace - Kromě zdrojových souborů je možné také do projektu přidat Test Bench soubor pro testování algoritmu. Často se používá technika simulací, kdy jsou porovnávány výsledky algoritmu v jazyce C a algoritmu upraveného pro syntézu na systémové úrovni. Po provedené simulaci jsou porovnány výsledky obou algoritmů, zda upravený algoritmus odpovídá referenčnímu. Tímto způsobem lze efektivně testovat algoritmus bez nutnosti provádět syntézu.

Analýza syntézy - Po syntéze na systémové úrovni jsou shrnuty její výsledky a potřebné zdroje (viz obrázek 6.2). Také je zde uvedeno, zda byla dosažena požadovaná perioda hodin a jaká je latence a interval komponenty. Pokud není některá hodnota

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	50
FIFO	48	-	810	2751
Instance	22	3	3333	8010
Memory	-	-	-	-
Multiplexer	-	-	-	54
Register	-	-	9	-
Total	70	3	4152	10865
Available	280	220	106400	53200
Utilization (%)	25	1	3	20

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00	4.375	0.63

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
942360	942360	942350	942350	dataflow

Obrázek 6.2: Výsledek C syntézy.

v požadovaném rozsahu, je vyznačena červeně. Po úspěšně provedené syntéze lze přejít do režimu analýzy. Zde je podrobně vidět naplánované operace v jednotlivých taktech hodinového signálu [4].

Export RTL - Jakmile je dosaženo požadovaných vlastností komponenty, je třeba vygenerovat její RTL implementaci, jež může být dále využita ve vývojovém prostředí Vivado.

Integrace do systému - Ve vývojovém prostředí Vivado je proveden import vytvořené komponenty, který je následován integrací IP bloku do blokového schématu. Po propojení s ostatními bloky je provedena validace designu. Pokud je vše správně spojeno, je provedena syntéza, následně implementace a poté generování bitstreamu.

Programování SoC - Vygenerovaný bitstream je importován do prostředí Xilinx SDK, pomocí něhož je nahrán například do programovatelné logiky SoC Zynq. Pokud systém využívá i procesorový systém Zynq, Xilinx SDK slouží k tvorbě kódu, jenž je poté nahrán právě do procesoru ARM, který je součástí SoC Zynq.

Kapitola 7

DMX512

7.1 Úvod

DMX512 je standardem pro digitální komunikaci a přenos řídicích informací vyvinutý pro řízení světelné jevištní techniky a světelných efektů. Specifikace vychází z průmyslového standardu EIA485, který je znám jako RS485. Jde o digitální náhradu analogového řízení, kde základní řídicí veličinou je konkrétní hodnota napětí.

DMX512 byl původně určen pro ovládání světelných stmívačů, které bývaly ovládány nekompatibilními proprietárními protokoly. V dnešní době se používá k propojování ovládacích pultů, stmívačů, světelných zařízení a speciálních efektů, jako jsou například výrobníky mlhy. Ovládání pyrotechnických efektů je však zakázáno, a to z důvodu absence ochrany signálu prostředky, jakými jsou například časové značky, kódování nebo zpětná vazba. V případě poruchy signálu by tak mohlo dojít k náhodnému odpálení pyrotechniky [8].

7.2 Topologie

DMX512 je založen na single master, multi slave topologii s jednotlivými uzly zapojenými za sebou, kdy se zapojuje výstup zařízení DMX OUT potažmo DMX THRU do vstupu následujícího zařízení DMX IN, což se nazývá daisy chain. Sběrnice je tvořena jedním ovladačem, který je master celé sběrnice a jedním nebo více slave zařízeními. Například světelný pult ovládá stmívače, výrobníky mlhy a inteligentní světla.

Specifikace vyžaduje terminátor, který se zapojí do portu DMX OUT nebo DMX THRU posledního zařízení v řadě. Terminátor je ve své podstatě rezistor o hodnotě 120 Ω, který se připojí na diferenciální signálové vodiče. Odpor rezistoru se shoduje s charakteristickou impedancí kabelu. V některých nenáročných aplikacích, například systémech, kde se používá malý počet zařízení na krátké vzdálenosti, se užití terminačního rezistoru zanedbává.

Sběrnice DMX512 o maximálním počtu 512 adres se nazývá DMX Universe. Každý výstup DMX OUT ovladače (jeden nebo i více) může ovládat pouze jeden DMX Universe.

7.3 Fyzická vrstva

Data jsou přenášena pomocí diferenciálního páru s napěťovými úrovněmi dle specifikace EIA-485. Logické úrovně jsou reprezentovány rozdílovým napětím mezi oběma vodiči. Detekce logického stavu založená na rozdílovém napětí mezi oběma vodiči je výhodná zejména kvůli eliminaci indukovaného rušivého signálu, který se většinou přičítá k oběma vodičům stejně [8].

Přijímač rozlišuje stav log.1 při rozdílu napětí $A - B < -200\,\mathrm{mV}$. Logický stav log.0 je při rozdílu napětí $A - B > +200\,\mathrm{mV}$. Vysílač by měl na výstupu při log.1 generovat na vodiči A napětí $-2\,\mathrm{V}$, na vodiči B $+2\,\mathrm{V}$, při log.0 by měl na vodiči A generovat $+2\,\mathrm{V}$, na vodiči B $-2\,\mathrm{V}$ [9].

Z hlediska kabeláže přenos probíhá diferenciálně po stíněné kroucené dvoulince s charakteristickou impedancí $120\,\Omega$. Na konci vedení je terminační rezistor proto, aby nedocházelo k odrazům signálu.

Dle specifikace jsou pro DMX512 používány konektory typu XLR s pěti piny, kdy samice je použita pro DMX OUT a samec je určen pro DMX IN. Jiné typy XLR konektorů jsou zakázány. Přesto u levnějších zařízení bývají používány XLR konektory s třemi piny. Zákaz používání XLR konektorů s třemi vývody vyplývá z toho, že by mohlo dojít k záměně s kabely, které jsou určeny pro audio. Na těch by se mohlo vyskytovat napětí o hodnotě $48\,\mathrm{V}$, které se používá pro napájení kondenzátorových mikrofonů, což by vedlo ke zničení světelného zařízení. Dále pak kabely určené pro audio neodpovídají specifikaci DMX512 a degradují přenášený signál. Pro stabilní aplikace, kde není nutné rozpojování konektorů se používají konektory typu RJ-45 [3].

Pro krátké vzdálenosti menší než 45 metrů a při použití pouze několika zařízení se občas vynechává terminační rezistor. Také je možnost při stejných podmínkách použít kabely s vyšší kapacitou a jinou charakteristickou impedancí, jako například mikrofonní kabely.

7.4 Datový formát

Přenosová rychlost protokolu DMX512 byla stanovena na $250\,\mathrm{kBit/s}$, což udává dobu trvání jednoho bitu na $4\,\mu\mathrm{s}$. Data jsou po sběrnici posílána sériově s paketem obsahujícím 512 datových bajtů. Po sběrnici se posílají pouze data bez adresy. Každé zařízení má nastavenou svou vlastní počáteční adresu a od této adresy přečte požadovaný počet bajtů. Počáteční adresa může tedy nabývat hodnotu 0 až 511. Budou-li mít dvě stejná

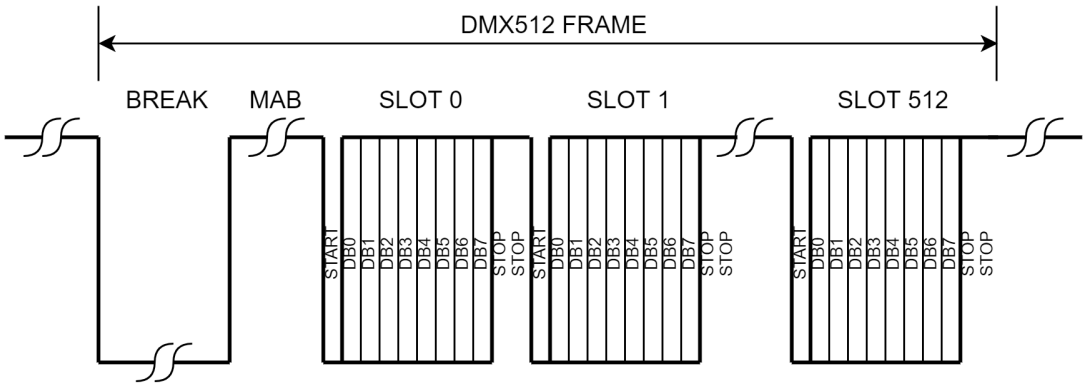
Popis	Min.	Typ.	Max.	Jednotka
Break	88	88	10^6	us
MAB	8	8	10^6	us
Rámec	43,12	44	44,48	us
Start bit	3,92	4	4,08	us
Data bit	3,92	4	4,08	us
Stop bit	3,92	4	4,08	us
MTBF	0	0	1	s
MTBP	0	0	1	s

Tabulka 7.1: Časování DMX512.

zařízení stejnou adresu, budou také na posílané povely reagovat současně. Časování v protokolu DMX512 je uvedeno na obrázku 7.1.

Klidový stav sběrnice je log.1. Přenos je realizován asynchronně a jeho začátek je synchronizován vysláním nulové úrovně Break, která musí trvat nejméně 88 μs, a následující synchronizační mezerou MAB (Mark After Break) s úrovní log.1 a minimální délkou trvání 8 μs. Dále následuje první poslaný rámec tzv. start byte, jehož hodnota je zpravidla nastavena na 0x00. V opačném případě je vysílaný rámec určen pro jinou než světelnou aplikaci. Start byte je následován zbývajícími 512 datovými rámci. Každý rámec se skládá z jednoho start bitu úrovně log.0, osmi datových bitů bez parity a dvěma stop bity log.1. Mezi jednotlivými rámci mohou být mezery MTBF (Mark Time Between Frames) a MTBP (Mark Time Between Packet) v délce nejvíce 1 s [10]

Datový rámec neboli kanál nebo také adresa, jak je nazýván v terminologii DMX512, slouží pro ovládání například červené barvy RGB LED světelného zařízení, které, jak již ze zkratky vyplývá, obsahuje červenou, zelenou a modrou barvu. Tudíž pro ovládání všech tří barev tohoto zařízení jsou potřeba 3 DMX512 adresy. Obdobně tomu je v případě zařízení typu RGBW, kdy čtvrtá barva je bílá, a tak budou potřeba k ovládání 4 DMX512 kanály. Při použití maximálního počtu RGBW zařízení, kdy každá barva má unikátní adresu může být v jednom DMX Universu až 128 zařízení.



Obrázek 7.1: Datový formát DMX512 [8]

(a) : LED PAR

(b) : Otočná hlavice

(c) : Striplight (LED Bar)

Obrázek 7.2: Světelná zařízení [11].

Pro regulaci jasu jednotlivých barevných kanálů slouží 256 úrovní, přičemž při postupném stmívání nejsou změny plynulé, což je způsobeno právě malou bitovou hloubkou o hodnotě 8. Některá zařízení proto mají bitovou hloubku zvětšenou na hodnotu 16 tak, že je tato hodnota rozdělena do dvou bytů a je vysílána dvěma kanály.

7.5 Světelná zařízení využívající DMX512

Jak již bylo zmíněno dříve, k zařízením, jež využívají protokol DMX512, patří různá světla, stmívače, speciální efekty, ovládací pulty a mnoho dalších. Ovládací pulty a speciální efekty, jejichž podmnožinou jsou například výrobníky mlhy, také využívají ke své funkci DMX512, avšak nepatří do skupiny světelných zařízení.

Jevištní světelnou techniku řízenou protokolem DMX512 lze rozdělit do několika skupin dle způsobu její funkce a konstrukce:

PAR (Parabolic Aluminized Reflector) - (viz obrázek 7.2a) je parabolický reflektor, jenž produkuje směrové paprsky světla. Zdroj světla je tvořen buď jednou lampou nebo maticí LED čipů, v tom druhém případě se před označení přidává LED. Za označením PAR bývá uveden průměr reflektoru například PAR 64, kde číslovka udává průměr v osminách palců.

Striplight - je světelné zařízení podlouhlého tvaru, v němž je po celé délce umístěno vedle sebe několik lamp. Vyzařované světelné paprsky odpovídají podlouhlému

tvaru světelného zařízení. Každou z lamp tohoto zařízení je možné ovládat jednotlivě. Využití LED u tohoto typu zařízení není výjimkou, tato zařízená jsou často nazývána LED Bar (viz obrázek 7.2c).

Otočná hlavice - je uvedena na obrázku 7.2b. Hlavní charakteristikou tohoto typu světelných zařízení je možnost řízeného pohybu celého přístroje a ovládání výstupních paprsků světla, jako je například změna jejich tvaru, zaostření nebo rozostření atd. Použitím tohoto typu světelných zařízení vzniká více dynamická světelná show. Otočné hlavice jsou nazývány také jako Inteligentní světla.

Dimmer - neboli Stmívač je zařízení, které slouží k regulaci světelné intenzity lamp, jež nemohou být jinak ovládány protokolem DMX512.

Všechny tyto typy světelných zařízení nabízí možnost ovládání jejich jednotlivých funkcí protokolem DMX512, od světelné intenzity jednotlivých barevných složek: červené, zelené, modré, bílé, jantarové. Přes ovládání všech barevných složek jednou adresou, úpravě tvaru světelného paprsku, až po rotaci světelného zařízení ve více osách.

Každá z těchto funkcí je přiřazena k jedné adrese protokolu DMX512. Zařízení má nastavenou svou vlastní počáteční adresu a od té čte další adresy v řadě v závislosti na počtu funkcí, jež může zařízení vykonávat.

Práce se zabývá detekcí světelných zařízení typu PAR a Striplight v obrazu. Použité postupy je možné využít i k detekci ostatních typů zařízení, avšak zde není zaručeno dosažení správných výsledků procesu. To je způsobeno zvoleným způsobem identifikace světelných zařízení, jenž je vhodný pro statické osvětlení, nikoliv však pro detekci přístrojů, které umožňují pohyb (Otočné hlavice).

Kapitola 8

Zpracování obrazu

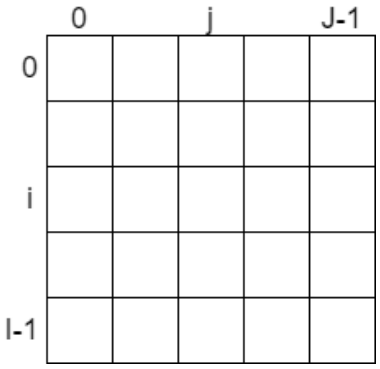
8.1 Lenna

Obrázek 8.1: Lenna [12].

Lenna (viz 8.1) je standardní testovací obrázek, který se používá pro demonstraci algoritmů a srovnání výsledků ve zpracování obrazu od roku 1973. Je to fotografie švédské modelky Lenny Sjööblom, která se objevila v roce 1972 v magazínu Playboy. Používaný výřez fotografie má rozměry 512 x 512 bodů a zachycuje pouze hlavu a ramena modelky.

8.2 Digitální obraz

Digitální obraz je možné získat snímáním kamerou nebo procesem digitalizace spojité dvojrozměrné obrazové informace do dvojrozměrné obrazové matice. Digitální obraz lze popsat jako diskrétní dvojrozměrnou obrazovou funkci $f(x, y)$, která představuje rozložení určité fotometrické veličiny (např. jasu) po ploše, a u které jak funkční hodnoty f, tak souřadnice x, y mohou nabývat diskrétních hodnot.



Obrázek 8.2: Pravoúhlá obrazová matice.

Obrazová matice (obrázek 8.2) může mít libovolnou geometrii, avšak v praxi se používá téměř výhradně pravoúhlá matice. Jeden prvek obrazové matice se nazývá obrazový bod neboli pixel. Poloha obrazového bodu je určena řádkovým indexem `i` a sloupcovým indexem `j` [13].

8.3 Bodové operátory

Nejjednodušší operátory, které se používají při zpracování obrazové informace, se týkají transformace hodnoty jasu v příslušném obrazovém bodě bez ohledu na hodnoty jasu ostatních obrazových bodů v prostorovém nebo časovém okolí.

Nechť hodnota jasu či jiné fyzikální veličiny a_I v obrazovém bodě o souřadnicích x, y je dána výrazem

$$a_I = a_I\,(x, y)\,. \tag{8.1}$$

Bodový operátor je pak definován funkční závislostí

$$a_O\,(x, y) = f\,(x, y)\,. \tag{8.2}$$

Tato funkce $f \in \mathbb{R}^2$ a může být tedy lineární, nelineární, spojitá, nespojitá a pod. V digitálních systémech je z hlediska volby převodní charakteristiky mnohem větší volnost volby, než u analogových systémů a lze vytvářet v podstatě libovolný průběh využitím tzv. transkódovací tabulky (look-up table) [13].

8.4 Diskrétní konvoluce

Konvoluce patří k základním nástrojům práce se signály. Pro zpracování digitálního obrazu se používá diskrétní konvoluce, jelikož je zpracovávána diskrétní jasová funkce.

$$f(x, y) * h(x, y) = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} f(x - i, y - j).h(i, j), \tag{8.3}$$

,

kde $f(x, y)$ je čtvercová vstupní obrazová matice, $h(x, y)$ je konvoluční jádro.

Konvoluční filtr se uvádí v podobě impulsní odezvy **h**. Zadaná matice se označuje různými názvy, např. konvoluční jádro (kernel), konvoluční matice, konvoluční maska, filtr. Významnou součástí takto definovaného filtru je konstanta vytknutá před matici, která slouží k zachování vstupního dynamického rozsahu obrazu. Její stanovení vychází z předpokladu dynamického rozsahu vstupu (0; 1) a nejhoršího možného případu, kdy výsledek konvoluce je dán součtem všech kladných prvků matice filtru resp. všech záporných. Dimenze matice **h** se liší podle požadovaných vlastností filtru, nejčastěji se používají matice 3×3, velmi zřídka větší než 9×9 prvků.

Tvar zadaného konvolučního jádra bývá:

$$\mathbf{h}(x, y) = \frac{1}{K} \cdot \begin{bmatrix} b_{11} & b_{12} & . & b_{1L} \\ b_{21} & b_{22} & & . \\ . & & & . \\ b_{L1} & . & . & b_{LL} \end{bmatrix} = K^{-1} \cdot [b_{kl}], \tag{8.4}$$

kde K^{-1} je konstanta úpravy dynamického rozsahu,
L je dimenze jádra filtru [13].

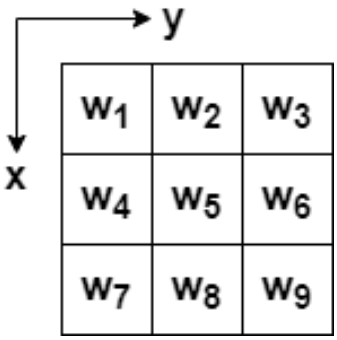
8.5 Konvoluční filtry

V praxi se nejčastěji používají již navržené a vyzkoušené banky filtrů, ze kterých se subjektivním posouzením výsledku vybírá ten nejvhodnější. V mnoha systémech nejsou uživateli vůbec dostupné hodnoty prvků matice 8.4 a pro volbu filtru jsou uvedeny pouze výsledky operací, např. "vyhlazení", "vylepšení" atd. Mezi nejčastěji používané filtry patří frekvenční filtry s charakteristikou horní, dolní nebo pásmové propusti.

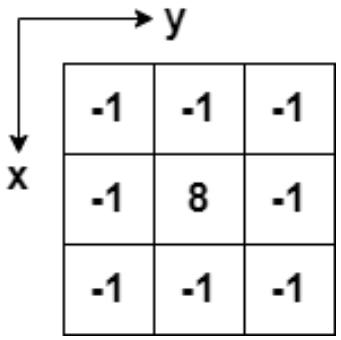
Dolní propust je filtr odstraňující z obrazu složky vyšších prostorových frekvencí. Takový charakter mají především detaily a šum. Konvoluční jádro typu dolní propust je charakteristické tím, že obsahuje pouze nezáporné členy. Výraz 8.4 pak představuje průměr. Dolní propust lze použít například pro odstranění šumu a rušivých zkreslení. Typickou dolní propustí je filtr:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{8.5}$$

Kromě těchto základních typů filtrů existují ještě speciální konvoluční filtry navržené například pro detekci hran, výpočty směrových prvních a vyšších derivací, rozdílové filtry atd.



Obrázek 8.3: Označení prvků obecné konvoluční masky o rozměru $3x3$.



Obrázek 8.4: Obecná konvoluční maska o rozměru $3x3$ pro detekci izolovaného bodu.

Horní propust je filtr odstraňující z obrazu složky nízkých prostorových kmitočtů a původní stejnosměrnou složku. Její charakteristickou ukázkou jsou následující filtry:

$$\frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}; \quad \frac{1}{8}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}. \tag{8.6}$$

Středový prvek horní propusti je vždy kladný a kolem něj jsou symetricky rozmístěné záporné prvky. Musí platit $\sum b_{kl} = 0$, jinak filtr zavádí novou stejnosměrnou složku. Horní propust se používá zejména ke zdůrazňování jemných detailů, například v operacích obecně označovaných jako "zaostření" obrazu.

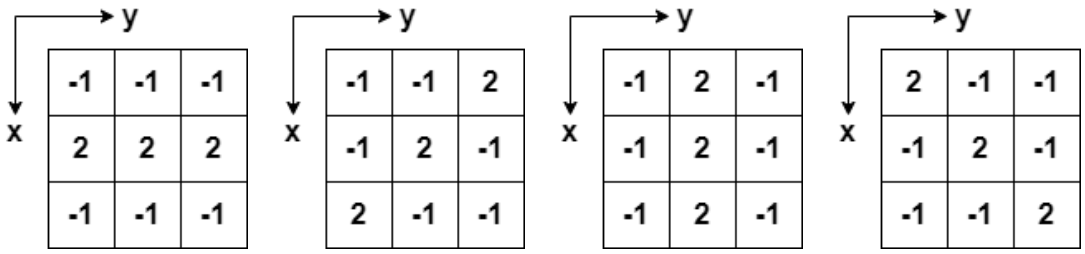
Pásmová propust propouští pouze vybrané složky prostorových frekvencí, zpravidla uprostřed frekvenčního rozsahu nebo jen v určitých oblastech spektra. Je často používána k rekonstrukci obrazu, např. k odstranění rušivých vertikálních interferencí vzniklých snímáním. Za pásmové propusti lze také považovat hranové detektory [13].

8.6 Detekce nespojitostí v obrazu

Nejpoužívanějším způsobem pro detekci bodů, přímek a hran v obrazu je využití konvoluce s vhodnou maskou v prostorové oblasti. Pro obecnou masku (viz obrázek 8.3) o rozměru 3×3 s koeficienty z_i umožňuje tento postup vypočítat součet součinů koeficientů masky s příslušnou hodnotou obrazového bodu v oblasti omezené touto maskou. Odezva masky v každém bodě obrazu je

$$R = \omega_1 z_1 + \omega_2 z_2 + \cdots + \omega_9 z_9 = \sum_{i=1}^{9} \omega_i z_i, \tag{8.7}$$

kde z_i je úroveň šedi obrazového bodu korespondujícího s koeficienty masky ω_i. Odezva masky je definována ke středu masky.



(a) : maska pro vodorovné přímky

(b) : maska pro přímky pod +45°

(c) : maska pro svislé přímky

(d) : maska pro přímky pod -45°

Obrázek 8.5: Konvoluční masky $3x3$ pro detekci přímek

8.6.1 Detekce bodů

Při detekci izolovaného bodu za použití masky na obrázku 8.4 bude bod detekován v pozici, na které je maska centrována, pokud

$$\mathsf{R} > T, \tag{8.8}$$

kde T je nezáporná prahová hodnota, tzv. práh a R je dáno rovnicí 8.7. V zásadě se jedná o vyjádření míry váhovaných rozdílů mezi středovým bodem a sousedními body. Základní myšlenkou je, že úroveň šedi izolovaného obrazového bodu bude dosti odlišná od úrovně šedi jeho sousedních bodů.

Maska na obrázku 8.4 je maska, která se používá jako filtr typu horní propust v prostorové oblasti, v tomto konkrétním případě je vyhrazena pouze pro detekci bodu. Znamená to, že pouze rozdíly, které jsou větší než hodnota T, mohou být spojovány s přítomností izolovaných bodů.

8.6.2 Detekce přímek

Pro detekci přímky jsou uvažovány masky $3x3$ viz obrázek 8.5. v případě masky a) bude největší odezva o šířce jednoho obrazového bodu na přímky orientované horizontálně. Při konstantním pozadí obrazu bude maximální odezva tehdy, pokud bude přímka procházet středním řádkem masky. Analogicky bude mít maska na obrázku b) největší odezvu pro přímky orientované pod úhlem 45°, maska na obrázku c) pro svislé přímky a maska d) bude mít největší odezvu pro přímky pod úhlem -45°.

8.6.3 Detekce hran

Nejpoužívanějším způsobem detekce nespojitostí je detekce hran. Hrana je hranice mezi dvěma oblastmi, kde se skokově mění hodnota jasu. Pro studium změn funkce $f : \mathbb{R}^2 \Rightarrow \mathbb{R}^2, f \in \mathbb{C}^1$, dvou proměnných se používají parciální derivace a změnu funkce udává její gradient

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}. \tag{8.9}$$

Gradient je vektorová veličina, určující směr největšího růstu funkce f (směr gradientu) a strmost tohoto růstu (velikost, modul gradientu). Pixely s velkým modulem gradientu se nazývají hrany.

Vektor gradientu má směr nejrychlejší změny f v (x, y). Pro obrazovou funkci $f(x, y)$ je velikost gradientu

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}. \tag{8.10}$$

Směr gradientu ψ je dán vztahem

$$\psi(x,y) = \arg\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right), \tag{8.11}$$

kde arg(x, y) je úhel (v radiánech) mezi souřadnou osou x a radiusvektorem k bodu (x, y) [14].

Laplaceův operátor

V některých případech je zkoumána pouze velikost gradientu (též velikost hrany) bez ohledu na její směr. Pro odhad velikosti se používá všesměrový lineární Laplaceův operátor - Laplacián ∇^2, který vychází z druhých parciálních derivací

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{8.12}$$

Digitální obraz je však diskrétní jasová funkce, tudíž jsou u Laplaceova operátoru derivace aproximovány pomocí diferencí.

Jelikož je Laplacián invariantní vůči rotaci, může být počítán konvolucí s jedinou maskou. Použitím tohoto operátoru je získána pouze velikost hrany a ne její směr. V digitálním obrazu je Laplacián aproximován diskrétní konvolucí. Dvě používaná konvoluční jádra 3×3 jsou [14].

$$\mathbf{h_1} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}; \quad \mathbf{h_2} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{8.13}$$

(a) : původní obrázek

(b) : po aplikaci Laplaceova operátoru

(c) : po aplikaci Sobelova operátoru ve směru osy x

(d) : po aplikaci Sobelova operátoru ve směru osy y

Obrázek 8.6: Aplikace gradientních operátorů s konvolučním jádrem o velikosti 3×3.

Nevýhodou Laplaceova operátoru je velká citlivost na šum, což lze vidět na obrázku 8.6b.

Operátor Prewittové

Operátor Prewittové, podobně jako Sobelův, Kirschův a Robinsonův, aproximuje první derivaci a není prostorově invariantní tak, jako Laplaceův operátor, z čehož vyplývá, že pro různá otočení jsou různé výsledky. Pro masku o velikosti 3×3 je až 8 možných natočení. Zde jsou uvedené 3 z nich [14]:

$$\mathbf{h_1} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}; \quad \mathbf{h_2} = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}; \quad \mathbf{h_3} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}. \tag{8.14}$$

Sobelův operátor

$$\mathbf{h_1} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}; \quad \mathbf{h_2} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}; \quad \mathbf{h_3} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{8.15}$$

Sobelův operátor se často používá pro detekci vodorovných a svislých hran, na což postačí masky $\mathbf{h_1}$, $\mathbf{h_3}$ [14]. Maska $\mathbf{h_1}$ je aplikována na obrázku 8.6d, maska $\mathbf{h_3}$ na obrázku 8.6c.

8.7 Prahování

Prahování patří k nejstarší a nejjednodušší metodě segmentace obrazu. Segmentace obrazu je proces extrakce, v němž jsou objekty separovány od nezajímavého pozadí. I

když má prahování svá široká omezení, co se týká nastavení a parametrů, řadí se k široce používané metodě. K jejím výhodám patří jednoduchost a tím pádem snadná implementace a časová nenáročnost.

Princip prahování a metody na něm založené spočívají v tom, že objekty a pozadí mají jinou úroveň intenzity jasu. Stačí tudíž určit tuto rozdílovou úroveň (práh) a poté každý pixel, který má menší hodnotu než zvolený práh, je určen jako pixel pozadí a všechny ostatní pixely jako pixely objektu, který má být segmentován.

Obecně je prahovaný obrázek g(x,y) definován jako [14]

$$g(x,y) = \begin{cases} 1, & f(x,y) > T \\ 0, & f(x,y) \leq T \end{cases} . \tag{8.16}$$

Výsledkem operace je binární obraz, kde pixely s menší nebo rovnou úrovní intenzity jasu než je hodnota prahu budou rovny 0, ostatní pixely budou nabývat hodnoty 1.

8.8 Matematická morfologie

Matematická morfologie je proces, kdy za použití matematických nástrojů dochází k extrakci požadovaných částí obrazu pro další zpracování. Princip matematické morfologie je založen na nelineárních operacích s obrazem za použití terminologie teorie množin. Nejčastěji se u morfologie pracuje s binárním obrazem, ale je možné zpracovávat i s obrazem ve stupních šedi.

V morfologických operacích se pracuje s obrazem X a strukturním elementem B. Strukturní element má význam jako maska u konvoluce, postupně jej přikládáme na jednotlivé pixely obrázku A. Výsledek morfologických operací je závislý na rozpoložení daných pixelů, nikoliv na jejich hodnotách [14].

Základní operace matematické morfologie:

- **Dilatace**
- **Eroze**
- **Otevření**
- **Uzavření**
- **Transformace**

8.8.1 Binární dilatace

Dilatace zvětšuje objekty v obraze a používá se často pro zaplňování děr a zálivů. Ve své podstatě dilatace skládá body dvou množin pomocí vektorového součtu. Dilatace

$X \oplus B$ je bodovou množinou všech možných vektorových součtů pro dvojice pixelů, vždy pro jeden z množiny X a jeden z množiny B [14].

$$X \oplus B = \{p \in \mathbb{E}^2 : p = x + b, x \in X \text{ and } b \in B\}, \tag{8.17}$$

kde p je bod v obrazu, $\mathbb{E}^2$ představuje binární obrazový prostor. X představuje původní obraz a B je strukturálním elementem.

Dilatace je Minkowského součet, tj. sjednocení posunutých bodových množin [15]

$$X \oplus B = \bigcup_{b \in B} X_b. \tag{8.18}$$

8.8.2 Binární eroze

Eroze je duální operace k dilataci $\oplus$. Při erozi dochází k odstranění slupky v obrazu a tudíž k jeho zmenšení. Eroze se využívá k odstranění drobných nerovností a vyhlazení obrazu.

$$X \ominus B = \{p \in \mathbb{E}^2 : p = x + b \in X \text{ for all } b \in B\}, \tag{8.19}$$

kde p je bod v obrazu, $\mathbb{E}^2$ představuje binární obrazový prostor. X představuje původní obraz a B je strukturálním elementem. Eroze skládá dvě bodové množiny s využitím rozdílů vektoru. Jak již bylo uvedeno, eroze je duální transformací k dilataci, ale není její inverzní transformací [14].

Eroze je Minkowského rozdíl, tj. průnik všech posunů obrazu X o vektory $-b \in B$ [15],

$$X \ominus B = \bigcap_{b \in B} X_{-b}. \tag{8.20}$$

Na obrázku 8.7 je porovnání binárních operací dilatace a eroze.

8.8.3 Otevření

Jedná se o kombinaci operací eroze $\ominus$ následovanou dilatací $\oplus$. Otevření obecně vyhlazuje kontury objektu, ruší úžiny a eliminuje úzké výčnělky.

$$A \circ B = (A \ominus B) \oplus B \tag{8.21}$$

Obrázek 8.7: Porovnání aplikace dilatace (uprostřed) a eroze (vpravo). Vlevo je původní obraz [14].

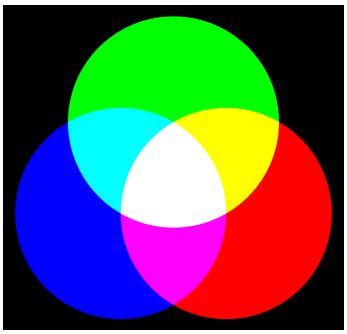
8.8.4 Uzavření

Jde o operaci dilatace $\oplus$ následovanou erozí $\oplus$. Uzavření také vyhlazuje kontury objektu, ale na rozdíl od otevření spojuje úzké mezery a dlouhé úzké zálivy, eliminuje malé díry a zaplňuje mezery v konturách [14].

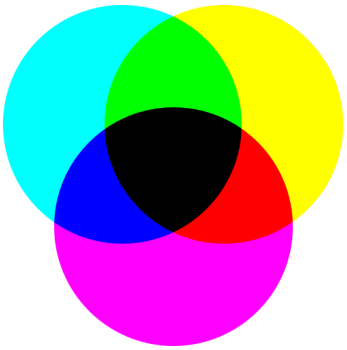
$$A \bullet B = (A \oplus B) \ominus B \tag{8.22}$$

8.9 Barevné modely

Aditivní barevný model (viz obrázek 8.8) pracuje na principu přidávání barev do černé. Čím více barev je přidáno, tím světlejší výsledek vznikne, tedy tím více se blíží bílé barvě. Aditivní barevné prostředí nepotřebuje vnější světlo, což je dáno technickými vlastnostmi monitorů, resp. použitými luminiscenčními prvky, které jsou samy zdrojem vyzařovaného barevného světla. Jeho typickým příkladem je barevný prostor RGB.



Obrázek 8.8: Aditivní barevný model RGB [16].



Obrázek 8.9: Subtraktivní barevný model [16].

Subtraktivní barevný model (viz obrázek 8.9) subtraktivní barevné prostředí odráží světlo, a proto potřebuje vnější zdroj světla. Jakákoliv předloha zobrazená na bílém papíře je příkladem subtraktivního barevného modelu. Základní barvy jsou odečítány od bílého světla. Čím více barev je z něj odečteno, tím více se výsledek blíží černé barvě. Každá barva totiž pohlcuje část světelného spektra. Bílá světlo odrazí, černá absolutně pohltí. Subtraktivní míchání barev je typické zejména pro tiskařské techniky. Mezi jeho zástupce patří barevný prostor CMYK [17].

8.9.1 RGB

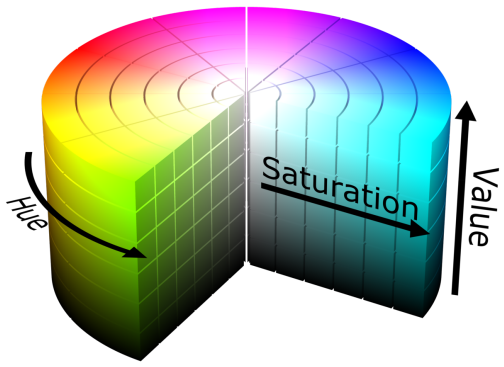
RGB (Red, Green, Blue) znázorněný na obrázku 8.8 je aditivní barevný model založený na faktu, že lidské oko je citlivé na tři barvy - červenou, zelenou a modrou. Každá barva je udána mohutností těchto tří základních barev - komponent. Základní barvy mají vlnové délky 630, 530 a 450 nm.

Mohutnost se udává buď v procentech (dekadický způsob) nebo podle použité barevné hloubky jako určitý počet bitů vyhrazených pro barevnou komponentu (pro 8 bitů na komponentu je rozsah hodnot 0 - 255), přičemž čím větší je mohutnost, tím s vyšší intenzitou se barva komponenty zobrazuje [18].

8.9.2 HSV

HSV (Hue, Saturation, Value), někdy také HSB (Hue, Saturation, Balance) je barevný model odpovídající lidskému intuitivnímu popisu barev. Má tři základní parametry: tón (odstín), sytost (saturace) a jas. Tento model se nepoužívá pro ukládání fotografií, ale má dobré uplatnění při jejich editaci. Model je znázorněn na obrázku 8.10.

Hue – odstín. Převládající barva odražená nebo procházející objektem. Měří se jako poloha na standardním barevném kole (0° až 360°). Obecně se odstín označuje názvem barvy.



Obrázek 8.10: Barevný model HSV [19].

Saturation – sytost barvy, příměs jiné barvy. Někdy též chroma, síla nebo čistota barvy, představuje množství šedi v poměru k odstínu, měří se v procentech od 0 % (šedá) do 100 % (plně sytá barva). Na barevném kole vzrůstá sytost od středu k okrajům.

Value – hodnota jasu, množství bílého světla. Relativní světlost nebo tmavost barvy. Jas vyjadřuje kolik světla barva odráží, dalo by se také říct přidávání černé do základní barvy [19].

8.9.3 Převody mezi modely

RGB ⇒ Úrovně šedi

$$Y = 0,299R + 0,587G + 0,114B \tag{8.23}$$

RGB ⇒ HSV

$$V = \max(R, G, B)$$

$$S = \begin{cases} \frac{V - \min(R,G,B)}{V} & \text{pokud } V \neq 0 \\ 0 & \text{jinak} \end{cases}$$

$$H = \begin{cases} \frac{60(G-B)}{V-\min(R,G,B)} & \text{pokud } V = R \\ 120 + \frac{60(B-R)}{V-\min(R,G,B)} & \text{pokud } V = G \\ 240 + \frac{60(R-G)}{V-\min(R,G,B)} & \text{pokud } V = B \end{cases} \tag{8.24}$$

Pokud $H < 0$, poté $H = H + 360$ [20].

Část II

Praktická část

Kapitola 9

Kamera

9.1 Úvod

Výběr kamery pro účely této práce byl omezen na dva různé způsoby jejího připojení k vývojovému kitu Zybo Z7-20. Vývojový kit je vybaven rozhraním HDMI, jenž je standardem pro mnohá zařízení, a rozhraním MIPI CSI-2 (Mobile Industry Processor Interface - Camera Serial Interface) neboli sériové rozhraní mezi kamerou a procesorem, které je využíváno u mobilních zařízení.

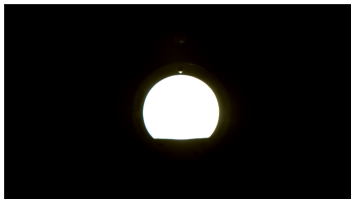
Výhodou rozhraní typu HDMI je rozšířenost tohoto standardu, což potvrzuje fakt, že výstupní konektor typu HDMI se v dnešní době vyskytuje skoro na všech digitálních kamerách, ať už na kamerách pro hobby užití, profesionálních kamerách a nebo akčních kamerách. Z čehož vyplývá, že by bylo možné použít jakoukoliv kameru z výše zmíněných ke snímání obrazu.

Nevýhodou u těchto kamer je, že obraz na HDMI výstupu z kamery může být již upraven vnitřním zpracováním obrazu, jako je například automatické vyvážení bílé barvy, automatické nastavení času expozice nebo automatické nastavení zisku kamerového senzoru. Nebo není možné vzdáleně upravovat nastavení kamerového senzoru a je nutno se spolehnout právě na automatická nastavení, která poskytuje ať už samotný kamerový senzor a nebo firmware kamery.

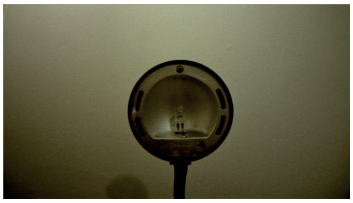
Výhodou rozhraní MIPI CSI-2 je možnost komunikace s kamerovým senzorem přes rozhraní I^2C. Přes rozhraní I^2C je možné přistupovat k jednotlivým registrům kamerového senzoru a upravovat nastavení samotného senzoru, což je velkou výhodou pro účely této práce, neboť je v průběhu jednotlivých operací možné měnit například kvalitu obrazu, rozlišení, expoziční čas, zisk senzoru a mnoho dalších nastavení.

Ze zmíněných důvodů byl vybrán kamerový senzor OV5640, který je poskytován jako součást modulu Digilent Pcam 5C, jenž nabízí Digilent jako rozšíření vývojového kitu Zybo Z7 (viz kapitola 5.6).

(a) : zapnutý světelný zdroj, automatické nastavení



(b) : zapnutý světelný zdroj, manuální nastavení kamerového senzoru



(c) : vypnutý světelný zdroj

Obrázek 9.1: Porovnání získaného obrazu při různém nastavení kamerového senzoru.

9.2 Nastavení kamerového senzoru pro detekci světelných zdrojů

Pro snímání světelných zdrojů bylo zvoleno rozlišení obrazu 720p při snímkové frekvenci 60 fps. Volba vychází z kompromisu mezi kvalitou obrazu a snímkovou frekvencí, kterou podporuje kamerový senzor. Zvolené rozlišení je dostatečné pro detekci jednotlivých světelných zdrojů. I když snímková frekvence byla nastavena na 60 fps, měřením bylo zjištěno, že reálná snímková frekvence je pouze 50 fps.

Přestože nejrychlejší změna podmíněná příkazem přes sběrnici DMX512 u světelného zdroje je omezena na maximální frekvenci sběrnice, jež je 44 Hz, je každý příkaz odesílán dvakrát na sběrnici DMX512 ve dvou za sebou jdoucích paketech. Tím lze uvažovat, že frekvence nejrychlejší změny je omezena na 22 Hz, čímž je zabráněno, aby docházelo k aliasingu při snímání světelných zařízení kamerou.

Základní nastavení kamerového senzoru, které vychází z aplikačních poznámek v katalogovém listu k zařízení, zahrnuje nastavení automatických korekcí obrazu jako jsou AGC, AEC atd. Při snímání světelného zdroje dochází k saturaci pixelů a není možné určit přesnou polohu daného světelného zdroje viz obrázek 9.1a. Na obrázku 9.1c je uveden obraz z kamery s vypnutým světelným zdrojem.

Z tohoto důvodu bylo nutné vypnout funkce automatického nastavení zisku senzoru a automatického nastavení expozičního času a nastavit tyto hodnoty manuálně. Nastavení příliš nízkého času expozice a nízkého zisku senzoru vede k zobrazení pouze nejsvětelnějších snímaných míst, přičemž slabší zdroje světla zanikají. V některých případech zanikají i požadované světelné zdroje.

Experimentálním přístupem byly zjištěny hodnoty pro nastavení registrů kamerového senzoru, které zaručují, že při snímání světelného zdroje bude docházet k saturaci pixelů maximálně v oblasti světelného zdroje. V místech, kde zdroj světla není, bude intenzita zachyceného světla minimální. Konfigurace kamerového senzoru viz obrázek 9.1b bude použita při detekci světelných zdrojů. Na obrázku lze vidět, že intenzita jednotlivých pixelů v oblastech, kde se nenachází světelný zdroj, se blíží nule.

Kapitola 10

Programovatelná logika

10.1 Úvod

Obrazový senzor je připojen k programovatelné logice SoC Zynq-7020 pomocí rozhraní MIPI CSI-2. Pro zpracování dat z obrazového senzoru a následného zobrazení bylo jako základ využito výukové demo poskytované od výrobce vývojového kitu Digilent [21]. Demo je určené pro vývojové prostředí Vivado 2018.2.

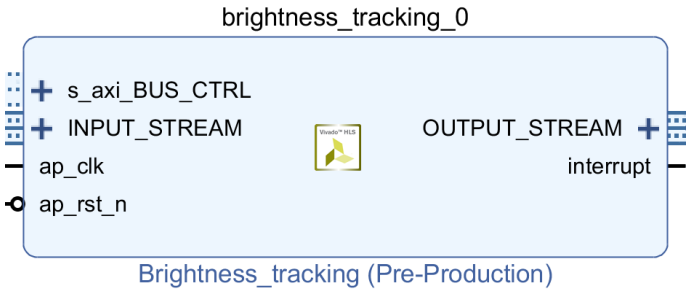
Výukové demo sestává z několika IP bloků, které zajišťují získání obrazových dat z obrazového senzoru a jejich zápis do paměti a následného zpracování pro projekci obrazu přes rozhraní HDMI. Součástí dema je mimo jiné IP blok zajišťující konfiguraci procesorového systému, jako je například frekvence procesoru, povolení funkce určitých periférií, mezi něž se řadí i I^2C, jež zde slouží pro komunikaci s kamerovým senzorem.

Demo obsahuje také firmware pro procesorový systém, který umožňuje měnit nastavení kamery a ovládat některé z IP bloků přes rozhraní AXI Lite. Část tohoto firmwaru, jenž nastavuje periférie pro komunikaci s kamerovým čipem byl ponechán a využit jako základ pro celý systém. Funkce samotného firmwaru v procesorovém systému bude rozebrána v kapitole 11.

10.2 IP blok pro detekci světelných zařízení

Blok nese název **Brightness tracking**, neboli sledování jasu. Název vychází z prvotní myšlenky návrhu celého systému, kdy je detekce světelných zařízení založena na detekci obrazových bodů, které jsou mnohem jasnější, než cokoliv jiného v okolí. Při rozšiřování možností pro detekci světelných zařízení, byla přidána také detekce pro různé barevné odstíny, tudíž název **Brightness tracking** není úplně přesný. Což také potvrzuje fakt, že IP blok slouží nejen k detekci světelných zařízení, ale také k vykreslování grafických obrazců. Nicméně název již zůstal takový, jaký byl zvolen na začátku.

IP blok pro detekci světelných zařízení, jenž byl vytvořen pomocí Vivado HLS,



Obrázek 10.1: IP blok Brightness tracking.

byl vložen do datové cesty mezi IP bloky, jež slouží ke zpracování dat z kamerového senzoru, a IP blok zajišťující zápis obrazových dat do paměti. Původní záměr byl vložit IP blok až za IP blok, který zapisuje data do paměti, nicméně v této konfiguraci nebylo z neznámého důvodu možné komunikovat s IP blokem přes rozhraní AXI Lite z procesorového systému.

Jelikož v IP bloku probíhá detekce obrazových bodů, které ohraničují dané světelné zařízení, bylo klíčové tato data získat. Řešením bylo umístit IP blok před IP blok, jenž zapisuje data do paměti. Po této úpravě již bylo možné komunikovat s IP blokem a získávat z něj potřebná data. Avšak nebylo zjištěno, proč byla původní kombinace chybná.

Další problém podobné podstaty se objevil, když byl za blok pro detekci obrazových bodů vložen další IP blok (také vytvořen pomocí Vivado HLS) s myšlenkou rozdělit celé zpracování obrazu do více IP bloků, kdy má každý blok jinou funkci. Nicméně znovu nastal problém v komunikaci s jednotlivými IP bloky. Tudíž bylo zvoleno řešení, jenž sdružuje celou část zpracování obrazu do jednoho IP bloku.

10.2.1 Struktura

Obrazová data vstupují do IP bloku (viz obrázek 10.1) portem *INPUT_STREAM* ve formě AXI4-Stream. Což je způsob přenosu dat, při kterém jsou vzorky dat posílány za sebou, čtení a zápis je sekvenční, tudíž přečtená data nemohou být přečtena podruhé. Signál TDATA je hlavním signálem pro přenos dat, kdy bitová hloubka obrazových dat je 24 bitů, což poskytuje 8 bitů pro každý RGB kanál.

AXI4-Stream protokol má také postranní kanály pro transport dodatečných dat. Mezi ně například patří: TVALID, TREADY, TUSER, TLAST. Postranní kanál TLAST udává konec přenášené posloupnosti dat. TVALID je v log.1, pokud jsou výstupní data validní. TREADY je v log.1, pokud IP blok dokončil svůj cyklus a je připravený na další. AXI4-Stream data vstupují do komponenty postupně s náběžnou hranou hodinového signálu. Uvnitř bloku jsou provedeny patřičné operace, jež budou uvedeny následně. Nakonec jsou obrazová data vyvedena z IP bloku ve formě již zmíněného protokolu AXI4-Stream portem *OUTPUT_STREAM*.

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	66
FIFO	48	-	1644	9272
Instance	78	2	32127	39141
Memory	-	-	-	-
Multiplexer	-	-	-	72
Register	-	-	12	-
Total	126	2	33783	48551
Available	280	220	106400	53200
Utilization (%)	45	~0	31	91

Obrázek 10.2: Použité zdroje pro IP blok Brightness tracking.

Celá komponenta je řízena přes rozhraní AXI LITE (port *s_axi_BUS_CTRL*) z procesorového systému, kdy řídící data slouží ke spouštění samostatného bloku, ale také jednotlivých funkčních oblastí. Přes stejné rozhraní jsou data odesílána z IP bloku do procesorového systému. Mezi tato data se také řadí informace o poloze světelných zdrojů, která však musí být ještě zpracovány.

Zdroje potřebné pro implementaci celého IP bloku jsou uvedeny na obrázku 10.2.

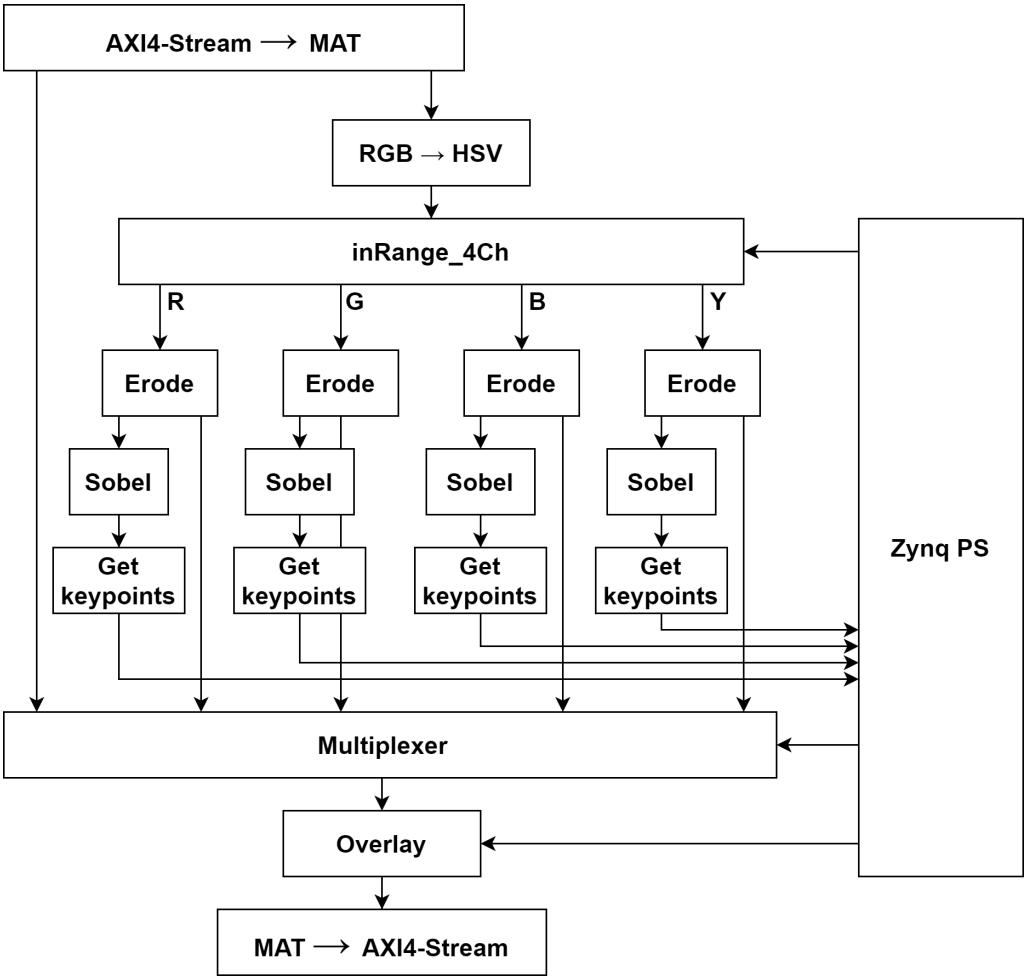
10.2.2 Tok obrazových dat

Ke zpracování obrazu byly využity funkce z knihovny, jež je součástí vývojového prostředí Vivado HLS. Vychází z knihovny OpenCV pro jazyky C++ a Python. Nicméně knihovna pro Vivado HLS obsahuje pouze část všech dostupných funkcí, které se nacházejí v původní knihovně, a to z důvodu, že není efektivní realizovat některé funkce na hardwarové úrovni nebo tyto funkce nebyly zatím vytvořeny pro Vivado HLS. Dále byly použity funkce, které byly vytvořeny pro účely této práce například pro snížení latence nebo zlepšení propustnosti. Tyto funkce jsou uvedeny v kapitole 10.3. Zjednodušené blokové schéma toku obrazových dat je znázorněno na obrázku 10.3.

Pro demonstraci bude použit obrázek světelného zdroje typu LED PAR, který vyzařuje modrou barvu. Obrázek je nasnímám kamerou OV5640, jenž je manuálně nastavena pro detekci světelných zdrojů tak, jak bylo zmíněno v kapitole 9.2. Na obrázku 10.4a je uveden obraz z kamery, který bude následně zpracován.

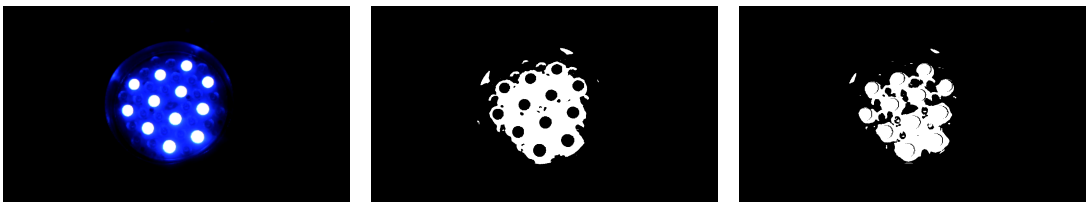
Obrazová data jsou na vstupu IP bloku převedena z AXI4-Stream na matici tří barevných kanálů RGB o rozlišení 1280×720 pixelů, přičemž jednotlivé pixely mají bitovou hloubku 8 bitů pro každý barevný kanál. Data jsou dále zpracovávána v maticové formě. Následně je tok dat rozdělen na dva stejné datové toky. Jeden slouží jako originál a bude využit až na konci procesu a druhý datový tok bude zpracováván.

Datový tok určený pro zpracování je převeden z barevného prostoru RGB (red, green, blue) do prostoru HSV (hue, saturation, value). Kanál hue udává odstín barvy, saturation udává sytost barvy a value hodnotu jasu.



Obrázek 10.3: Zjednodušené blokové schéma toku obrazových dat.

Dále obrazová data vstupují do funkce, která byla speciálně vytvořena a sdružuje několik dílčích procesů za účelem snížení latence. Uvnitř této funkce, jež nese název **InRange_4Ch**, se vstupní tok dat rozdělí na čtyři shodné tříkanálové obrazové matice, přičemž na každém barevném kanálu (HSV) z každé matice je provedeno prahování současně zdola, tak i shora.



(a) : obraz před zpracováním

(b) : bitová maska vzniklá prahováním pro detekci modré barvy

(c) : bitová maska vzniklá prahováním pro detekci jasové složky

Obrázek 10.4: Porovnání obrazu před a po prahování.

Tedy, pokud je hodnota prahovaného pixelu větší než dolní práh T_1 a zároveň je hodnota menší než horní práh T_2, bude výslednému pixelu přiřazena maximální hodnota tj. 255, v opačném případě bude danému pixelu přiřazena hodnota 0, z toho vyplývá, že lze vzniklé obrazce nazývat bitovými maskami.

Meziproduktem tohoto procesu jsou tři bitové masky pro každou tříkanálovou obrazovou matici. Na všechny tři bitové masky, z nichž každá zachycuje prahování na jednotlivých barevných kanálech, je poté aplikován bitový součin. Čímž jsou masky spojeny do jedné. Výstupem z této funkce je jedna bitová maska pro každý ze čtyř detekčních kanálů. Detekčními kanály jsou myšleny obrazové matice pro detekci základních barev R, G, B a jasové složky Y.

Důvodem k rozdělení vstupního toku dat do čtyř datových toků a následného prahování jednotlivých obrazových matic, byla potřeba paralelní detekce více barev světelných zdrojů. První z prahovaných matic bude mít nastavené prahy pro detekci červené barvy světelného zdroje, což je zajištěno prahováním na kanálu Hue z barevného prostoru HSV. Zbylé dva barevné kanály jsou také prahovány. Kanál Saturation je prahován tak, aby nedetekoval příliš tmavé odstíny barvy, kanál Value je prahován tak, aby naopak nedetekoval příliš světlé odstíny, jelikož k detekci světlých míst v obrazu bude využita speciální obrazová matice.

Druhá z obrazových matic je určena pro detekci zelené barvy a způsob prahování je zde obdobný jako u červené barvy. To samé platí pro barvu modrou. Čtvrtá obrazová matice je speciální v tom, že je prahována na detekci všech jasných obrazových bodů. Takže zde prahování pro kanály Hue a Saturation není využito, což znamená, že rozmezí jejich hodnot není omezeno, a je prahován pouze kanál Value, tudíž jasová složka.

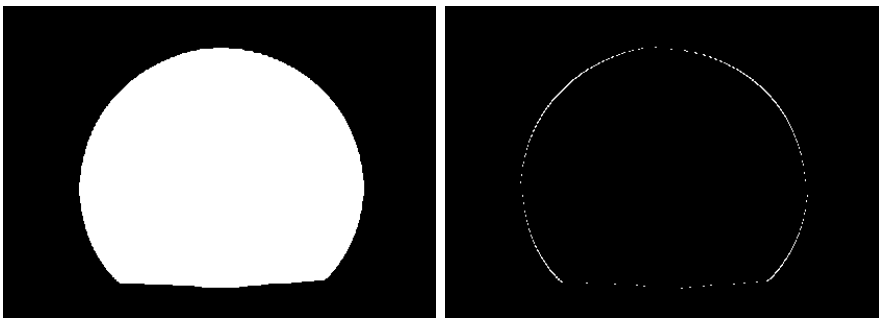
Na obrázku 10.4b je bitová maska, jenž vznikla zmíněným procesem. Prahování je nastaveno na detekci modré barvy. Pro úplnost je na obrázku 10.4c uvedena bitová maska, jenž je výsledkem prahování obrazové matice pro detekci jasové složky. Body masek vzniklých detekcí červené a zelené barvy jsou rovny hodnotě 0 v případě modrého zdroje.

Pro každý z těchto kanálů bylo vytvořeno propojení do procesorového systému, čímž následně bude možné měnit jednotlivé prahy T pro různé kanály, což je vhodné například pro ladění parametrů nebo pro detekci světelného pozadí detekované scény.

(a) : aplikace Eroze

(b) : aplikace Sobelova filtru (přiblíženo)

Obrázek 10.5: Operace pro detekci modré barvy.



(a) : před aplikací Sobelova filtru **(b) :** po aplikací Sobelova filtru

Obrázek 10.6: Aplikace Sobelova filtru.

Na jednotlivé bitové masky, jež jsou výsledkem prahování, jsou aplikovány tři iterace operace Eroze se strukturálním elementem o rozměru 3×3, jenž zmenší detekovanou oblast a odstraní náhodné pixely, které by mohly způsobovat chybu detekce. Což je znázorněno na obrázku 10.5a.

Následně je na každou bitovou masku aplikován Sobelův filtr s konvoluční maskou o rozměru 3×3, jenž detekuje hrany mezi černými a bílými oblastmi masky. Tím vznikne obrazová matice viz obrázek 10.5b, ve které jsou body ohraničující světlé a tmavé části světelného zdroje, jež byly detekovány a mohou být použity k získání polohy světelného zdroje.

Je vhodné poznamenat, že uvedený světelný zdroj typu LED PAR je tvořen skupinou LED čipů. Což po aplikaci prahování a Eroze vytváří ne zcela pravidelný tvar. To je v tomto případě způsobené tím, že je světelný zdroj nasnímán z krátké vzdálenosti. Kdyby byl zdroj snímán z velké vzdálenosti, mezery mezi jednotlivými LED čipy by v obrazu zanikly a získaný obraz by byl více méně pravidelný. Na druhou stranu i v tomto případě jsou získaná data validní a je možné je zpracovat, neboť výsledná poloha světelného zařízení je počítána jako aritmetický průměr souřadnic detekovaných bodů v každé ose, viz kapitola 11.

Pro demonstraci ideální aplikace Sobelova filtru je ještě uveden jiný světelný zdroj (stolní lampička), jenž má téměř pravidelný tvar, tudíž získané výsledky po aplikaci filtru jsou ukázkové a jak je vidět na obrázku 10.6b, získané body ohraničují oblast světelného zdroje.

Funkce **Get keypoints** slouží k vyhledání tzv. klíčových bodů, jež nesou informaci o poloze bodů získaných Sobelovým filtrem a zapsání jejich pozice do pole, které bude odesláno do procesorového systému, kde budou tato data následně zpracována a vyhodnocena.

Před aplikací Sobelova filtru byl každý z datových toků znovu rozdělen na dva shodné, kdy jeden z nich byl určen právě pro aplikaci zmíněného filtru a druhý pokračoval dále beze změny do funkce, jenž nese název **Multiplexer**. Podstatou této funkce je výběr obrazového datového toku, jenž bude zobrazován na displeji.

Do této funkce vstupují všechny čtyři bitové masky, které byly vytvořeny v předcho-

zích krocích a současně s nimi vstupuje také původní nezměněný obraz. Na základě signálu pro výběr výstupu, jenž přichází z procesorového systému, bude zvolený vstup převeden na výstup. Mezi možnosti volby zobrazení jednotlivých bitových masek, je zde i volba zobrazení všech masek najednou, a to samostatně nebo také s originálním obrazem. Tyto možnosti zobrazení jsou dále využity v různých částech procesu analýzy světelných zařízení.

Nakonec jsou přes obrazovou matici vykreslovány grafické obrazce nazývané **Overlay**, které slouží k překrytí obrazu v požadovaných místech. Vykreslovány jsou pouze čtyři druhy obrazců. První z vykreslovaných obrazců je obdélník, jenž slouží k ohraničení oblasti, ve které se nachází detekovaný zdroj světla. Barva obdélníku je odvozena od detekované barvy daného světelného zařízení, tudíž červená, zelená, modrá barva. Bílá barva je zvolena pro jakýkoliv světelný zdroj, jehož barva není z již zmíněných, a nebo u kterého nebyla barva rozeznána. Pozici vykreslovaného obdélníku je možné měnit z procesorového systému, stejně tak i rozměr v ose x a ose y a také barvu obdélníku.

Druhý typ vykreslovaných grafických obrazců je pravidelná mřížka o třech vertikálních a třech horizontálních čárách (při výchozím nastavení), která slouží k usnadnění umístění a nasměrování kamery na jeviště. Počet vykreslovaných čar v obou směrech je možné měnit z procesorového systému, nicméně mezi jednotlivými čárami bude vždy stejná mezera.

Třetím typem obrazců jsou čtyři digity pro vykreslení čísel a písmen. Tyto slouží například k informování uživatele o aktuálním stavu daného procesu nebo k vypsání kódu chyby. Jednotlivé části procesu analýzy a detekce světelných zdrojů a zobrazované informace k nim budou rozebrány v kapitole 11.

Čtvrtý druh vykreslovaných obrazců je až 16 obdélníků, které vymezují oblasti, v nichž se nacházejí detekovaná zařízení. Tyto oblasti vykazují shodné předdefinované vlastnosti, které jsou využity pro následné ovládání detekovaných zařízení. Funkce těchto oblastí je rozebrána v kapitole 11.3.5.

Poslední část v toku dat u tohoto IP bloku převádí obrazová data z maticové formy zpět na AXI4-Stream datový tok.

10.3 Realizace vybraných funkcí ve Vivado HLS

Jak již bylo zmíněno v kapitole 10.2.2, některé funkce zpracování obrazu byly vytvořeny speciálně pro účely této práce kvůli snížení latence nebo zvýšení propustnosti. Zde jsou uvedeny vybrané z nich:

- inRange
- Multiplexer

10.3.1 inRange

Funkce **inRange** slouží k prahování vstupní tříkanálové obrazové matice současně shora i zdola. Výstupem je jednokanálová obrazová matice (bitová maska). Pro každý ze vstupních barevných kanálů je definován dolní práh *lower_band* a horní práh *upper_band*. Pokud se hodnoty pixelů všech tří obrazových matic nacházejí mezi zmíněnými prahy, je výsledná hodnota rovna maximální hodnotě 255 v ostatních případech je hodnota výsledného pixelu 0. Tato funkce je aplikována na každý z detekčních kanálů.

```
void inRange(RGB_IMAGE& src, SINGLE_CHANNEL_IMAGE& mask,
   int* lower_band, int* upper_band)
{
   hls::Scalar<3, unsigned char> s, m;
   hls::Scalar<1, unsigned char> d;
   HLS_SIZE_T rows = src.rows;
   HLS_SIZE_T cols = src.cols;
   for(HLS_SIZE_T i = 0; i < rows; i++)
   {
      for(HLS_SIZE_T j= 0; j < cols; j++)
      {
#pragma HLS loop_flatten off
#pragma HLS pipeline II=1
         src >> s;
   for (char k = 0; k < 3; k++){
   if (s.val[k] >= *(lower_band + k) && s.val[k] <= *(upper_band + k)){
       m.val[k] = 255;
   }else{
       m.val[k] = 0;
   }
   }
   if (m.val[0] == 255 && m.val[1] == 255 && m.val[2] == 255){
       d.val[0] = 255;
   }else{
            d.val[0] = 0;
        }
        mask << d;
      }
   }
}
```

K optimalizaci kódu pro syntézu na systémové úrovni byly použity direktivy preprocesoru *HLS loop_flatten off* a *HLS pipeline II=1* pro snížení latence a zvýšení propustnosti (viz kapitola 6.2).

10.3.2 Multiplexer

Funkce **Multiplexer** je určena k zobrazení požadovaného datového toku na displeji. Na výběr je z jednotlivých bitových masek detekčních kanálů pro červenou, zelenou, modrou barvu a jasovou složku. Dále je možné vybrat původní obraz z kamery a nebo kombinaci zmíněných datových toků.

Stejně jako v předchozím případě byly k optimalizaci kódu pro syntézu na systémové úrovni použity direktivy preprocesoru.

```
void multiplexer(RGB_IMAGE& src_0, RGB_IMAGE& dst,
                 SINGLE_CHANNEL_IMAGE& src_1,
    SINGLE_CHANNEL_IMAGE& src_2,
                 SINGLE_CHANNEL_IMAGE& src_3,
    SINGLE_CHANNEL_IMAGE& src_4, char select) {
    hls::Scalar<3, unsigned char> s, d;
    hls::Scalar<1, unsigned char> s_1, s_2, s_3, s_4;
    HLS_SIZE_T rows = src_0.rows;
    HLS_SIZE_T cols = src_0.cols;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
#pragma HLS loop_flatten off
#pragma HLS pipeline II=1
      src_0 >> s;
      src_1 >> s_1;
            src_2 >> s_2;
            src_3 >> s_3;
      src_4 >> s_4;

      switch (select) {
          case 0:
            d = s;
      break;
    case 1:
      d.val[0] = 0;
      d.val[1] = 0;
      d.val[2] = s_1.val[0];
      break;
    case 2:
      d.val[0] = 0;
      d.val[1] = s_2.val[0];
      d.val[2] = 0;
      break;
    case 3:
      d.val[0] = s_3.val[0];
      d.val[1] = 0;
      d.val[2] = 0;
      break;
    case 4:
      d.val[0] = s_4.val[0];
      d.val[1] = s_4.val[0];
      d.val[2] = s_4.val[0];
      break;
    case 5:
      d.val[0] = s_3.val[0] / 2 + s_4.val[0] / 2;
      d.val[1] = s_2.val[0] / 2 + s_4.val[0] / 2;
      d.val[2] = s_1.val[0] / 2 + s_4.val[0] / 2;
      break;
    case 6:
      d.val[0] = s.val[0] / 3 + s_3.val[0] / 3 + s_4.val[0] / 3;
      d.val[1] = s.val[1] / 3 + s_2.val[0] / 2 + s_4.val[0] / 2;
      d.val[2] = s.val[2] / 3 + s_1.val[0] / 2 + s_4.val[0] / 2;
      break;
    default:
      d = s;
```

```
56          break;
57          }
58                  dst << d;
59            }
60      }
61  }
```

Kapitola 11

Procesorový systém

11.1 Konfigurace

Současně s bitstreamem, což je soubor, který obsahuje data pro programování programovatelné logiky, jsou vygenerovány funkce v jazyku C++, jež slouží k nastavování parametrů jednotlivých IP bloků celého systému z PS a jelikož je procesorový systém konfigurován také jako IP blok, tak i jeho. Nutno zmínit, že mezi těmito funkcemi se nacházejí i takové, jejichž návratová hodnota udává stav, v němž se IP blok nachází. Pokud blok vykoná svoji činnost, funkce s přívlastkem *IsDone* vrací hodnotu log.1, v opačném případě je hodnota log.0. Tato funkce bude dále využívána k synchronizaci systému.

Dále je vytvořen soubor, který obsahuje parametry, jež byly nakonfigurovány ve vývojovém prostředí Vivado, jako je například frekvence procesorového systému, adresy v paměti pro jednotlivé IP bloky a periférie atd. Takto vygenerované parametry jsou dále používány ve formě maker například k inicializaci periférií.

11.2 Inicializace

Jak již bylo zmíněno v kapitole 10.1, jako základ pro firmware celého systému byla použita část kódu z výukového dema, která slouží k inicializaci kamerového senzoru, systému přerušení, periférií UART, GPIO, HDMI a I^2C, jenž slouží pro komunikaci s kamerovým senzorem. Mezi další periférie, které bylo nutné inicializovat, patří systémový časovač, jenž bude generovat přerušení pro odesílání DMX512 dat, a poté samostatné rozhraní DMX512.

Pro inicializaci protokolu DMX512 bylo třeba vytvořit ovladač, jenž umožňuje komunikaci mezi SoC Zynq-7020 a modulem Pmod RS485. Jak již bylo zmíněno, DMX512 používá stejné elektrické úrovně pro logické signály jako RS485, a proto byl použit modul Digilent Pmod RS485, na jehož primární straně jsou přiváděna data

o elektrických úrovních 0 V pro log.0 a 3,3 V pro log.1. Na sekundární straně jsou elektrické úrovně odpovídající specifikaci RS485.

Jelikož jsou datové rámce DMX512 obdobné, jako používá UART komunikace, bylo původně zamýšleno odesílat data protokolu DMX512 s využitím hardwarové periférie UART. Nicméně dané řešení vedlo k nefunkčnosti jiných periférií, kdy řešení problému nefunkčnosti ostatních periférií bylo časově velmi náročné. Tudíž se nakonec přistoupilo k softwarovému řešení odesílání dat protokolu DMX512 pomocí GPIO pinu a systémového časovače, jenž generuje přerušení pro odeslání každého bitu.

Nakonec je potřeba inicializovat IP blok vytvořený pomocí Vivado HLS a povolit jeho automatický restart, jakmile dokončí vykonávaný proces. Procesem je zde míněno zpracování jednoho obrazového snímku a odeslání klíčových bodů do procesorového systému, v případě, že nějaké byly detekovány. Dále se jedná o počáteční nastavení hodnot jednotlivých proměnných parametrů, jež byly vytvořeny pro řízení činnosti IP bloku. Mezi nastavitelné parametry patří:

band lower, band upper - dolní a horní práh pro prahování obrazových matic, jež jsou určeny pro detekci jednotlivých barev R, G, B a jasové složky Y

keypoints - detekované nespojitosti ve filtrovaném obrazu s citlivostí na dané barvy R, G, B a jasovou složku Y

output select - výběr výstupu obrazového multiplexeru

segment, segment position - slouží pro nastavení jednotlivých segmentů sedmisegmentových digitů a pozice samotných digitů

center x, center y, x size, y size, box color - parametry pro nastavení pozice, velikosti a barvy obdélníku, jenž ohraničuje detekované světelné zdroje

scene left top, scene right bot, scene opacity - nastavení tvarů simulujících požadované rozmístění světelných zdrojů

number of vertical lines, number of horizontal lines - počet vertikálních a horizontálních čar mřížky překrývající obraz

11.3 Firmware

Firmware procesorového systému je rozdělen do několika dílčích částí. Při spuštění zařízení proběhne výše zmíněná inicializace systému. Poté přichází na řadu běh samotného programu pro analýzu světelných zařízení, jenž je rozdělen na následující fáze:

- **Idle**
- **Setup**

- **Process**
- **Check results**
- **Match scene**
- **Error**
- **Tuning**

11.3.1 Idle

Idle je první fází, do které vstupuje program po inicializaci celého systému. V této fázi je zobrazován pouze originální obraz překrytý mřížkou a nápisem **IdLE** (viz obrázek 11.1a) v levém dolním rohu. Kamerový senzor je nastaven na výchozí nastavení, automatická nastavení senzoru a obrazu jsou aktivní.

Účelem fáze je zobrazit to, co snímá kamera a umožnit její umístění před jevištěm tak, aby bylo v zorném poli jeviště celé, popřípadě i prostor kolem jeviště, kde by se mohly vyskytovat světelné efekty. Pravidelná mřížka, jež překrývá obraz, napomáhá právě k nasměrování kamery.

11.3.2 Setup

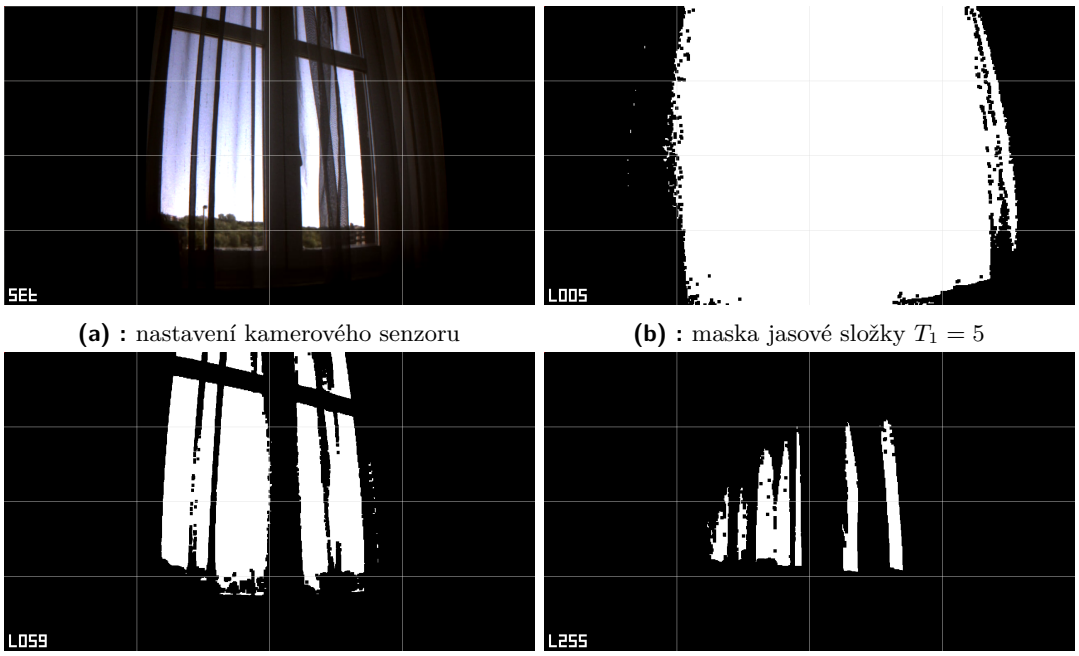
Druhá fáze je určena pro detekci maximálního okolního světelného záření za podmínek, že nesvítí žádné z detekovaných zařízení. Na displeji je zobrazována maska detekčního kanálu pro jasovou složku Y. V levém dolním rohu je nápis **SEt** (viz obrázek 11.1b), jenž informuje o tom, v jaké fázi se program nachází.

Na počátku této fáze jsou přenastaveny hodnoty kamerového senzoru, a to hlavně čas expozice a zisk kamerového senzoru, jak bylo zmíněno v kapitole 9.2. U obrazové matice, jež je filtrována tak, aby detekovala jasovou složku v obrazu, je změněn dolní práh T_1 pro operaci prahování na hodnotu 5 pro kanál Value. To způsobí, že jsou detekovány pixely, které mají větší hodnotu, než je právě dolní hodnota prahu. Horní práh T_2 je nastaven na maximální hodnotu, což je 255.

Tímto nastavením se docílí toho, že je v obrazu detekováno mnoho útvarů, čímž jsou generovány klíčové body, které tyto útvary ohraničují. Z čehož vyplývá, že pokud

(a) : fáze Idle **(b) :** fáze Setup **(c) :** úroveň jasu **(d) :** fáze Process **(e) :** detekovaná adresa

Obrázek 11.1: Nápisy na displeji I.



(a) : nastavení kamerového senzoru

(b) : maska jasové složky $T_1 = 5$

(c) : maska jasové složky $T_1 = 59$

(d) : maska jasové složky $T_1 = 255$

Obrázek 11.2: Demonstrace velké intenzity okolního osvětlení.

by byla při následné detekci světelných zařízení hodnota dolního prahu stejná, budou působit okolní světelné zdroje jako rušení pro proces detekce. Tudíž je hodnota dolního prahu zvýšena o 2 a znovu je kontrolováno, jestli jsou detekovány nějaké útvary. V případě, že jsou útvary v obrazu opět detekovány, je hodnota dolního prahu opět zvýšena. O aktuálně nastavené hodnotě dolního prahu informuje nápis v levém dolním rohu **LXXX**, kde za *XXX* je dosazena právě zmíněná hodnota tak, jak je znázorněno na obrázku 11.1c.

Tento proces je opakován do té doby, než se dosáhne stavu, kdy v obrazu nejsou detekovány žádné útvary. Tím je získána hodnota pro nastavení dolního prahu kanálu Value, jež je použita i pro ostatní obrazové matice, které slouží pro detekci červeného, zeleného a modrého světelného zdroje. Avšak není použita přímo detekovaná hodnota, ale hodnota o 10 větší, čímž se sníží pravděpodobnost výskytu chyby způsobené náhodným odrazem.

Pokud by intenzita okolního světelného záření byla vysoká tak, že budou v obrazu detekovány útvary i v případě, že je hodnota dolního prahu 255, přechází zařízení do chybového stavu viz kapitola 11.3.6. Důvodem přechodu do chybového stavu je fakt, že není vhodné detekovat světelná zařízení za daných podmínek. Při detekci se předpokládá, že detekovaný světelný zdroj bude mít v obrazu větší intenzitu jasu, než vykazují ostatní snímané objekty. Což právě zajišťuje nastavení dolního prahu kanálu Value, neboť objekty, jež mají menší hodnotu jasu, nebudou detekovány. Z experimentů vyplývá, že vhodná úroveň dolního prahu pro kanál Value je jakákoliv hodnota menší než 150.

Na obrázku 11.2 je demonstrován průběh navyšování dolního prahu T_1 pro kanál Value. Na obrázku 11.2d je vidět, že byla intenzita okolního osvětlení příliš velká a tudíž i po dosažení hodnoty 255 pro dolní práh T_1 se v obrazu nacházejí útvary, které generují klíčové body.

11.3.3 Process

V předchozí fázi byly nastaveny počáteční podmínky pro proces detekce světelných zdrojů, dále přichází na řadu samotná detekce světelných zařízení. Nejprve je však třeba zmínit, že celý proces je synchronizován se snímkovou frekvencí kamerového senzoru tj. 50 fps. Se stejnou frekvencí také zpracovává IP blok **Brightness tracking** jednotlivé snímky, tudíž se stejnou frekvencí jsou odesílány klíčové body z IP bloku do procesorového systému.

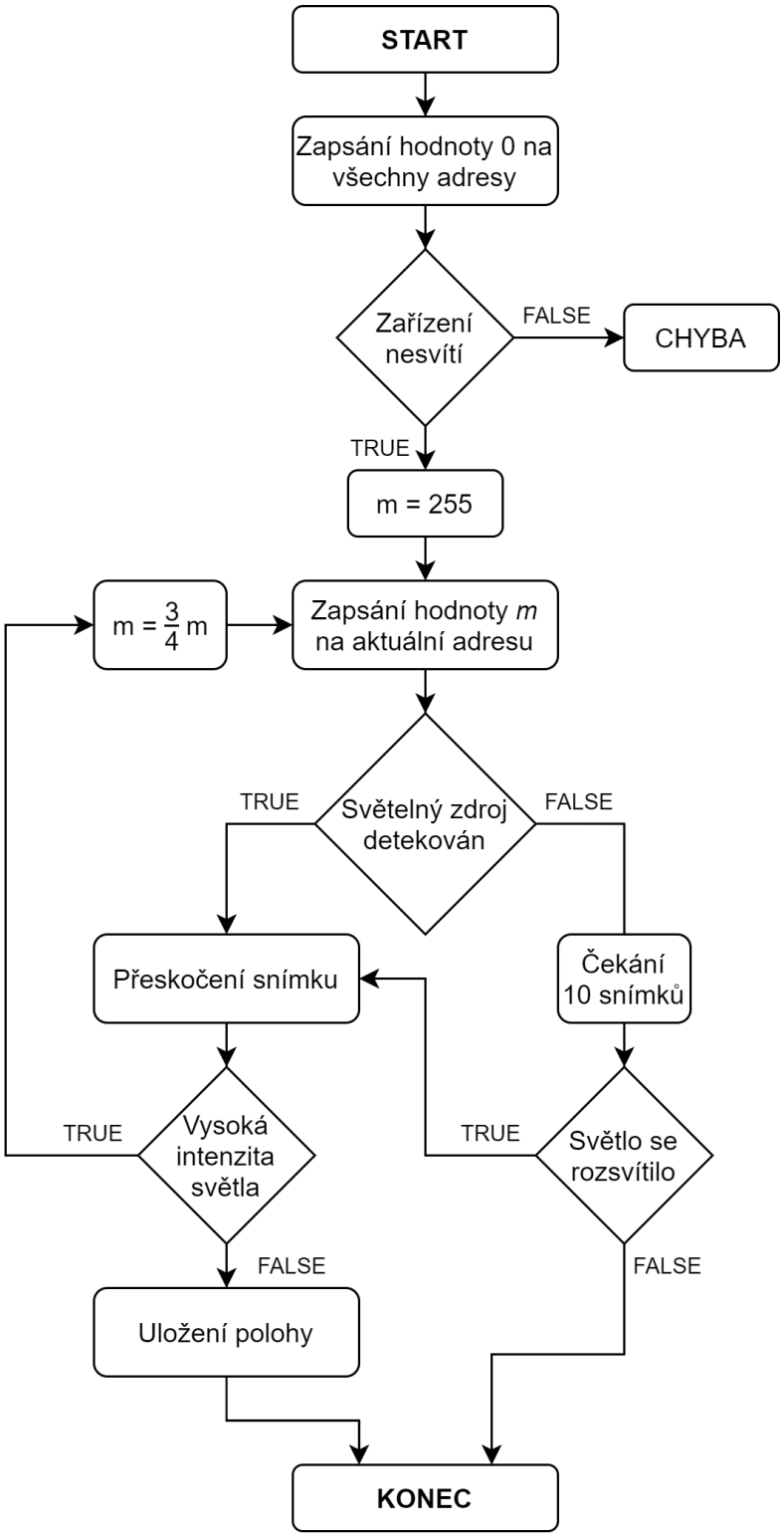
Kamerový senzor je nastaven stejně jako v předchozí fázi, na displeji je zobrazován jak původní obraz, tak i bitové masky všech detekčních kanálů. Tyto masky mají stejnou barvu, jako je barva, kterou tento kanál detekuje. Pro jasovou složku byla zvolena bílá barva. V levém dolním rohu je označení aktuální fáze **PrOC** viz obrázek 11.1d.

Proces probíhá v několika krocích a je opakován pro každou z adres protokolu DMX512. Zjednodušený průběh procesu je znázorněn obrázku 11.3. Na začátek je odeslána hodnota 0 na všechny DMX512 adresy, pro zajištění, nečinnosti všech světelných zdrojů. Poté probíhá kontrola, jestli jsou všechna zařízení nečinná tak, že se kontroluje zda jsou v obrazu detekovány klíčové body, jak pro červenou, zelenou a modrou barvu, tak i pro jasovou složku.

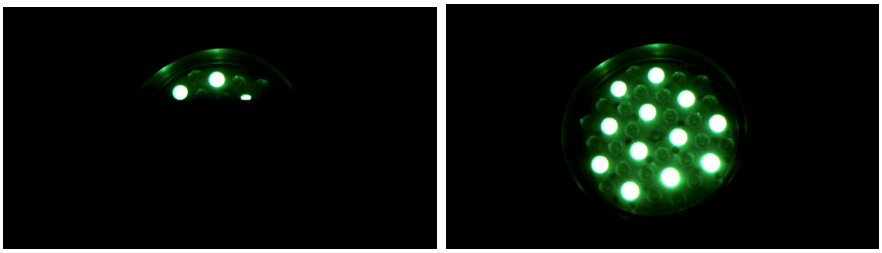
Tato kontrola je prováděna jednak z toho důvodu, že některá světelná zařízení reagují na příkazy opožděně nebo chvíli trvá než se zhasnou, což by mohlo při detekci na další adrese způsobovat chyby. Dalším důvodem ke kontrole je případ, kdy se v průběhu detekce objeví v zorném poli nečekaný zdroj světla, což ve většině případů vede k chybě detekce. Avšak zmíněnou kontrolou se tomuto případu předejde. Pokud se objeví v zorném poli nečekaný zdroj světla, je proces detekce pozastaven a čeká se přibližně 4 s. V případě, že zdroj světla nepomine, přechází program do chybového stavu viz kapitola 11.3.6.

Po kontrole nečinnosti všech zařízení je vyslána na aktuálně detekovanou DMX512 adresu hodnota 255, za účelem rozsvícení světelného zařízení. Tato adresa je také uvedena v levém dolním rohu ve tvaru **PXXX**, kde *XXX* je právě daná adresa (viz obrázek 11.1e). Následující periodu se detekuje, jestli byly na nějakém z detekčních kanálů detekovány klíčové body.

Pokud byly v aktuálním snímku detekovány klíčové body, neurčuje se z tohoto snímku poloha světelného zařízení. Důvodem k tomu je fakt, že se světelný zdroj mohl rozsvítit v průběhu vykreslování daného snímku (viz obrázek 11.4a). Tím by mohla být zaznamenána pouze část plochy, kterou zdroj v obrazu zabírá, a to by mohlo způsobit, že bude jeho poloha špatně určena. Určení polohy světelného zdroje probíhá



Obrázek 11.3: Zjednodušený průběh detekce na jedné adrese.



(a) : neúplně vykreslený světelný zdroj **(b) :** úplně vykreslený světelný zdroj

Obrázek 11.4: Částečně vykreslený světelný zdroj.

až v následujícím snímku, kde se počítá s tím, že bude oblast, kterou světelný zdroj v obrazu zabírá, kompletně vykreslena (viz obrázek 11.4b).

Poloha světelného zdroje je určena na základě získaných klíčových bodů, které již byly znázorněny na obrázcích 10.5b a 10.6b. Z těchto bodů je vypočítán střed světelného zařízení jako aritmetický průměr jejich souřadnic v ose x a v ose y. Rozměr obdélníku, jenž ohraničuje oblast světelného zdroje, v ose x je počítán jako dvojnásobek vzdálenosti mezi středem a nejvzdálenějším klíčovým bodem v dané ose. To stejné platí i pro rozměr obdélníku v ose y. Vynásobením těchto dvou rozměrů je získán přibližný obsah plochy, kterou světelný zdroj v obrazu zabírá.

V případě, že je oblast, kterou zabírá světelný zdroj v obrazu, větší než čtvrtina plochy obrazu, předpokládá se, že intenzita zdroje je příliš vysoká a určení polohy by nebylo přesné. Tudíž je snížena intenzita tak, že je na danou DMX512 adresu odeslána hodnota o čtvrtinu menší. Poté je opakována detekce klíčových bodů.

Pokud je oblast, kterou světelný zdroj v obrazu zabírá, již menší než čtvrtina plochy obrazu, je uložena poloha světelného zdroje. V opačném případě je intenzita snížena opakovaně. Může nastat situace, kdy je oblast stále větší, než je požadováno, a to i při hodnotě 1. Za těchto okolností je již uložena poloha i informace o tom, že má světelný zdroj velkou intenzitu, což je důležité znát při dalším zacházení s daným světelným zařízením.

Další z možností je, že se na sběrnici DMX512 nenachází žádné zařízení s právě detekovanou adresou. V průběhu detekce, při kontrole, jestli byly získány klíčové body, se v případě, že nebyly detekovány žádné klíčové body čeká 10 period. Takto nastavená doba je jednak z důvodu, že některá světelná zařízení reagují na příkazy se zpožděním nebo chvíli trvá, než se rozsvítí. Pokud je po uběhnutí zmíněné doby detekce neúspěšná, je aktuální adresa považována za neobsazenou.

Jakmile je poloha určena nebo je zjištěno, že je adresa neobsazená, běh programu se vrací na pozici, kdy je na všechny adresy odeslána hodnota 0 a celý proces detekce je opakován pro další adresu. Tímto způsobem probíhá detekce na každé z 512 adres protokolu DMX512. Proces je možné kdykoliv přerušit a pokračovat do další fáze analýzy.

Doba detekce

Doba trvání detekce světelných zařízení je ovlivněna několika faktory, přičemž největší podíl na výsledném času má doba čekání systému na odezvu, pokud je detekovaná adresa neobsazena. Jak již bylo zmíněno, pokud nepřichází odezva na odeslaný povel, čeká se 10 period, poté se přechází na detekci dalšího zařízení. Když se k tomuto času přičtou i operace před čekáním, je výpočet následující:

$$t = z + c + l + w + j + d, \tag{11.1}$$

kde t je celkový počet period pro detekci jednoho zařízení,
z je fáze, kdy je na všechna zařízení odeslána hodnota `0`,
c je fáze, v níž probíhá kontrola, jestli jsou všechna světelná zařízení nečinná,
l je fáze odeslání hodnoty 255 na danou adresu,
w je doba čekání na odezvu,
j je fáze v níž jsou detekovány klíčové body, ale detekce neprobíhá,
d je doba detekce.

Nejdelší doba detekce světelných zařízení by byla dosažena v případě, pokud by se na sběrnici nenacházelo žádné světelné zařízení, čímž by převažovaly doby čekání. Doba pro detekci na jedné adrese je vypočtena dosazením následujících hodnot za dané proměnné:

$$t = 1 + 1 + 1 + 10 + 0 + 0.$$

Periody pro detekování klíčových bodů jsou rovny hodnotě 0, jelikož žádné klíčové body detekovány nejsou. Detekce na jedné adrese tedy trvá 13 period. Jak již bylo zmíněno, frekvence procesu je 50 Hz, tudíž perioda je 20 ms, z čehož vyplývá, že detekce na všech adresách DMX512 celkově trvá v nejhorším případě 133, 12 s.

Nejkratší doby detekce by bylo možné dosáhnout v případě, že by každá z adres DMX512 byla obsazena světelným zařízením, tím by dosazení do rovnice 11.1 vypadalo následovně:

$$t = 1 + 1 + 1 + 0 + 1 + 1.$$

Celková doba trvání procesu detekce je v tomto případě 51, 2 s. Výsledná doba trvání celého procesu je někde mezi těmito extrémy, jelikož s největší pravděpodobností nebudou nikdy obsazeny všechny adresy. Mezi další faktory, které prodlužují dobu trvání, patří například taky případ, kdy je intenzita světelného zdroje vysoká a musí být snižována, což prodlužuje dobu pro detekci na jedné adrese. Nicméně tento případ nastává jen občas.

(a) : fáze Check results | **(b)** : zobrazené zařízení | **(c)** : fáze Match scene | **(d)** : fáze Tuning | **(e)** : zkoušená adresa | **(f)** : kód chyby

Obrázek 11.5: Nápisy na displeji II.

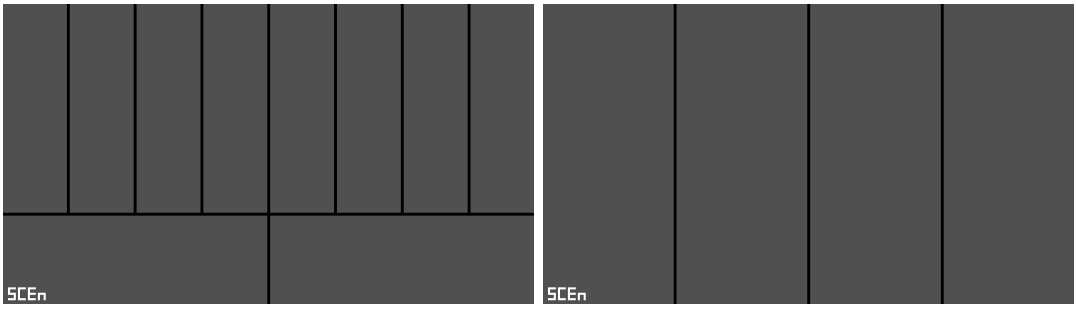
11.3.4 Check results

Po ukončení detekce světelných zařízení na každé DMX512 adrese, je možné zkontrolovat výsledky detekce. Na začátku této fáze je v levém dolním rohu nápis **CHEC** (viz obrázek 11.5a. Kamerový senzor je nastavený na výchozí nastavení tak, jako ve fázi **Idle**. Na displeji je zobrazen originální obraz, který zachycuje celou scénu. Ze všech DMX512 adres jsou vybrány pouze ty, na kterých byly detekovány světelné zdroje. Tyto adresy je možné procházet.

Adresa na níž byl detekován světelný zdroj je označena **CXXX**, kde *XXX* je nahrazeno jako v předchozích případech danou adresou. Přes obraz je vykreslován obdélník, ohraničující polohu, kde byl na dané adrese detekován světelný zdroj. Obdélník má barvu odpovídající té, která byla detekována. Také je možné aktuálně prohlížené světelné zařízení rozsvítit stisknutím tlačítka *Test* (viz kapitola 12).

11.3.5 Match scene

Touto fází končí celý proces analýzy světelných zařízení. Kamerový senzor zůstává nastavený jako ve fázi předchozí. To stejné platí i u zobrazeného obrazu na displeji. V levém dolním rohu je nápis **SCEn**. Přes obraz jsou vykresleny průhledné obdélníky, které vymezují oblasti, jež mají určité vlastnosti. Pokud se v některém z těchto obdélníků nachází detekovaný světelný zdroj, je tomuto zdroji přiřazena informace o tom, že se nachází v dané oblasti. Rozložení obdélníků je nazýváno *Předdefinovaná scéna* (viz obrázek 11.6). Mezi těmito scénami lze přepínat a vybrat, které rozložení uživatel preferuje.



(a) : rozložení scény s deseti oblastmi | **(b)** : rozložení scény se čtyřmi oblastmi

Obrázek 11.6: Příklad předdefinované scény.

Získané informace o jednotlivých zařízeních jsou odeslány přes rozhraní UART do externího zařízení, které tato data zpracuje. Externí zařízení ovládá světelnou show na základě předem naprogramovaného sledu pokynů, které jsou vázané na Předdefinovanou scénu. A to konkrétně na polohu daného obdélníku (Scene segment). Pokud tedy bude mít externí zařízení informaci o tom, že se zařízení nachází v oblasti, která má být například rozsvícena, bude toto zařízení rozsvíceno.

Příklad analyzovaných dat, jež jsou odesílána přes rozhraní UART:

```
RESULTS_START
Analysed devices: 12
CH: 1, Color: RED, X: 357, Y: 193, X size: 71, Y size: 75, Area: 5325,
    Intensity: 255, Scene segment: 2
CH: 2, Color: GREEN, X: 348, Y: 175, X size: 59, Y size: 61, Area:
    3599, Intensity: 255, Scene segment: 2
CH: 3, Color: BLUE, X: 357, Y: 194, X size: 81, Y size: 79, Area: 6399,
     Intensity: 255, Scene segment: 2
CH: 11, Color: RED, X: 508, Y: 194, X size: 159, Y size: 157, Area:
    24963, Intensity: 255, Scene segment: 3
CH: 12, Color: GREEN, X: 505, Y: 187, X size: 103, Y size: 112, Area:
    11536, Intensity: 255, Scene segment: 3
CH: 13, Color: BLUE, X: 511, Y: 195, X size: 181, Y size: 191, Area:
    34571, Intensity: 255, Scene segment: 3
CH: 21, Color: RED, X: 747, Y: 174, X size: 159, Y size: 173, Area:
    27507, Intensity: 255, Scene segment: 7
CH: 22, Color: GREEN, X: 744, Y: 177, X size: 133, Y size: 133, Area:
    17689, Intensity: 255, Scene segment: 7
CH: 23, Color: BLUE, X: 754, Y: 178, X size: 193, Y size: 203, Area:
    39179, Intensity: 255, Scene segment: 7
CH: 31, Color: RED, X: 899, Y: 188, X size: 79, Y size: 95, Area: 7505,
     Intensity: 255, Scene segment: 6
CH: 32, Color: GREEN, X: 892, Y: 182, X size: 67, Y size: 73, Area:
    4891, Intensity: 255, Scene segment: 6
CH: 33, Color: BLUE, X: 897, Y: 187, X size: 105, Y size: 107, Area:
    11235, Intensity: 255, Scene segment: 6
RESULTS_END
```

11.3.6 Error

Do této fáze bývá přesměrován běh programu v případě, že se někde vyskytla chyba. Systém rozeznává několik druhů chyb, kdy každé chybě je přidělen kód, který je znázorněn na displeji viz obrázek 11.5f. Mezi chybové stavy se řadí:

- **Exceeded light intensity**
- **Unexpected light source**

Exceeded light intensity neboli překročení intenzity světla. Chyba s kódem **Er01** upozorňuje na příliš vysokou světelnou intenzitu okolí, jež by při následné detekci světelných zdrojů způsobovala chyby v určení jejich polohy. Za těchto podmínek není vhodné provádět detekci a je třeba snížit světelnost okolí.

Unexpected light source neboli neočekávaný světelný zdroj. Chyba s kódem **Er02** nastává, když je na všechny adresy DMX512 odeslána hodnota 0 a některý ze světelných zdrojů zůstane svítit, nebo když se v průběhu detekce vyskytne jiný světelný zdroj, který by měl být zhasnutý.

11.3.7 Tuning

Do této fáze je možné se dostat z fáze *Idle* stisknutím tlačítka *Test* (význam jednotlivých tlačítek je uveden v kapitole 12). Kamerový senzor zůstává nastavený stejně jako v předchozí fázi. Na displeji je zobrazován původní obraz. V levém dolním rohu se zobrazí nápis **tunE** (viz obrázek 11.5d). Při přepínání mezi jednotlivými adresami je zobrazen nápis **tXXX**, kde místo *XXX* je jako obvykle uvedena vybraná adresa (viz obrázek 11.5e).

Pokud je stisknuto tlačítko *Test*, je na aktuální adresu odeslána hodnota 255, pokud je tlačítko uvolněno, je na stejnou adresu odeslána hodnota 0. V obou případech je na ostatní adresy odesílána hodnota 0. Tato fáze je opuštěna po stisknutí tlačítka, jenž slouží pro přepínání mezi jednotlivými fázemi.

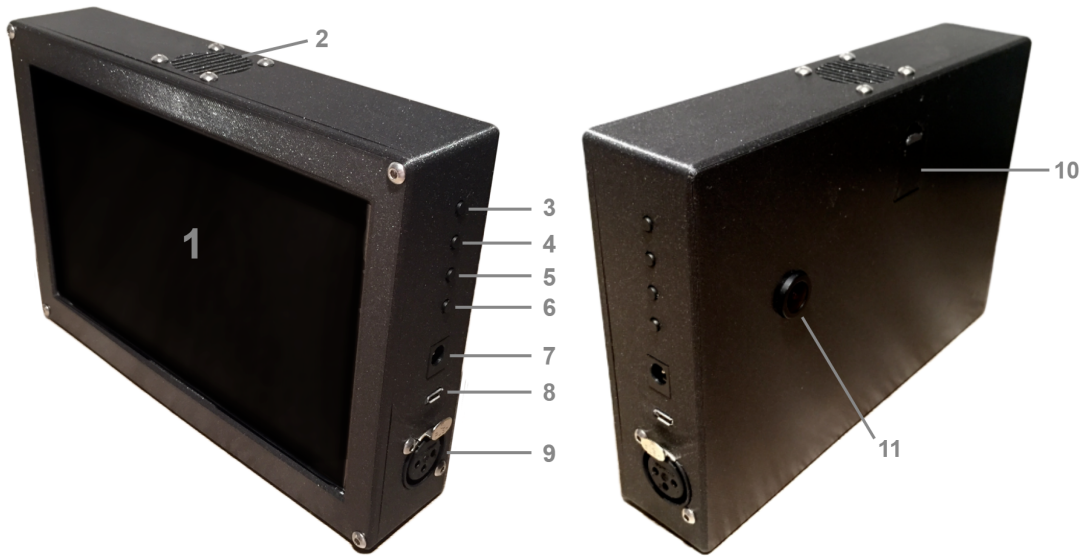
Kapitola 12

Realizace

Pro snadnou manipulaci a používání celého zařízení byly všechny použité komponenty (vývojová deska Zybo Z7, Pcam 5C, Pmod RS485, displej, tlačítka, konektory, ventilátor) umístěny do krabičky tištěné na 3D tiskárně materiálem typu PLA (viz obrázek 12.1).

Z přední strany zařízení je umístěn displej (1), na horní straně krabičky se nachází otvor pro ventilátor a pod ním také samotný ventilátor (2). Na boční straně jsou umístěna čtyři tlačítka (3, 4, 5, 6). První z tlačítek slouží ke spuštění procesu analýzy světelných zařízení a také k přepínání mezi jednotlivými fázemi tohoto procesu.

Tlačítka 4 a 5 slouží k procházení adres detekovaných zařízení a výběru Předdefinovaných scén, kdy tlačítko 4 je pohyb nahoru a tlačítko 5 je pohyb dolů. Také je možné těmito tlačítky ve fázi *Idle* měnit počet horizontálních a vertikálních čar, jež



(a) : přední pohled **(b) :** zadní pohled

Obrázek 12.1: Realizované zařízení.

jsou vykreslovány přes obraz.

Tlačítko označené číslem 6 má funkcí více. Ve fázích *Check results* a *Tuning* je toto tlačítko nazváno *Test* a slouží k rozsvícení světelného zdroje, jenž se nachází na aktuální adrese. Po stisknutí tlačítka ve fázi *Idle* je běh programu přesunut do fáze *Tuning*.

Číslem 7 je označen souosý konektor $5,5\,\text{mm}/2,1\,\text{mm}$ (střední pin je +), jenž slouží k napájení celého zařízení. Napájecí napětí je $5\,\text{V}$ a potřebný proud je $1\,\text{A}$. Dále se zde nachází USB konektor (8), který slouží k přenášení dat získaných na základě detekce z *Stage Light Analyzer* do externího zařízení přes rozhraní UART.

Poslední konektor v řadě (9) je třípinový konektor typu XLR, jenž slouží pro připojení sběrnice DMX512, na níž se nachází zařízení, která mají být detekována. Přestože dle standardu by se měl používat konektor s pěti piny viz kapitola 7, byl použit konektor s třemi piny, jelikož se v dnešní době vyskytuje mnoho zařízení s tímto druhem konektoru. Pokud by bylo potřeba použít konektor s pěti piny, může být použita kabelová redukce.

Na zadní straně zařízení se nachází pod číslem 10 servisní otvor a pod číslem 11 objektiv kamerového modulu Pcam 5C. Na spodní straně zařízení je umístěna matice se závitem (UNC 1/4”), který se používá u fotoaparátů. Tato matice slouží k přichycení zařízení ke stativu.

Kapitola 13

Omezení

Realizované zařízení je schopné detekovat světelná zařízení ovládaná pomocí protokolu DMX512 pouze za následujících podmínek:

Detekované zařízení využívá společný dimmer kanál

Pokud světelné zařízení využívá společný dimmer kanál (tj. kanál, který reguluje intenzitu všech ostatních kanálů dohromady), není na základě realizovaného postupu detekce světelných zařízení možné detekovat toto zařízení. Neboť k rozsvícení tohoto světelného zařízení je nutné odesílat pokyny na dvě DMX512 adresy současně, a to na adresu barevného kanálu (R, G, B, atd), jenž má být ovládán a taky na adresu dimmer kanálu. Přičemž *Stage Light Analyzer* provádí detekci pouze na jedné adrese DMX512. Adresy, které obsazuje toto zařízení, budu detekovány jako neobsazené.

Tento problém by mohl být vyřešen například detekcí na více adresách, za předpokladu, že má společný dimmer kanál adresu o 1 vyšší než poslední z barevných kanálů (tato úvaha vychází ze získaných informací o různých DMX512 zařízeních). Nicméně to není pravidlem a společný dimmer kanál se může nacházet na adrese o 1 menší než první z barevných kanálů a nebo na jiné adrese.

Aplikace tohoto způsobu detekce by navyšovala celkový čas potřebný k celému procesu, neboť by bylo nutné zkoušet různé kombinace současného rozsvícení více kanálů. Což by vedlo také k tomu, že výpočet uvedený v kapitole 11.3.3 by musel být pozměněn.

Z čehož vyplývá, že realizace toho způsobu detekce světelných zařízení je možná, nicméně není cílem této práce a bude řešena při dalším vývoji.

Detekce základních barev

V současné verzi *Stage Light Analyzer* je umožněna detekce pouze základních barev, jež jsou používány ve většině zařízení. Jsou to barvy červená, zelená, modrá a bílá.

Některá zařízení používají například jantarovou barvu. Nicméně tato barva není tolik rozšířená a v procesu detekce bude klasifikována jako bílá. To stejné bude platit i pro jiné neobvyklé barevné složky světelných zařízení.

Detekované zařízení umožňuje pohyb a rotaci

Jak již bylo zmíněno v kapitole 7.5, způsob detekce světelných zařízení, jenž je realizován u *Stage Light Analyzer* není vhodný pro identifikaci zařízení typu Otočná hlavice, které kromě svícení umějí pohybovat a rotovat zdrojem světla, zaostřovat světelný paprsek, tvarovat jej a mnoho dalších funkcí. Tyto funkce jsou také přiřazeny na DMX512 adresy. V případě detekce budou adresy pro pohyb a rotaci zařízení detekovány jako neobsazené, což by v rámci detekce světelných zařízení nevadilo. Nicméně není doporučeno u těchto zařízení detekci provádět, neboť pokud by byl v jejím průběhu zdroj světla daného zařízení otočen na opačnou stranu, než je umístěna kamera, nebyl by tento světelný zdroj detekován.

Jak již bylo uvedeno, detekce pohybů světelných zařízení není u SLA v rámci této práce implementována. Navržené řešení tohoto problému je následující a ne zcela kompletní.

V prvním případě, pokud by toto světelné zařízení bylo na počátku procesu detekce otočené na kameru a byly detekovány jeho barevné kanály, bylo by dále možné detekovat pohyb tak, že by jeden z kanálů, jenž byl detekován, svítil a zároveň by se posílaly pokyny na další adresy. V obrazu by se detekovalo, jestli daný světelný zdroj mění pozici nebo ne.

V druhém případě, pokud by světelné zařízení nebylo otočené na kameru, by analýza tohoto zařízení byla komplikovanější a časově náročnější na realizaci.

Kapitola 14

Závěr

Cíl práce, navrhnutí a realizace funkčního zařízení založeného na systému na čipu Zynq-7000 za využití vývojového prostředí Vivado HLS, jenž umožňuje v reálném čase identifikovat vybraná světelná zařízení řízená protokolem DMX512 ve snímaném obrazu, byl splněn dle zadání.

Byla vybrána vhodná kamera pro snímání světelných efektů a rozšiřující modul k vývojové desce ZYBO Z7-20, jenž zajišťuje fyzickou vrstvu pro protokol DMX512. Na základě informací o zařízeních, která jsou používána ve světelné technice, za využití zpracování obrazu a platformy Zynq byl navržen způsob detekce světelných zdrojů typu PAR a Striplight, jež nevyužívají společný stmívací kanál. Navíc byl navržen způsob přiřazení předprogramované světelné show k detekovaným zařízením, na základě pozice, na které byla světelná zařízení detekována.

Zařízení funguje dle předpokladů, jakmile je správně nasměrována kamera na světelná zařízení, jež mají být detekována, je možné spustit proces detekce. Po spuštění procesu jsou nastaveny počáteční podmínky změřením světelné intenzity okolí. Následuje samotná detekce světelných zařízení postupně na všech adresách protokolu DMX512. Pokud zařízení reaguje rozsvícením na právě detekované adrese, reakce je zachycena kamerou a následně je vyhodnocena. Po prozkoumání všech adres jsou výsledky rekapitulovány a prezentovány uživateli. Nakonec je možné přiřadit detekovaná zařízení k předprogramované scéně.

Realizované zařízení může být v budoucnu vylepšeno například rozšířením detekce o zařízení, která využívají společný stmívací (dimmer) kanál. Možné řešení tohoto vylepšení již bylo navrženo v kapitole 13. Dále by mohl být realizován způsob detekce pro světelná zařízení typu Otočná hlavice. Další možností vylepšení je využití operačního systému Linux, díky němuž by mohla být práce s grafikou uživatelského rozhraní přívětivější. Také by bylo možné z realizace vyjmout vývojovou desku ZYBO Z7-20 a navrhnout DPS speciálně pro tohle zařízení, čímž by se zmenšily geometrické rozměry krabičky.

Bibliografie

1. Kotlil, Pjoter [online] [cit. 2020-05-13]. Dostupné z: https://www.barrak-club.cz/cz/pronajem/.
2. [online] [cit. 2020-05-13]. Dostupné z: https://www.ottawaspecialevents.com/events-concerts-stage-shows-lighting-designers-ottawa.htm.
3. Crockett, Louise H.; Elliot, Ross A.; Enderwitz, Martin A. *The Zynq Book: Embedded Processing Withe ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC*. Strathclyde Academic Media, 2014.
4. Husák, Jiří. *Využití syntézy na systémové úrovni pro aplikace s platformou ZYNQ*. 2015. Diplomová práce. VUT FIT, Brno.
5. Digilent. *Zybo Z7: Zynq-7000 ARM/FPGA SoC Development Board* [online] [cit. 2020-05-05]. Dostupné z: https://store.digilentinc.com/zybo-z7-zynq-7000-arm-fpga-soc-development-board/.
6. Digilent. *Pmod RS485 Reference Manual* [online] [cit. 2020-04-27]. Dostupné z: https://reference.digilentinc.com/reference/pmod/pmodrs485/reference-manual.
7. Digilent. *Pcam 5C: 5 MP Fixed Focus Color Camera Module* [online] [cit. 2020-05-05]. Dostupné z: https://store.digilentinc.com/pcam-5c-5-mp-fixed-focus-color-camera-module/.
8. Wikipedie. *DMX512 — Wikipedie: Otevřená encyklopedie* [online]. 2020 [cit. 2020-04-27]. Dostupné z: https://en.wikipedia.org/wiki/DMX512.
9. Staněk, Jan; Řehák, Jan. *RS 485 422* [online]. 1998 [cit. 2020-04-27]. Dostupné z: https://vyvoj.hw.cz//teorie-a-praxe/dokumentace/rs-485-422.html.
10. Nušl, Jaroslav. *Protokol DMX512* [online] [cit. 2020-04-27]. Dostupné z: http://www.soh.cz/podpora/teorie.
11. [online] [cit. 2020-05-15]. Dostupné z: https://www.thomann.de/cz/index.html.
12. *Lenna* [online] [cit. 2020-05-19]. Dostupné z: http://www.lenna.org/full/len_full.html.

13. Klíma, Miloš; Bernas, Martin; Hozman, Jiří; Dvořák, Pavel. *Zpracování obrazové informace*. ČVUT, 1999.

14. Hájovský, Radovan; Pustková, Radka; Kutálek, František. *Zpracování obrazu v měřicí a řídicí technice*. Ediční středisko VŠB – TUO, 2012.

15. Hlaváč, Václav. *Matematická morfologie* [online] [cit. 2020-05-05]. Dostupné z: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/71-3MatMorpholBinCz.pdf.

16. Wikipedie. *Barevný model — Wikipedie: Otevřená encyklopedie*. 2020. Dostupné také z: https://cs.wikipedia.org/wiki/Barevn%C3%BD_model.

17. [online] [cit. 2020-05-06]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=772.

18. Wikipedie. *RGB — Wikipedie: Otevřená encyklopedie*. 2019. Dostupné také z: https://cs.wikipedia.org/w/index.php?title=RGB&oldid=17919929.

19. Wikipedie. *HSV — Wikipedie: Otevřená encyklopedie*. 2019. Dostupné také z: https://cs.wikipedia.org/wiki/HSV.

20. OpenCV. *Color conversions* [online] [cit. 2020-05-06]. Dostupné z: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html.

21. Digilent. *Zybo Z7 Pcam 5C Demo* [online] [cit. 2020-04-28]. Dostupné z: https://reference.digilentinc.com/learn/programmable-logic/tutorials/zybo-z7-pcam-5c-demo/start.