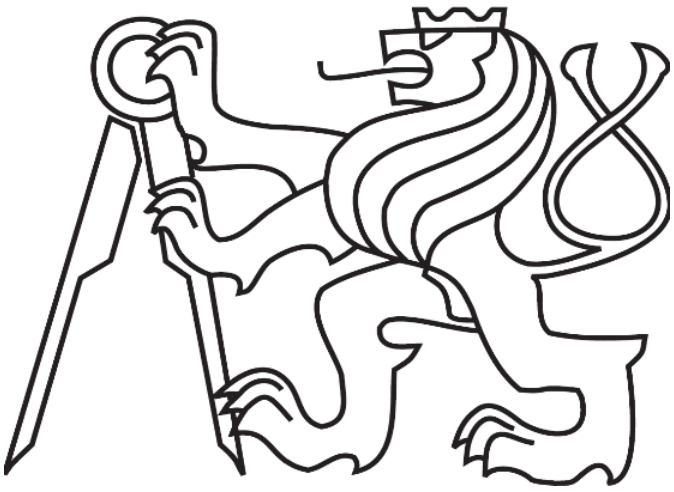


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra mikroelektroniky

Systém pro záznam a zpracování videa využívající systém na čipu

Video signal processing using the system on chip

Diplomová práce

Studijní program: Elektronika a komunikace.
Studijní obor: Elektronika.

Vedoucí práce: prof. Ing. Pavel Hazdra, CSc.

Bc. David Kazák

Praha 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kazák** Jméno: **David** Osobní číslo: **434765**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Elektronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Systém pro záznam a zpracování videa využívající systém na čipu

Název diplomové práce anglicky:

Video signal processing using the system on chip

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 způsoby kódování, přenosu a záznamu videa.
3) Navrhněte a realizujte systém využívající SoC Zynq, který umožní zpracování (např. vyvážení obrazu, doplnění značek, apod.) a záznam videa na SD kartě v reálném čase. 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] P. Wilson, Design Recipes for FPGAs, Elsevier, 2015.

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: **11.02.2020** 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

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 dne 13. 08. 2020

……..…….
Podpis autora

Poděkování

Rád bych tímto poděkoval svému vedoucímu, panu prof. Ing. Pavlu Hazdrovi CSc., za zapůjčení vývojové desky Zybo Z7-20, za čas a rady, které mi při vypracování práce věnoval. Rodině, přátelům a všem kteří mě při tvorbě této práce podporovali.

Abstrakt

Tato diplomová práce se zabývá zpracováním a záznamem video signálu v reálném čase s využitím systémů na čipu Xilinx Zynq 7000. Cílem je vytvořit model záznamového zařízení, který bude zaznamenávat video signál z počítače s rozhraním HDMI.

Abstract

This master thesis is focused on video signal processing and capturing in real time using system on chip Xilinx Zynq 7000. The goal is to develop model of capturing device, which will be able to capture video signal from computer with HDMI interface.

Klíčová slova

Zpracování videosignálu, záznam videosignálu, HDMI, Matroska, mkv, EBML, systémy na čipu, Zynq 7000, Zybo Z7-20.

Key words

Video signal processing, video signal capturing, HDMI, Matroska, mkv, EBML, system on chip, Zynq 7000, Zybo Z7-20.

Obsah

Seznam obrázků 10

Seznam tabulek 12

Seznam použitých symbolů 12

Seznam zkratek 13

1 Úvod 15

2 Programovatelné logické obvody 17

2.1 Historie programovatelných logických obvodů 17

2.2 Programovatelné hradlové pole 19

2.3 Systémy na čipu 20

2.3.1 Xilinx Zynq 7000 22

3 Vivado Design Suite 27

4 Přenos a zpracování videosignálu 31

4.1 Barevné prostory 31

4.1.1 RGB 31

4.1.2 YUV a YCbCr 32

4.1.3 Barevné podvzorkování 33

4.2 HDMI 34

4.3 Kódování videosignálů 35

4.4 Datový kontejner Matroska 37

4.5 Současné možnosti záznamu videosignálu z počítače 39

5 Návrh a realizace modelu 41

5.1 Vývojová deska Zybo Z7-20 42

5.2 Blok zpracovávající videosignál v reálném čase 44

5.3 Záznamový systém 50

5.4 Zdrojové kódy 54

6 Dosažené výsledky 57

6.1 Návrh dalšího postupu práce. 62

7 Závěr 63

8 Literatura 64

Seznam příloh 66

Seznam obrázků

Obrázek 1: Programmable Logic Array – PLA – Signetics – 1975 [1]. 17

Obrázek 2: Programmable Array Logic – PAL – MMI – 1978 [1]. 18

Obrázek 3: Architektura komplexních programovatelných logický obvodů – CPLD [1]. 18

Obrázek 4: Architektura programovatelných hradlových polí – FPGA – Xilinx – 1985 [1]... 19

Obrázek 5: Typy logických buněk v FPGA. Vlevo logická buňka typu SRAM, vpravo jednorázově programovaná buňka [1]. 20

Obrázek 6: Jednoduché porovnání SoB a SoC. Vlevo systém na desce plošných spojů, vpravo systém na čipu [3]. 21

Obrázek 7: Zjednodušený model SoC Xilinx Zynq 7000 [3]. 21

Obrázek 8: Zjednodušené blokové schéma APU – ARM Cortex-A9 [3]. 23

Obrázek 9: Schéma Xilinx Zynq 7000 [4]. 24

Obrázek 10: Postup návrhu ve Vivado Design Suite [6]. 27

Obrázek 11: Příklad blokového designu. Blokový design obsahuje PS Zynq 7020, resetovací systém procesoru, propojení AXI 3 (PS) a AXI 4 (IP) a AXI BRAM kontrolér. Součást modelu vytvořeného v kapitole 5. 28

Obrázek 12: Barevný prostor RGB [13]. 31

Obrázek 13: Běžné typy barevného podvzorkování [14]. 33

Obrázek 14: Rozhraní HDMI – blokové schéma přenosu [15]. 34

Obrázek 15: TMDS režimy pro jeden snímek v rozlišení 720 x 480 px [15]. 35

Obrázek 16: Hybridní video kodér [16]. 36

Obrázek 17: Vnoření EBML prvků. 37

Obrázek 18: Zjednodušené blokové schéma navrženého modelu záznamového zařízení. 41

Obrázek 19: Digilent Zybo Z7-20 [24].. 43

Obrázek 20: Blokové schéma systému přidávání objektů – Object Adder System. Zeleně jsou označené vstupy, oranžově výstupy. 45

Obrázek 21: Aplikace Pixel Masking. 46

Obrázek 22: White Balancer System – blokové schéma. 48

Obrázek 23: Recorder System – zjednodušené blokové schéma. Zeleně jsou označeny vstupy a oranžově jsou označeny výstupy. 52

Obrázek 24: Ukázka záznamu s vloženým logem na bílém pozadí. S převodem bílého pozadí na transparentní. 57

Obrázek 25: Příklad možností změny vyvážení bílé. Vpravo obraz posunutý k teplejším barvám. Vlevo ke studenějším. 58

Obrázek 26: Ukázka problémů s rychle pohybujícími se objekty při prokládání. 59

Obrázek 27: Vlevo zdrojový obrázek v rozlišení 1280 x 720 px. Svislé pruhy 4x široké. Vpravo záznam tohoto obrázku, se slitými barvami vlivem snížení rozlišení a barevným pod vzorkováním 4:1:1. Obrázek je v plném rozlišení v přílohách. 60

Obrázek 28: Rozlití barev při barevné pod vzorkování. 61

Obrázek 29: Porovnání textů na různých barvách pozadí. Plné rozlišení v přílohách. 61

Seznam tabulek

Tabulka 1: Parametry vybraných programovatelných logických polí v Zynq 7000 [5]. 22

Tabulka 2: Zápis velikosti EBML elementu. 38

Tabulka 3: Tabulka Top-Level elementů [20]. 38

Tabulka 4: Technické parametry Xilinx Zynq-7020 [4, 5]. 42

Tabulka 5: Technické parametry vývojové desky Digilent Zybo Z7-20 [24]..... 44

Tabulka 6: Seznam popisovaných bloků a k nim přiřazených zdrojových souborů. 55

Seznam použitých symbolů

B	(-)	Hodnota modrého kanálu.
B_M	(-)	Modifikovaná hodnota modrého kanálu.
BpP	(B)	Bytes per Pixel – Počet bajtů na jeden pixel.
BW	($B \cdot s^{-1}$)	Datový tok video signálu.
f_s	(Hz)	Snímková frekvence.
f_p	(Hz)	Pixelová frekvence.
G	(-)	Hodnota zeleného kanálu.
H	(px)	Výška obrazu v pixelech.
H_b	(px)	Výška vertikální zatmívané oblasti v pixelech.
k_B	(-)	Násobící koeficient pro modrou barvu.
k_R	(-)	Násobící koeficient pro červenou barvu.
R	(-)	Hodnota červeného kanálu.
R_M	(-)	Modifikovaná hodnota červeného kanálu.
U	(-)	Chromatická složka signálu.
V	(-)	Chromatická složka signálu.
W	(px)	Šířka obrazu v pixelech.
W_b	(px)	Šířka horizontálně zatmívané oblasti v pixelech.
Y	(-)	Jasová složka signálu.

Seznam zkratek

APU	Application Processor Unit – Aplikační procesorová jednotka.
ARM	Advanced RISC Machine – Architektura procesorů a obchodní značka společnosti Arm Holdings.
ASIC	Application Specific Integrated Circuit – Aplikačně specifický integrovaný obvod.
ASSP	Application Specific Standard Part – Aplikačně specifický standardní integrovaný obvod.
AVC	Advanced Video Coding – Pokročilé video kódování. Standard podle ISO a ITU.
CABAC	Context-Adaptive Binary Arithmetic Coding.
CPLD	Complex Programmble Logic Device – Komplexní programovatelný logický obvod.
CPU	Central Processing Unit – Centrální procesorová jednotka.
DPS	Deska plošných spojů.
DVI	Digital Visual Interface – Rozhraní pro přenos video signálu.
EEPROM	Electrically Erasable Programmable Read-Only Memory – Elektronicky vymazatelná paměť pouze pro čtení.
EBML	Extensible Binary Meta Language.
FPGA	Field Programmable Gate Array – Programovatelné hradlové pole.
FPU	Floating Point Unit – Jednotka pro výpočty s plovoucí desetinnou čárkou.
HDL	Hardware Description Language – Jazyky pro popis hardware.
HDMI	High-Defininition Multimedia Interface – Rozhraní pro přenos video signálu.
HLS	High Level Synthesis – Vysoká úroveň syntézy.
IP	Intellectual Property.
LUT	Look-up Table.
MIO	Multiplexed Input/Output – Multiplexované vstupy a výstupy.
MMI	Monolithic Memories, Inc – Výrobce integrovaných obvodů.
MMU	Memory Management Unit – Jednotka správy paměti.
PAL	Programmable Array Logic – Programovatelné pole logických hradel.
PL	Programmable Logic – Programovatelné logické pole v Zynq 7000 ekvivalentní k FPGA Artix 7.
PLA	Programmable Logic Array – Programovatelné logické pole.
PS	Processing System – Procesorová část čipu Zynq 7000.
OCM	On-Chip Memory – SRAM paměť integrovaná v SoC Zynq 7000.
RISC	Reduced Instruction Set Computer – Počítač (Procesor) s redukovanou instrukční sadou.
SIMD	Single Instruction Multiple Data.
SCU	Snoop Control Unit.
SoB	System on Board – Systém na desce plošných spojů.
SoC	System on Chip – Systém na čipu.
SPLD	Simple Programmable Logic Device – Jednoduchý programovatelný logický obvod.
TMDS	Transition-minimized differential signaling.
UEFI	Unified Extensible Firmware Interface.
VHDL	VHSIC-HDL – Very High Speed Integrated Circuit – Hardware Description Language – Jazyk pro popis velmi rychlých integrovaných obvodů.

1 Úvod

Cílem této diplomové práce je demonstrovat možnosti systémů na čipu Xilinx Zynq 7000, při zpracování video signálů v reálném čase. Výsledkem bude model záznamového zařízení, který bude schopný zpracovávat videosignál v rozlišení 1280 x 720 px při snímkové frekvenci 60 Hz a zároveň ukládat tento videosignál na microSD kartu ve formátu přehratelném v běžném počítači. Model bude dále podporovat vkládání dvou obrázků, uložených v SoC, do obrazu. Systém bude také umožňovat korekci barev. Zdroj videosignálu bude počítač s rozhraním HDMI. Model bude realizovaný na vývojové platformě Digilent Zybo Z7-20.

Model bude zacílen především na výuková videa a aplikace, u kterých není možné používat softwarový záznam videosignálu. Praktickým příkladem může být naučné video o konfiguraci UEFI počítače nebo serveru. Dalším možným využitím jsou aplikace, kde softwarový záznam může negativně ovlivnit celkový výkon aplikace.

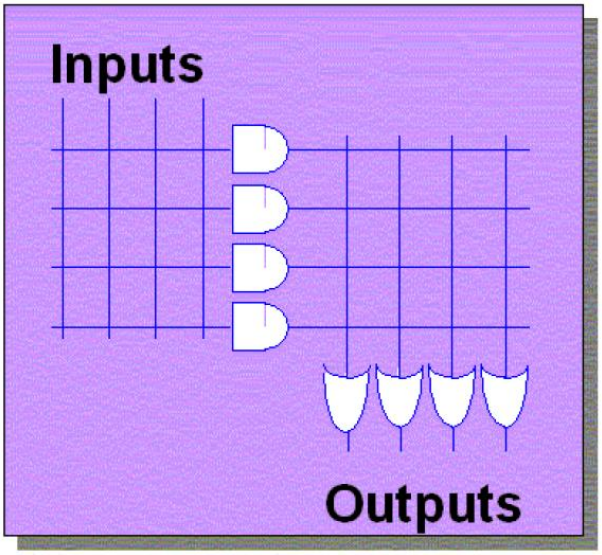
První část práce představuje základní pohled na problematiku systému na čipu a záznamu videosignálů. Druhá část se zabývá návrhem, realizací a testováním modelu záznamového zařízení.

2 Programovatelné logické obvody

Tato kapitola má za cíl shrnout vývoj programovatelných logických obvodů, od jejich počátků v sedmdesátých letech 20. století až po systémy na čipu typu Xilinx Zynq.

2.1 Historie programovatelných logických obvodů

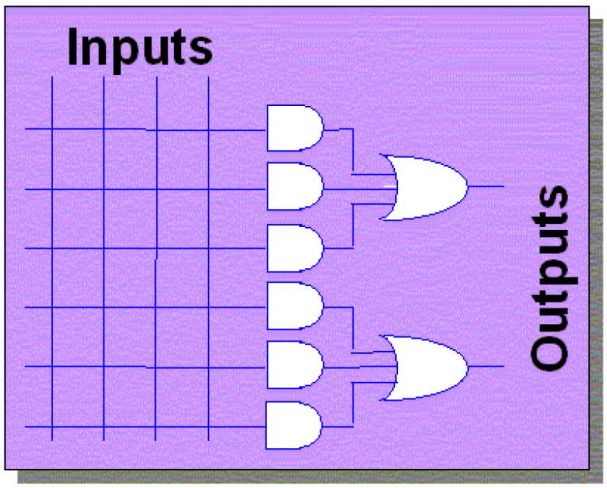
Typický logický obvod začátku sedmdesátých let minulého století byla deska plošných spojů s diskrétními logickými členy. V té době se začala rozvíjet myšlenka jednoho většího zařízení, které by v sobě kombinovalo větší množství logických členů AND a OR, které by návrhářům umožnilo integrovat velké AND-OR struktury do jednoho čipu. Tento přístup zkracuje dobu vývoje, umožňuje menší zařízení, protože integrovaný logický obvod zabírá výrazně menší plochu na desce plošných spojů, než když je složen z diskrétních logických členů AND a OR. Mezi další výhody patří větší flexibilita vývoje, nižší spotřeba a vyšší rychlost [1].



Obrázek 1: Programmable Logic Array – PLA – Signetics – 1975 [1].

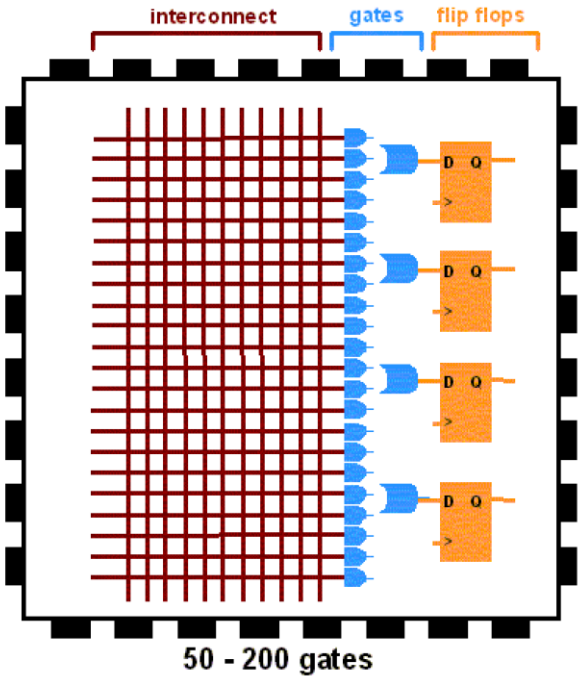
V roce 1975 vzniklo, ve společnosti Signetics, první programovatelné logické pole, nebo-li PLA, které umožňovalo libovolnou kombinaci členů AND a OR. Programovatelné byly obě propojovací pole, které jsou znázorněna na obrázku 1. Další architekturou byly PAL, nebo-li programovatelné pole logických hradel, která mají na rozdíl od PLA programovatelné jen propojovací pole na straně AND členů a propojení OR členů je fixní, jak je vidět na obrázku 2. Z tohoto důvodu nabízely PAL menší množství kombinací zapojení AND-OR než PLA. Toto omezení bylo vykompenzováno vyšší rychlostí těchto obvodů. PAL a PLA, souhrnně někdy

označované jako SPLD – jednoduchý programovatelný logický obvod, byly vhodné pro menší specifické digitální obvody. [1, 2].



Obrázek 2: Programmable Array Logic – PAL – MMI – 1978 [1].

Další vývojovým krokem byly komplexní programovatelné obvody – CPLD, jejichž myšlenka tkví ve spojení několika SPLD nebo jiných makro buněk do jednoho komplexnějšího obvodu. Hlavní výhody, podobně jako u předchozích obvodů, byly zjednodušení, zrychlení a zlevnění návrhu obvodů [1, 2].



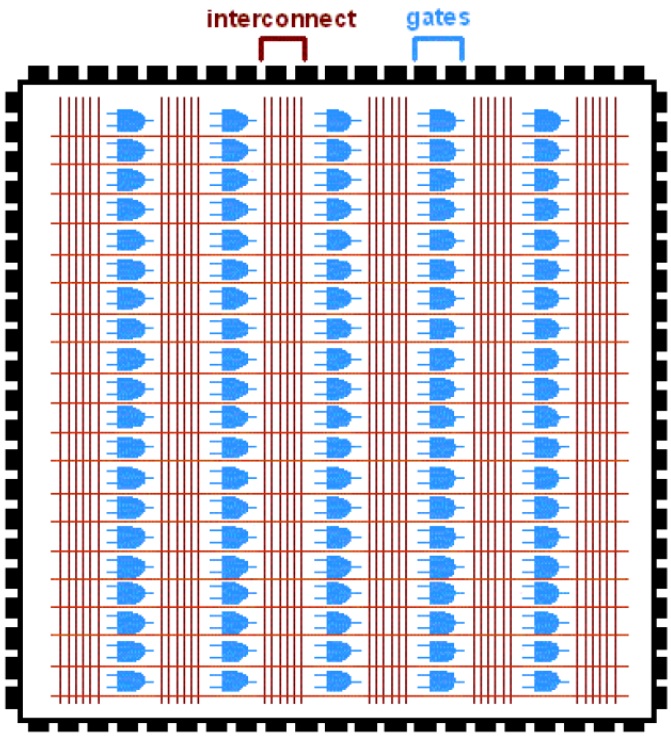
Obrázek 3: Architektura komplexních programovatelných logický obvodů – CPLD [1].

Zapojení CPLD je vidět na obrázku 3, obsahuje jedno velké centrální propojovací pole, pomocí kterého se propojují vstupy obvodů a logické bloky mezi sebou v případě návrhu většího logického obvodu [1].

CPLD obvody jsou programovatelné pomocí HDL, které umožňují snadný vývoj a simulaci číslicových obvodů, to umožňuje snížit počet potřebných prototypů a snížit tím dobu vývoje a náklady na systém [1].

2.2 Programovatelné hradlové pole

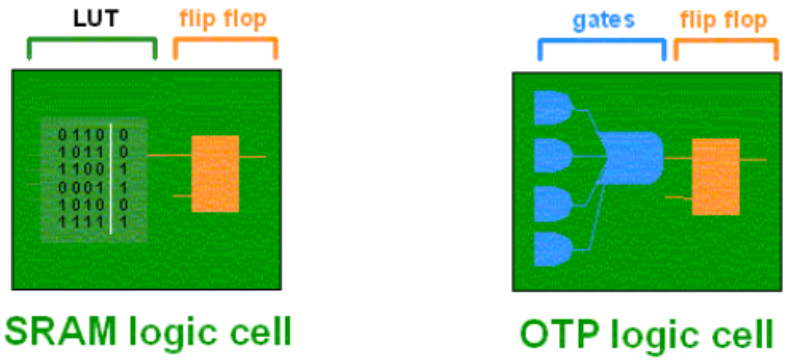
Jedna z nevýhod CPLD je nižší hustota integrace oproti hradlovým polím, která nejsou flexibilně programovatelné. Jejich zapojení je dáno z výroby specifickou vrstvou kovových propojek. Výhody obou řešení kombinují programovatelné hradlové pole, nebo-li FPGA. První FPGA vyrobila již v roce 1985 společnost Xilinx [1].



Obrázek 4: Architektura programovatelných hradlových polí – FPGA – Xilinx – 1985 [1].

FPGA mají místo centrálního propojovacího pole, kanály, sloupce nebo řádky pomocích, kterých se propojují logické buňky. Buňky se rozdělují na dva základní typy. Na jednorázově programovatelné buňky, které jsou tvořeny hradly a klopnými obvody. Druhý, běžnější typ, jsou buňky na bázi SRAM, která nepoužívají klasické logické členy, ale Look-Up Table, což

jsou uživatelky nastavitelné pravdivostní tabulky, které definovaným vstupům přiřazují definované výstupy [1, 2].



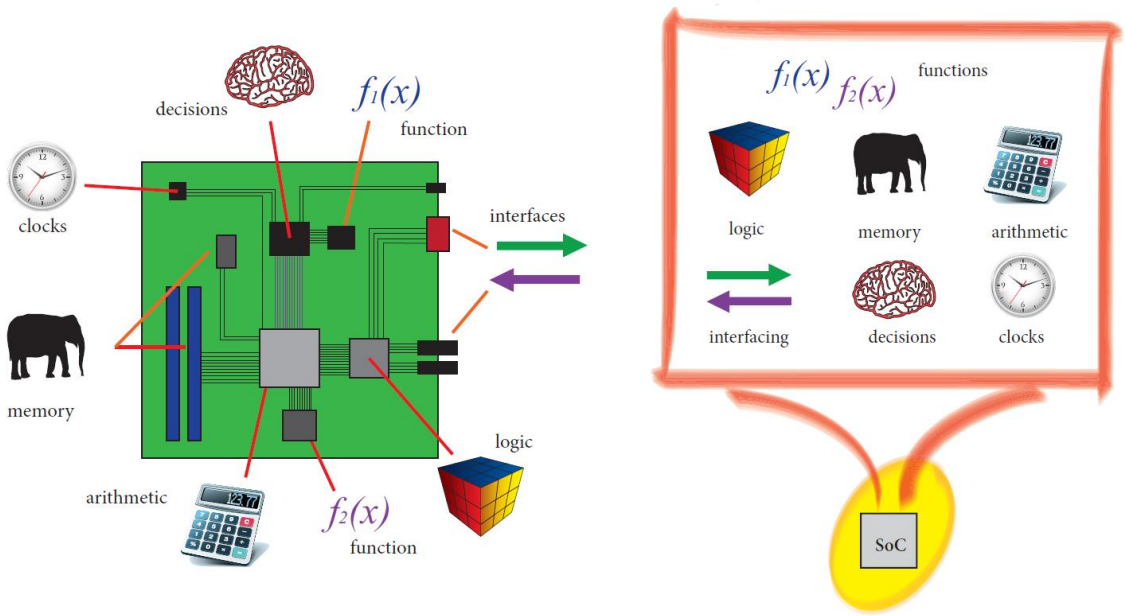
Obrázek 5: Typy logických buněk v FPGA. Vlevo logická buňka typu SRAM, vpravo jednorázově programovaná buňka [1].

FPGA na bázi SRAM logických buněk se musí programovat po každém připojení k napájecímu napětí z připojené EEPROM nebo jiné nevolatilní paměti [1].

Současná programovatelná hradlová pole mají tisíce až statisíce logický buněk, obsahují také další integrované speciální obvody jako například digitální signálové procesory nebo blokové paměti. Lze s nimi vytvořit velmi komplexní systémy, které dokáží paralelně zpracovat velké množství informací rychlostmi v řádech gigabitů za sekundu [1, 2, 3]. Pro komplexnost a výkon, se FPGA stávají také základem systémů na čipu.

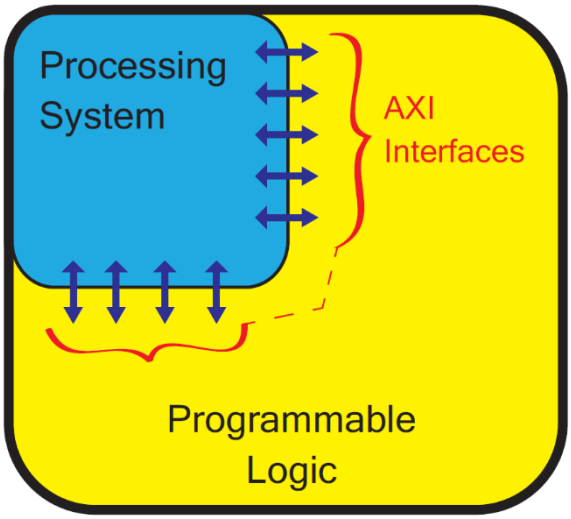
2.3 Systémy na čipu

Systémy na čipu integrují kompletní funkčnost obvodu nebo celého systému na jeden čip. Často se spojují systémy na čipu s realizací pomocí aplikačně specifických obvodů, nebo-li ASIC, které v sobě integrují všechny potřebné komponenty dané aplikace. Systémy na čipu jsou rychlejší, energeticky úspornější a menší než srovnatelné systémy na desce plošných spojů (SoB) [3].



Obrázek 6: Jednoduché porovnání SoB a SoC. Vlevo systém na desce plošných spojů, vpravo systém na čipu [3].

Nevýhodu ASIC jsou vysoké náklady na vývoj, a to jak finanční, tak i časové. Vyplatí se je tedy vyrábět jen ve velkých sériích. To znamená, že systém na čipu založené na ASIC nejsou vhodné pro celou řadu aplikací, které by jinak mohly využívat benefitů systémů na čipu. V těchto oblastech se dlouhou dobu používají FPGA, která jsou řádově flexibilnější než aplikačně specifické obvody [3].



Obrázek 7: Zjednodušený model SoC Xilinx Zynq 7000 [3].

V systémech na čipu založených na FPGA je obtížné realizovat procesor s vysokým výkonem. Dostupné soft[1] procesory mají obvykle velmi nízký výpočetní výkon, v porovnání s hard[2] procesory typu ARM Cortex-A9 a podobnými, které mohou být integrované v ASIC. Tuto nevýhodu řeší řada systémů na čipů Xilinx Zynq 7000, která integruje programovatelné logické pole, odvozené od FPGA Xilinx řad Artix 7 nebo Kirtex 7 s jednojádrovým nebo dvoujádrovým procesorem ARM Cortex-A9 [3, 4]. Více o systémech na čipu řady Zynq 7000 v následující kapitole.

2.3.1 Xilinx Zynq 7000

Řada systémů na čipu Xilinx Zynq 7000 v sobě kombinuje flexibilitu a škálovatelnost FPGA systémů s výkonem typickým pro ASIC a ASSP[3]. Mají k dispozici jednojádrový nebo dvoujádrový 32 bitový procesor ARM Cortex-A9 na frekvenci 667 MHz až 1 GHz, který se společně se svými periferiemi nazývá Processing System – PS a programovatelné logické pole – PL, které je ekvivalentní k FPGA Xilinx Artix 7 nebo Kirtex 7 [4]. Možné parametry PL v tabulce 1.

Tabulka 1: Parametry vybraných programovatelných logických polí v Zynq 7000 [5].

Název	Z-7007S	Z-7010	Z-7020	Z-7030	Z-7100
Programovatelné logické buňky	23000	28000	85000	125000	444000
Look-Up tabulky	14400	17600	53200	78600	277400
Klopné obvody	28800	35200	106400	157200	554800
Bloková RAM (36 kb bloky)	1,9 Mb (50)	2,1 Mb (60)	4,9 Mb (140)	9.3 Mb (265)	26,5 Mb (755)
Digitální signálové procesory – DSP	66	80	220	400	2020

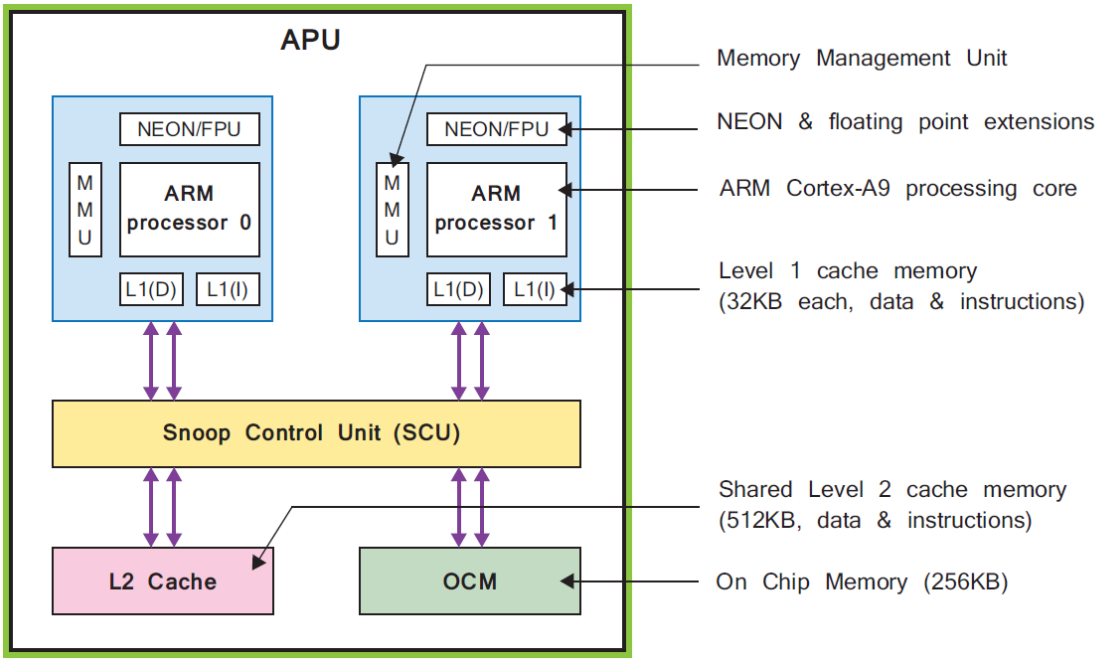
Processing System Zynq obsahuje kromě APU – Aplikační procesorové jednotky, která obsahuje samotný procesor ARM Cortex-A9, také celou řadu periferii, které můžete vidět na obrázku 9. Mezi ně patří paměťové kontroléry pro DDR2, LPDDR2, DDR3 a DDR3L s podporou až 1 GB paměti a pro připojení nevolatilních pamětí typu SRAM/NOR, NAND a

[1] Soft procesory jsou procesory, které jsou distribuované jako IP jádra pro FPGA. Jedná se o konfigurovatelné procesory Xilinx MicroBlaze, ARM Cortex-M1 a další podobné procesory

[2] Hard procesory jsou procesory, které jsou v daném čipu vytvořené přímo ve výrobě. Příkladem může být ARM Cortex-A9 v čipech řady Xilinx Zynq 7000 zmíněné v následující kapitole.

[3] ASSP – Application Specific Standard Part – Aplikačně specifické standardní integrované obvody jsou podmnožinou ASIC. Postup návrhu je stejný jako ASIC ale jedná se o univerzálnější integrované obvody, které jsou využitelné v širším spektru aplikací.

Q-SPI. Dále 2x USB, 2x Ethernet (1000BASE-T – IEEE 802.3ab), 2x SD SDIO, GPIO, 2x UART, 2x CAN, 2x I^2C, 2x SPI vyvedené pomocí 54 multiplexovaných vstupů a výstupů – MIO a i v neposlední řadě AXI sběrnice pro připojení periferii v PL [4].



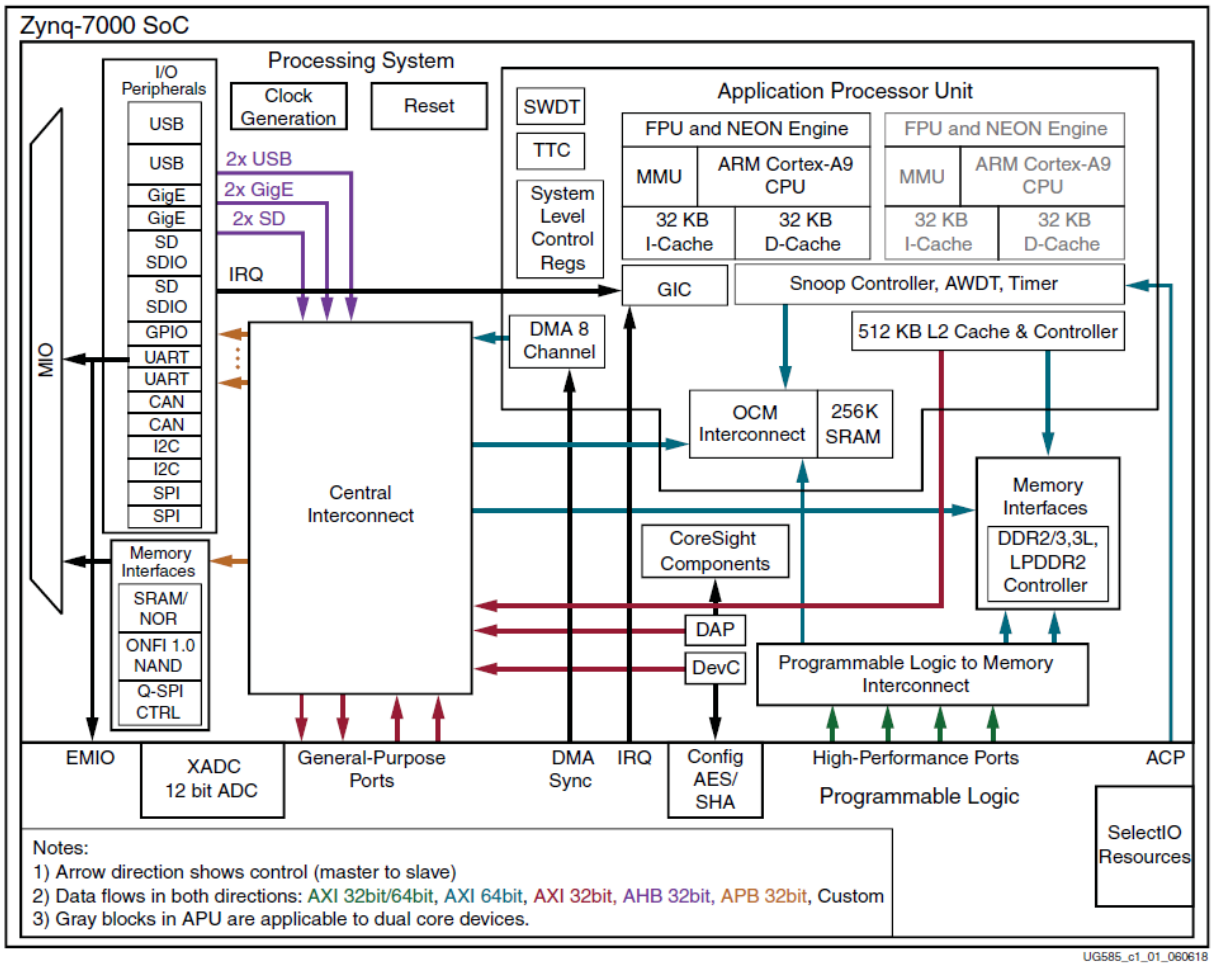
Obrázek 8: Zjednodušené blokové schéma APU – ARM Cortex-A9 [3].

Blokové schéma APU je na obrázku 8. APU obsahuje dvě jádra ARM Cortex-A9, každé z nich má 32 kB L1 mezipaměti pro data a 32 kB L1 mezipaměti pro instrukce, MMU, jejichž primární činnost je zajištění překladu virtuálních adres na fyzické při využití Linuxových operačních systémů. Dále má každé jádro svoji jednotku na výpočty s plovoucí desetinnou čárkou – FPU a 128 bitovým NEON Media Processing Engine, které zajišťují výpočty typu SIMD – Single Instruction Multiple Data pro algoritmy používané při zpracování médií a digitálním zpracování signálů [3, 4].

Komponenty sdílené mezi jádry řídí SCU – Snoop Control Unit, která zajišťuje propojení ARM jader s 512 kB sdílené L2 mezipaměti a 256 kB SRAM paměti (OCM), které jsou integrované přímo v Zynq a zároveň umožňuje přístup do OCM z programovatelného logického pole přes sběrnici ACP [3, 4].

Využití systémů na čipu Zynq 7000 je velmi široké například ve zpracování videa, v rozpoznávání objektů – strojové vidění, v softwarově definovaném radiu a v dalších výpočetně náročných aplikacích. V každém designu je potřeba individuálně zvážit, na jakou platformu

aplikaci cílit. Roli hrají celkové náklady na vývoj a materiál, požadavky na výpočetní výkon, paměti, škálovatelnost, flexibilitu a další. V následujících odstavcích shrnu základní porovnání mezi systém na čipu Zynq, FPGA a procesory [3].



Obrázek 9: Schéma Xilinx Zynq 7000 [4].

Hlavní rozdíl mezi Zynq 7000 a ARM Cortex-A9, nebo podobnými procesory, je v programovatelném poli, ve kterém je možné realizovat periferie, které daný procesor nemá integrované. Je také možné některé úlohy, které je možné paralelizovat, přesunout do programovatelného pole a ušetřit tak čas procesoru [3].

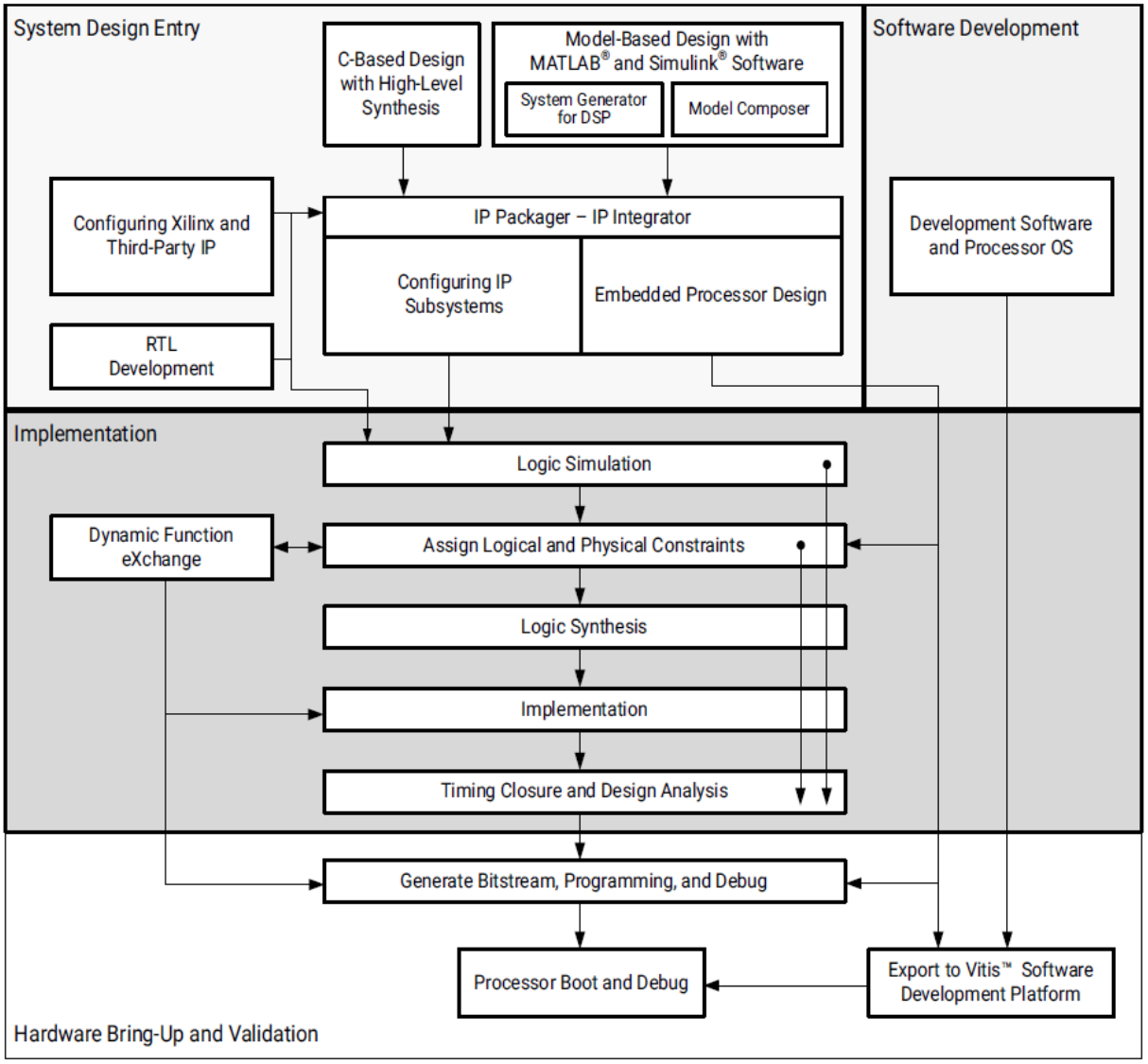
Výhoda SoC Zynq nad srovnatelnými FPGA je v integrovaném ARM Cortex-A9 procesoru, který je přibližně desetkrát výkonnější než soft procesory, které je možné realizovat v FPGA a i v PL části SoC Zynq [3].

Diskrétní spojení FPGA a procesoru, ve formě systému na desce plošných spojů, umožňuje rozdělení úloh mezi procesor a FPGA podle potřeb jako u Zynq. Nevýhodou je větší plocha na

DPS dva čipy oproti jednomu. Zároveň může být design limitovaný omezenou propustností sběrnice mezi procesorem a FPGA ve srovnání s vysokou propustností sběrnic AXI v SoC Zynq [3].

3 Vivado Design Suite

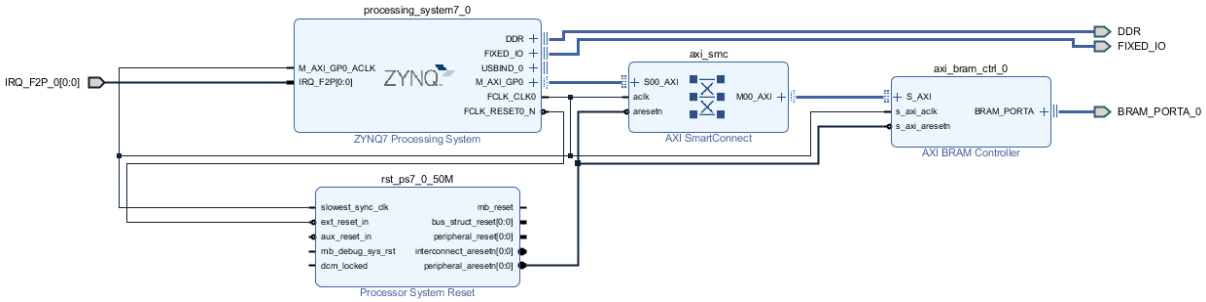
Tato kapitola shrnuje vývojářské nástroje pro vývoj na FPGA a SoC Xilinx. První část této kapitoly se zabývá možnostmi vývojářské sady Vivado Design Suite. Druhá část se se zaměřuje na použitím IP jader a znovu využitelnost kódu při vývoji.



Obrázek 10: Postup návrhu ve Vivado Design Suite [6].

Xilinx pro podporu vývoje na svých FPGA a SoC dodává Xilinx Vivado Design Suite, v základní verzi zdarma s možností koupit plné verze s pokročilejší funkcionalitou. Vivado Design Suite obsahuje tři základní nástroje. První nástroj je Xilinx Vivado, který slouží k syntéze digitálních obvodu vytvořených pomocí VHDL nebo Verilog, integraci IP jader, simulaci designů a generování bitstreamu pro naprogramování FPGA nebo PL části SoC.

Vivado také generuje hardwarovou platformu pro Xilinx SDK. Druhý nástroj v balíků Vivado Design Suite Xilinx SDK slouží k tvorbě firmwaru a softwaru pro ARM procesory a pro soft procesory Xilinx MicroBlaze v jazycích C a C++. SDK, podporuje vývoj pro Linux, FreeRTOS nebo samostatné aplikace. Pro oba operační systémy, a i pro samostatné aplikace, jsou dostupné board support packages, ovladače a knihovny zajišťující základní práci s hardwarem. SDK dále podporuje nástroje pro ladění kódu [3, 6].



Obrázek 11: Příklad blokového designu. Blokový design obsahuje PS Zynq 7020, resetovací systém procesoru, propojení AXI 3 (PS) a AXI 4 (IP) a AXI BRAM kontrolér. Součást modelu vytvořeného v kapitole 5.

Třetím nástrojem ve Vivado Design Suite je Xilinx Vivado HLS[4]. Vivado HLS je vývojové prostředí pro vysokou úroveň syntézy, podporuje syntézu do RTL z jazyků C, C++ a System C. To umožňuje implementovat velmi pokročilé algoritmy, snadnější přenositelnost mezi cílovými zařízeními. Vivado HLS umožňuje testovat vytvořené návrhy pomocí testů psaných v jazycích C a C++, to ještě před syntézou do RTL, a tím zásadně snížit čas, který je nutný věnovat verifikaci designu. Podle [7] může být vývoj ve Vivado HLS až 4 krát rychlejší, než kdyby byl vyvíjen běžným postupem ve Vivado. Typicky se ve Vivadu HLS vyvíjí bloky designu, které mají složitější algoritmy. Z těchto bloků se generují IP jádra, které se se zbytkem designu integrují ve Vivado [3, 6, 7, 8].

IP jádra umožňují znovu použít jednou syntetizovaný a verifikovaný blok v jiném projektu a tím ušetřit čas, který by byl potřebný na opětovný vývoj existujícího prvku. IP jádra je také možné získat z Vivado IP Catalog, který je distribuovaný společně s Vivado Design Suite. Tento repositář obsahuje základní sadu IP jader, například pro blokové paměti, jádro pro využití integrovaných DSP a mnoho dalších. Dále je možné využít IP jádra třetích stran, které mohou být buď volně dostupné, příkladem mohou být IP jádra, které dodává Digilent ke svým vývojovým kitům, nebo licencovaná. IP jádra, která jsou distribuována jako soft jádra, jež

[4] Vivado a Vivado HLS jsou dvě samostatné aplikace z Vivado Design Suite.

možné konfigurovat podle potřeb a můžou být k nim dostupné i zdrojové kódy. Dále existují i hard IP jádra, které není možné modifikovat a jsou dodavatelem obvykle i enkryptovány, typicky se takto distribuují licencovaná jádra [3, 6, 9].

Blokové designy umožňují spojit více IP jader do jednoho bloku, kterému se vytvoří HDL obálka[5], pomocí které je možné vložit blokový design do RTL projektu. Blokové designy se vytváří pomocí grafického uživatelského prostředí a dostupných repositářů IP jader. Přiklad využití je snadná integrace PS části Zynq do designu, jak je znárorněno na obrázku 11. Pomocí blokového designu je možné velmi rychle přidat všechny potřebné AXI periferie a vygenerovat HDL obal, ke kterému je možné přidat další bloky vytvořené v HDL, další IP jádra a blokové designy [6, 7, 9].

[5] HDL Wrapper.

4 Přenos a zpracování videosignálu

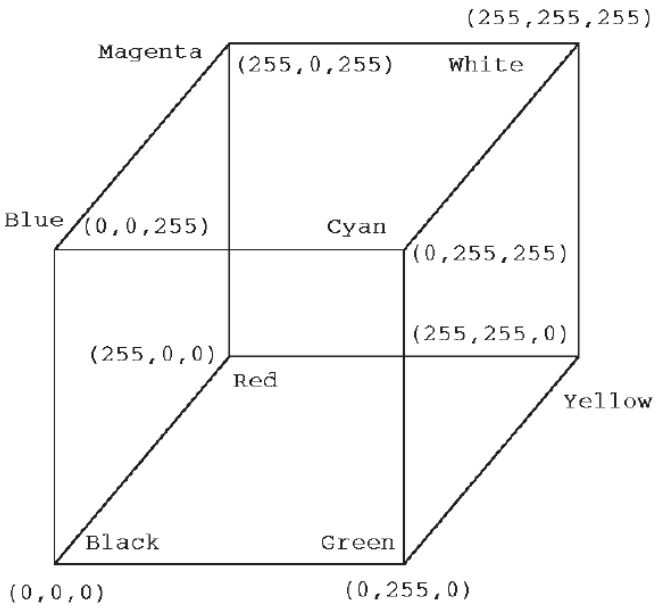
Tato kapitola si klade za cíl shrnout základy přenosu a zpracování videosignálu. V první části jsou shrnuty barevné prostory pro reprezentaci barev v digitálních systémech. V druhé části je popsán přenos signálu přes rozhraní HDMI. Ve třetí části je úvod do kódování videosignálů. Ve čtvrté části je popsán datový kontejner Matroska, který je jedním z kontejnerů, do kterých je možné ukládat zaznamenané videosignály. V poslední části této kapitoly jsou shrnuty současné možnosti záznamu videosignálu.

4.1 Barevné prostory

Barevný prostor je obvykle tří dimenzionální prostor, ve kterém se dají vyjádřit pro lidské oko viditelné barvy. Všechny barevné prostory vychází z vlastností lidského zraku a snaží se o co nejvěrnější vyjádření barev, které je nezávislé na zobrazovacím zařízení [10, 11, 12].

4.1.1 RGB

Nejběžnější barevný prostor je RGB – červená, zelená a modrá [10, 11]. Založení barevného prostoru na těchto barvách má základ ve třech typech čípků v lidském oku, které jsou nejvíce citlivé na vlnové délky odpovídající přibližně těmto barvám [11, 12].



Obrázek 12: Barevný prostor RGB [13].

Z barev RGB je možné namíchat libovolnou barvu a většina výpočetní techniky zobrazuje v tomto barevném prostoru, i kdyby se signál přenášel a zpracovával v jiném barevném prostoru [10, 11]. Jedna z nejběžnějších variant je RGB24bpp[6], která barvy reprezentuje 24 bity na pixel, tzn. 8 bitů pro každou barvu. V některých případech se používá ještě alfa kanál označený písmenem A, který nese informaci o transparentnosti [11].

Většina barevných systémů se odvozuje od RGB. Například barevný prostor CMY[7] – tyrkysová, purpurová a žlutá, který je na rozdíl od aditivního RGB, subtraktivní barevný prostor, je možné transformovat pomocí rovnice (4.1) [12].

$$[R \quad G \quad B] = [1 \quad 1 \quad 1] - [C \quad M \quad Y] \tag{4.1}$$

4.1.2 YUV a YCbCr

Jak již bylo zmíněno RGB využívá toho, že lidské čípky jsou nejcitlivější na vlnové délky přibližně odpovídající červené, zelené a modré barvě. Oproti tomu barevné prostory YUV a YCbCr využívají toho, že je lidský zrak citlivější na informaci o jasu, na monochromatické vidění tyčinek, než na barevném vidění pomocí čípků [11].

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}597 & 0{,}114 \\ -0{,}147 & -0{,}289 & 0{,}436 \\ 0{,}613 & -0{,}515 & 0{,}100 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.2}$$

Systémy pochází z barevné analogové televize, kde bylo potřeba k černobílému jasovému signálu přidat informace o barevných složkách. Složka Y je jasová složka, složky U a V, respektive Cb a Cr, nesou informaci o barvách v závislosti na jasovém signálu. [10, 11, 13]. Hlavní rozdíl mezi barevnými prostory YUV a YCbCr je v tom, že složky U a V jsou celočíselné a složky Cb a Cr mohou nabývat pouze kladných hodnot [10]. Složky YUV a

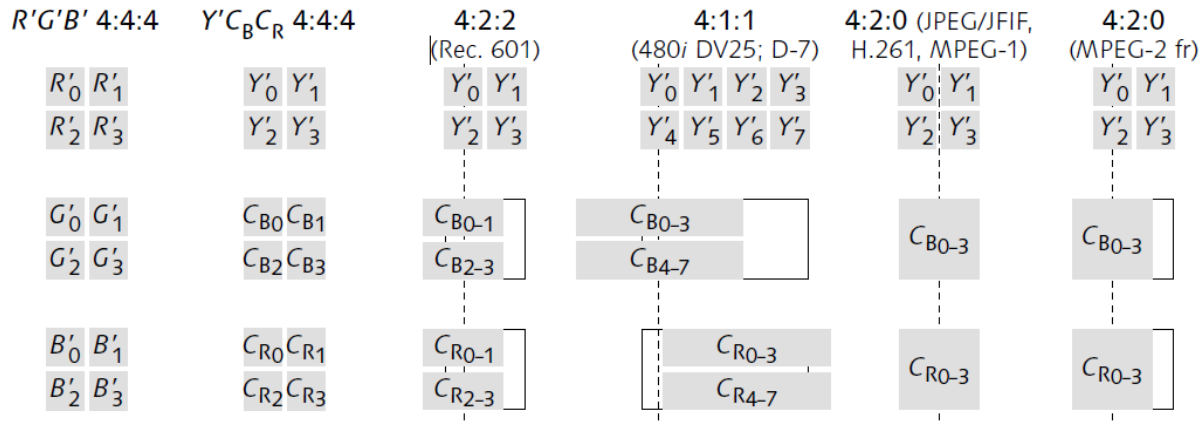
[6] Někdy může být také označována jako RGB24, RGB888 nebo jen jako RGB.
[7] Také známý jako CMYK, kde písmeno K je Key, barva, která vznikne smícháním všech barev. V tomto případě černá. Přidává se z ekonomických důvodů, aby se například při tisku nepoužívaly všechny barvy na tvorbu černé.

YCbCr je možné určit podle rovnic (4.2) a (4.3) [10, 11]. I v odborné literatuře se tyto dva barevné prostory nesprávně zaměňují[8].

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0{,}257 & 0{,}504 & 0{,}098 \\ -0{,}148 & -0{,}291 & 0{,}439 \\ 0{,}500 & -0{,}419 & 0{,}081 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{4.3}$$

4.1.3 Barevné podvzorkování

Barevné podvzorkování, nebo-li Chroma subsampling, je technika, které se využívá k redukci datového toku YUV a YCbCr signálů. Je založená na zmíněné nižší citlivosti lidského zraku na barevnou informaci než na jasovou informaci. Vzorkování se udává pomocí tří čísel oddělených dvojtečkami. První číslo je typicky 4 a udává jasovou horizontální vzorkovací referenci, od které se odvozují další dvě čísla. Druhé číslo je horizontální faktor podvzorkování, třetí číslo je stejné jako druhé nebo nula a udává, jestli byl signál vertikálně podvzorkován v poměru 2:1. Pokud je třetí číslo stejné jako první, znamená to, že signál nebyl podvzorkován vertikálně. Nula naopak naznačuje, že byl vertikálně podvzorkován v poměru 2:1. Některé běžné typy podvzorkování můžete vidět na obrázku číslo 13 [14]. Například podvzorkováním 4:1:1 nebo 4:2:0 lze signál ztrátově zkomprimovat na polovinu. Je možné signál zkomprimovat i více v případě použití méně běžných typů podvzorkování jako je 4:1:0.

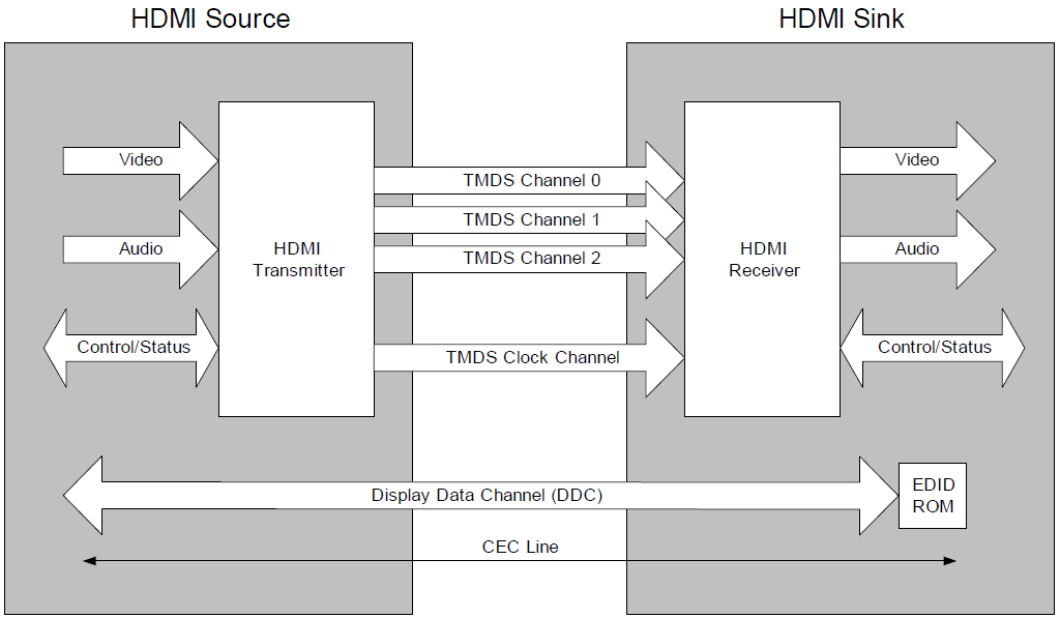


Obrázek 13: Běžné typy barevného podvzorkování [14].

[8] V následujících kapitolách budu popisovat ukládání video signálu ve formátu YUV4:1:1, který je ale podle definice YCbCr. V práci ho budu ale označovat jako YUV, abych se držel terminologie literatury, ze které jsem čerpal informace o tomto formátu.

4.2 HDMI

HDMI společně s DisplayPort, DVI a DE-15 VGA patří mezi čtyři v současné době nejpoužívanější rozhraní pro připojení externí obrazovky. HDMI se skládá z HDMI zdroje a HDMI přijímače, které jsou propojeny čtyřmi diferenčními páry. Tři TMDS páry jsou pro přenos audio a video signálů a čtvrtý je pro hodinové signály. Dále je k dispozici datový kanál, který slouží ke konfiguraci zdroje a cílové externí obrazovky [15].

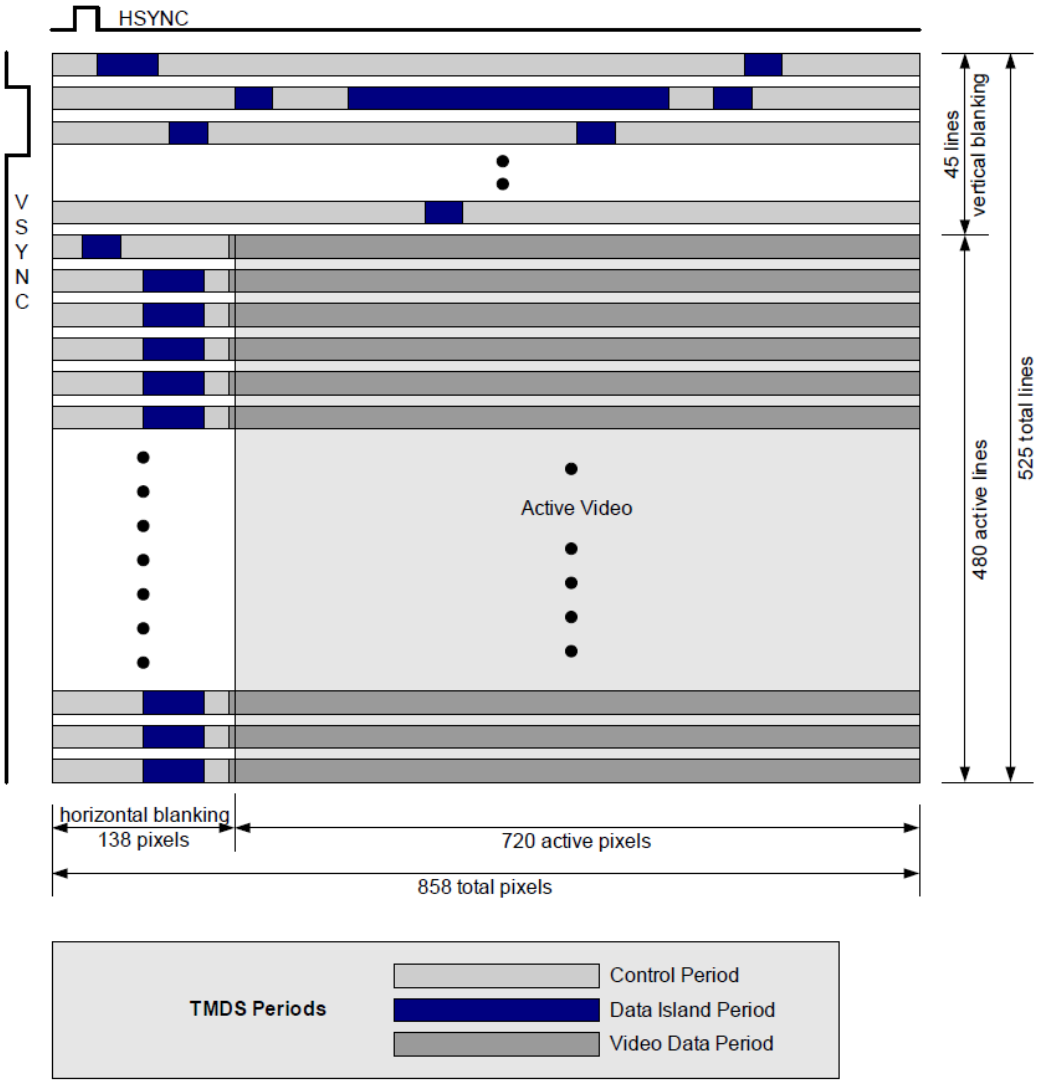


Obrázek 14: Rozhraní HDMI – blokové schéma přenosu [15].

HDMI rozhraní pracuje ve třech režimech. Režim video dat je doba, kdy se přenáší pixely obrazu po řádcích zleva doprava. Mód datových ostrovů přenáší audio a další signály. Posledním režimem je řídící doba, která se využívá jako výplň, když se nepřenáší video signál a ani není potřeba přenášet další data, používá se k synchronizaci a určuje také jaký další režim bude následovat. Režim video dat a datový ostrov musí být rozděleny pomocí řídící doby. Data se přenášejí kódovaná 10 bitovými slovy. Data video signálu se doplňují do 10 bitů ochranným pásem. Datové ostrovy jsou kódovány pomocí TERC4. Příklad jednoho snímku můžete vidět na obrázku 15 [15].

Podporované jsou tři typy formátů pixelů, a to RGB 4:4:4, YCrCb 4:4:4 a YCrCb 4:2:2. Barevná hloubka může být kromě standardních 24 pixelů také 30, 36 nebo i 48 pixelů v režimu Deep Color [15].

Všechny HDMI zdroje a přijímače musí být kompatibilní s DVI 1.0, které také používá TMDS. Pokud přijímač nebo zdroj pracuje v režimu DVI 1.0, je formát pixelů omezen pouze na RGB, nepoužívají se datové ostrovy a ochranné pásmo videa, kterým v běžném HDMI režimu začíná každý úsek přenosu videosignálů [15].

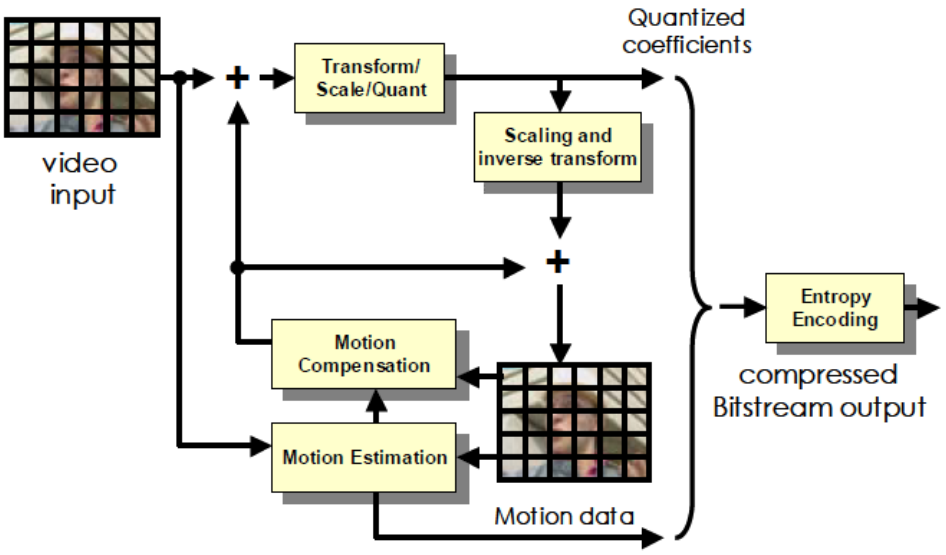


Obrázek 15: TMDS režimy pro jeden snímek v rozlišení 720 x 480 px [15].

4.3 Kódování videosignálů

Nekomprimovaný videosignál nese velké množství redundantních a irelevantních informací a má velmi vysoký datový tok. Určit je ho možné podle rovnice (5.4), podle rozlišení a snímkovací frekvence se přenosové rychlosti videosignálu pohybují v řádech stovek $MB \cdot s^{-1}$. Rozvíjí se techniky, jak datový tok videosignálu redukovat s co nejmenšími dopady na vjem koncového uživatele. Mezi první techniky, které se používaly, patří převod do barevných

prostorů YUV nebo YCbCr a redukce barevné informace, jak je popsáno v kapitole 4.1.2. Tento postup se používal při přenosu v analogových systémech NTSC a PAL a je základem i velké části digitálních kodérů.



Obrázek 16: Hybridní video kodér [16].

Digitální kódování se začalo standardizovat již v roce 1984. V roce 1990 standard ITU H.261 implementoval hybridní video kódování, které je základem mnoha kodérů i v dnešní době. Hybridní kodéry v sobě kombinují dvě metody. První je odhad pohybu mezi snímky, ze snímků které byly kódovány dříve. Druhá dekoreluje rozdíly po predikci převodem do frekvenční oblasti, typicky diskrétní kosinovou transformací, a poté koeficienty transformace kóduje kompresními kódy, obvykle odvozenými z Huffmanova kódu [16].

ISO a ITU standardizovaly MPEG-2 – H.262. MPEG-2 byl adoptován v DVB-T/S/C, v DVD, v kamerách a další elektronice. Obsahuje 4 úrovně a 6 profilů. Tyto profily a úrovně určují maximální rozlišení videa, použitý typ barevného vzorkování a typy použitých snímků [17].

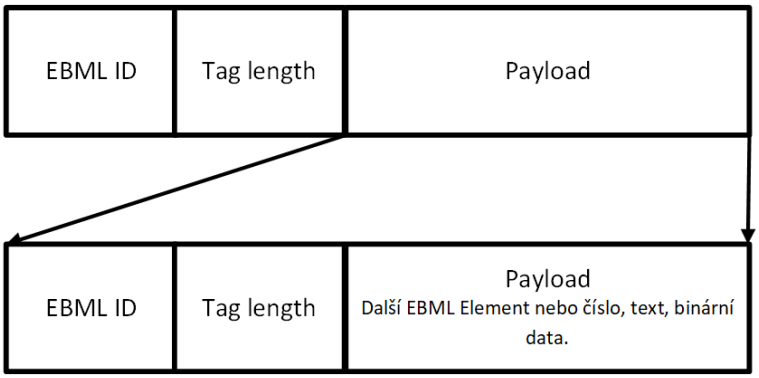
MPEG-2 tvoří z pixelů makrobloky 16 krát 16. Tyto makrobloky se převádí do frekvenční oblasti diskrétní kosinovou transformací. Koeficienty DCT se poté kódují dvourozměrným Huffmanovým kódem. V makroblocích se hledá podobnost s předchozími snímky pro predikci snímků. Predikovat se dají bloky, které mezi snímky zůstaly stejné. Hledají se bloky, které se mezi snímky posunuly, pro vytvoření pohybových vektorů, které se kódují místo DCT koeficientů. Snímky se rozdělují na I-snímky, které jsou pouze vnitřně kódované, nevyužívají

predikci a slouží jako reference pro predikci a také pro případnou synchronizaci signálu, aby se jedna chyba nekumulovala přes celý videozáznam. Druhý typ snímku je P-snímek, nebo-li snímek predikovaný z předchozího snímku. Posledním typem snímků jsou B-snímky, které používají obousměrnou predikci ze snímku před a potom [18].

Dalším velmi používaným kódovacím standardem je MPEG-4 část 10 AVC – H.264. Je stále založený na principech hybridního kodéru. H.264 používá pokročilejší aritmetické entropické kódování – CABAC. Zavádí variabilní velikost bloku pro kompenzaci pohybu až o velikosti 4 x 4. Predikované snímky používají více snímků jako referenci, MPEG-2 používá pouze jeden. H.264 umožňuje váhovat a posouvat signály kompenzující pohyb [19].

4.4 Datový kontejner Matroska

Matroska je datový kontejner, známý přidruženými soubory s příponou mkv, sloužící k uložení jednoho nebo více video a audio záznamů. Zaznamenané stopy je možné komprimovat různými kompresními algoritmy, v případě potřeby i každou stopu jiným algoritmem. Podporované je i nekomprimované audio a video. Matroska není kompresním algoritmem, cílí na to být univerzálním pouzdrem pro audio a video signály. K uloženým stopám je možné přidávat meta data, sady titulků. Je možné vytvořit menu pro výběr stop, obdobně jako je známé z DVD [20].



Obrázek 17: Vnoření EBML prvků.

Matroska je postavená na jazyku Extensible Binary Meta Language – EBML, který byl původně vyvinut pro potřeby Matrosky. Vychází z principů XML[9]. Každá značka má svoje unikátní EBML ID, velikost v bajtech a informaci, která je nesená uvnitř elementu. Elementy

[9] XML – Extensible Markup Language je značkovací jazyk sloužící univerzálnímu přenosu dat ve standardizovaném formátu více viz https://www.w3.org/TR/xml/

je možné do sebe vnořovat. Každý element Matrosky má definovanou úroveň zanoření, ve které se má nacházet a kolikrát se může vyskytovat v prvku, ve kterém je zanořen, jestli je jeho použití povinné a v jakém formátu a rozsahu mohou mít data v něm nesena. Velikost EBML značky je vyjádřena v bajtech a je kódovaná systémem podobným UTF-8. Počet nul na začátku udává, kolik bajtů je vyhrazeno pro velikost tagu po první jedničce. Možné varianty zápisu délky tagu jsou v tabulce 2 [20, 21].

Tabulka 2: Zápis velikosti EBML elementu.

Zapsané bity (Big Endian)	Rozsah
1xxx xxxx	0 až 2^{7}-2
01xx xxxx xxxx xxxx	0 až 2^{14}-2
001x xxxx xxxx xxxx xxxx xxxx	0 až 2^{21}-2
0001 xxxx xxxx xxxx xxxx xxxx xxxx xxxx	0 až 2^{28}-2
0000 1xxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	0 až 2^{35}-2
0000 01xx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	0 až 2^{42}-2
0000 001x xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	0 až 2^{49}-2
0000 0001 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	0 až 2^{56}-2

Podporované typy dat v EBML elementech jsou celá a přirozená čísla, číslo s plovoucí desetinnou čárkou, textové řetězce ASCII a UTF-8, datum, binární data a master element, který slouží pouze k uchovávání dalších prvků [21].

Tabulka 3: Tabulka Top-Level elementů [20].

Info	Informace o segmentu.
Meta Seek Information	Informace o pozici dalších Top-Level elementů.
Tracks	Informace o video, audio stopách, každá stopa reprezentována tagem TrackEntry.
Cluster	Blok audio-video dat
Cues	Informace pro vyhledávání v rámci Segmentu.
Attachments	Přílohy.
Chapters	Informace o kapitolách, pokud je daná stopa obsahuje.
Tags	Metadata k elementům

EBML dokument musí obsahovat EBML hlavičku a Segment, který je kořenovým elementem. Do něj jsou vnořeny všechny další EBML značky. Jejich využití a pořadí není pevně dané, jejich umístění záleží na konkrétní implementaci Matrosky. Jediný element, který je povinný je Segment Info, který nese základní informace o daném segmentu a o případných navazujících segmentech. Aby bylo video přehratelné, je potřeba, aby Segment obsahoval alespoň jednu stopu (element TrackEntry) a alespoň jeden cluster, který obsahuje bloky obrazový dat.

Nevhodné umístění může vést k degradaci výkonu například při rychlém posunu ve videozáznamu [20, 21].

Přehled Top-Level značek v rámci segmentu je k dispozici v tabulce 3, přehled zanořených elementů můžete najít v [20], stejně jako přehled EBML ID přidružených k daným elementům.

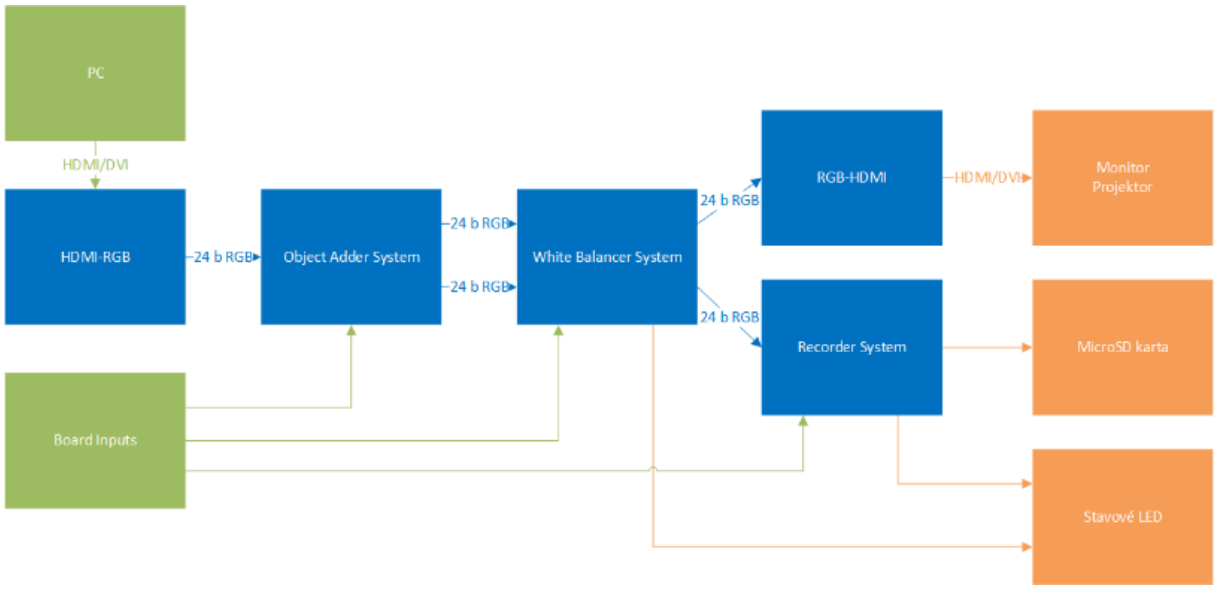
4.5 Současné možnosti záznamu videosignálu z počítače

V současnosti existují tři kategorie způsobů záznamu videosignálů z běžného počítače v reálném čase. První kategorii tvoří softwarové enkodéry, příkladem může být open source knihovna x264 nebo x265. Zpracování videosignálu probíhá na CPU zdrojového počítače. Druhá kategorie jsou kodéry s dedikovaným hardwarem, který je stále součástí počítače. Vyžadují podpůrnou aplikaci v počítači na ukládání záznamu a ovládání kodéru. Kódování, což je nejnáročnější operace, už probíhá na vyhrazeném hardwaru. Příkladem může být NVENC integrovaný v grafických kartách NVIDIA, nebo AMD VCE v grafických kartách AMD a nebo Intel Quick Sync integrovaný v procesorech Intel. Třetí kategorii tvoří hardwarové zařízení, které dokážou pracovat nezávislé na zdrojovém počítači a ukládat na videozáznam na SD kartu. Druhou možností je připojení záznamového zařízení ke zdrojovému nebo jinému nezávislému počítači pomocí USB, Thunderbolt nebo PCI-Express a ukládat záznam na jeho pevný disk, například střihové zařízení od společnosti AVerMedia nebo Elgato.

Jak bylo změřeno v [22] softwarové kódování má vyšší spotřebu elektrické energie a výrazně více zatěžuje hlavní procesor počítače. Řešení druhé a třetí kategorie požívají dedikovaný hardware, a tím omezují nebo úplně eliminují využití času CPU počítače. Řešení spadající do druhé kategorie jsou závislé na zdrojovém počítači a stále využívají čas CPU, i když výrazně méně než čistě softwarové kódování. Oproti třetí kategorii mají výhodu, že dedikovaný hardware je součástí procesoru nebo grafického procesoru. který je standardní součástí počítače a není proto potřeba kupovat další hardware, jehož cena se pohybuje v současné době v jednotkách tisíců českých korun. Výhodou třetí kategorie je, že dokáže zaznamenávat videosignál bez využití zdrojového počítače a jedy možné zaznamenávat z počítačů, které nemají dostatečný výkon pro běh aplikace, kterou je zapotřebí zaznamenat a softwarové kódování a zároveň nemají dedikovaný kodér. Je také možné zaznamenávat i dobu kdy není počítač schopný běhu obslužného hardwaru, například během POST.

5 Návrh a realizace modelu

V této kapitole popíši platformu, na které je model realizovaný, postup návrhu a realizace bloků, ze kterých se celé řešení skládá. Celé řešení se rozděluje do dvou základních bloků. První blok, detailněji se mu budu věnovat v kapitole 0, se stará o zpracování videosignálu z počítače a jeho úpravu v reálné čase. Výstup z tohoto bloku jde jednak výstupním HDMI konektorem na externí monitor, tak i do druhé bloku, který se stará o záznam videosignálu, tomuto bloku se podrobněji budu věnovat v kapitole 5.3. Blokové schéma modelu je vidět na obrázku 18.



Obrázek 18: Zjednodušené blokové schéma navrženého modelu záznamového zařízení.

Model je realizovaný pomocí volně dostupného vývojového studia Xilinx Vivado 2019.1, ve kterém jsou jazykem VHDL realizované všechny modré bloky na obrázku 18. Výjimku tvoří blok Recorder System, blokové schéma na obrázku 23, jehož blok Matroska Writer je realizovaný pomocí jazyka C ve vývojovém studiu Xilinx SDK 2019.1. Matroska Writer je jako jediný blok realizovaný v PS Zynq. Podpůrná aplikace pro počítač (více v kapitole 5.2) je vyvíjena pomocí Microsoft Visual Studio 2019 a Microsoft Framework 4.7.2 v jazyce Visual Basic.

Vzhledem k potřebě splnit podmínky časování syntetizovaného obvodu se ve Vivado používá Vivado Implementation 2019 Performance ExtraTimeOpt implementační strategie, která oproti základní strategii používá pokročilejší algoritmy pro optimalizaci časovaní obvodu, které při

vyšších frekvencích podávají lepší výsledky. Nevýhodou této implementační strategie je delší výpočetní čas potřebný k implementaci.

Pro ladění během vývoje se používala kombinace simulací a testů na hardwaru s využitím testovacích Integrated Logic Analyzer IP jader [23], které umožňují po splnění specifikované podmínky zaznamenat stavy vybraných sběrnic a signálů v designu. Generované mkv soubory se validovaly pomocí volně dostupných nástrojů mkvalidator a MKVToolNix [20].

Nejprve byl vytvořen základní koncept modelu, který je na obrázku 18. Bloky poté byly vyvíjené v pořadí, ve kterém jimi prochází videosignál a ve kterém jsou popisovány v kapitolách níže. Postupně jak byly bloky vyvíjeny, tak byla průběžně testována jejich funkčnost a kompatibilita se zbytkem modelu.

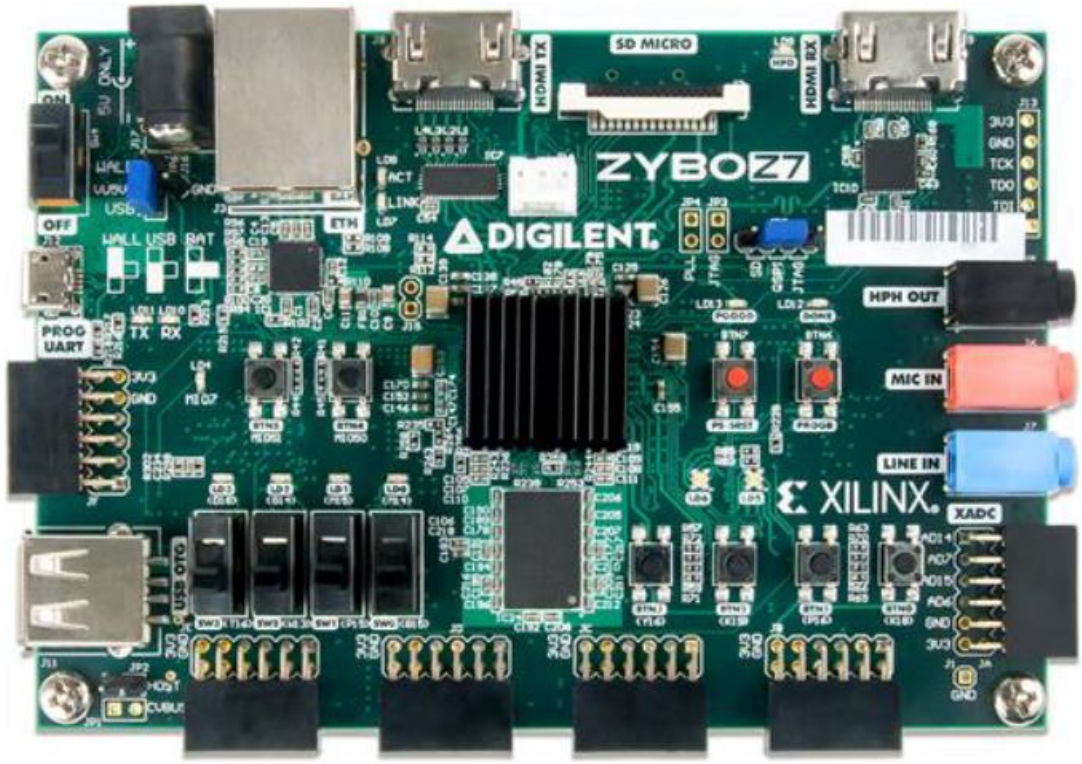
5.1 Vývojová deska Zybo Z7-20

Pro tuto práci mi byla katedrou mikroelektroniky zajištěna vývojová deska Zybo Z7-20 od společnosti Digilent. Tento kit obsahuje potřebné periferie pro vývoj záznamového zařízení. Srdcem tohoto vývojového přípravku je systém na čipu Zynq 7020 od firmy Xilinx, který v sobě kombinuje dvoujádrový procesor ARM Cortex-A9 MPCore na frekvenci 667 MHz a programovatelnou logickou část, která je ekvivalentem FPGA řady Artix-7. Více technických parametrů a informací o Zynq 7020 v tabulce 4 a v kapitole 2.3.1.

Tabulka 4: Technické parametry Xilinx Zynq-7020 [4, 5].

Procesorová jádra	2x 667 MHz ARM Cortex-A9 MPCore
On-Chip Memory	256 kB
Podpora externí paměti	1 GB – DDR3, DDR3L, DDR2, LPDDR2
Podpora statické externí paměti	2x Quad-SPI, NAND, NOR
Periferie	2x UART, 2x CAN 2.02B, 2x I2C, 4x 32 b GPIO 2x USB 2.0 (OTG), 2x Gigabit Ethernet, 2x SD
Rozhraní mezi PS a PL	2x AXI 32b Master/Slave, 4x AXI 64/32 b Memory, AXI 64 b ACP, 16 kanálů přerušení.
Xilinx 7 Series Ekivalent	Artix-7 FPGA
Programovatelné logické buňky	85000
Look-Up tabulky	53200
Klopné obvody	106400
Bloková RAM (36 kb bloky)	4,9 Mb (140)
Digitální signálové procesory – DSP	220

K Zynq 7020 je na Zybu Z7-20 připojená celá řada periferii. Nejdůležitější jsou dva HDMI konektory, kde jeden slouží jako vstup a druhý jako výstup. To umožňuje přijmout videosignál z počítače, zpracovat, uložit a odeslat dále na monitor. Mezi další používané periferie patří slot na microSD karty, který se používá na ukládání zaznamenaného videa. Pro interakci s uživatelem slouží sada přepínačů a tlačítek. Další parametry a periferie Zybo Z7-20 jsou vidět v tabulce 5.



Obrázek 19: Digilent Zybo Z7-20 [24]..

Vývojový kit Zybo Z7-20 má tři možnosti napájení, první je pomocí MicroUSB portu (J12), druhou možností je externím zdrojem připojený pomocí jack konektoru s vnitřním průměrem 2,1 mm. Poslední možnost je připojení napájecího napětí přímo na středový pin propojky J16, která v ostatních případech slouží k přepínání mezi prvními dvěma zmíněnými variantami. I přesto, že se MicroUSB port nabízí jako nejjednodušší způsob napájení, je potřeba myslet na to, že napájení přes MicroUSB port je limitováno na 750 mA a zároveň standardní USB2.0 porty počítají s maximálním odběrem proudu 500 mA. Proto je doporučeno použít externí zdroj, pro který je proudové omezení nastavené na 4 A [24].

Dvoujádrový procesor ARM Cortex-A9 a jeho integrované periferie se v terminologii společnosti Xilinx nazývají Processing System – PS, v této práci se budu této terminologie

držet. Stejně tomu bude u programovatelného logického pole ekvivalentnímu k FPGA Artix-7, které je v rámci rodiny čipů Zynq 7000 nazývané PL – Programmble Logic [4]. Toto spojení procesoru a programovatelného hradlového pole přináší celou řadu výhod, které jsem popisoval v kapitole 2.3.1, ale zároveň i nevýhody a komplikace při vývoji. Jako hlavní nevýhodu uvedu, že některé periferie jsou na vývojové desce Zybo Z7-20 zapojené buď přímo do PS, nebo do PL, kvůli tomu je potřeba například při zápisu na microSD kartu z bloku, který je syntetizován v PL, zapisovat přes procesorovou část PS. Omezení je samozřejmě i opačným směrem. Bloky, mezi PS a PL, lze propojit vysoko rychlostní sběrnici AXI [4, 24].

Tabulka 5: Technické parametry vývojové desky Digilent Zybo Z7-20 [24].

Systém na čipu	Xilinx Zynq-7020
Paměť	1 GB DDR3L - 32 bit sběrnice s efektivní frekvencí 1066 MHz
Paměť	16 MB Quad-SPI Flash
Paměť	Slot na microSD kartu
USB	USB-JTAG, USB-UART, USB2.0 OTG
Síť	1 Gbps Ethernet RJ-45.
HDMI	1x vstupní (sink) a 1x výstupní (source) port
Kamera	Pcam konektor pro připojení kamery.
Audio	Stereo sluchátkový, pro mikrofon, Line-in 3,5 mm Jack
Přepínače	4x posuvný přepínač
Tlačítka	4x tlačítko zapojené do PL, 2x tlačítko zapojené do PS
LED diody.	2x RGB LED do (PL), 4x LED (PL), 1x LED (PS)
Pmod	3x Vysokorychlostní Pmod (PL), 1x standardní Pmod (PL), 1x Pmod k ADC (PL), 1x Pmod (PS)

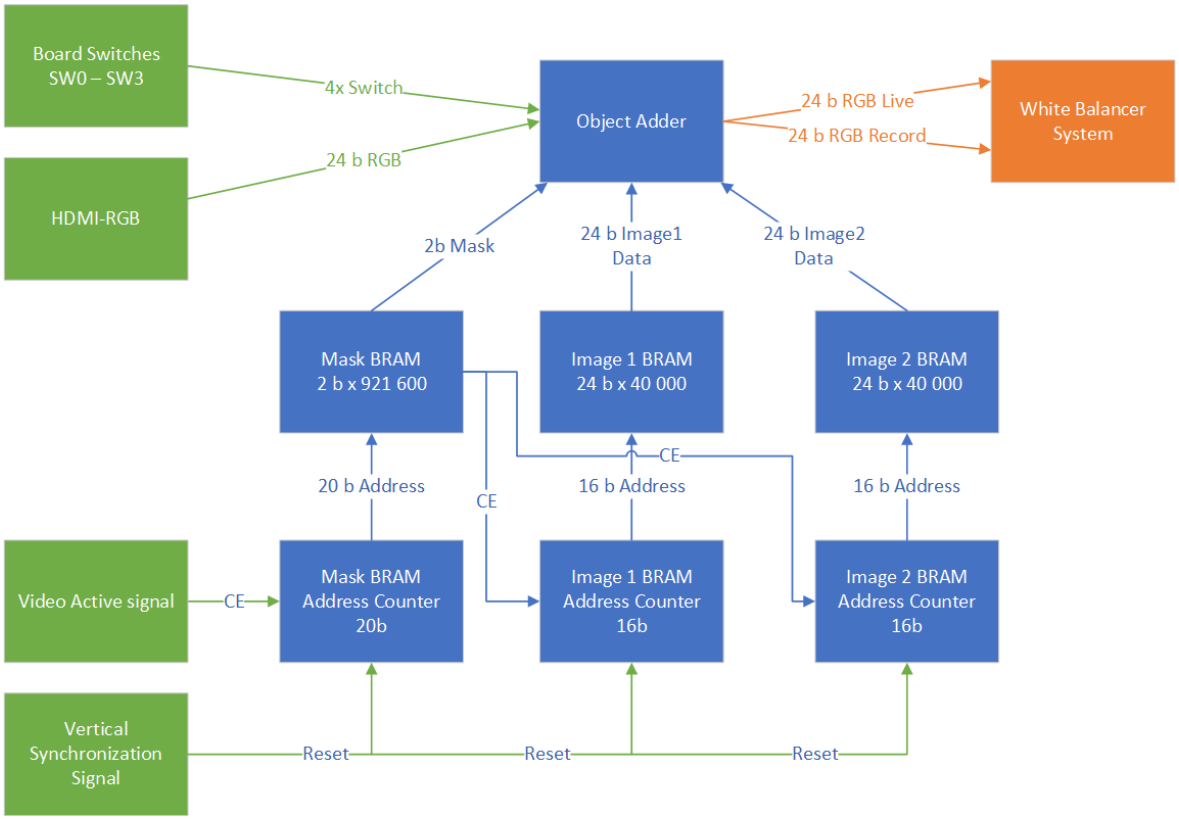
5.2 Blok zpracovávající videosignál v reálném čase

Jak již bylo zmíněno výše, cíl této práce je zpracovávat videosignál v rozlišení 1280 na 720 pixelů při obnovovací frekvenci 60 Hz. Z toho vyplývá, podle rovnice (5.1) při započtení zatmavených intervalů, pixelová frekvence 74,25 MHz [25]. To znamená, že je potřeba zpracovat datový tok 1,236 $Gb \cdot s^{-1}$, snímková perioda je 16,67 ms a perioda pixelové frekvence je 13,47 ns.

$$f_p = (W + W_b) \cdot (H + H_b) \cdot f_s$$
$$= (1280 + 370) \cdot (720 + 30) \cdot 60 \quad (5.1)$$
$$= 74{,}25 \text{ MHz}$$

Kde W (px) je šířka viditelného obrazu, W_b (px) je šířka horizontální zatmívané oblasti, H (px) je výška obrazu, H_b (px) je výška vertikální zatmívané oblasti a f_s (Hz) je snímková frekvence.

První krok, který se provádí se vstupním videosignálem, je jeho převod z HDMI TDMS signálu na 24bitový RGB signál, synchronizační signály, pixelovou frekvenci a další pomocné signály. Tuto úlohu zajišťuje blok HDMI-RGB, který se skládá z IP jádra Digilent DVI-to-RGB (Sink) 1.9 IP, které sice implementuje jen DVI 1.0 [26]. DVI 1.0 je ale povinnou součástí HDMI [15] a pro účely této práce dostatečné. Blok HDMI-RGB dále obsahuje převodník z RBG na RGB, protože IP jádro DVI-to-RGB má, i přes svůj název, výstup ve formátu RBG [26]. To znamená, že ve 24bitovém slově je prohozena pozice zelené a modré složky.

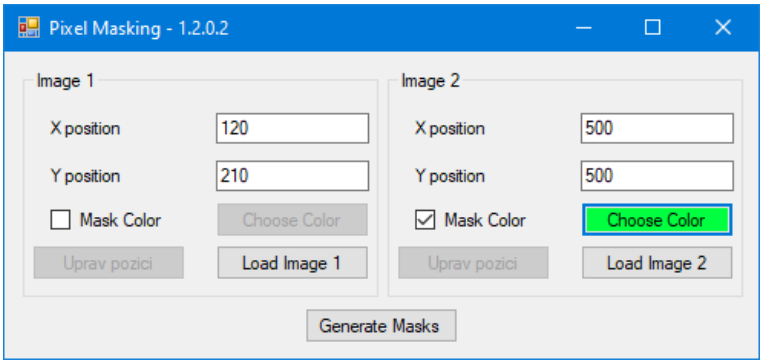


Obrázek 20: Blokové schéma systému přidávání objektů – Object Adder System. Zeleně jsou označené vstupy, oranžově výstupy.

Dvaceti čtyř bitový videosignál je pak dále zpracován v systému Object Adder System, jehož blokové schéma je na obrázku 20. Tento systém přidává do videosignálu objekty, loga, ale i text na transparentním pozadí, maskováním konkrétních pixelů ve videosignálu a jejich nahrazení pixely vkládaného objektu. Systém umí vložit až dva nezávislé objekty, oba až v rozlišení 40 kpx. Pixely, které mají transparentní barvu, se do celkového počtu možných pixelů nepočítají. Object Adder System je implementovaný v PL jazykem VHDL. Je k němu přidružená obslužná aplikace pro počítač Pixel Masking vyvinutá programovacím jazykem Microsoft Visual Basic 2019. Obě části detailněji popíši v odstavcích níže.

Object Adder System v době, kdy je aktivní signál Video Active, čte pro každý pixel dvou bitovou masku z blokové paměti Mask BRAM. Nejméně významný bit (LSB) této masky nese informaci o tom, jestli má být pixel přepsán pixel objektem 1 (log. 1), a nebo se má na výstup přenést vstupní RGB signál (log. 0). Nejvíce významný bit (MSB) této dvou bitové masky má stejný význam, jen určuje přepsání pixelu daty objektu 2. Signály dvou bitové masky se také používají k inkrementaci čítačů, které adresují blokové paměti Image 1 BRAM a Image 2 BRAM, tak aby měly na svém výstupu připraveny data pro následující pixel daného objektu. Čítače se resetují na začátku každé snímku pomocí vertikálního synchronizačního impulsu.

Spínače SW0 až SW3, slouží k ovládání výstupů bloku Object Adder. SW0 ovládá zobrazování objektu 1 do videosignálu procházejícího na externí monitor. SW1 ovládá vkládání objektu 1 do záznamu. Stejně tak SW2 vkládá objekt 2 do signálu procházejícího na monitor a SW3 ho přidává do záznamu.



Obrázek 21: Aplikace Pixel Masking.

Pixel Masking je aplikace pro počítače s Microsoft Windows v 64 bitové verzi s Microsoft .NET Framework 4.7.2 nebo vyšším, vyvinutá v jazyce Microsoft Visual Basic 2019. Slouží k vygenerování coe souborů, které se dají vložit do bitstreamu pro Xilinx

Zynq 7020 jako výchozí obsah blokových pamětí Mask BRAM, Image 1 BRAM a Image 2 BRAM. Do programu je možné nahrát dva obrázky ve formátech PNG, JPG, BMP nebo GIF. Jejich rozlišení, po odečtení pixelů s transparentní barvou, nesmí přesáhnout 40 kpx, aby je bylo možné uložit do příslušných blokových pamětí. Je možné zvolit umístění objektu ve výsledném obraze zadáním souřadnice levého horního rohu vkládaného obrázku. Systém validuje, aby obrázek nebyl umístěn mimo cílovou obrazovku. Pokud se nahrává obrázek s jednobarevným pozadím je možné zvolit libovolnou barvu, které se bude klíčovat na transparentní.

Po nahrání obrázků a kontrole a nastavení barev, které mají být transparentní, validaci velikosti a umístění vložených obrázků. Aplikace Pixel Masking vytvoří dvourozměrné pole velikosti 1280 x 720 objektů třídy Pixel obsahující pouze transparentní pixely. Toto pole objektů reprezentuje celou obrazovku. Poté se na vybrané místa vloží data pixelů vkládaných obrázků. Třída Pixel má čtyři veřejné vlastnosti. Vlastnost Type je výčtového typu PixelType, který nese informaci o tom, jestli je pixel transparentní nebo patří k obrázku 1 anebo k obrázku 2. Třída Pixel má dále vlastnosti datového typu Byte pro červenou, zelenou a modrou barvu.

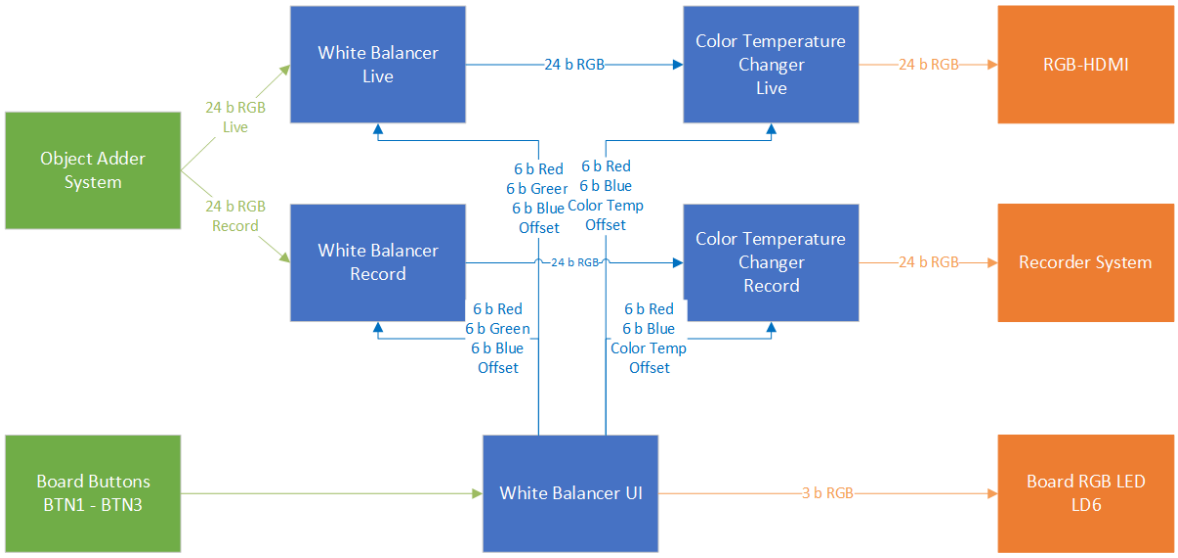
Z tohoto dvourozměrného pole objektů třídy Pixel se generují tři coe soubory. Pro zrychlení procesu každý samostatným výpočetním vláknem. PrimaryMask.coe pro blokovou paměť Mask BRAM, obsahuje dvou bitová slova pro každý pixel obrazu. „00“, pokud je pixel transparentní a bude se na jeho místě zobrazovat data získaná z HDMI, „01“, pokud lze daný pixel nahradit pixelem z obrázku 1 a „10“, pokud je možné pixel nahradit daty z obrázku 2. Image1BRAM.coe a Image2BRAM.coe obsahují 24 bitová slova, ve kterých jsou RGB data pixelů vkládaných obrázků.

Tyto tři coe soubory se importují do Vivado, jako výchozí stav blokových pamětí, a poté se jako součást bitstreamu nahrají do Zybo Z7-20. Po připojení zdroje signálu a cílové obrazovky je možné začít zobrazovat vložené objekty na vybraných místech a vkládat je do záznamu.

V bloku Object Adder System se také vstupní RGB signál rozděluje na dva RGB signály, kde jeden je určen pro zobrazení na monitoru a druhý je určen pro záznam. Důvodem je to, že je možné zobrazit objekt 1 v záznamu ale v živém obrazného nechat skrytý, nebo opačně. To samé platí i pro objekt 2. Následující bloky úpravy barev jsou zdvojené, aby mohly upravovat jak zaznamenávaný videosignál, tak i signál procházející na monitor. Bloky jsou takto uspořádané, aby se případné úpravy barev aplikovaly i na vložené objekty.

Za blokem Object Adder System je blok úpravy barev White Balancer System, který je navržen pro kompenzaci barev cílového monitoru a posun obrazu k teplejším nebo studenějším barvám. Ovlivňovat barvy je možné dvěma způsoby.

První způsob, blok White Balancer, je přičtení nebo odečtení 31 úrovní od každé barvy. Tyto úrovně mají stejnou velikost jako úrovně RGB signálu. Systém je ošetřen tak aby se po přičtení požadované úrovně nedostal přes maximální hodnotu 255 na daném barevném kanále. Obdobně se při odečítání kontroluje, aby se hodnota barevných kanálů nedostala do záporných hodnot.



Obrázek 22: White Balancer System – blokové schéma.

Druhá možnost, blok Color Temperature Changer, je posun teploty obrazu o 31 kroků směrem k teplejším nebo studenějším barvám. Jeden krok směrem k teplejším barvám znamená vynásobení červené složky signálu hodnotou 1,03125 a vynásobení modré složky signálu hodnotou 0,96875. To znamená: jeden krok je ovlivnění hodnoty barevné složky o přibližně 3 %. Pokud je zvolen posun směrem ke studenějším barvám násobící hodnoty se prohodí a modrá složka se násobí hodnotou větší než jedna a červená naopak hodnotou menší než jedna. V obou variantách není zelená složka ovlivněna. Násobení je realizované tak, že se v neutrální pozici hodnota daného barevného kanálu vynásobí 32 a poté pomocí bitového posunu vydělí 32. V případě, kdy se má hodnota barevného kanálu zvýšit, tak se k prvnímu koeficientu přičte 1 a hodnota barevného kanálu se tedy násobí číslem 33 a dělí číslem 32. Tím se ve výsledku dojde k vynásobení hodnotou 1,03125. Obdobně, pokud se hodnota barevného kanálu snižuje, tak se odečte od prvního koeficientu 1 a hodnota barevného kanálu se násobí číslem 31 a poté dělí číslem 32, a to odpovídá násobení hodnotu 0,96875. Tento postup umožňuje pracovat jen

s celými čísly. Výpočty jsou popsány rovnicemi (5.2) a (5.3). Opět je ošetřeno, aby se výsledná hodnota nedostala mimo rozsah 0 až 255.

$$R_M = \frac{k_R}{32} \cdot R \tag{5.2}$$

Kde R_M (-) je modifikovaná hodnota červeného barevného kanálu, k_R (-), je násobící koeficient pro červenou barvu a R (-) je zdrojová hodnota červeného kanálu.

$$B_M = \frac{k_B}{32} \cdot B \tag{5.3}$$

Kde B_M (-) je modifikovaná hodnota modrého barevného kanálu, k_B (-), je násobící koeficient pro modrou barvu a B (-) je zdrojová hodnota modrého kanálu.

Oba výše zmíněné systémy na úpravu barev jsou zapojeny sériově za sebou, a to pro průchozí i zaznamenávaný videosignál. White Balancer, který realizuje úpravu sčítáním a odčítání je zapojen první a za ním je Color Temperature Changer, který úpravu realizuje pomocí násobení. Celý blok White Balancer System je vidět na obrázku 22.

Celý systém na úpravu barev je ovládán pomocí tlačítek BTN1 až BTN3. Tlačítko BTN1 slouží pro výběr barvy, která se bude upravovat. RGB LED LD6 signalizuje vybranou barvu. Bílá barva signalizuje, že je vybraný systém Color Temperature Changer. Tlačítko BTN3 se používá pro přidání úrovně vybrané barvy a tlačítko BTN2 snižuje úroveň dané barvy.

Posledním blokem v části zajišťující průchod videosignálu na externí obrazovku, je blok RGB-HDMI. Tento blok obsahuje převodník z RGB na RBG a IP jádro Digilent RGB-to-DVI (Source) 1.4, které zajišťuje potřebný převod na TMDS signál, který se může přenášet po DVI nebo, v tomto případě přes HDMI [27]. Druhý výstupní signál z bloku White Balancer System je přiveden na vstup na bloku Recorder System.

5.3 Záznamový systém

V této kapitole se budu věnovat návrhu bloků, které zajišťují redukci datového toku videosignálu a starají se o jeho uložení na microSD kartu v kontejneru Matroska.

Již v úvodu je potřeba říct, že se mi pro tuto práci, oproti původnímu předpokladu, nepodařilo sehnat, žádné IP jádro, které by bylo schopné zajistit kódování MPEG v jakékoliv jeho verzi a které by bylo dostupné volně k použití a kompatibilní ze SoC Xilinx Zynq-7020. Zároveň Zynq 7020, na rozdíl od řady Zynq Ultrascale+ EV, neobsahuje integrovaný kodér H.264/265. Komerční IP jádra jsou bohužel mimo finanční možnosti této práce, jelikož se jejich ceny pohybují v řádech tisíců amerických dolarů. Vývoj MPEG enkodéru je svou komplexností mimo rozsah na této práce. Proto jsem se rozhodl pro níže uvedenou cestou redukce datového toku videosignálu, která je jen ztrátová a výsledný záznam bude mít vždy výrazně nižší kvalitu, než které by bylo možné dosáhnout použitím H.264 kodéru nebo jiného enkodéru.

Zdrojový videosignál má rozlišení 1280 na 720 px a snímkovou frekvenci 60 Hz, to po dosazení do rovnice (5.4) odpovídá datovému toku 158,2 MB·s^{-1}. Při ukládání videa na hardwarovém platformě Zybo Z7-20 je úzké hrdlo slot na microSD karty, který je na vývojové desce pouze ve verzi high speed, která má maximální teoretickou rychlost 25 MB·s^{-1} [24, 28]. Nepomůže tedy ani použití, dnes už velmi běžných, microSD karet kategorie UHS-I (Ultra High Speed Phase I), které dosahují rychlostí až 104 MB·s^{-1} [28].

$$BW = W \cdot H \cdot f_s \cdot BpP \tag{5.4}$$

Kde BW (B/s) je přenosová rychlost obrazu, W (px) je šířka obrazu, H (px) je výška obrazu, BpP (B) je počet bajtů na pixel (Byte per Pixel).

Jak bylo zmíněno výše je zapotřebí zásadně snížit datový tok videosignálu, aby ho bylo možné uložit na microSD kartu. Prvním krokem je snížení snímkové frekvence na 30 Hz, to odpovídá snížení datového toku na polovinu. Druhý krok je snížení rozlišení z 1280 x 720 px na 640 x 360 px. To je snížení počtu pixelů na čtvrtinu a tím i datového toku na čtvrtinu. Třetí krok je převedení videosignálu z barevného prostoru RGB, který má 24 bitů na pixel tj. 96 bitů na 4 pixely, do barevného prostoru YUV 4:1:1, který má 48 bitů na makro pixel, který se skládá ze 4 pixelů v řádce. Tím dosáhneme redukce datového toku o polovinu. Čtvrtý a poslední krok

v redukci datového toku je převod videosignálu z progresivního snímkování na prokládané, tím se sníží datový tok na polovinu. Výsledný videosignál, který se bude ukládat na microSD kartu, má rozlišení 640 na 360 pixelů při 30 půl snímcích za sekundu v barevném prostoru YUV 4:1:1. Po dosazení do rovnice (5.4) vychází datový tok na 4,94 $MB \cdot s^{-1}$, tedy na 3,125 % původního datového toku. To poskytuje dostatek času na zpracování a uložení dat i s rezervou pro případ použití pomalejší microSD karty. V následujících odstavcích detailněji popisuji bloky zajišťující záznam video signálu.

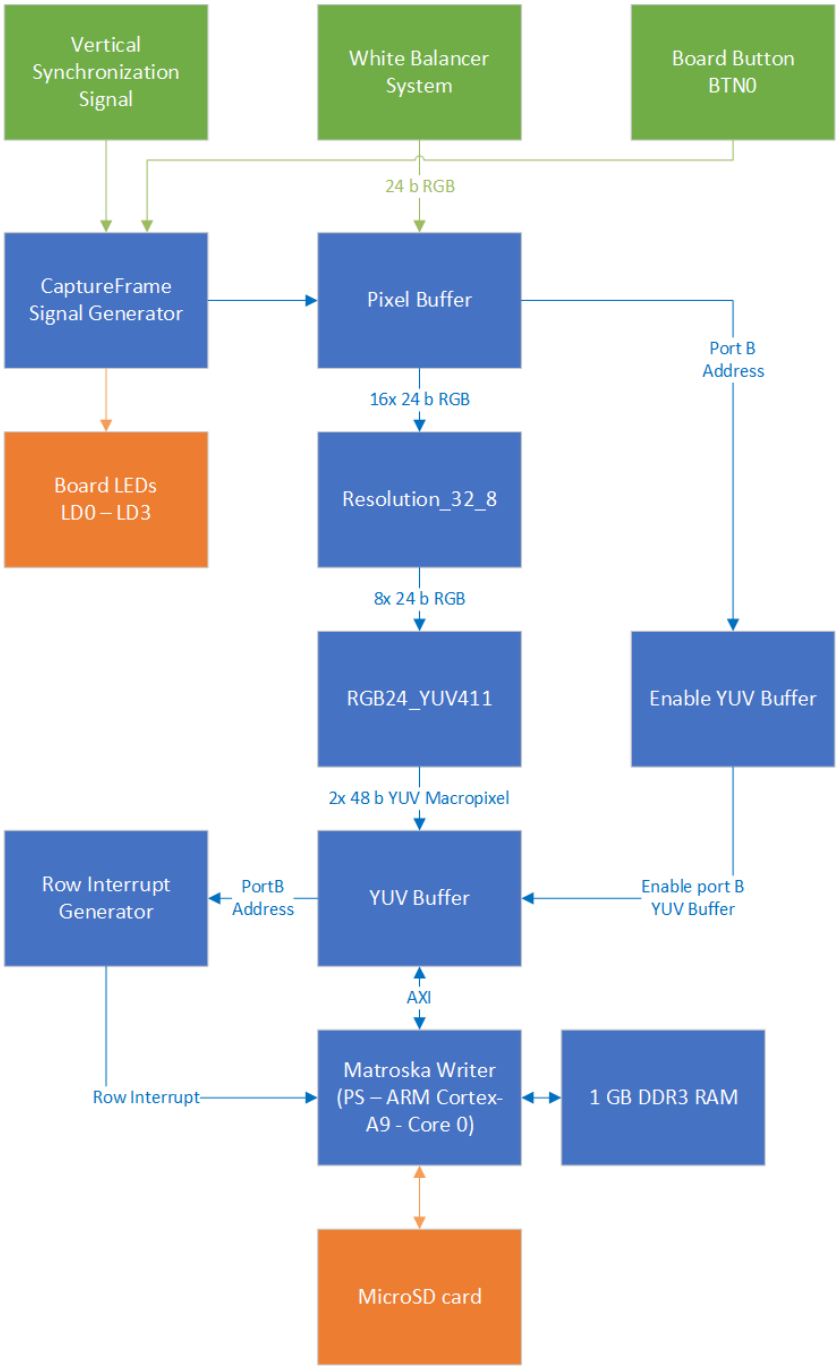
Signál, který se zaznamenává od signálu, který se v reálném čase přenáší na externí monitor, se odděluje již v bloku Object Adder System a poté se na něj aplikují stejné úpravy barev jako na signál pro monitor, ale v samostatných blocích (viz kapitola 5.2). Dále už se videosignál dostane do bloku Recorder System, jehož zjednodušené blokové schéma je na obrázku 21.

V bloku Recorder System je řídící blok CaptureFrame Signal Generator, který po zapnutí nahrávání tlačítkem BTN0, na každou náběžnou hranu vertikálního synchronizačního impulsu, změní hodnotu signálu CaptureFrame z logické 0 na logickou 1 nebo z logické 1 na logickou 0, v závislosti na předchozím stavu. Výchozí stav před začátkem nahrávání je logická 0. Signál CaptureFrame povoluje činnost dalších bloků v systému Recorder potřebných pro zaznamenání snímku. Tímto přepínáním signálu CaptureFrame je snížení snímkové frekvence na polovinu, tedy z 60 Hz na 30 Hz.

Když je signál CaptureFrame v logické 1 a zároveň je signál VideoActive také v logické 1, povolí se zápis do blokové paměti Pixel Buffer na portu A, do které se zapisují pixely po 24 bitových slovech. Paměť Pixel Buffer má kapacitu na uložení 4 řádek po 1280 pixelech při 24 bitech na pixel. Vždy po uložení dvou řádek se generuje signál Enable_B_Pixel_Buffer, který povoluje čtení z portu B blokové paměti Pixel Buffer. Z portu B se čte po 16 pixelech, nebo-li 384 bitech, první čtení se provádí z liché řádky a následující čtení ze sudé. Na vstupu následujícího bloku Resolution_32_8 je pak segment obrazu 16 pixelů široký a 2 pixely vysoký.

Blok Resolution_32_8 snižuje rozlišení z 32 pixelů na 8 pixelů. Skládá se z 8 bloků Resolution_4_1, které provádějí samotné snížení rozlišení. Blok Resolution_4_1 průměruje jednotlivé barevné složky 4 pixelů uspořádaných do čtverce a tyto průměry pro každou barevnou složku jsou výsledné barevné složky nově vzniklého pixelu. Pomocí bloku Resolution_32_8 se snižuje rozlišení z původních 1280 x 720 px na 640 x 360 px.

Osm pixelů uspořádaných v řádce, které jsou výstupem bloku Resolution_32_8 se dále zpracovávají v bloku RGB24_YUV411, který převádí pixely z barevného prostoru RGB do barevného prostoru YUV4:1:1. Tento blok se skládá z 8 bloků Y, 2 bloků U a 2 bloků V. Prvky U a V se vzorkují z pixelu na pozici 0 a 4 v 8 pixelovém makropixelu. Složky Y, U a V jsou vypočítávány podle rovnic (5.5), (5.6) a (5.7) [29]. Tento způsob umožňuje výpočet složek YUV bez nutnosti výpočtů s plovoucí desetinou čárkou.



Obrázek 23: Recorder System – zjednodušené blokové schéma. Zeleně jsou označeny vstupy a oranžově jsou označeny výstupy.

$$Y = \big((66 \cdot R + 128 \cdot G + 25 \cdot B + 128) \gg 8\big) + 16 \tag{5.5}$$

Kde Y (-) je jasová složka pixelu, R (-) je červená složka, G (-) je zelená složka a B (-) je modrá složka pixelu.

$$U = \big((-38 \cdot R - 74 \cdot G + 112 \cdot B + 128) \gg 8\big) + 128 \tag{5.6}$$

Kde U (-) je chromatická složka pixelu, R (-) je červená složka, G (-) je zelená složka a B (-) je modrá složka pixelu.

$$V = \big((112 \cdot R - 94 \cdot G - 18 \cdot B + 128) \gg 8\big) + 128 \tag{5.7}$$

Kde V (-) je chromatická složka pixelu, R (-) je červená složka, G (-) je zelená složka a B (-) je modrá složka pixelu.

Makropixely převedené do YUV4:1:1 se ukládají do blokové paměti YUV Buffer. Když se ukládá lichý snímek, tak pro každou lichou řádku generuje žádost o přerušení do PS do procesoru ARM Cortex-A9 (blok Matroska Writer). Při ukládání sudého snímku se generuje žádost o přerušení pro každou sudou řádku. Tím to bude procesor číst jen každou druhou řádku a bude moct uložit video jako prokládané. Port A paměti YUV Buffer je připojený pomocí IP jádra AXI BRAM Controller v4.1 a sběrnice AXI k procesoru [30].

Blok Matroska Writer při startu v PS vygeneruje hlavičku kontejneru Matroska a uloží ji na microSD kartu. V rámci obsluhy přerušení se pak ukládají řádky z blokové paměti YUV Buffer na svoji pozici v clusteru kontejneru Matroska, který je uložená v DDR3 paměti. Obrazová data se ukládají ve formátu Y41P – YUV4:1:1 planární, což znamená, že v rámci jedno snímku jsou nejprve uloženy všechny jasové Y složky snímku, za nimi všechny chromatické U složky a nakonec všechny V složky snímku. Jeden cluster je v této konkrétní implementaci 1 minuta videa. Poté co se naplní celý cluster, tak se z RAM uloží na microSD kartu a v RAM se začne vytvářet nový cluster. Velikost jednoho mkv kontejneru je limitovaná na 4 GB, což je maximální velikost souboru v souborovém systému FAT32 [31].

Kontejner Matroska, který je popsán v kapitole 4.4, má, v této implementaci, pořadí elementů EBML Header ve kterém je označena EBML verze 1 a typ dokumentu 2. Dále je Segment, ve kterém je definovaný TimestampScale, na 2 ms, to znamená, že časové značky v blocích a clusterech jsou v násobcích 2 ms. Segment obsahuje jednu video stopu – Track, ve které je zaznamenáno, že se jedná o prokládané video v rozlišení 640 na 360 px. Kódované pomocí Codec_ID = V_UNCOMPRESSED, v barevném prostoru ColourSpace = Y41P, což jsou nekomprimovaná data ve formátu planární YUV 4:1:1. Za tagem Tracks už následují jednotlivé clustery. Cluster zapouzdřuje 900 SimpleBlock, kde každý blok obsahuje data lichého a sudého půlsnímku. Vzhledem k relativně nízkému počtu Top-Level prvků (Info, Tracks a 13 clusterů) nebyl implementován SeekHead blok.

5.4 Zdrojové kódy

Kompletní zdrojové kódy a kompletní projekt je možné najít na přiloženém DVD nebo v online repositářích: https://dkkk@dev.azure.com/dkkk/CaptureCard/_git/CaptureCard. Pro repositář CaptureCard, který obsahuje zdrojové kódy vytvořené ve Vivado a Xilinx SDK, je finální comit s SHA1 hashem: cf85d4ad8d059eab3e5574464c98420f837c6099 v branche Thesis_Final. Pro repositář CaptureCard_SupportProjects, který obsahuje projekt s aplikací PixelMasking, je finální comit s SHA1 hashem: 387ba98b5ea1d24c719ab5505b2db4ee1173e313 v branche Thesis_Final. Všechny verze, které by byly v repositářích vytvořeny po těchto výše zmíněných jsou už další postup práce. Příslušnost zmíněných bloků ke zdrojovým souborům je v tabulce 6.

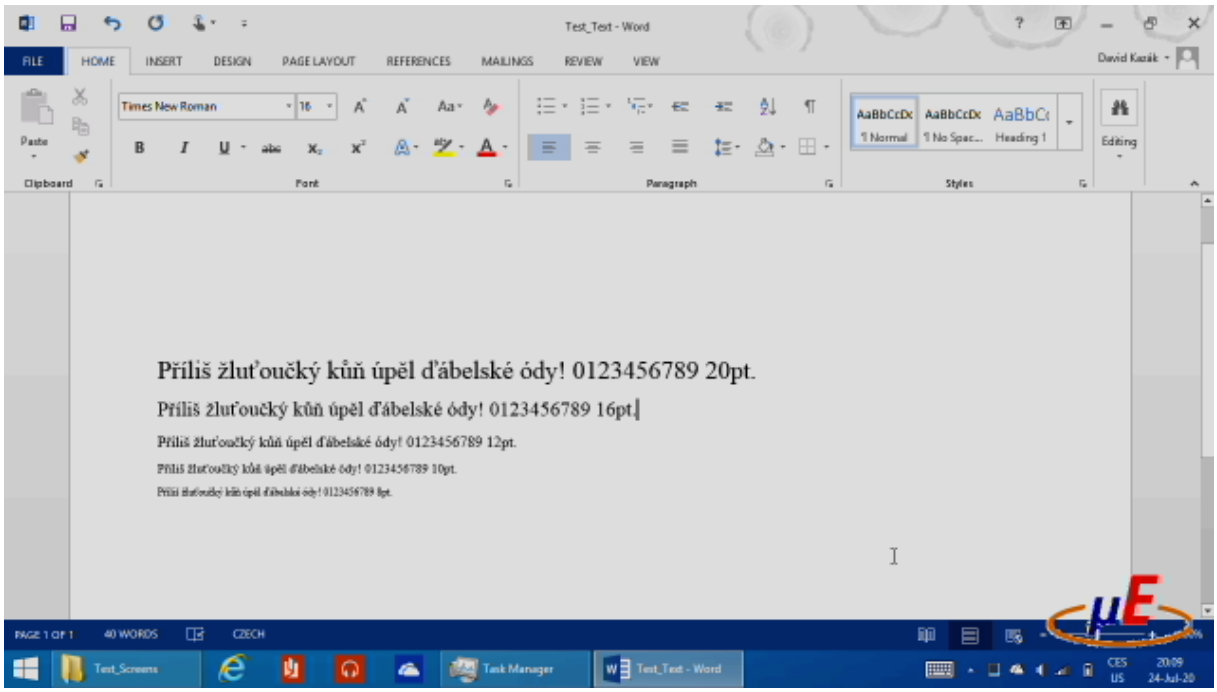
Tabulka 6: Seznam popisovaných bloků a k nim přiřazených zdrojových souborů.

Název bloku	**Název souboru**	**Jazyk**	**Nadřízený blok**
Capture Card	Capture_Card.vhd	VHDL	Top level blok
HDMI-RGB	HDMI_RGB.vhd	VHDL	Capture Card
DVI2RGB	Digilent DVI-to-RGB (Sink) 1.9 IP [26]	IP Core	HDMI-RGB
RBG_RGB	RBG_RGB.vhd	VHDL	HDMI-RGB
Object Adder System	Object_Adder_System.vhd	VHDL	Capture Card
Object Adder	Object_Adder.vhd	VHDL	Object Adder System
Mask RAM	Block Memory Generator v8.4 LogiCORE IP [32]	IP Core	Object Adder System
Image1 RAM	Block Memory Generator v8.4 LogiCORE IP [32]	IP Core	Object Adder System
Image2 RAM	Block Memory Generator v8.4 LogiCORE IP [32]	IP Core	Object Adder System
WhiteBalancer System	WhiteBalancer_System.vhd	VHDL	Capture Card
WhiteBalancer_UI	WhiteBalancer_UI.vhd	VHDL	WhiteBalancer System
WhiteBalancer	WhiteBalancer.vhd	VHDL	WhiteBalancer System
Color Temperature Changer	Color_Temperature_Changer.vhd	VHDL	WhiteBalancer System
RGB-HDMI	RGB_HDMI.vhd	VHDL	Capture Card
RGB_RBG	RGB_RBG.vhd	VHDL	RGB-HDMI
RGB2DVI	Digilent RGB-to-DVI (Source) 1.4 [27]	IP Core	RGB-HDMI
Recorder System	Record_System.vhd	VHDL	Capture Card
CaptureFrame Signal Generator	CaptureFrame_Signal_Generator.vhd	VHDL	Recorder System
Pixel Buffer	Block Memory Generator v8.4 LogiCORE IP [32]	IP Core	Recorder System
Resolution_32_8	Resolution_32_8.vhd	VHDL	Recorder System
Resolution_4_1	Resolution_4_1.vhd	VHDL	Resolution_32_8
RGB24_YUV411	RGB24_YUV411.vhd	VHDL	Recorder System
Y	Y.vhd	VHDL	RGB24_YUV411
U	U.vhd	VHDL	RGB24_YUV411
V	V.vhd	VHDL	RGB24_YUV411
YUV Buffer	Block Memory Generator v8.4 LogiCORE IP [32]	IP Core	Recorder System
Matroska Writer	Main.c, SimpleBlock.c a Clusters.c	C	Recorder System
PixelMasking	CaptureCard_SupportProjects.sln, Main.vb, Picture.vb, Mask.vb a Pixel.vb	Visual Basic 2019	Nezávislý blok

6 Dosažené výsledky

V této kapitole vyhodnotím dosažené výsledky: vlastnosti, výhody, nevýhody zvoleného postupu. Dále se pokusím navrhnout úpravy a zlepšení mého řešení.

Model záznamového zařízení podporuje zpracování videosignálu o rozlišení 1280 x 720 px a snímkové frekvenci 60 Hz, dále umožňuje korigovat barvy toho signálu a vkládat do něj dva malé objekty, každý o rozlišení 40 kpx. Tento videosignál je možné zaznamenávat na microSD kartu do kontejneru Matroska v rozlišení 640 x 360 px a půlsnímkové frekvenci 30 Hz.



Obrázek 24: Ukázka záznamu s vloženým logem na bílém pozadí. S převodem bílého pozadí na transparentní.

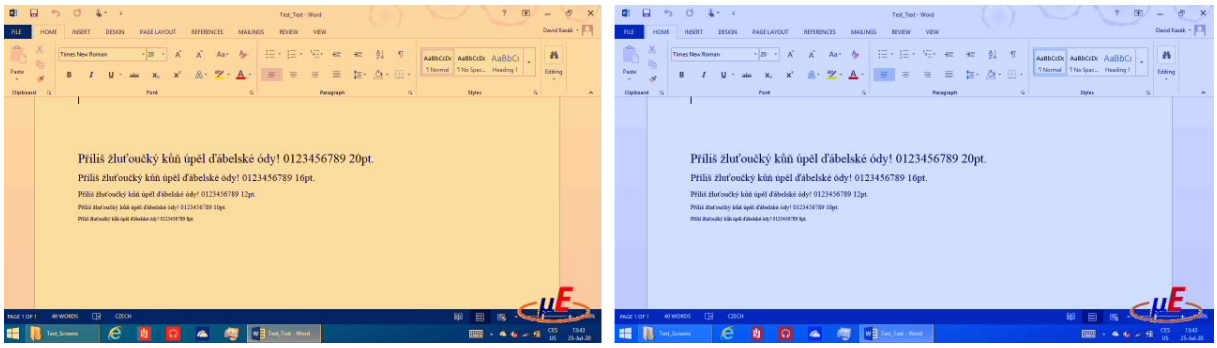
Realizovaný design má odhadovanou spotřebu SoC 2,3 W[10]. V době, kdy není k vývojovému kitu Zybo Z7-20 připojený žádný zdroj HDMI signálu, odpojí se většina bloků od hodinového signálu a model záznamového zařízení se přesune do úspornějšího režimu.

Model využívá 6140 LUT, což je 11,5 % dostupných LUT v Zynq 7020, 6686 klopných obvodů, to je 6,3 % dostupných klopných obvodů. Dále využívá 132,5 bloků blokových pamětí,

[10] Odhad po implementační strategii Vivado Performance_ExtraTimingOpt 2019, reportovaný pomocí Vivado Implementation Defaults Reports 2019.

což odpovídá 95 % dostupných blokových pamětí. Systém také používá dva ze čtyř dostupných MMCM a jeden ze čtyř dostupných fázových závěsů.

Na obrázku 24 je ukázka loga katedry mikroelektroniky na bílém pozadí. Bílé pozadí bylo odstraněno pomocí aplikace Pixel Masking. Barva, která se používá jako klíč, musí být monochromatická, protože pokud je potřeba jako v příkladě výše, změnit bílé pixely obrázku na transparentní, tak se změní pouze pixely, které mají přesně vybranou hodnotu, v tomto případě RGB 0xFFFFFF. Ale například barva 0xFFFFFE, kde je jen o LSB, méně modré barvy, v obrázku zůstane. Beze změny 0xFFFFFF na transparentní by tato nedokonalost nebyla pro lidské oko rozpoznatelná, ale v případě že jsou okolní pixely odstraněny a v obraze zůstanou pixely s nedokonalostmi, které mohou působit rušivě. Příklad tohoto je vidět na obrázku 24, kde vzniká kolem loga katedry mikroelektroniky zdánlivě bílý rámeček, který by měl být také odstraněn.



Obrázek 25: Příklad možností změny vyvážení bílé. Vpravo obraz posunutý k teplejším barvám. Vlevo ke studenějším.

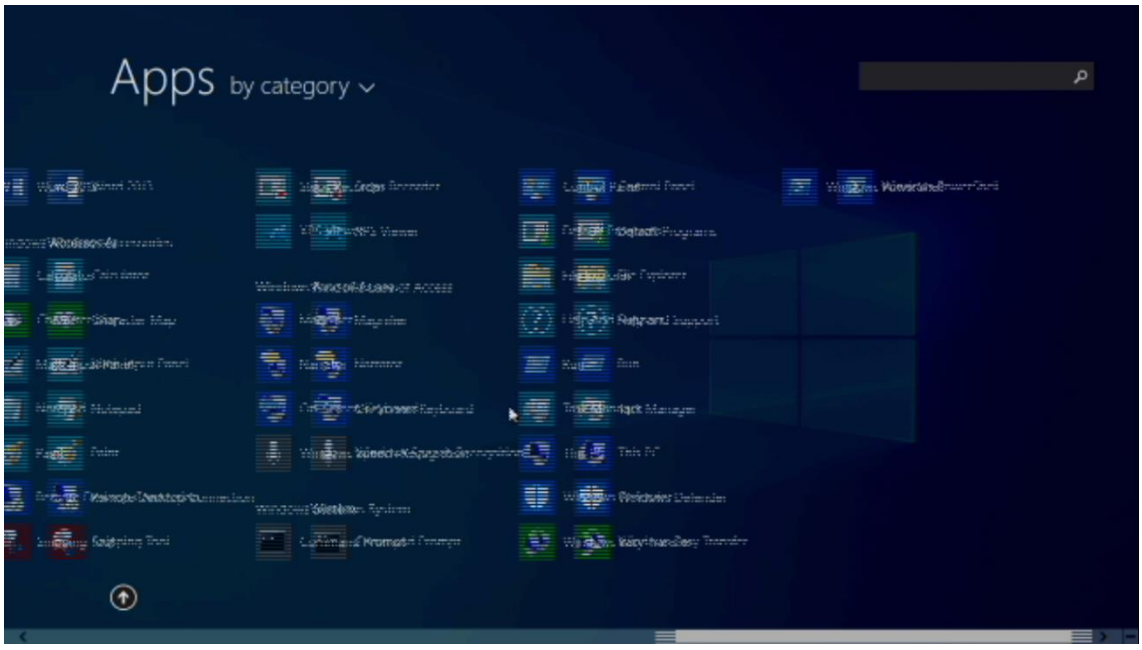
Vkládání objektů do obrazu pomocí bloků je uživatelsky velmi jednoduché, ovládá se pouze čtyřmi přepínači. Až do doby, kdy je potřeba s objektem posunout, aby se zobrazoval v jiné části obrazovky. V tomto případě je nutné v počítači v aplikaci Pixel Masking znovu importovat oba objekty, vygenerovat coe soubory. Tyto coe soubory se poté v Xilinx Vivado použijí jako výchozí stav pro IP jádra blokových pamětí. Poté je nutné provést syntézu a implementaci a vygenerovat nový bitstream celého řešení, včetně opětovné syntézy zmíněných Out-of-Context IP jader blokových pamětí. Z dat z Vivado je nutné vygenerovat hardware specifikaci pro Xilinx SDK, kde je po jejich importu potřeba vygenerovat nové board support packages a provést novou kompilaci programu. Poté je možné naprogramovat vývojovou desku Zybo Z7-20 a zobrazit objekt na novém místě. Celý tento proces trvá přibližně 15 až 20 minut. Zybo Z7-20 nemá dostatek tlačítek na implementaci kurzorových kláves, které by bylo možné použít na jako vstupy od uživatele pro posun objektů. Jedna možnost, jak posun objektů

zjednodušit, je vytvořit mezi Zybo Z7-20 a počítačem konzoli přes sériový port, přes kterou by bylo možné posílat příkazy, jako právě zmíněný posun objektů. Druhá možnost je dokoupit k vývojovému kitu Zybo Z7-20 moduly s tlačítky a přepínači do Pmod konektoru.

Tato sériová konzole by byla také vhodná pro snadnější ovládání bloku WhiteBalancer System, který má své ovládací tlačítka, ale uživatel nemá dostatečnou zpětnou vazbu o tom, jakou úroveň má zvolenou. Vidí pouze výsledek na externím monitoru, což je ve většině případů dostatečné.

Možnosti bloku WhiteBalancer System jsou demonstrovány na obrázku 25. Vlevo je video obraz posunutý do teplejších barev, větší množství červené barvy a modrá je naopak utlumená. Na pravé straně obrázku 25 je naopak obraz posunutý ke studenějším barvám a zde je přidána modrá barva a podíl červené barvy je snížen. Jedná se o demonstrační ukázky, při běžném používání není doporučená takto velká změna barev. Neutrální stav je vidět na obrázku 24.

Postup ukládání videosignálu je popsán v kapitole 5.3, vlastnosti a nedostatky tohoto řešení jsou zmíněny v následujících odstavcích.

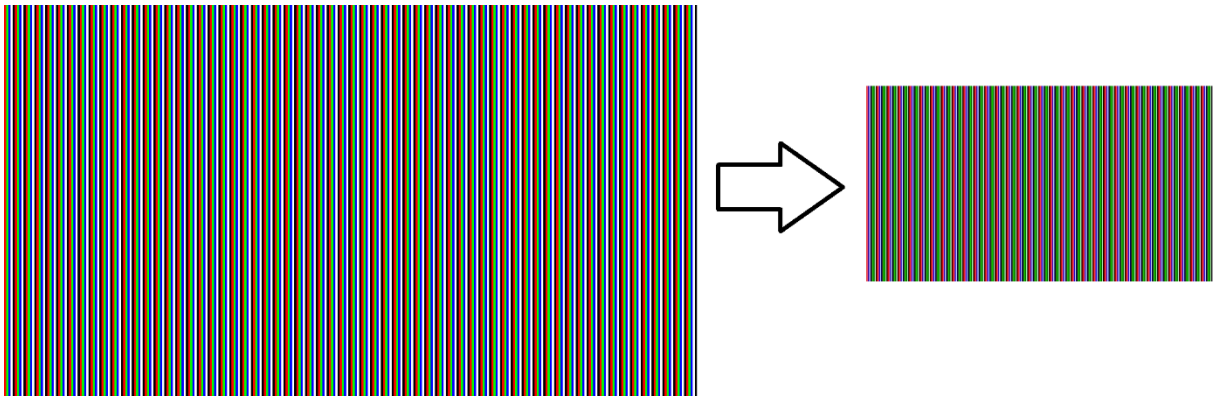


Obrázek 26: Ukázka problémů s rychle pohybujícími se objekty při prokládání.

Datový tok videa je podle rovnice (5.4) 39,55 $Mb \cdot s^{-1}$, což v kombinaci s maximální velikostí souboru v souborovém systému FAT32 a jedno minutovou implementací Matroska clusteru znamená, že je možné uložit 13 minut videozáznamu. Pokud by byl cluster kratší, bylo by

možné získat ještě dalších přibližně 30 sekund videa, pak už by se stejně projevil limit souborového systému FAT32. Každý cluster má zároveň svoji hlavičku, takže v případě velmi krátkých clusterů by se ukládalo větší množství neužitečné informace. Možnosti, jak zvýšit délku ukládaného záznamu, spočívají ve snížení datového toku videa, což je možné realizovat buď implementací nějakého video kodeku, nebo například dalším snížením rozlišení nebo barevné hloubky. Další možnost je pokračování do dalšího mkv souboru a provázat je spolu pomocí tagů NextFilename, NextUID a PrevFilename, PrevUID. Poté by byla délka záznamu limitovaná velikostí microSD karty.

Jedna z technik, která se používá při redukci datového toku videa, je snížení snímkové frekvence a zavedení prokládání snímků. Z původní snímkové frekvence 60 Hz na půl snímkovou frekvenci 30 Hz, které odpovídá snímková frekvence 15 Hz. Pokud jsou ve videosignálu objekty pohybující se nízkou rychlostí, je tato nízká snímková frekvence nerušivá a nezaznamenatelná. Ale na rychle pohybující se objektech se projevuje prokládání, kdy je objekt na lichých řádcích v jiné části snímku než jeho části na sudých řádcích. Příklad je vidět na obrázku 26, jedná se o zachycený snímek ze záznamu, který byl nahrán s úmyslem zachytit tento problém. Vzhledem k zaměření tohoto modelu záznamového zařízení pro potřeby výuky se dá předpokládat, že rychle pohybující se objekty se budou vyskytovat jen výjimečně.



Obrázek 27: Vlevo zdrojový obrázek v rozlišení 1280 x 720 px. Svislé pruhy 4x široké. Vpravo záznam tohoto obrázku, se slitými barvami vlivem snížení rozlišení a barevným pod vzorkováním 4:1:1. Obrázek je v plném rozlišení v přílohách.

Snímkové frekvenci 15 Hz odpovídá doba trvání snímku $66\frac{2}{3}$ ms. TimestampScale je zvolen na 2 ms, to znamená, že dvě nejbližší časové značky, které je možné zvolit, jsou 66 ms a 68 ms. Zvolil jsem 66 ms, protože je rozdíl od správné hodnoty nejmenší. Při přehrávání záznamu je tedy každý snímek o $\frac{2}{3}$ ms kratší a video se přehrává nepatrně rychleji, než bylo zaznamenáno.

Kumulativní chyba na minutě záznamu je 600 ms, což je 1 %. Tato chyba je neznatelná. Bylo by ji možné průběžně kompenzovat tím, že by každý třetí snímek měl periodu 68 ms.

Obrázek 28: Rozlití barev při barevné pod vzorkování.

Pokud se používá barevné podvzorkování, vznikají na hraně dvou odlišných barev pixely, které by již měly mít druhou barvu, ale barevné složky mají na vzorkovanou ještě z první barvy. Kombinací jasové složky z druhé barvy a barevných složek z první barvy vznikne třetí barva, která může být velmi odlišná od barev, jejichž smícháním vznikla. V tomto případě se při použití YUV 4:1:1 vzorkuje jasová složka pro každý pixel a chromatické složky se vzorkují každé 4 pixely. Oblast, kde vznikne třetí barva, která ve zdrojovém signálu nebyla obsažena, může být tedy až 3 pixely široká. Tento nedostatek nejvíce vyniká v extrémním případě, jako je na obrázku 27, nebo na hranách jako je na obrázku 28. Dále to také ovlivňuje čitelnost textu na barevném pozadí.

Obrázek 29: Porovnání textů na různých barvách pozadí. Plné rozlišení v přílohách.

Nejlepší čitelnost má černý text na bílém pozadí, protože bílá a černá jsou nechromatické barvy a liší se od sebe pouze jasovými složkami, barevné složky mají stejné. Text na bílém pozadí je čitelný i ve velikosti 10 pt[11]. Oproti tomu text na modrém pozadí má z testovaných pozadí

[11] Při rozlišení 1280 x 720 px.

nejhorší čitelnost a text není čitelný ani při velikosti 20 pt. Všechny testované texty můžete vidět na obrázku 29 a v přílohách.

Tyto problémy s barvami jsou vlastností barevného podvzorkování. Bylo by možné tento vliv částečně eliminovat zvolením jiného typu barevného podvzorkování. Například 4:2:0, která podvzorkovává jak horizontálně, tak i vertikálně a redukuje datový tok stejně jako 4:1:1. Takže chrominanční složky by byly společné pro čtverec se stranou 2 px, místo 4 pixelů v řádce. Nevýhoda podvzorkování 4:2:0 oproti 4:1:1 je komplikovanější realizace.

Příklad zaznamenaného videa je dostupný v přílohách.

6.1 Návrh dalšího postupu práce.

Další postup ve vývoji tohoto modelu záznamového zařízení se dá rozdělit do dvou směrů podle časové dotace na další vývoj. Směr, který je časově méně náročný, je cesta zlepšování uživatelské přívětivosti zařízení a úpravy některých nedokonalostí popsaných v kapitole 6. Například použití konzole přes sériový port pro snadnější ovládání nebo změna z YUV 4:1:1 na 4:2:0.

Pokud by byla časová dotace vyšší, bylo by možné kompletně přepracovat Recorder System blok, který je odpovědný za ukládání videozáznamu. Buď přesunem designu na platformu Xilinx Zynq Ultrascale+ SoC řady EV, která má H.264/H.265 enkodér a dekodér integrovaný jako pevné jádro. Recorder System blok by se vytvořil v PL kolem tohoto enkodéru. Přesun na novou platformu by trval odhadem řádově 2 až 3 měsíce práce. Další možnost je zůstat na platformě Zybo Z7-20 se SoC Xilinx Zynq 7020 a vyvinout vlastní H.264 enkodér. To je velmi náročný úkol, vývoj enkodéru by trval hrubým odhadem 1 rok práce. V obou případech by bylo možné zachovat bloky zpracovávající videosignál procházející na externí monitor a využít část bloků z Recorder System, například převodník z RGB na YUV.

Dalšími možnostmi rozšíření práce je zpracovávat i audio signály, které k záznamu patří. Bylo by možné zpracovávat audio signál z datových ostrovů zdrojového HDMI a z externího mikrofonu připojeného pomocí standardního 3,5 mm jack konektoru. Do datové kontejneru Matroska by bylo možné ukládat každou audio stopu samostatně, nebo je spojit do jedné stopy, podle potřeb a přání uživatele.

7 Závěr

V této práci jsem navrhnul a realizoval model záznamového zařízení na systému na čipu Xilinx Zynq 7020. Tento model cílí na využití jako studijní pomůcka a na oblasti, kde není možné zaznamenávat obraz softwarovou cestou.

Model se umisťuje mezi počítač a externí monitor nebo projektor a je s nimi propojen přes rozhraní HDMI. Vstupní signál je podporován v rozlišení 1280 x 720 px a snímková frekvence 60 Hz. Model umožňuje zaznamenat až 13 minut nekomprimovaného videosignálu v rozlišení 640 x 360 px při půlsnímkové frekvenci 30 Hz, ve formátu pixelů YUV 4:1:1 do datového kontejneru Matroska. Limity jsou detailněji popsány v kapitole 6.

Hlavním problém při tvorbě práce byl způsobený tím, že systémy na čipu Xilinx řady Zynq 7000 nemají na rozdíl od řady Zynq Ultrascale+ EV integrovaný video kodér. Existují licencované kodéry, které jsou ale licenčními poplatky mimo možnosti této práce a je vhodné je používat pouze u komerčních řešeních. Vývoj vlastního video kodéru, pro H.264 nebo pro podobný video kodek, je mimo rozsah této práce.

Pro další postup této práce je doporučeno vyvinout vlastní video enkodér nebo změnit platformu na Xilinx Zynq Ultrascale+.

8 Literatura

[1] K. Parnell a N. Mehta, Programmable Logic Design Quick Start Handbook, Xilinx, 2004.

[2] P. Wilson, Design Recipes for FPGAs Using Verilog and VHDL, Elsevier Ltd., 2016.

[3] L. H. Crockett, R. A. Elliot, M. A. Enderwitz a R. W. Stewart, The Zynq Book Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC, Glasgow, Scotland, UK: University of Strathclyde, 2014.

[4] Xilinx, „Zynq-7000 SoC Technical Reference Manual (UG585) (v1.12.2),“ 1. 7. 2018. [Online]. Available: https://www.xilinx.com/.

[5] Xilinx, „Zynq-7000 SoC Data Sheet: Overview (DS190) (v1.11.1),“ 2. 7. 2018. [Online]. Available: https://www.xilinx.com/.

[6] Xilinx, „Vivado Design Suite User Guide: Design Flows Overview (UG892) (v2020.1),“ 8. 7. 2020. [Online]. Available: https://www.xilinx.com/.

[7] Xilinx, „UltraFast Vivado HLS Methodology Guide (UG1197) (v2020.1),“ 3. 6. 2020. [Online]. Available: https://www.xilinx.com/.

[8] D. O'Loughlin, A. Coffey, F. Callaly, D. Lyons a F. Morgan, „Xilinx Vivado High Level Synthesis: Case studies,“ v *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies*, Limerick, Ireland, 26. - 27. 7. 2014.

[9] Xilinx, „Vivado Design Suite User Guide: Designing with IP (UG896) (v2020.1),“ 13. 7. 2020. [Online]. Available: https://www.xilinx.com/.

[10] M. H. Asmare, V. S. Asirvadam a L. Iznita, „Color Space Selection for Color Image Enhancement Applications,“ v *International Conference on Signal Acquisition and Processing*, 2009.

[11] M. Podpora, G. P. Korbaś a A. Kawala-Janik, „YUV vs RGB – Choosing a Color Space for Human-Machine Interaction,“ *Federated Conference on Computer Science and Information Systems*, pp. 29-34, 2014.

[12] G. H. Joblove a D. Greenberg, „Color Spaces for Computer Graphics,“ Cornell University.

[13] C. Reiter, „With J: Image Processing 2: Color Spaces,“ 6. 2004. [Online].

[14] C. Poynton, „Chroma Subsampling notation,“ 2002. [Online]. Available: http://vektor.theorem.ca/graphics/ycbcr/Chroma_subsampling_notation.pdf.

[15] Hitachi, Ltd., Matsushita Electric Industrial Co., Ltd., Philips Consumer Electronics, International B.V., Silicon Image, Inc., Sony Corporation, Thomson Inc. a Toshiba Corporation, „High-Definition Multimedia Interface Specification Version 1.3a,“ 10 11 2006. [Online].

[16] M. Jacobs a J. Probell, „A Brief History of Video Coding,“ ARC International, 2007.

[17] ITU-T, „Recommendation H.262: INFORMATION TECHNOLOGY – GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION: VIDEO,“ 2000.

[18] D. Ruiu, „An Overview of MPEG-2,“ v *The 1997 Digital Video Test Symposium*, 1997.

[19] T. Wiegand, G. J. Sullivan, G. Bjøntegaard a A. Luthra, „Overview of the H.264/AVC Video Coding Standard,“ v *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 13, NO. 7*, 2003.

[20] The Matroska association, „Matroska Media Container,“ 2005 - 2020. [Online]. Available: https://www.matroska.org/index.html. [Přístup získán únor až červenec 2020].

[21] R. Lhomme, D. Rice a M. Bunkus, „Extensible Binary Meta Language - draft-ietf-cellar-ebml-17,“ 27. 1. 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-cellar-ebml/. [Přístup získán 15. 3. 2020].

[22] M. A. Wilhelmsen, H. K. Stensland, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz a P. Halvorsen, „Using a Commodity Hardware Video Encoder for Interactive Video Streaming,“ v *IEEE International Symposium on Multimedia*, 2014.

[23] Xilinx, „Integrated Logic Analyzer v6.2 LogiCORE IP Product Guide (PG172),“ 5. 10. 2016. [Online]. Available: https://www.xilinx.com/.

[24] Digilent Inc., „Zybo Z7 Board Reference Manual,“ 12 02 2018. [Online]. Available: https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/start. [Přístup získán 19 03 2020].

[25] ITU-R, „Recommendation ITU-R BT.1543-1: 1 280 x 720, 16:9 progressively-captured image format for production and international programme exchange in the 60 Hz environment,“ 2015.

[26] Digilent Inc., „DVI-to-RGB (Sink) 1.9 IP Core User Guide,“ 13. 11. 2017. [Online]. Available: https://github.com/Digilent/vivado-library/blob/e7c8f2132fef5ca2d09c3856292af4c45e398ee0/ip/dvi2rgb/docs/dvi2rgb.pdf.

[27] Digilent Inc., „RGB-to-DVI (Source) 1.4 IP Core User Guide,“ 6. 9. 2017. [Online]. Available: https://reference.digilentinc.com/.

[28] Technical Committee SD Card Association, „SD Specifications Part 1: Physical Layer Specification Symplified Specification, Version 7.10,“ 25. 3. 2020. [Online]. Available: https://www.sdcard.org/downloads/pls/index.html.

[29] Microsoft Corporation, „Converting Between YUV and RGB,“ 30. 6. 2006. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/embedded/ms893078(v=msdn.10).

[30] Xilinx, „AXI Block RAM (BRAM) Controller v4.1 LogiCORE IP Product Guide (PG078),“ 22. 5. 2019. [Online]. Available: https://www.xilinx.com/.

[31] Microsoft Corporation, „Microsoft FAT Specification,“ 30. 8. 2005. [Online]. Available: http://read.pudn.com/downloads77/ebook/294884/FAT32%20Spec%20%28SDA%20Contribution%29.pdf.

[32] Xilinx, „Block Memory Generator v8.4 LogiCORE IP Product Guide (PG058),“ 9. 12. 2019. [Online]. Available: https://www.xilinx.com/.

[33] Xilinx, „Clocking Wizard v6.0 LogiCORE IP Product Guide (PG065),“ 25. 2. 2019. [Online]. Available: https://www.xilinx.com/.

[34] Xilinx, „Vivado Design Suite User Guide: Using Constraints (UG903) (v2018.3),“ 5. 12. 2018. [Online]. Available: https://www.xilinx.com/.

Seznam příloh

1. 01_Capture_Card_6.0.2.rar – Projekt ve Xilinx Vivado 2019.1 a Xilinx SDK 2019.
2. 02_CaptureCard_SupportProjects_1.2.0.3.rar – Solution ve Microsoft Visual Studio 2019 – Aplikace PixelMasking
3. 03_Příloha č. 03 - Příklad zaznamenaného videa.mkv – Ukázka záznamu.
4. 04_Příloha č. 04 - Obrázek 27.png – Obrázek 27 v plné velikosti.
5. 05_Příloha č. 05 - Obrázek 29.png – Obrázek 29 v plné velikosti.