

České Vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra mikroelektroniky

Diplomová práce

**Zpracování videosignálu
s pomocí FPGA**

Diplomant :

Pavel Typl

Vedoucí práce :

Doc. Ing. Pavel Hazdra, CSc.

Školní rok :

2004 / 2005

Děkuji Doc. Ing. Pavlu Hazdrovi CSc. za vedení, podporu a cenné rady v průběhu zpracování diplomové práce.

Abstract

Typl, P. Videosignal processing using FPGA.

This text discusses design and realization of a real time video compression system. The system has to digitalize and then process analog videosegnal in order to decrease a total bandwidth necessary to transfer videosegnal, while keeping an acceptable image quality. Such system can be proved useful in many applications, as presented in the introduction part of this text. Several ways how to decrease the transmitted data rate are discussed in following section. Next, the JPEG algorithm is described, since it makes a basis for majority of contemporary algorithms for image and video compression. Then, the Field Programmable Gate Arrays (FPGA) Virtex II, as the main implementation platform, are introduced together with VHSIC Hardware Description Language (VHDL) which is used for description and definition of designed system. The main part is aimed to describe the actual implementation, partitioning of the whole JPEG algorithm into sub-blocks and their own functionality. There one can find many useful block diagrams and tables, which helps him to understand functionality of each block. The realization itself is reviewed in the ending of this thesis, together with ideas for further development of the designed system.

Abstrakt

Typl, P. Zpracování videosegnálu pomocí FPGA.

Diplomová práce, Praha 2005

Práce se zabývá rozborem a návrhem systému pro kompresi videosegnálu v reálném čase. Systém má digitalizovat analogový videosegnál, a následně ho zpracovat pomocí obvodu typu FPGA tak, aby byl snížen datový tok digitalizovaného signálu při zachování jeho přijatelné kvality. V úvodní kapitole se může čtenář seznámit širokými možnostmi využití takového systému. Dále jsou uvedeny možné způsoby snížení datového toku a je popsán algoritmus JPEG, který je základním stavebním kamenem většiny algoritmů pro snížení objemu obrazových dat. V další části je představen obvod FPGA řady Virtex II, který je základem implementační platformy. V hlavní části práce pak hloubavější čtenář nalezne popis vlastní implementace, rozdělení funkcí mezi jednotlivé části systému a množství blokových schémat ilustrujících jednotlivé bloky. V poslední části je zhodnocena realizace celého systému a je navržen směr, kterým by se měla práce dále ubírat.

Obsah

1 Seznam použitých symbolů a zkratk.....	V
2 Úvod.....	1
3 Rozbor zadání.....	3
4 Komprese dat.....	5
4.1 Komprese obrazu a videa.....	8
4.1.1 Historie.....	8
4.1.2 Metody.....	8
4.1.3 Komprese JPEG.....	8
4.1.4 Rozšíření algoritmu JPEG.....	14
5 Návrh systému.....	15
5.1 Úvod.....	15
5.2 Implementační platforma.....	15
5.2.1 Kompresní algoritmus.....	17
5.2.2 Upřesnění.....	17
Analýza velikosti a charakteru pamětí.....	18
5.2.3 Vybraná platforma.....	21
FPGA.....	21
Programovatelné obvody Virtex II.....	23
5.2.4 Přípravek.....	25
Vývojový kit XSV-300 s FPGA Virtex.....	25
Vývojový kit V2LC1000 s FPGA Virtex II.....	25
Kompletní implementační platforma.....	26
5.3 Způsob návrhu.....	26
VHDL.....	28
6 Popis realizace.....	29
6.1 Návrhový systém.....	29
6.2 Popis systému na nejvyšší úrovni.....	29
6.3 Popis realizovaných bloků.....	31
6.3.1 Vývojová deska XSV-300.....	31
Externí bloky.....	31
Bloky realizované v FPGA.....	33
6.3.2 Propojení desek.....	37
BusDriver a Video Receiver.....	38
6.3.3 Vývojová deska V2LC1000.....	38
Paměť.....	38
DDR controller.....	40
Realizace DDR kontroléru.....	41
Memory Arbiter.....	44
JPEG kodér/dekodér.....	48
Analyzer.....	54
VGA.....	55
6.4 Zhodnocení realizace.....	56
7 Závěr.....	58
7.1 Dosažené výsledky.....	58
7.2 Doporučení pro další vývoj.....	58

1 Seznam použitých symbolů a zkratek

ABEL	Advanced B oolean E quation L anguage
ASIC	A pplication S pecific I ntegrated C ircuit
BPP	B its P er P ixel
CCTV	C losed C ircuit T V
CNN	C able N ews N etwork
CPLD	C omplex P rogrammable L ogic D evice
CVBS	C omposit V ideo B urst S ignal
DCM	D igital C lock M anager
DCM	D igital C lock M anager
DCT	D iscrete C osine T ransform
DDR RAM	D ual D ata R ate R andom A ccess M emory
DSP	D igital S ignal P rocessor
DV	D igital V ideo
FIFO	F irst I n F irst O ut
FPGA	F ield P rogrammable G ate A rray
FPS	F rames P er S econd
FSM	F inite S tate M achine
H	H eight - výška
I ² C	I nter- I ntegrated C ircuit
IO	I ntegrovaný O bvod
IOB	I nput O utput B lock
JFIF	J PEG F ile I nterchange F ormat
JPEG	J oint P hotographic E xpert G roup
KLT	K arhuhnen- L oève T ransform
LSB	L east S ignificant B it
LSI	L arge S cale I ntegration
MAC	M ultiply A CCumulate
MSB	M ost S ignificant B it
MSps	M ega S amples P er S econd
NTSC	N ational T elevision S ystem C omittee
PAL	P hase A lternating L ine
PHY	P H Y sical L ayer
RTL	R egister T ransfer L evel
SECAM	S ysteme E lectronique C ouleur A vec M emoire
TARGA	T ruvision A dvanced R aster G raphics A dapter
VGA	V ideo G raphics A rray
VHDL	V HSIC H ardware D escription L anguage
VHSIC	V ery H igh S peed I ntegrated C ircuit
W	W idth – šířka
XGA	E X T ended G raphics A rray

2 Úvod

V dnešní době se stále zvyšuje počet aplikací, kde je potřeba přenos videosignálu v reálném čase na střední až větší vzdálenost a/nebo z velkého množství zdrojů. Jsou to různé typy dohledových systémů jako například sledování provozu na silnicích, ostraha objektů i otevřených prostranství.

Zároveň se podstatně zvětšil počet kamer, které jsou instalovány na veřejných prostranstvích - jako prevence a odhalování zločinu, nebo v souvislost s hrozbou terorismu. Podle informací CNN z roku 2002 bylo v Anglii nainstalováno *jeden a půl milionu* dohledových kamer a průměrný Angličan byl zachycen kamerou více než 300 krát za den!

Navíc, se zvyšující se úrovní automatizace, se přenos obrazu stále více uplatňuje i při dálkovém řízení a kontrole výrobních procesů, zvláště pak v nebezpečných provozech, kde může být ohroženo lidské zdraví.

Realizace přenosu klasickým analogovým způsobem je náročná a může přinést řadu problémů (šum zesilovačů, rušení, zemní smyčky). Rovněž množství kabelů ve "sběrném bodě" může neúměrně narůstat a rozšiřování nebo přestavba systému je složitým krokem. Všechny tyto problémy může vyřešit digitální přenos obrazu. Umožňuje využití moderních technologií strukturované kabeláže nebo může dokonce s úspěchem využít stávající infrastruktury.

Výhodným řešením je přenos digitalizovaného videosignálu ve standardním formátu prostřednictvím běžných datových sítí. Tento přístup má následující vlastnosti v porovnání s analogovým přenosem:

Výhody :

- **Vyšší odolnost digitálního přenosu proti rušení**
- **Zjednodušení propojení**
 - Možnost připojit teoreticky neomezený počet kamer na standardní lokální síť Ethernet
 - Využití stávající infrastruktury lokální sítě Ethernet pro připojení kamer.
- **Modularita systému**
 - Přidání další kamery do místa s přístupem k síti je bezproblémovým a rychlým krokem.
- **Kamera je plnohodnotným členem sítě**
 - Sledování obrazu z kamery je možné na libovolném vybraném počítači v síti
 - Je možné dálkově řídit nastavení kamery, jako třeba natočení, ostření, závěrku atd.

Nevýhody :

- **Velký objem dat nutných přenést v reálném čase**
- **Nutnost dalšího HW**

Z výše uvedeného vyplývá, že výhody plynoucí z přenosu obrazu po síti je mnoho. Nicméně pokud se nesníží celkový objem přenášených dat, je systém nepoužitelný. Z těchto požadavků, snížení datového toku nutného pro efektivní přenos videosignálu v reálném čase, vychází zadání této diplomové práce. Jejím cílem je návrh elektronického systému, který v reálném čase zdigitalizuje a zkomprimuje videosignál na datový tok maximálně 5 Mbit/s se zpožděním maximálně 100 ms.

Svou práci jsem proto orientoval směrem k nalezení vhodného kompresního algoritmu, platformy a následně i implementaci tohoto algoritmu.

V první etapě nebyly k dispozici tak výkonné součástky jako nyní a většina úsilí směřovala k optimální implementaci výpočtu DCT. Tato práce byla prezentována a oceněna na konferenci POSTER[1] v roce 2003. Poté byla práce přerušena stáží v zahraničí. Po návratu byla k dispozici výkonnější platforma, která je nyní jádrem celého systému.

V první části jsou podrobněji rozebrány aspekty, které je nutné vzít v úvahu při návrhu systému podle zadání. V druhé části jsou uvedeny způsoby datové komprese, její klasifikace a příklady. Dále se práce zaměřuje specificky na kompresi obrazu - na algoritmu JPEG jsou představeny základní stavební kameny využívané kompresními algoritmy obrazového signálu a videa.

V hlavní části je představena vybraná implementační platforma a celý návrh systému. Textová část práce není a ani nemůže být detailním popisem navrženého systému, spíše popisuje principy funkce jednotlivých bloků a tok dat mezi nimi. Celý systém je dostatečně zdokumentován v elektronické podobě zdrojovými kódy a projektovými soubory návrhového systému na přiloženém CD.

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

3 Rozbor zadání

Zadání požaduje převedení analogového videosignálu z černobílé videokamery s rozlišením 720x576 pixelů na digitální datový tok 5Mbit/s. V následující analýze si ukážeme, jak velký je nekomprimovaný digitální datový tok z jedné kamery.

Analýza:

ČB kamera s těmito parametry :

Rozlišení (WxH)	720x576
Počet snímků za sekundu (fps)	25
Počet stupňů šedi (bits per pixel)	256 (8bitů)

Tab. 1

$$W = 720 \quad H = 576 \quad fps = 25 \quad bpp = 8$$

$$bitrate = H \cdot W \cdot fps \cdot bpp = 720 \cdot 576 \cdot 25 \cdot 8 \text{ Mbit/s} \approx \mathbf{83 \text{ Mbit/s}}$$

Rovnice 1

Z výsledků analýzy vyplývá že datový tok z jedné kamery je ohromný. Pokud by byla tato data přenášena po standardní 100Mbitové síti Ethernet, využila by v podstatě celou její kapacitu.

Výhody takového systému jsou minimální – modularita systému je stále malá, množství kabeláže velké a využití stávající infrastruktury je diskutabilní, jelikož data přenášená z kamery by drasticky snížila přenosové rychlosti mezi ostatními členy sítě.

Je proto zřejmé že aby byl systém použitelný, je *nutné* snížit datový tok. Snadno vypočteme že pro splnění zadání, které omezuje datový tok na 5Mbit/s, musíme dosáhnout kompresního poměru¹ 6% (Rovnice 2).

$$\frac{5 \text{ Mbit/s}}{83 \text{ Mbit/s}} \approx 6\%$$

Rovnice 2

Tohoto cíle lze dosáhnout několika způsoby uvedenými v Tab. 2.

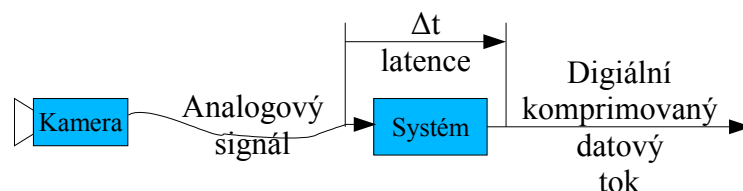
	Způsob	Změna parametru
1	Snížení rozlišení	180 x 144
2	Snížení snímkové frekvence	1,5 fps
3	Snížení počtu stupňů šedi	0,5(N/A)
4	Vhodnou kompresí obrazových dat	?

Tab. 2 - Způsoby snížení datového toku videosignálu a jejich dopad na parametry obrazu

1 Kompresní poměr je definován jako poměr $\frac{\text{komprimovaná velikost}}{\text{originální velikost}}$

Všechny uvedené varianty snižují kvalitu přenášeného obrazu. Degradace parametrů podle bodů jedna až tři nebo jejich kombinací je markantní ale i přesto může být pro některé aplikace dostačující. Avšak tam kde je potřeba vysoká kvalita přeneseného signálu, nebo rychlá odezva je nutné zvolit vhodnou kompresi obrazových dat, která sníží objem přenášených dat na požadovanou úroveň a zachová přitom kvalitní obraz.

Proto ve své práci použiji snížení datového toku pomocí bodu čtyři. Požadavek aby komprese probíhala v *reálném čase* klade nároky na její rychlost – Všechny bloky musí zpracovávat data alespoň takovou rychlostí, jakou jsou dodávána.



Obr. 1 - Latence

Maximální zpoždění (specifikované zadáním na 100ms) limituje latenci celého systému. V tomto případě je definována jako čas který uběhne od navzorkování vstupního analogového signálu do jeho zakódování.

Latence je dána zpožděním v kompresní datové cestě signálu. Podle toho s jakými bloky dat pracuje kompresní algoritmus je určitá latence vnesena již nutností uspořádat vstupních dat. Orientační hodnoty této části latence jsou uvedeny v Tab. 3.

Základní kompresní blok	Typická latence
Celý snímek	40 ms
Půlsnímek	20 ms
8 řádek v půlsnímku	35 us

Tab. 3 - Latence vnesená uspořádáním vstupních dat

Jak je vidět i při zpracování po celých snímcích je zpoždění vnesené uspořádáním vstupních dat maximálně 40 ms. Na vlastní algoritmus tak připadá latence 60 ms, což je více než dostačující.

Více limitujícím faktorem je spíše požadavek na kompresi dat v reálném čase a dosažení požadovaného kompresního poměru.

4 Komprese dat

Komprese je metoda, která umožňuje zmenšit objem dat a přitom zachovat jejich informační hodnotu.

I přes neustálé zvyšování propustnosti datových sítí (a kapacity záznamových médií) je šířka pásma stále cennou komoditou a nelze s ní plýtvat. Proto je při přenosu i záznamu dat snaha odstranit jakoukoliv nadbytečnou informaci. Tento krok je označován jako komprese.

Nadbytečná informace ve vstupních datech nemusí být na první pohled zřejmá a liší se podle charakteru dat – jeho znalost je proto je při volbě vhodného algoritmu nutná.

Snaha přenášet informace co nejefektivněji je již velmi stará. Již v devatenáctém století Samuel Morse v dobře známé Morseově abecedě² použil jeden ze základních přístupů pro efektivní přenos informací – často přenášené symboly jsou reprezentovány kratšími kódy. Písmena která se často vyskytují v anglickém jazyce, jako například písmena E a T jsou kódovány jen jednou tečkou/čárkou, zatímco například H které se nevyskytuje v angličtině tak často je kódováno čtyřmi tečkami. Telegrafický přenos zpráv tak bylo rychlejší než kdyby byla všechna písmena kódována stejným počtem teček/čárek.

Klasifikace

Hlavní parametr komprese, který je nutné zvolit je, zdali je přípustné, aby se v přenosovém řetězci nějaká data “ztratila”. Podle toho dělíme kompresi na

- **Bezeztrátovou (lossless)** – Pouze odstraňuje redundance - Rozkomprimovaná data jsou naprosto identická s daty vstupními.
- **Ztrátovou (lossy)** – Není nutné aby byla data identická, ale je důležité zda je výsledný dojem po komprimaci - dekomprimaci co nejlíže vstupním datům z hlediska pozorovatele

Podle toho jestli je náročnost kompresního a dekompresního algoritmu stejná nebo ne, dělíme kompresi na

- **Symetrickou** - Komprese i dekomprese je stejně náročná
- **Nesyetrickou** – Rozdílná výpočetní náročnost kompresního a dekompresního algoritmu (v naprosté většině případů je náročnější proces komprese)

Bezeztrátová komprese

Bezeztrátová komprese se používá se všude tam, kde nelze vynechat žádnou část informace, což je je typické pro binární data, textové nebo spustitelné soubory, řídicí signály apod.

V současné době jsou nejrozšířenější bezeztrátové komprimační metody založeny na následujících algoritmech, jejich nejrůznějších variantách kombinacích a modifikacích :

- Run Length Encoding (RLE) – Sekvence stejných symbolů ve vstupním řetězci jsou nahrazeny počtem jejich opakování [2].
- Huffmanovo kódování – Symboly které se vyskytují ve vstupních datech nejčastěji jsou kódovány menším počtem bitů než symboly které se objevují zřídka [2].

2 Původní morseova abeceda nepoužívala jen tečku, čárku a mezeru, ale i dlouhou čárku a dlouhou mezeru. Morseova abeceda jak ji známe dnes je modifikace té původní - “mezinárodní morseova abeceda”



- Aritmetické kódování – Podobné jako Huffmanovo kódování, efektivnější ale výpočetně náročnější [2].
- Lempel-Ziv-Welch (LZW) – Zaměňuje opakující se řetězce symbolů za speciální symboly [2].

Ztrátová komprese

Ztrátová komprese je typická svým využitím pro přenos informací určených pro zpracování lidskými smysly – sluchem a zrakem. Tyto informace se vyznačují obrovským datovým tokem, ale lidské smysly nejsou schopny využít všechny informace v něm obsažené. Objem těchto dat lze řádově zmenšit (Tab. 4) při minimálním vlivu na subjektivní vjem pozorovatele resp. posluchače. Ztrátová komprese tedy využívá znalosti nejen charakteru kódovaného signálu, ale i schopností pozorovatele k tomu, aby byla vypuštěná informace co nejméně znatelná.

Příklady komprese




V Tab. 4 je uvedeno několik příkladů komprese, kde byla komprese vybrána s ohledem na komprimovaná data.

<i>Typ komprese</i>	<i>Ztrátová</i>	<i>Originální velikost</i>	<i>Velikost po kompresi</i>	<i>Kompresní poměr</i>
PKZip "Regedit.exe"	Ne	74'736 bytes	32'892	44 %
MPEG Layer 3 (zvuk) "Enya – Wild Child"	Ano	40 Mbytes	5,5 MBytes	13 %
JPEG (obraz) "lena.bmp"	Ano	50 kBytes 	3 kBytes 	6 %

Tab. 4 - Příklady komprese

Hodnoty v Tab. 4 jsou pouze orientační. Bohužel poměr komprese – subjektivní vjem se nedá přesně kvantifikovat (existují ale přístupy které s o to pokoušejí [3]) V dnešní době má například s formátem MP3 vlastní zkušenosti téměř každý a může si udělat vlastní názor co se týče kvality.

Na příkladu uložení ručně psaných poznámek si ukážeme že nevhodnou volbou kompresního algoritmu je možné data znehodnotit a ni tím nic nezískat oproti jinému algoritmu.

<i>Originál</i>	<i>JPEG</i>	<i>Run Length Encoding (TIFF)</i>
 114 kBytes	 3 kBytes	 2,6 kBytes

Tab. 5

Pozn. : Tento příklad lze brát i jako varování. Formát JPEG si získal takovou popularitu, že je používán téměř každým a na všechno. Pro fotografie je jeho použití ideální, ale pro dvoubarevné dokumenty je naprosto nevhodný. V okolí ostrých hran vznikají artefakty, které znepříjemňují čtení a dosažený kompresní poměr není o nic lepší než při použití formátu TIFF.

4.1 Komprese obrazu a videa

4.1.1 Historie

Začátkem osmdesátých let vznikl kodek pro přenos videosignálu COST211. Ten byl založen na principu DPCM(diferenční pulzní kódová modulace) a byl standardizován CCITT(Comite Consultatif Internationale de Telegraphie et Telephonie) jak standard H.120. Stanovený maximální datový tok byl 2MBit/sec(pro Evropu). Pro zvýšení kvality bez překročení požadovaného datového toku by bylo potřeba kódovat jeden pixel méně než jedním bitem, což vedlo ke kodekům založeným na kódování bloku.

Koncem osmdesátých let bylo v ITU-T(dříve CCIT) přijato 15 návrhů na kodeky pro videoconferencing. Čtrnáct z nich bylo založeno na Diskrétní Cosinové Transformaci(DCT) a jeden na Vektorové Kvantizaci. DCT bylo tudíž zvoleno jako základ komprese pro videokonferencing. Ne náhodou jiná organizace, JPEG(Joint Photographic Expert Group) , zvolila DCT taktéž jako základ pro kompresi statického obrazu.

Kodeky založené na DCT prokázaly velké zvýšení kvality oproti H.120. V roce 1989 byl standardizován H.261 - Data v jednotlivých snímcích jsou komprimována s pomocí DCT, rozdíl mezi snímky pomocí DPCM.

Začátkem devadesátých let MPEG(Motion Picture Expert Group) začalo vyvíjet kodek pro kompresi, záznam a přehrávání videa. Jelikož v tomto případě není nutná komprese v reálném čase, bylo možné použít komplexnější kompresní algoritmus. Algoritmus použitý H.261 byl rozšířen o "odhad pohybu" a tím se podařilo zvýšit korelaci mezi snímky a dal tak vzniknout standartu MPEG-1

V současnosti jsou stále nejpoužívanější standardy založené na MPEG-2 [4], což je rozšíření algoritmu MPEG-1. Pro zvýšení kompresního poměru jsou používány i dvouprůchodové algoritmy, ale jejich použití pro přenos signálu v reálném čase je nemožné. Další, zatím nepřilíš rozšířenou variantou, je komprese založená na tzv. wavelet transformaci.

4.1.2 Metody

Většina metod používaných pro kompresi obrazu, ať již statického nebo dynamického, využívá základní algoritmy stejné jako je metoda pro kompresi statického obrazu algoritmem JPEG. Proto nejdříve vysvětlím jednotlivé kroky JPEG a teprve následně uvedu rozšíření která dala vzniknout dalším algoritmům pro kompresi videa.

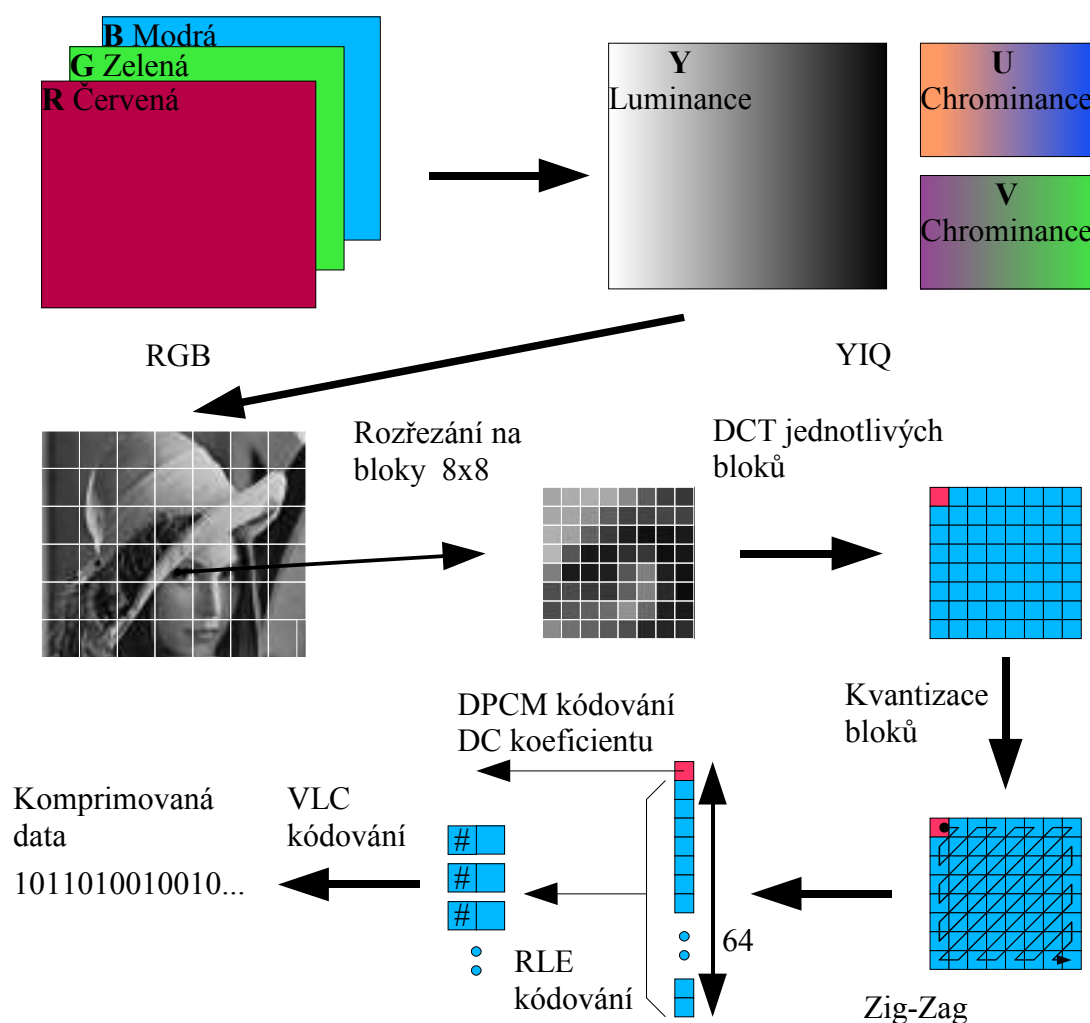
4.1.3 Komprese JPEG

Zkratka JPEG označuje **J**oint **P**hotographic **E**xpert **G**roup. Jedná se o soubor algoritmů (standart po formát souborů obrazu komprimovaného touto metodou je označen jako JFIF[5]) určených pro kompresi jak barevného tak grayscale obrazu. Komprese je symetrická a může být ztrátová i bezztrátová, ovšem síla JPEGu je kompresi ztrátové.

Přehled algoritmu

Algoritmus JPEG, který je popsán diagramem na Obr. 2 sestává z následujících kroků :Konverze obrazu z prostoru RGB do prostoru YUV

1. Rozdělení obrazu na bloky o velikosti 8x8 pixelů, odstranění stejnosměrné složky
2. Dvourozměrná Diskrétní Cosinová Transformace jednotlivých bloků
3. Kvantizace výsledných koeficientů
4. Zig-zag kvantizovaných koeficientů
5. Zakódování AC i DC koeficientů



Obr. 2 Přehled jednotlivých kroků algoritmu JPEG

Převod barevného prostoru

Konverze barevného prostoru z klasické reprezentace pomocí tří hlavních barev červené, modré a zelené do reprezentace pomocí jasové složky (Y-luminance) a dvou barevných složek (U,V - chrominance). Využívá se zde nedokonalosti lidského oka, které je nejcitlivější na jasovou složku a barevné složky vnímá hůř (To si snadno ověříme za šera, kdy ještě není problém rozeznávat předměty, ale přesné rozpoznávání barev je již značně oslabeno).

Této nedokonalosti je využito již ve standardu PAL (Phase Alternating Line) pro televizní vysílání a videozáznam. U televizního přenosu toto umožnilo použít nižší šířku pásma pro složky chrominance. Při digitální kompresi to umožňuje komprimovat tyto složky s nižší přesností, což vede na vyšší kompresní poměry.

Rozřezání

Pro další zpracování je obraz “rozřezán” na bloky o velikosti 8x8 pixelů a další operace jsou prováděny identicky s každým z těchto bloků.

Hodnoty jasové složky i barevných složek jsou v intervalu 0 až 255. Pro optimální hodnoty koeficientů po transformaci je vhodnější reprezentace -128 až 127, což provedeme odečtením 128 od hodnoty každého pixelu.

2D Diskrétní Kosinová Transformace

Dalším rysem oka je jeho vyšší citlivost na pozvolné změny obrazu než na změny “náhlé”. Pro usnadnění rozdílného nakládání s různými složkami je vhodné obraz převést transformací která má

- ✓ Optimální koncentraci energie (informace)
- ✓ Kompletně dekoreluje signál

Takovéto vlastnosti právě splňuje transformace KLT (*Karhunen-Loève Transform, Principal Component Analysis,...*) která má ale bohužel i další následující negativní vlastnosti :

- x Bázové funkce závisí na vstupním signálu
- x Je neseparovatelná
- x Výpočetně velmi náročná

Nevýhody KLT jsou značné (zvláště pak její výpočetní náročnost), což její použití v praxi značně omezuje. Optimální je použít transformaci, která se co nejvíce blíží pozitivním vlastnostem KLT, ale je výpočetně jednodušší.

Tou je právě Diskrétní Cosinová Transformace. Ta ve dvourozměrném prostoru přiřazuje reálnému vzoru $f(i,j)$ reálný obraz $F[u,v]$ a její osmibodová forma používaná pro kompresi je definována jako

$$F[u, v] = \frac{1}{4} \cdot \sum_{i,j=0}^{N-1} \lambda(u) \cdot \lambda(v) \cdot \cos\left(\frac{(2i+1) \cdot u \cdot \pi}{16}\right) \cdot \cos\left(\frac{(2j+1) \cdot v \cdot \pi}{16}\right) \cdot f(i, j)$$

$$\lambda(x) = \begin{cases} \frac{1}{\sqrt{2}} & \dots x=0 \\ 1 & \dots \text{jinak} \end{cases}$$

Rovnice 3 - 2D DCT

Transformace je separovatelná – rovnici (Rovnice 3) můžeme převést na

$$F[u, v] = \frac{\lambda(u)}{2} \cdot \sum_{i=0}^{N-1} \cos\left(\frac{(2i+1) \cdot u \cdot \pi i}{16}\right) \cdot \frac{\lambda(v)}{2} \cdot \sum_{j=0}^{N-1} \cos\left(\frac{(2j+1) \cdot v \cdot \pi j}{16}\right) \cdot f(i, j)$$

Rovnice 4 Separovaná 2D DCT

Vlastnosti DCT jsou

- Dekoreluje signál typický pro obrazová data
- Koncentrace energie je nižší než u KLT, ale pro typická obrazová data dostačující
- Výpočetní složitost je přijatelná

Kvantizace

V tomto kroku dochází ke ztrátě informace. Blok je skalárně vydělen empiricky zjištěnou kvantizační maticí (Obr. 3) a zaokrouhlen. Informace na které je oko citlivé jsou tedy zachovány s větším rozlišením než informace které jsou vnímány méně.

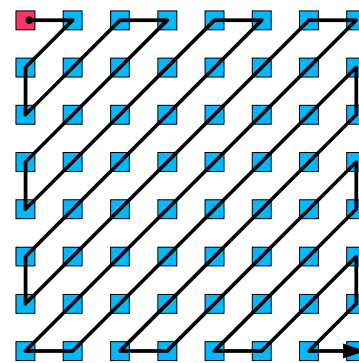
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Obr. 3 JPEG Kvantizační matice

Zigzag

V této fázi je blok převeden na vektor o délce 64. A to pomocí specifického postupu zvaného *Zig-Zag scan* (Obr. 4). Právě tento způsob převodu zajistí, že koeficienty budou ve výsledném vektoru seřazeny podle důležitosti. Na prvním, místě je stejnosměrná (DC) složka celého bloku.

Vysoké hodnoty koeficientů v pravé dolní části kvantizační matice (společně s vlastnostmi obrazových dat) mají ten efekt, že se ve výsledném vektoru se objeví posloupnosti nul. Nuly na konci vektoru které nepředcházejí žádnému koeficientu (trailing zeroes) jsou odstraněny (ignorovány).



Obr. 4

Differential Pulse Code Modulation

Velikost DC složky není ani tak moc závislá na ostatních koeficientech stejného bloku, jako spíše na velikosti stejnosměrné složky v sousedních blocích. Proto je stejnosměrná složka kódována jako rozdíl vzhledem k velikosti DC složky v předchozím bloku. Právě kódování rozdílu místo absolutní hodnoty je nazýváno *Differential Pulse Code Modulation* (DPCM)

Příklad:

Data 50 55 53 60 62 61 72 75 jsou DPCM zakódována jako +50 +5 -2 +7 +2 -1 +11 +3

Run Length Encoding

Run length encoding (RLE) je bezztrátová komprese. Kóduje sekvence stejných symbolů pomocí jednoho symbolu a počtu jeho opakování. Tato komprese je efektivní na specifický druh dat jako například

AAAAAA	BBBB	A	BBBBBBBB	AAAAAA	B	AAAAAA	...
6A	4B	1A	8B	6A	1B	7A	...

V následujícím případě k ekompresi vůbec nedojde, ba právě naopak.

A	B	A	B	A	BB	A	...
1A	1B	1A	1B	1A	2B	1A	...

Pro kompresi dat po zigzag scanu se používá varianta RLE, která bere v úvahu pouze sekvence nul. Pro tato data jsou totiž typické právě sekvence nul. Data jsou reprezentována jako počet předcházejících nul a hodnota. Počet nul je limitován šestnácti. Pokud je na vstupu sekvence šestnácti nul je kódována jako tzv. *zero overrun* a dále je pokračováno jako by začala nová.

15	12	0 0 6	0 0 0 16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 7	
0-15	0-12	2-6	3-16	0-0 (=zero overrun)	2-7	

Variable length kódování

Posledním krokem algoritmu JPEG je kódování s proměnnou délkou - Variable Length Coding (VLC). Bývá označováno buď obecně VLC, nebo jako huffmanovo kódování.

Jde o bezztrátovou kompresi založenou na kódování častých symbolů menším počtem bitů. Rozdíl DC koeficientu je kódován samostatně, AC koeficienty jsou kódovány jako dvojice. První člen představuje počet předcházejících nul a druhý hodnotu koeficientu. Zero overrun je kódován vlastní značkou dlouhou jedenáct bitů.

Pro získání výsledného binárního kódu se nejdříve zjistí tzv. kategorie (viz Tab. 6) podle absolutní velikosti AC koeficientu respektive difference DC.

<i>Hodnota</i>	<i>Kategorie</i>	
	<i>DC</i>	<i>AC</i>
0	0	-
-1,1	1	1
-3,-2,2,3	2	2
-7..,-4,4,..,7	3	3
-15,..,-8,8,..,15	4	4
..

Tab. 6 - Kategorie koeficientů

<i>Kategorie (počet bitů pro kódování hodnoty) C</i>	<i>Prefix kategorie</i>	<i>Celkový počet bitů</i>
0	010	3
1	011	4
2	100	5
3	00	5
4	101	7
5	110	8
6	1110	10
7	11110	12
..

Tab. 7 - Prefix DC kódů

Celkový počet bitů nutný pro uchování rozdílu DC koeficientu je nejnižší pro malé rozdíly. Stejnosečná složka se totiž v případě obrazových dat mění většinou pozvolna – je větší pravděpodobnost že rozdíl DC složky bude malý a tudíž kódován menším počtem bitů. Výsledná reprezentace kódovaného DC koeficientu se vytvoří sloučením prefixu kategorie a C LSBitů difference.

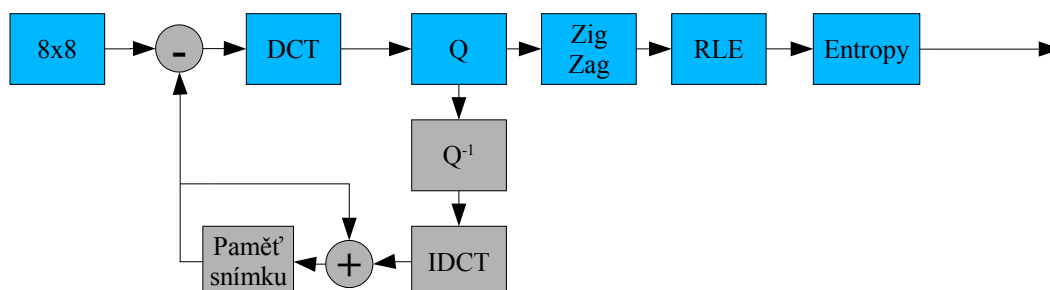
Příklad:

Aktuální DC koeficient je 200, DC koeficient minulého bloku byl 214. Rozdíl -14(binárně 11110010) spadá do kategorie 4. bude tedy kódován jako 101 0010.

Pro AC složku je přístup obdobný, s tím rozdílem že prefix je dán nejen kategorií, ale i počtem předcházejících nul. Ve chvíli, kdy jsou již všechny zbývající koeficienty nulové je blok ukončen značkou End Of Block (EOB).

4.1.4 Rozšíření algoritmu JPEG

JPEG je určen pro kompresi statického obrazu. Videosignál se dá komprimovat jako sekvence JPEG komprimovaných snímků – tzv. Motion – JPEG. Jelikož ten ale nebere v úvahu žádné souvislosti mezi jednotlivými snímky, není komprese tak velká jak by mohla být. Ve videosignálu obecně se často jednotlivé snímky velice podobají a toho se dá využít při kompresi.



Obr. 5 - Rozšíření algoritmu JPEG

Na Obr. 5 je uvedeno blokové schéma JPEG kodéru (modře) rozšířeného o další bloky, které jsou používány při kompresi videosignálu, ne statických obrázků.

První snímek je nejprve zkomprimuje a jednak odešle, ale zároveň je ihned dekomprimován a uložen do paměti snímku. Druhý příchozí snímek je nejprve odečten od prvního a teprve tento, takzvaný rozdílový snímek je zkomprimován a odeslán. Zároveň je opět dekomprimován, sečten s původním snímkem a uložen do paměti snímku.

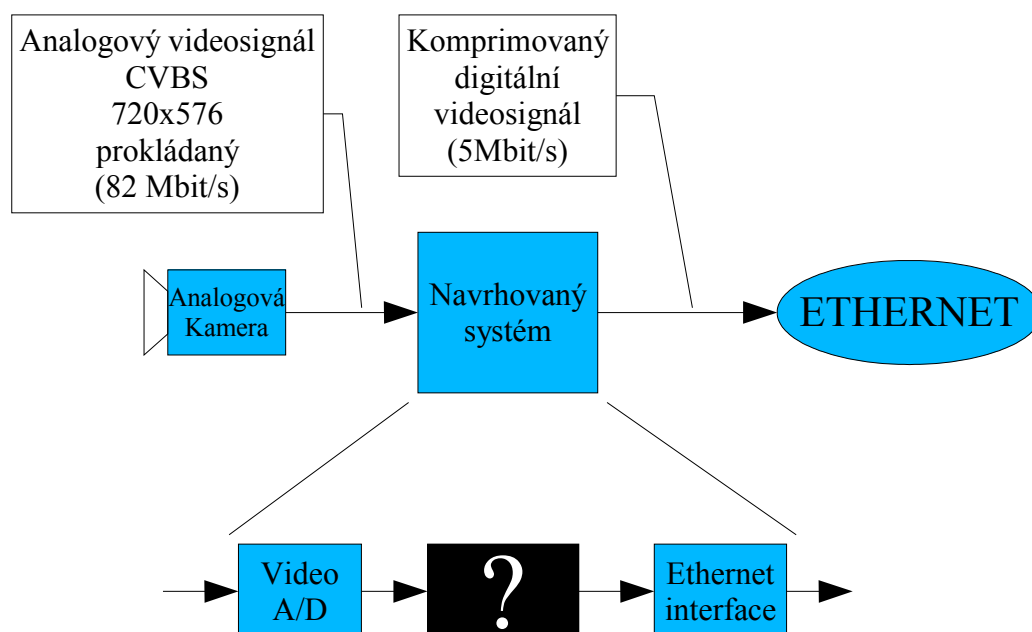
Rozdílový snímek je nutný dělat z nového snímku a snímku původního, již degradovaného kompresí-dekompresí. Jen tak se dá zaručit, že kodér i dekodér mají v paměti stejný minulý snímek a nedochází ke kumulaci chyby.

Existuje celá řada rozšíření, které vedou na vyšší kompresní poměr/kvalitu obrazu. Je to například kvalifikace bloku a podle toho zvolená kvantizační matice. Nebo zavedení vektorů pohybu(motion vector), kdy je zohledněn pohyb ve videosignálu atd. To vede na komprese typu H.263, MPEG [4] a další.

5 Návrh systému

5.1 Úvod

Dle zadání je cílem návrhu je převést analogový videesignál z černobílé kamery o rozlišení 720x576 pixelů (prokládaně) na digitální datový tok (maximálně 5Mbit/s). Přestože implementace bloku pro připojení k síti Ethernet není cílem této práce, uvedu ho zde. Může totiž hrát důležitou roli při výběru platformy, jelikož celý návrh směřuje k využití sítě Ethernet jako média pro přenos digitálních dat.



Obr. 6 - Kontextové schéma

5.2 Implementační platforma

Existuje několik různých variant, co může být jádrem systému. V Tab. 8 je shrnuto několik základních vlastností různých variant řešení.

	<i>Rychlost vývoje</i>	<i>Náklady na vývoj</i>	<i>Flexibilita</i>	<i>Další</i>
Osobní počítač	velmi vysoká	velmi nízké	velmi vysoká	objemný, drahý, velká spotřeba
ASIC	velmi malá	velmi vysoké	velmi nízká	výborné parametry
FPGA	střední	střední	vysoká	
DSP	vysoká	nízké	střední	
Dedikovaný IO	vysoká	nízké	nízká	

Tab. 8 - Varianty řešení a jejich vlastnosti

Z Tab. 8 je vidět, že co se týče vývoje a flexibility návrhu, je ideálním řešením použití osobního počítače. Jeho použití ovšem brání především velké množství potřebného prostoru a dále jeho vysoká cena. Také jeho výpočetní schopnosti jsou limitovanější než u ostatních, protože jde o obecný systém který není optimalizovaný pro prováděné operace. Nicméně pro zpracování M-JPEG je jeho rychlost dostačující, ale při využití pokročilejší komprese nemusí být schopen zpracování v reálném čase.

Použitím aplikačně specifického IO můžeme teoreticky dosáhnout nejlepších parametrů především co se týče rychlosti komprese (ne její kvality, ta je závislá na použitém algoritmu!), spotřeby a dalších. To je naneštěstí vykoupeno velmi nízkou flexibilitou a dlouhým, pracným, tudíž i nákladným vývojem. Jelikož výpočetní schopnosti dalších diskutovaných možností jsou na dostatečné úrovni pro danou aplikaci, použití ASIC nepřináší žádnou výhodu.

Další možnou volbou jsou obvody typu FPGA. Oproti obvodům ASIC jsou však reprogramovatelné, tudíž podstatně flexibilnější - vývoj systému s jejich použitím je rychlejší a levnější. Přitom si zachovávají stále velmi dobré parametry, hlavně z hlediska výpočetních schopností. Dalším rysem FPGA (a ASIC) je, že v případě dostatečné kapacity jsou schopny "absorbovat" další, původně samostatné externí IO. To bylo vlastně původní hlavní funkcí FPGA – tzv. Glue Logic.

V aplikacích se signálovými procesory se již dlouhou dobu používají vyšší programovací jazyky jako například C. Díky tomu je vývoj rychlejší a tím pádem i levnější. Jsou flexibilní, neboť jejich funkce se dá měnit prostou změnou software. Pevná architektura DSP může sice vést na vyšší operační frekvence, ale DSP pracuje v zásadě sekvenčně. Proto musí DSP zpracovat algoritmus krok po kroku, zatímco v FPGA i ASIC je možné provádět jednotlivé kroky algoritmu současně.

FPGA je pomalejší, protože propojení jednotlivých bloků není "hard-wired", ale nabízí možnost nezávislé a naprosto paralelní činnosti různých částí "čipu".

Implementace pomocí specifického integrovaného obvodu (kodeku) nabízí vcelku rychlý návrh celého systému. Ten ale musí obsahovat vlastní kodek a pak ještě minimálně řídicí mikroprocesor. Při uvážení komunikace po Ethernetu se systém rozroste o další integrované obvody. Nevýhodou takového řešení je hlavně jeho omezená flexibilita

Zadáním je určeno že pro tuto aplikaci má být použito FPGA. Po zvážení výše uvedených skutečností jsem dospěl k názoru, že jeho použití pro účely tohoto projektu je opodstatněné. K tomuto poznatku mě vedly jeho následující vlastnosti

1. Je flexibilní, rychlé a má dostatečné výpočetní schopnosti.
2. Jako designér si mohu volit mezi využitím systémových prostředků a rychlostí implementovaného algoritmu.
3. V jednom FPGA může spolupracovat několik návrhů – například kodér obrazu a blok pro komunikaci po síti Ethernet.

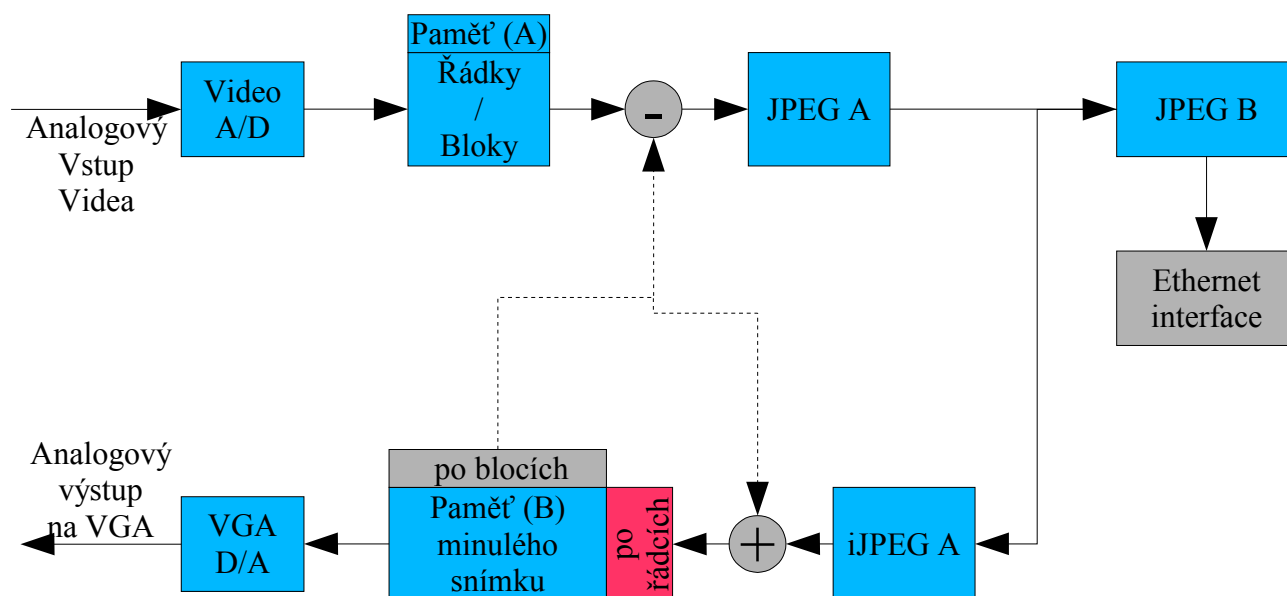
5.2.1 Kompresní algoritmus

Jako výchozí kompresní algoritmus jsem zvolil Motion – JPEG. Při jeho použití se dá dosáhnout požadované komprese alespoň na 6% původní velikosti. Dalším a hlavním důvodem pro volbu tohoto algoritmu je, že většina současných postupů při kompresi videosignálu je založena právě na stejných základních algoritmech jako JPEG. Díky tomu je možné systém lehce rozšířit a získat tak implementaci výkonnějšího algoritmu. Pokud to bude časově možné provedu první krok tímto směrem spočívající v použití rozdílového snímku.

5.2.2 Upřesnění

Nyní podrobíme analýze systém jehož klíčovým prvkem bude FPGA. Dodefinoval jsem některé požadavky na systém

- Testovatelnost – výstup na VGA
- Přípravenost na eventuální rozšíření systému na algoritmus s větším kompresním výkonem



Obr. 7 - Výchozí blokové schéma

Základem výchozího blokového schématu (Obr. 7) jsou bloky JPEG A a B. První provádí potřebné základní kroky algoritmu JPEG (DCT, Kvantizace, Zig-Zag). Druhý pak provádí bezztrátové komprese (RLE a VLC) a vytváří výsledný datový formát.

Protože je vstupní signál v analogové formě, je nutné ho nejprve zdigitalizovat, což provádí Video A/D převodník.

Navíc algoritmus pracuje s obrazovými bloky 8x8 což si vynucuje konverzi vstupního videosignálu (který je v řádkovém formátu) na blokový formát. To je funkce převodníku Řádky/Bloky, který nutně potřebuje paměť.

Jak požadavek na testovatelnost systému, tak požadavek na možnost rozšíření v budoucnu, mě vedl na vytvoření zpětnovazební větve. Umožňuje přímé sledování degradovaného obrazu na VGA monitoru. Z toho důvodu je ve zpětnovazební větvi blok iJPEG A, který provádí inverzní kroky vzhledem k bloku JPEG A. Zobrazení na VGA monitoru probíhá opět v řádkovém formátu, je tedy nutné signál opět konvertovat což je provedeno v paměti snímku.

Eventuální rozšíření systému je naznačeno šedivými bloky – jednak jde o možnost využití rozdílový snímek - bloky (+) a (-). Dále možnost připojení k síti Ethernet reprezentována blokem Ethernet interface.

Analýza velikosti a charakteru paměti

Celý systém musí nezbytně obsahovat paměť pro převod videosignálu v řádkovém formátu na bloky 8x8. Pro splnění dodatečného požadavku výstupu na VGA monitor je nutné použít i Paměť B. Provedu analýzu potřebných velikostí jednotlivých pamětí.

Paměť A

Jedná se o paměť, které slouží k převedení řádkového formátu na bloky pixelů 8x8, což jsou elementy se kterými se pracuje v dalších částech řetězce. Rozdělení do bloků je možné provést “dvěma variantami algoritmu”.

1. Bloky se tvoří zvlášť ze sudých zvlášť z lichých pulsů
2. Bloky se tvoří z kompletního snímku sestaveného z jednotlivých dvou pulsů

První varianta klade nižší nároky na velikost paměti a zároveň má nižší zpoždění (data je možné zpracovávat ihned po přijetí jednoho pulsů³). Paměť musí být schopna uchovat informaci minimálně o osmi řádcích. Výsledná minimální velikost je tedy

$$C = W \cdot 8 \cdot \text{bpp bitů} = 46\,080 \text{ bitů}$$

Protože data jsou zapisována podle jiného adresového schématu než jsou čtena, je zde nebezpečí přepsání nevyčtených dat. Předpokládejme že algoritmus je schopen zpracovávat data stejnou rychlostí jako je obdrží, pak musíme kapacitu paměti zdvojnásobit (čistě teoreticky může být kapacita o něco málo menší než dvojnásobek, ale potíže s řízením za ty komplikace nestojí).

V případě první varianty je konečná kapacita interní paměti 92 160 bitů. V použitém FPGA jsou k dispozici interní blokové paměti o kapacitě 16,384 kbitů. Na převodník řádky/bloky by bylo potřeba šest blokových pamětí. Použitá součástka jich má 40, jde tedy o využití 15ti procentní využití.

Ve variantě druhé je nutné aby paměť pojala celý snímek. Za použití stejného předpokladu jako u bodu jedna, totiž, že algoritmus zpracovává data stejnou rychlostí jakou je obdrží, musíme kapacitu opět zdvojnásobit. Výsledná kapacita je tedy

$$W \cdot H \cdot 2 \cdot \text{bpp} \approx 6,6 \text{ Mbitů}^4$$

V tomto případě již není možné použít vnitřní paměť a je nutné použít paměť externí.

Paměť B

Ať už je tato paměť použita v základní konfiguraci jen pro zobrazení na monitoru, nebo jako paměť minulého snímku, je bezpodmínečně nutné aby měla kapacitu minimálně 3,3 Mbitů (popřípadě 6,6 Mbitů, záleží jestli chceme mít na výstupu tzv. double buffering). Důvodem je odlišné časování monitoru VGA. Ten vykresluje obecně jinou část snímku než jaká je právě na výstupu z “algoritmu”.

3 Respektive po přijetí prvních osmi řádek

4 Pro úplnost musím uvést že tato kapacita není nejmenší možná. Její použití je ale přímočaré a rozdíl není “dramatický”.

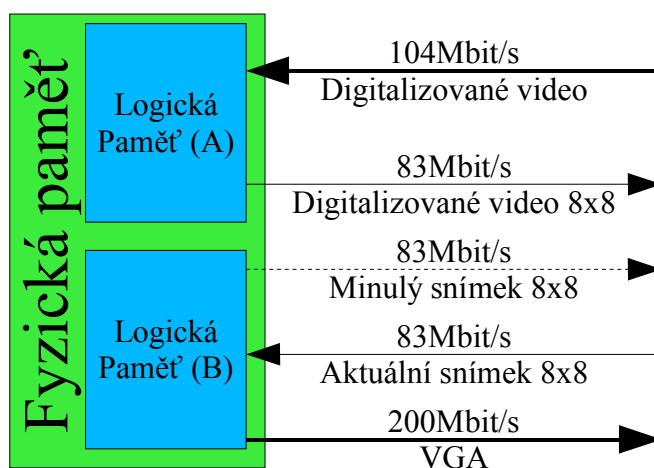
Závěr

Jako snímkovou je nutné použití paměť externí. Použití blokových pamětí pro převodník blok/řádek by spotřebovalo množství interních blokových pamětí, ale hlavně by zamezilo použití algoritmů pracujících s celým snímkem (místo pulsímku). Tyto algoritmy mají vyšší latenci, ale zároveň i vyšší kompresi protože jednotlivé pixely v bloku spolu přímo sousedí - jsou jsou tudíž více korelovány, což vede na vyšší kompresi. Proto je i pro převodník řádky bloky použita externí paměť.

Použití dvou externích pamětí by bylo neúnosné, ať již z hlediska ceny nebo kvůli omezenému množství vývodů FPGA. Řešením je použít paměť, která má dostatečnou kapacitu a zároveň je dostatečně rychlá, aby mohla být maskována jako několik samostatných pamětí.

Požadovaná kapacita paměti je tedy 3x velikost jednoho úplného snímku, tj. přibližně 13 Mbitů, při použití double buffering pro VGA.

Pro stanovení požadované propustnosti slouží diagram na



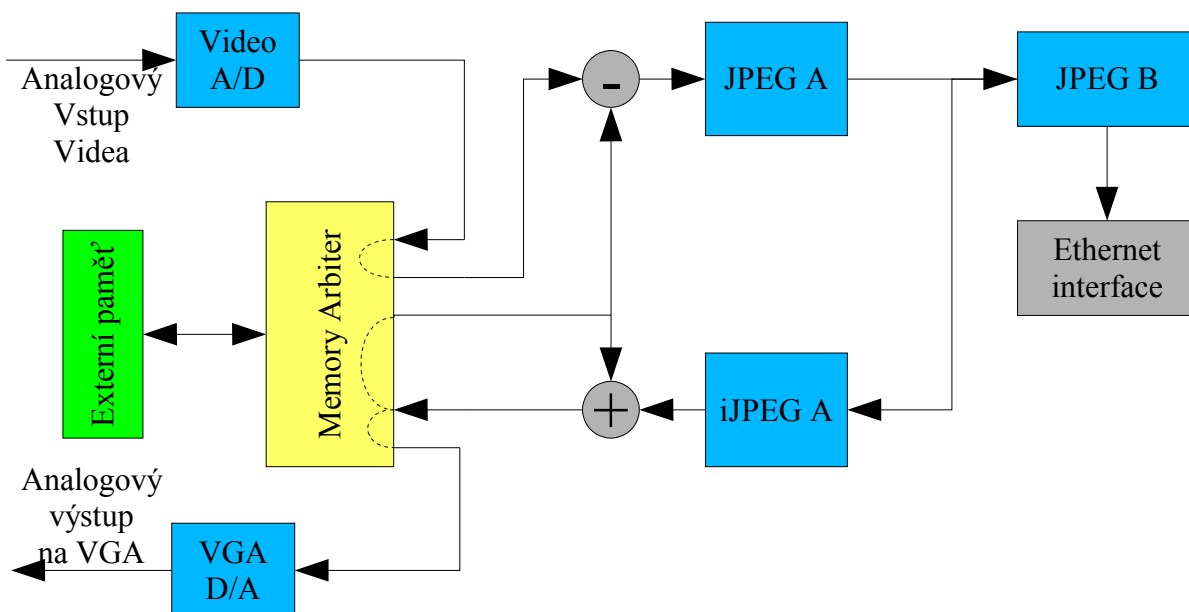
Obr. 8 Mapování fyzické paměti

Tučně vyvedené “kanály” jsou kritické, to znamená že data musí být do paměti uložena a paměť nemá možnost odložit jejich obsluhu. V těchto větvích musí být paměť umožňovat datovou propustnost na úrovni špičkového toku.

U ostatních větví stačí dodržet průměrnou datovou propustnost. Zde nedojde ke ztrátě dat, ale pouze ke zpoždění jejich zpracování. A dokud toto zpoždění nepřekročí kritickou hranici, pracuje systém bez problémů.

Sečteno a podtrženo, potřebná datová propustnost paměti musí být *minimálně* 906 Mbitů/s. Připojení paměti do designu nelze udělat přímo, ale musí být provedeno přes blok který obsahuje potřebné vyrovnávací paměti a řízení.

Tento blok, který jsem nazval “Memory Arbiter”, slouží k maskování jedné fyzické paměti jako dvou logických pamětí s několika nezávislými přístupovými kanály. Jeho začlenění do schématu je vidět na Obr. 9. Uvnitř bloku memory arbiter je pro přehlednost naznačen tok dat “skrz paměť”. Bloky pro převod mezi řádkovým a blokovým formátem byly pohlceny blokem memory



Obr. 9 Blokové schéma po přidání bloku Memory Arbiter

arbiter, protože je možné je realizovat pouze jako rozdílné adresovací schéma při čtení z paměti, než při zápisu.

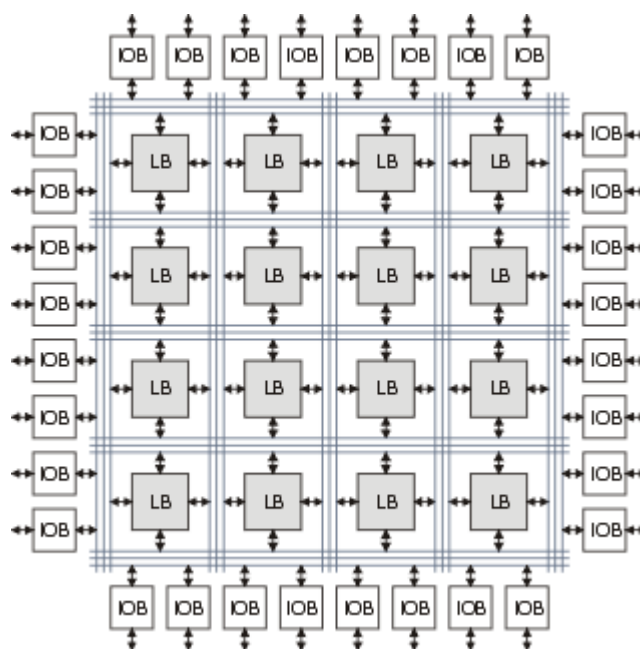
5.2.3 Vybraná platforma

FPGA

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

První FPGA měli kapacitu pouze v řádu tisíců ekvivalentních hradel a buď nahrazovali několik dříve hojně používaných obvodů typu LSI a dále byl využívány jako tzv. Glue Logic (přizpůsobení komunikační sběrnice mezi jednotlivými IO).

Typická struktura FPGA je znázorněna na Obr. 10



Obr. 10 - Obecná architektura FPGA

Základem každého FPGA je

- Matice Logických Bloků - Tyto bloky provádí jednoduché logické funkce. Složitější funkce se realizují ve více LB(Logický Blok) propojených najednou. Označení a těchto elementů se liší podle výrobce. Například Xilinx je označuje CLB(Configurable Logic Block), Altera jako LE(Logic Element) a Lattice jako PLC(Programmable Logic Cell)
- Vstupně/výstupní bloky(IOB) – komunikace s okolním světem
- Propojovací matice – Slouží jednak k propojení LB mezi sebou a také k propojení mezi LB a IOB

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

- Bloky interní paměti
- Bloky provádějící DSP operace
- Optimalizované propojení LB pro často používané funkce

- IOB podporujícím různé standardy.

Kapacita FPGA se udává *počtu systémových hradel*. Počet systémových hradel je pouze orientační – jejich počet závisí na zvoleném způsobu výpočtu a navíc výrobcům jde často i o marketing a z toho důvodu se snaží získat co největší číslo. Při znalosti struktury logických bloků má pro návrháře vyšší vypovídající hodnotu počet těchto logických bloků.

V současné době z přibližně deseti výrobců programovatelných obvodů vynikají dva výrobci a to Xilinx Inc. a Altera Corp. Ti pokrývají 60% trhu a vyrábějí FPGA na nejvyšší technické úrovni (dnes 90nm proces) schopné pokrýt požadavky mé aplikace.

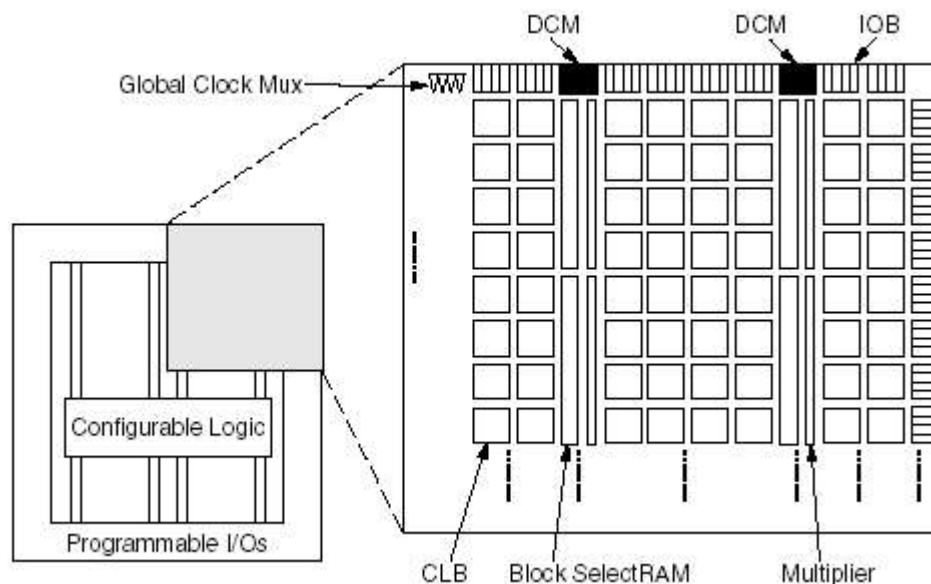
Jako realizační platforma byly vybrány FPGA řady Virtex a Virtex II od fy. Xilinx [6] z následujících důvodů :

- Měl jsem již nějaké zkušenosti s obvody firmy Xilinx
- Návrhový systém byl k dispozici
- Obvody Xilinx mají dobré vlastnosti pro DSP operace

Programovatelné obvody Virtex II

V této kapitole jsou popsány jen naprosto základní vlastnosti obvodů Virtex II. Mnoho dalších potřebných a užitečných informací je v katalogovém listu [7].

Jedná se o obvody typu FPGA od fy. Xilinx které mají kapacitu 40 tisíc až 8 miliónů ekvivalentních hradel. Jejich maximální kapacita současně s dedikovanými bloky v architektuře je předurčuje pro náročné aplikace, zejména pak zpracování signálu. Jejich vstupně-výstupní bloky je možné nastavit pro funkci podle různých IO standardů, což usnadňuje připojení různých periférií bez dalších komplikací.



Obr. 11 - Přehled architektury Virtex II

V náčrtu architektury je vidět, kromě již zmiňovaných logických bloků, zde nazývaných CLB (Configurable Logic Block) a IOB i několik dalších elementů :

- Block Select RAM – paměť s kapacitou 16kbitů s konfigurovatelnou šířkou slova. Přístup do paměti je možný dvěma zcela nezávislými bránami.
- Multiplier – Hardwarová násobička 18x18 bitů
- Digital Clock Manager(DCM) – Umožňuje kompenzaci zpoždění na hodinovém rozvodu, násobení a dělení a fázový posuv hodinového signálu.
- Global Clock Mux

Logický blok obsahuje čtyři ekvivalentní subbloky, nazývané “slice”. Které po dvou sdílejí ovládací signály (Clk,CE) pro registry. Struktura jedné poloviny jednotky slice (je až na drobné rozdíly symetrická) je znázorněna na Obr. 12.

5.2.4 Přípravek

Pro vývoj a realizaci systému byly použity dva vývojové kity osazené FPGA fy. Xilinx.

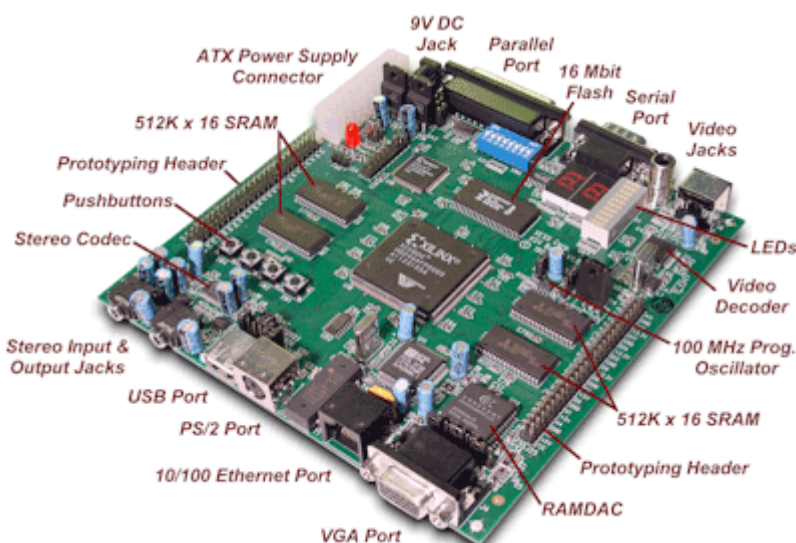
- XSV-300 s FPGA Virtex
- V2LC1000 s FPGA Virtex II

První jmenovaný obsahuje velké množství periférií, druhý je osazen výkonnějším hradlovým polem a dynamickou pamětí.

Vývojový kit XSV-300 s FPGA Virtex

Vývojová deska XSV-300 fy. XESS[8] obsahuje :

- FPGA Virtex 300 v pouzdře PQFP 240
- CPLD XC95108
- Programovatelný oscilátor
- 16 Mbit paměť typu FLASH
- Dvě nezávislé paměti 512k x 16 typu SRAM
- Videodigitalizér SAA7113 fy. PHILIPS
- RAMDAC převodník pro VGA výstup
- Stereo kodek pro digitalizaci audiosignálu
- Ethernet PHY
- a další (převodníky pro USB, RS232, konektory PS2,...)



Obr. 13 - Fotografie XSV-300

Tento přípravek obsahuje všechny potřebné periférie pro danou aplikaci – Videodigitalizér, VGA převodník a popřípadě i IO pro komunikaci po síti Ethernet. Tento vývojový kit byl použit v první fázi vývoje. V čase kdy byl zakoupen to byla nejlepší dostupná varianta.

Bohužel velké množství periférií s sebou přinášelo i nevýhody. Připojení velkého množství periférií vyžadovalo sdílení sběrnic, které zabraňuje současnému používání některých “periferií”. Jako omezující se ukázalo sdílení sběrnice SRAM a videodigitalizéru, které nedovoluje použít obě paměti současně s digitalizérem. V tomto případě je již celková kapacita paměti na této desce nedostačující (při rozšířených požadavcích definovaných v sekci 5.2.2)

Vývojový kit V2LC1000 s FPGA Virtex II

Vývojová deska V2LC1000 fy. Insight[9] je osazena :

- FPGA Virtex II 1000 v pouzdru BGA 240
- DDR 32MBytes

Za deskou XSV-300 sice zaostává co se týče periférií, avšak z pohledu výpočetního ji velice rychle předčí. Důvodů je hned několik :

1. Je osazena druhou generací řady Virtex – Ta nejen že může pracovat na vyšší operační frekvenci, ale má i další vylepšení pro efektivnější zpracování DSP úloh.
2. Paměť má větší kapacitu a vyšší propustnost sběrnice.

Kompletní implementační platforma

Pro vývoj jsem zvolil kombinaci obou vývojových kitů využívaje předností každého z nich. Deska XCV-300 slouží ke správnému nakonfigurování všech periférií, digitalizaci videosignálu a zobrazení na VGA, zatímco deska V2LC1000 provádí všechny operace s již zdigitalizovaným videosignálem. Paměť DDR, kterou je osazena deska V2LC1000 má dostatečnou datovou propustnost pro danou aplikaci.

Toto řešení dobře využívá dosažitelných zdrojů a způsobuje pouze minimální komplikace :

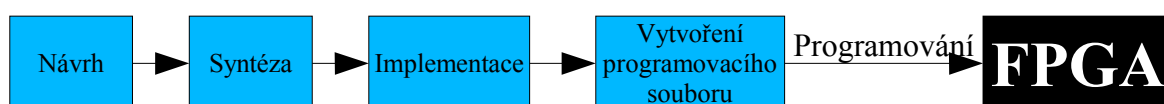
- Z hlediska vývoje : Vytvoření sběrnice s omezenou rychlostí mezi jednotlivými deskami.
- Z hlediska budoucího návrhu celého designu do jednoho FPGA : sloučení návrhu dvou FPGA do jednoho.

Řešení prvního bodu není náročné - k dispozici je dostatečné množství pinů pro propojení a rychlost sběrnice tudíž nedosahuje hodnot vyžadujících komplikované řešení.

Druhý bod komplikací vlastně ani není – sloučení designu z více FPGA do jednoho je elementárním krokem.

5.3 Způsob návrhu

Výrobce FPGA většinou dává k dispozici návrhové systémy, které provedou všechny kroky (orientačně jsou vidět na Obr. 14) od definice až po finální realizaci systému, která je reprezentována programovacím souborem (soubor obsahující všechny potřebné informace o nastavení jednotlivých programovatelných míst FPGA).



Obr. 14

Možnosti způsobu zadání návrhu se postupně vyvíjely - První systémy pro popis jednoduchých programovatelných součástek bylo možné navrhovat pouze přímým nastavením jednotlivých propojek. S přechodem na výkonnější součástky se objevila podpora jazyka ABEL, dále pak VHDL nebo Verilog a během posledních let se pak ještě přidaly systémy popisující chování FPGA na ještě vyšší úrovni.

Možnosti popisu funkce FPGA jsou

- **Přímé programování propojek** - Nepřipadá v úvahu pro obvody s kapacitou v řádu tisíců nebo dokonce miliónů ekvivalentních hradel. A o portabilitě takového návrhu se už vůbec nedá hovořit – je striktně vázaný na jednu jedinou součástku.

- **Schéma s použitím knihovních bloků** – Tento přístup umožňuje úplnou kontrolu nad uspořádáním v hardwaru. Nevýhodou návrhu je nejen jeho omezená přenositelnost, nedostatečné možnosti pro použití složitějších logických funkcí, ale i nízká flexibilita při provádění změn v návrhu.
- **ABEL** – Jednoduchý programovací jazyk na nižší úrovni. Má omezené možnosti a není vhodný pro součástky s velkým počtem systémových hradel.
- **VHDL, Verilog** – Programovací a simulační jazyky na úrovni RTL (Register Transfer Level – popisují tok dat mezi jednotlivými registry). Poskytují velmi dobrou kontrolu nad finálním uspořádáním v hardwaru, obsahují množství funkcí které usnadňují návrh a zároveň umožňují dobrou přenositelnost návrhu na mezi různými součástkami.
- **System-C, Handel-C** – Jedná se o vyšší jazyky. V době výběru byly teprve ve fázi léčení dětských nemocí, nyní se však již začínají uplatňovat i v praxi. I při použití těchto jazyků je jako mezikrok při implementaci použito VHDL a jeho znalost alespoň v základní podobě je nezbytná.
- **System Generator** – Nadstavba pro program Matlab-Simulink. Umožňuje návrh na vysoké úrovni. Obsahuje množství bloků převážně pro DSP operace, a další je možné dokoupit. Jeho výhodou je stejné prostředí pro návrh i simulaci a také možnost využít všech funkcí Matlabu (pro simulaci).

System Generator je komerční software a i jednotlivé bloky jsou obchodním artiklem. Navíc pro něj platí samé co pro vyšší programovací jazyky. Stále je tu jako mezikrok využito VHDL a podle mých zkušeností s tímto softwarem ze stáže na univerzitě v Linköpingu, je doladění na této úrovni nezbytné.

Po uvážení všech uvedených skutečností jsem dospěl k názoru že v dnešní době je pro digitálního návrháře znalost jazyka VHDL/Verilogu nutností ale zároveň stále ještě silným nástrojem pro návrh a simulaci digitálních obvodů. Rozhodl jsem se proto použít při návrhu právě tento programovací jazyk.

VHDL

VHSIC (Very High Speed Integrated Circuits) **H**ardware **D**escription **L**anguage. Volně přeloženo : *Jazyk pro popis hardwarového uspořádání velmi rychlých integrovaných obvodů.* Slouží k popisu chování a struktury elektronických systémů, a je vhodný a používaný pro popis struktury a chování FPGA nebo obvodů ASIC.

Jazyk VHDL byl vyvinut v osmdesátých letech na popud US. Ministerstva Obrany a v roce 1987 bylo ustanoveno jako IEEE 1076 standard. V roce 1994 byl tento standart rozšířen a dnes je znám jako IEEE standard 1076 1993. VHDL je stále dále vyvíjeno s pracovním označením VHDL-200x [10].

Simulace a syntéza jsou dvě základní funkce pro které se VHDL používá. Pro simulaci je možné využít všeho, co jazyk nabízí. Pro syntézu už jsme bohužel omezeni jen na podmnožinu VHDL. Z toho vyplývá že psát VHDL kód pro simulaci a syntetizovatelný kód jsou dvě *diametrálně odlišné záležitosti!* Omezení se týkají hlavně zatím nesyntetizovatelných konstrukcí jako například přístupu do souboru, operace s pointery a mnohé další.

Přestože samotná syntaxe VHDL je podobná rozšířeným programovacím jazykům, existuje zde *zásadní* rozdíl – jakožto HDL jazyk musí i VHDL nabízet paralelní zpracování. Tuto skutečnost si musíme uvědomit hned na začátku!

Co a jak napíšeme není v hardwaru reflektováno pouze *funkčně*, ale i *strukturně*. Výhodou je kontrola nad finálním uspořádáním v hardwaru. Zároveň můžeme bohužel nevhodným nebo laickým zadáním funkčně identických bloků zvětšit potřebnou plochu na čipu, zhoršit časové charakteristiky nebo dokonce obojí. Syntetizátory se sice stále zlepšují a jsou schopny identifikovat a optimalizovat zadání, ale zatím ne neomezeně.

VHDL je jazyk s velmi důslednou typovou kontrolou. Může se to zdát jako přítěž, komu by se pořád chtělo konvertovat typy přestože je to “úplně jasné”. Tato vlastnost je v konečném důsledku kladem protože se vyvarujeme mylného zpracování syntetizérem.

Existuje mnoho literatury popisující jazyk VHDL i několik kvalitních manuálů i tutoriálů na internetu[11]. Z nich je možné získat informace nejen o jazyku jako takovém, ale i rady co používat a čeho se naopak vyvarovat pro efektivní psaní syntetizovatelného kódu [12].

6 Popis realizace

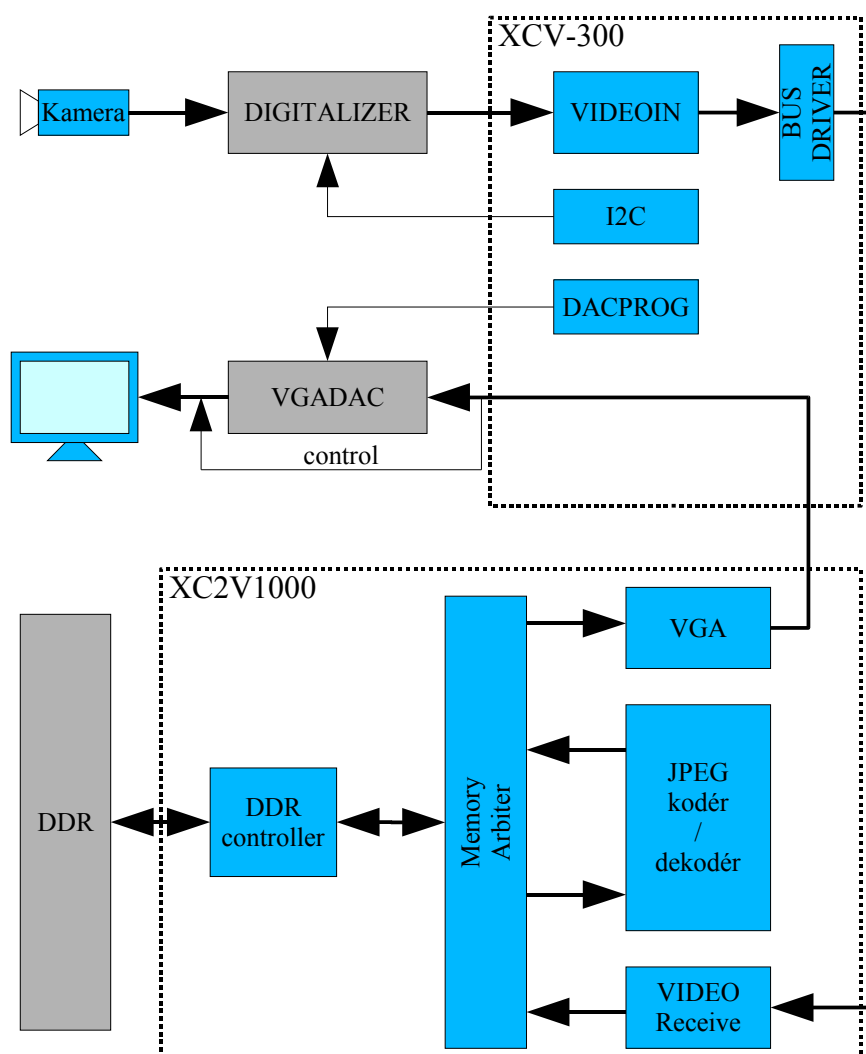
6.1 Návrhový systém

Návrh je proveden v plné verze systému Xilinx ISE, finální implementace používá verzi 6.1. Tento systém obsahuje všechny nástroje potřebné pro návrh, syntézu, implementaci i konečné programování. ISE sice neobsahuje vlastní simulátor, ale podporuje vazbu na externí, který je pro efektivní návrh nezbytný. Je použit Modelsim od fy. Modeltech. Pro kontrolu a simulaci na nejvyšší úrovni byl použit program MATLAB a Maple.

Celý návrh je vytvořen kompletně ve VHDL a je na příloženém CD. Správná funkčnost vybraných bloků byla ověřena v simulátoru (tam kde to rychlost simulace dovozovala).

6.2 Popis systému na nejvyšší úrovni

Celý systém je rozdělen mezi dvě vývojové desky. První, která obsahuje potřebné periferie jako je digitalizér videosignálu a video DA převodník a druhá, která provádí vlastní kompresi. Komunikace mezi oběma deskami probíhá po externí sběrnici. Na Obr. 15 je znázorněno zjednodušené blokové schéma celého zapojení.



Obr. 15 - Blokové schéma systému

Externí integrované obvody

- DIGITALIZER – Integrovaný obvod SAA7113H od fy Philips převádí analogový videosignál na formátovaný digitální tok (XSV-300)
- VGADAC – video analogové/digitální převodník (XSV-300)
- Paměť DDR TC59WM815BFT (V2LC1000)

Bloky které jsem realizoval v FPGA na desce XSV-300

- VIDEOIN - extrakce řídicích dat videosignálu a vlastní data videosignálu z datového toku produkovaného videodigitalizérem.
- I2C – jednorázová inicializace videodigitalizéru po sběrnici I2C.
- DACPROG – jednorázová inicializace rychlého, digitálně-analogového převodníku pro VGA.
- BUS DRIVER – Formátování dat pro přenos po externí sběrnici do druhé desky.

Bloky které jsem realizoval v FPGA na desce V2LC1000

- DDR controller – Řadič paměti typu DDR
- Memory arbiter – Do paměti přistupuje několik bloků. Arbiter zařizuje že všechna data pro/z každý blok jsou včas načtena/zapsána.
- VIDEO receive – Příjem zdigitalizovaných dat a jejich zápis do paměti
- VGA – Čtení dat z paměti a generování potřebných horizontálních a vertikálních synchronizačních impulsů pro VGA monitor.
- JPEG kodér

6.3 Popis realizovaných bloků

V této kapitole nejprve popíši nejprve funkci externích integrovaných obvodů a kamery. Poté popíši bloky které jsem realizoval v FPGA na desce XSV300, propojení s deskou V2LC1000 a nakonec bloky které jsem realizoval v FPGA na desce V2LC1000. Popis systému sleduje logický tok signálu .

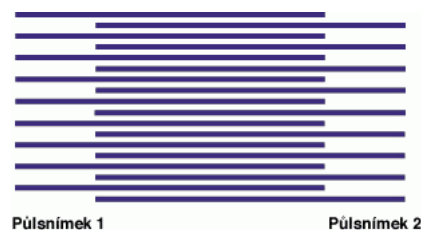
Kompletní bloků implementovaných v jednotlivých FPGA je proveden jazykem VHDL. Kompletní zdrojové kódy jsou k dispozici na příloženém CD.

6.3.1 Vývojová deska XSV-300

Externí bloky

Kamera

Signál z použité analogové kamery(MC420C) je přenášen ve formátu CVBS (Composite Video Burst Signal). Použité horizontální rozlišení je 720 pixelů, vertikální je 576 pixelů. Kamera má snímkovou frekvenci 25, ale obraz je přenášen jako 50 pulsnímků(Obr. 16).



Pulsnímek 1
Obr. 16

Pulsnímek 2

Videodigitalizér

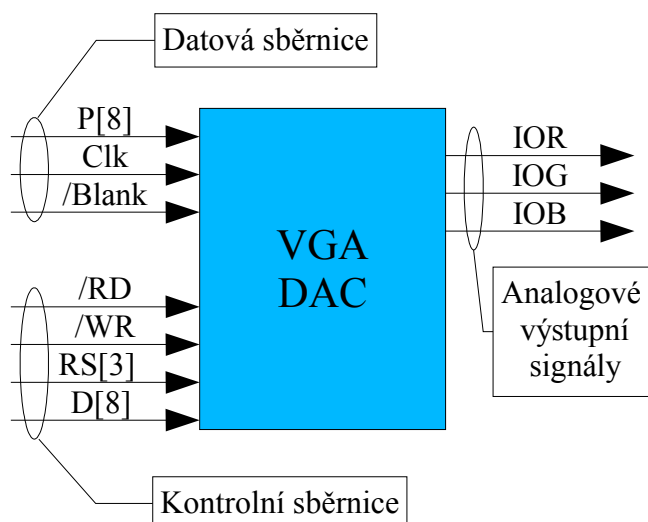
Jedná se o integrovaný obvod SAA7113H, což je multistandardní dekodér/digitalizér podporující standardy PAL, SECAM, NTSC. Na jeho vstup je připojen videosignál z kamery ve formátu CVBS, výstupem je digitalizovaný videosignál ve specifickém formátu(6.3.1/Tab. 11).

Obvod má velké množství registrů kterými je možné nastavit způsob zpracování videosignálu, jeho preprocessingu i formátu dat. Nastavení těchto registrů se provádí po sběrnici I²C. Popis jejich funkce je v dokumentaci obvodu [13], jejich nastavení je ve zdrojovém kódu (viz blok I²C).

VGADAC

Pro kontrolu funkce celého systému slouží monitor VGA. Pro zobrazení všech 256ti stupňů šedi je nutné použít DA převodník. Vývojová deska XESS obsahuje obvod určený přesně pro tento účel – BT481A vyráběný firmou Brokettree. Jedná se o trojnásobný 110 MHz DA převodník navržený na generování RGB signálů pro grafické adaptéry - každý převodník slouží pro jeden barevný kanál.

Navíc obsahuje paměť pro paletu 256ti barev plus 16ti barev pro overlay paletu. Piny pro její použití ale nejsou vyvedeny.



Obr. 17 VGA DAC

Hodnoty DA převodníků mohou být nastaveny několika způsoby :

- 5:5:5 – TARGA, 32k barev
- 5:6:5 – XGA, 64k barev
- 8:8:8 – True Color 16,8M barev
- 256 pseudo barev

První tři módy mají ještě několik variant nastavení sběrnice pro přenos intenzit jednotlivých barevných komponent (single edge, dual edge.. ap) Těmito formáty se zaobírat nebudu, ale lze je najít v dokumentaci obvodu [14].

Množství barev které tyto módy nabízejí je pro tuto aplikaci nadbytečné a navíc nás nutí k složitějšímu řízení a trojnásobné rychlosti sběrnice (Barevné složky jsou přenášeny sekvenčně). To je nepříjemné zvláště ve zvolené konfiguraci, kdy jsou data pro převodník přenášena po externí sběrnici mezi deskami.

Ideální pro tuto aplikaci je pseudobarevný mód. V tomto módu přivedeme na vstup převodníku číslo barvy a hodinový signál. Po náběžné hraně si převodník z vnitřní paměti vyčte odpovídající intenzity jednotlivých barevných složek a opět synchronně nastaví výstupní úroveň.

Pro práci v pseudobarevném módu je ovšem nutné předem nahrát do vnitřní paměti převodníku barevnou palte. Ta je naprogramována pomocí níže uvedeného bloku DacProg tak aby odpovídala 256ti úroňové stupnici šedi, což je provedeno blokem DacProg popsáním dále.

Bloky realizované v FPGA

Celá vývojová deska XSV-300 pracuje na frekvenci 27 MHz, která je získána z integrovaného obvodu videodigitalizéru. Díky tomu je fixní fázový vztah mezi hodinovým signálem využívaným FPGA a vstupními daty z videodigitalizéru, což usnadňuje jejich vzájemnou komunikaci.

Do každého bloku je zaveden globální resetovací signál a globální hodinový signál. Pro přehlednost nejsou tyto signály uváděny u každého bloku jednotlivě.

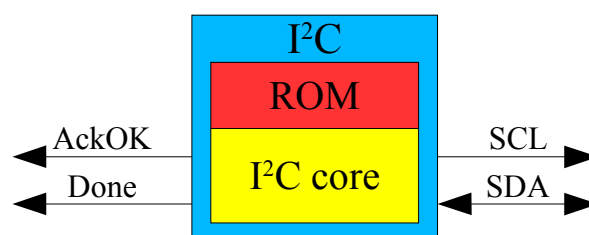
I²C

Blok I²C Slouží k inicializaci zařízení po sběrnici I²C [15] (Inter-Integrated Circuit). Tato sběrnice I²C byla vyvinuta firmou Philips a je dodnes velmi používaná nejen v procesorových systémech. Obsahuje pouze dva aktivní vodiče a to

- SCL – Serial Clock (jednosměrný)
- SDA – Serial Data (obousměrný)

Ve chvíli zapnutí nebo resetu celého systému vyše předem připravenou sekvenci (která je uložena v paměti ROM uvnitř FPGA) po sběrnici I²C. Signál Done indikuje ukončení celého konfiguračního procesu a signál AckOK jeho úspěšnost.

Tabulka (Tab. 9) zobrazuje formát dat v paměti ROM.



Obr. 18 - I²C

Adresa	0	1	2..1+n ₁
Data	Celkový počet bloků	Počet dat v prvním bloku (n ₁)	n ₁ x data prvního bloku	Počet dat ve druhém bloku(n ₂)	n ₂ x data druhého bloku	Počet dat ve třetím bloku(n ₃)	n ₃ x data třetího bloku	..

Tab. 9 - Formát dat v paměti ROM por inicializaci zařízení po sběrnici I²C

Start	Data prvního bloku	Ack Check	Stop	Start	Data druhého bloku	Ack Check	Stop	..

Tab. 10 - Sekvence dat vysílaná po sběrnici I²C

Blok nejdříve nastaví na sběrnici podmínku start a poté začne postupně vysílat data prvního bloku, přičemž po každém odeslaném bytu kontroluje jestli byl potvrzen(Ack Check). Po odeslání celého bloku ukončí přenos nastavením podmínky stop, poté znovu nastaví podmínku start a začne vysílat data z druhého bloku. toto se opakuje pro všechny bloky. Tímto způsobem můžeme efektivně inicializovat i více zařízení připojených na stejnou sběrnici.

V kódu VHDL je ROM definována jako bloková paměť s pevnými inicializačními hodnotami. Tyto hodnoty jsou nastaveny pomocí atributů Init_00, Init_01,...Při editaci těchto dat je nutné mít na paměti, že byte co bude v paměti na nejnižší adrese je v řetězci init_00 vpravo!

Přesné specifikace formátu I²C můžete nalézt v [15].

Videoin

Tento blok provádí extrakci dat ze “streamu” který je dodáván externím videodigitalizérem.

Základem celého bloku je stavový automat (na konci kapitoly je popsán i zobrazen na Obr. 20) který extrahuje jednak řídicí informace jako jsou

- Frame Start/End – začátek/konec pulsnímku
- Line Start/End – začátek/konec řádky
- Field – adresa pulsnímku (lichý nebo sudý)

a také označuje signálem DataValid jasové složky obrazu (kamera je černobílá, obě chrominance budou nulové).

Při použité konfiguraci videodigitalizéru vypadají přijímaná data následovně.

Blanking Period	Timing Reference Code						720 pixels Y:U:V 4:2:2						Timing Reference Code				Blanking Period	
	. 80 10	FF 00 00	SAV	C _b 0	Y0	C _r 0	Y1	C _b 2	Y2	. C _r 718	Y719	FF 00 00	EAV	80 10 .				

Tab. 11 - Formát dat z videodigitalizéru

Tab. 12 - Význam zkratek použitých v Tab. 11

SAV	Start of Active Video
EAV	End of Active Video
Y _n	Luminance pixelu n
C _b n	Chrominance _b pixelu n
C _r n	Chrominance _r pixelu n

Bit	7	6	5	4	3-0
Hodnota	1	lichý/sudý pulsnímeček	Vertikální zatemnění	0 pro SAV 1 pro EAV	rezervované bity

Tab. 13 - Formát značek SAV/EAV

Popis funkce stavového automatu(Obr. 20)

Stavy označené jako S_x_00 jsou pouze zpoždovací, vstupní hodnota v těchto stavech je nulová.

Po aktivním signálu Rst je stavový automat ve stavu “search” a čeká na hodnotu FF, která je předzvěstí značky”. Přes stavy S1_00 a S2_00 automat počká na vlastní značku. Hodnota “10” na bitech 5 a 4 značí začátek vertikální zatemňovací části videosignálu a právě pouze v tomto případě se stavový automat posune do další části.

Zde pomocí stejného způsobu jako v předchozí části najde značku a analyzuje jí ve stavu Video_VBLANK. Pokud má značka význam SAV – znamená to začátek viditelné části snímku, nastaví automat signály FS(Frame Start), LS(LineStart). Bit 6 označuje, jestli jde o lichý nebo sudý pulsniček a je uložen na výstup Field.

V této části videodigitalizér alternuje data pro jasovou a pro barevnou složku. Ve stavovém automatu je to reprezentováno smyčkou mezi stavy C (chrominance) a Y(luminance). Jasová data jsou “značkována” signálem DV (DataValid). Tuto smyčku automat opustí ve chvíli předzvěsti značky do stavu S5_00, zároveň nastaví signál LE(Line end)

Přes další stavy se dostane až do S8010 a v tom setrvá po celou dobu zatmívací periody. Příchod další značky dovede automat postupně až do stavu SAV, kde je podle zatemňovacího bitu (5) rozhodnuto jestli jde o začátek další řádky, pak se automat vrátí do smyčky C<->Y nebo jde o konec celého snímku - automat do stavu StartVBlank. Tady už se situace opakuje a automat čeká na začátek dalšího snímku.

DacProg

Tento blok konfiguruje po resetu VGA D/A převodník fixní paletou – stupnicí šedi.

To je prováděno po kontrolní sběrnici převodníku tvořenou signály D – Data, RS – Register Select a WR – write. Po ukončení konfigurace nastaví převodník signál Done.



Obr. 19 DacProg



Obr. 20 - Stavový automat Video IN

6.3.2 Propojení desek

Vzájemné propojení vývojových desek XSV-300 a V2LC1000 je externí a je provedeno 22mi asi 10 cm dlouhými vodiči (Příloha 1), které slouží k přenosu dat mezi jednotlivými deskami

1. Digitalizovaná data z desky XSV-300 do desky V2LC1000. Tato data jsou přenášena mezi bloky BusDriver a VideoReceiver

VIDEODATA[8]	Digitalizované hodnoty luminance jednotlivých pixelů
DataValid	Potvrzuje platnost dat na sběrnici VIDEODATA
FrameEnd	Označuje konec videosnímku

Tab. 14 - Signály XSV-300 -> V2LC1000

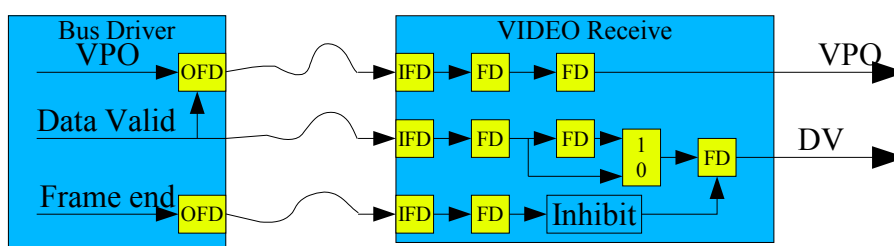
2. Ovládací a datové signály pro VGA z desky V2LC1000 do desky XSV-300 – v tomto případě jsou signály pouze propojeny skrz FPGA na desce XSV-300, nejsou tam ani registrovány, ani modifikovány.

PixClk	Hodinový signál přenosu
P[8]	Jas aktuálního pixelu
/VSync	Vertikální synchronizační signál
/HSync	Horizontální synchronizační signál
/Blank	Zatmívací signál

Tab. 15 - Signály V2LC1000 -> XSV-300

BusDriver a Video Receiver

BusDriver slouží k přizpůsobení formátu digitálního videosignálu (VPO) do formy vhodné pro přenos. Jde sice o přenos mezi jednotlivými deskami z nichž každá má vlastní operační frekvenci, ale kmitočet přijímací strany je několikrát vyšší než operační kmitočet vysílací strany, nejde tedy o náročný proces jako v případě dvou blízkých frekvencí.



Obr. 21 - Propojení vývojových desek

Na desce Insight je vstupní branou blok Video Receiver. Ten má na vstupu každý signál dvakrát registrovaný, čímž předejdeme nežádoucím efektům které by mohly být způsobeny metastabilitami klopných obvodů. Dále detekujeme hranu na signálu DataValid a po dvou hodinových cyklech (bezpečnostní prodleva) vnitřním signálem data valid potvrdí jejich platnost. Ten pouze přijímá digitalizovaný videosignál a předává ho dál do memory arbiteru k uložení do paměti. Propojení BusDriver – Video Reciver je znázorněno na Obr. 21

6.3.3 Vývojová deska V2LC1000

Paměť

Použití paměti DDR je výhodné, obzvláště z hlediska její vysoké datové propustnosti a velké kapacity. Toto je ovšem vykoupeno komplikacemi při práci s tímto typem paměti oproti typické paměti SRAM. V Tab. 16 jsou tyto paměti porovnány.

<i>Vlastnost</i>	<i>SRAM</i>	<i>DDR</i>
Maximální operační frekvence	nižší	vyšší
Kapacita	nižší	větší
Datová propustnost na pin	nižší	vyšší
Nutnost provádět refresh	ne	ano
Rozdělení datového prostoru	ne	ano
Využití obou hran hodinového signálu pro přenos dat	ne	ano

Tab. 16 - Porovnání vlastností pamětí typu DDR a SRAM

Z Tab. 16 je zřejmé, že paměť typu DDR nabízí podstatně vyšší datovou propustnost i kapacitu, což je ovšem vykoupeno náročnějším řízením. V následující části odůvodním jednotlivé rozdíly mezi typy pamětí a jaký mají dopad na celý systém.

V paměti DDR je informace uložena jako elektrický náboj na kondenzátoru. Tento náboj má však tendenci se vybíjet i v době, kdy je paměť připojena ke zdroji elektrického napájení. Aby nedošlo k jeho vybití a tím i ke ztrátě uložené informace, je nutné periodicky provádět tzv. refresh, tj. ožívování paměťové buňky. Instrukce refresh zajistí obnovení náboje na všech buňkách v rámci jedné řádky, jejíž adresa je dána vnitřním čítačem paměti. Pokud by byl překročen maximální interval udávaný výrobcem, může se stát že náboj (napětí) poklesne pod rozhodovací úroveň a informace je nenávratně ztracena.

Z důvodu úspory vývodů paměťového čipu, jsou paměťové buňky v DRAM uspořádány v matici a adresování buňky proto probíhá ve dvou fázích - adresace řádku a adresace sloupce. Adresové piny čipu jsou společné pro řádek i sloupec a jejich význam se určuje signály RAS a CAS (Row Address Select a Column Address Select).

Datový přenos mezi pamětí a kontrolérem probíhá s dvojnásobnou frekvencí než je taktovací signál. Právě díky tomu je dosaženo vyšší datové propustnosti. To ale na druhé straně klade vyšší nároky na kontrolér, který musí být schopen tuto informaci zpracovat.

- Kontrolér tedy musí být schopen pracovat s dvojnásobnou frekvencí pro přenos dat.
 - Virtex II obsahuje speciální klopný obvod DDR FF (Dual Data Rate Flip Flop) který je určen přesně pro tyto aplikace.
- Musí automaticky provádět refresh
- Musí zajistit otevírání a zavírání jednotlivých řádků

Dále z principu adresování vyplývá, že přechod na jinou řádkovou adresu je náročnější na režijní operace než přechod na jinou sloupcovou adresu. To už sice není záležitost samotného kontroléru, ale slouží to jako doporučení pro bloky které čtení/zápis vyžadují.

Vývojový kit je osazen pamětí DDR TC59WM815BFT od fy. Toshiba. Její kapacita je 32 MBytů a maximální operační frekvence je 125 MHz, při latenci 2,5. Datová šířka je 16 bitů a paměťové buňky jsou organizovány jako 4 banky x 8192 řádek x 512 sloupců.

Komunikace s DDR

Přenos příkazů k paměti je přes několik signálů, a to RAS, CAS, WE (Write Enable) a CS (Chip Select). Tyto signály jsou synchronně s náběžnou hranou hodinového signálu paměti načítány a určují operaci kterou bude paměť provádět..

Pro vlastní přenos jsou místo hodin použity takzvané “data stroby”(DQS). Jde o obousměrné signály generované společně s daty (DQ), které jsou při zápisu generovány kontrolérem a při čtení paměti – takové rozhraní se obecně nazývá “source synchronous” a umožňuje dosáhnout vyšších přenosových rychlostí.

Je pravidlem, že strobovací signál není pouze jeden, ale ke každému určitému počtu datových vodičů (většinou 4, 8, 16) je jeden strobovací. To je výhodné zvláště v případě širokých sběrnic – nejsou kladeny tak velké nároky na ekvivalentní délky jednotlivých tras na plošném spoji. postačuje odpovídající délka (resp. doba šíření) v rámci jedné sady datových signálů a jednoho strobovacího.

Platnost dat je určena náběžnou i sestupnou hranou strobovacího signálu. Při zápisu vyžaduje paměť aby byla hrana strobovacího signálu přibližně uprostřed dat (center aligned). Při čtení paměť strobovací signály generuje současně s daty (edge aligned).

Zápis i čtení probíhá v takzvaných burstech. Délka burstu určuje počet po sobě jdoucích lokací na které jsou zapsána data. Tato délka může být naprogramovaná na jednu z hodnot 2, 4, nebo 8. Správným časováním příkazů pro zápis (Write) je možné řadit bursty těsně za sebe, čímž dosáhneme maximální propustnosti.

Vlastní zápis nebo čtení se pak provádí posloupností příkazů(viz. Obr. 34) :

1. Otevření řádku (Activate)
2. Teoreticky neomezené množství operací čtení/zápis v rámci jednoho řádku
3. Uzavření řádku (Precharge)

Podrobnější popis viz. [16].

DDR controller

Paměť typu DDR je náročná na obsluhu, je proto naprosto nutné použít modul který samostatně zajistí komunikaci na nejnižší úrovni.

Měl jsem možnost volby mezi třemi možnostmi. Buď blok kontroléru koupit, použít nějaký volně dostupný nebo vytvořit svůj vlastní. Na nákup bloku by byly potřeba další investice. Jako jednodušší ze zbylých dvou variant se samozřejmě zdálo převzít již hotový, volně dostupný návrh. Buď přímo ze webové stránky Xilinx [6], nebo z některé “otevřené“ knihovny, například Opencores [17].

Zkušenosti s převzatými jádry nebyly nejlepší. Nepochybuji o jejich funkčnosti, ale pro konkrétní aplikaci je bylo nutné jádro “doladit” aby fungovalo. Právě to se u většiny ukázalo jako kámen úrazu, protože nikdo příliš nepočítal s jejich parametrizací a úpravy vyžadovaly zásah do samotného kódu, což není často úplně nejjednodušší.

Po několika dnech experimentování se různými jádry jsem se rozhodl, že vytvořím kontrolér vlastní, nicméně vycházel jsem většinou z ideí obsažených v dokumentaci jednotlivých jader. Kompletní návrh kvalitního parametrizovatelného DDR kontroléru by vydal na samostatnou diplomovou práci, proto jsem vytvořil kontrolér, který je “ušitý na míru” pro mojí aplikaci. Přijal jsem tato omezení :

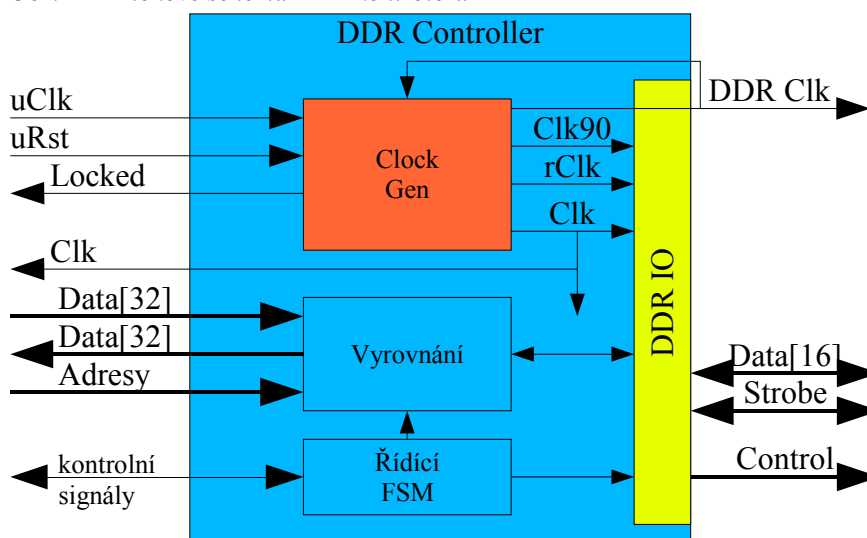
- Fixní nastavení paměti
 - Bursty o délce 8
 - Latence paměti 2.5
- Přístup do paměti bude probíhat v tzv. multiburstech⁵ o nastavitelné délce
- Celý multiburst bude adresovat pouze jednu řádku paměti.
- Před každým multiburstem je řádka otevřena(Activate) a po jeho skončení uzavřena (Precharge).
- Při čtení je použit pouze fázově posunutý hodinový signál, strobovací signály jsou ignorovány.

Po přijetí těchto předpokladů se návrh zjednodušil na únosnou míru, ale přitom jeho parametry stále dostačují jeho aplikaci v tomto projektu.

Realizace DDR kontroléru

Základem DDR kontroléru je stavový automat (řídicí FSM, 41/Obr. 25) Ten na základě požadavků od nadřazeného bloku řídí zápis/čtení z/do DDR paměti. Pro generaci všech potřebných hodinových signálů slouží blok ClockGen. Zároveň je zdrojem hodinového a resetovacího signálu pro celé FPGA. Na konci kapitoly je ilustrována funkce kontroléru (42/Obr. 26)

Obr. 22 - Blokové schéma DDR kontroléru

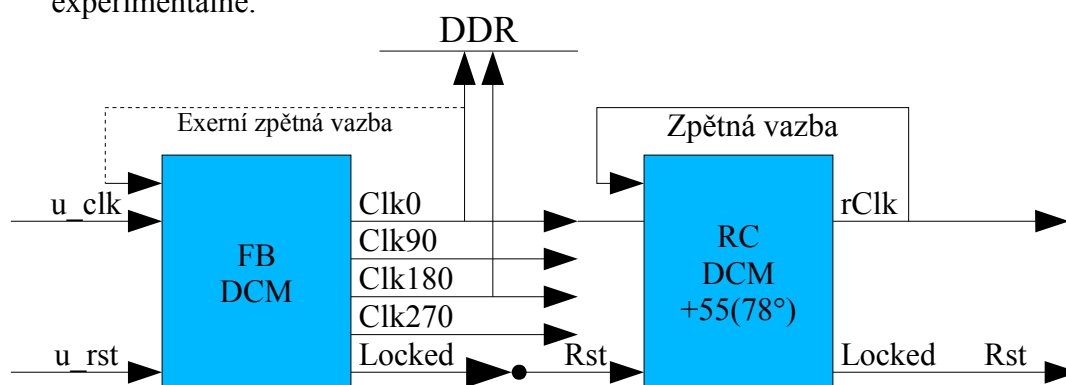


5 Sekvence na sebe vzájemně těsně navazujících burstů(Obr. 34).

ClockGen

Interface s DDR vyžaduje velké množství hodinových signálů. Jen tak se dá zaručit správné časování nutné pro funkci paměti. Proto ClockGen (Obr. 23) generuje následující signály

- Hodinové signály s fázovým posuvem 0° a 180° – pro taktování datových signálů
- Hodinový signál s fázovým posuvem 90° , respektive 270° - Pro taktování strobovacích signálů (center aligned s daty) je potřeba
- Hodinový signál s obecným vhodným fázovým posunem – Jsou použity místo strobovacích signálů pro zachytávání dat při čtení z paměti. Fázový posuv byl nalezen experimentálně.



Obr. 23 - Clock Gen

Vestavěný blok architektury Virtex II – Digital Clock Manager je určen pro práci s hodinovými signály [18].

V bloku ClockGen jsou využity dva DCM. FB_DCM synchronizuje hodiny DDR s hodinami FPGA a generuje i jejich fázově posunuté varianty s krokem čtvrt periody. RC_DCM fázově posouvá hodinový signál, který je použit k zachytávání dat ze sběrnice DDR.

Aby se zamezilo špatné funkčnosti je RC_DCM resetováno do doby, než se FB_DCM zavěsí. Zároveň je generován resetovací signál pro celé FPGA do doby, než jsou všechny hodinové signály stabilní.

DDR IO

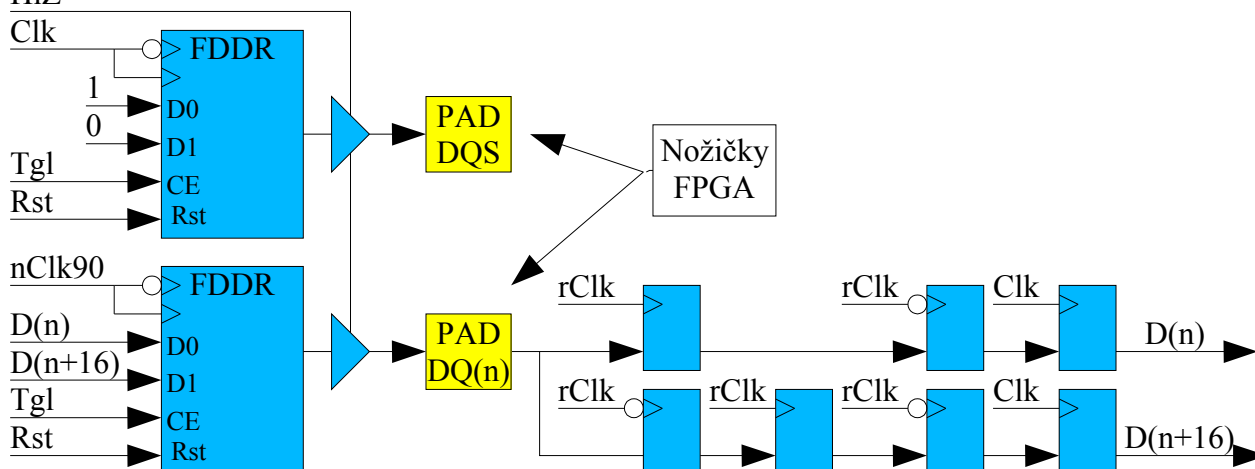
Blok DDR IO obsahuje speciální registry Dual Data Rate Flip Flop(FDDR). Jde o registry typu D, které se chovají jako by pracovaly s náběžnou i sestupnou hranou [1]. Tyto jsou používány při zápisu do paměti. Schéma IO bloku je na Obr. 24, přičemž je zobrazen jen jeden datový signál a jeden strobovací. Ve skutečnosti připadá na osm datových signálů jeden strobovací.

FDDR jsou použity i při generování zápisového signálu. To zaručuje precizní polohu hrany strobovacího signálu přesně v době kdy jsou data na sběrnici ustálená.

Při čtení jsou strobovací signály ignorovány, ale data jsou neustále načítána s hranou hodin rClk a /rClk, posléze převedena do domény Clk s níž pracuje celé FPGA. Platnost načítaných dat je dána určena stavovým automatem (mezi požadavkem na čtení a přichozími daty je pevný časový vztah).

Zapisovaná a čtená data mají dvojnásobnou datovou šířku oproti šířce datové sběrnice. Důvod je zřejmý – Vnitřní sběrnice pracují pouze s jednou aktivní hranou hodinového signálu, sběrnice DDR s oběma.

HiZ

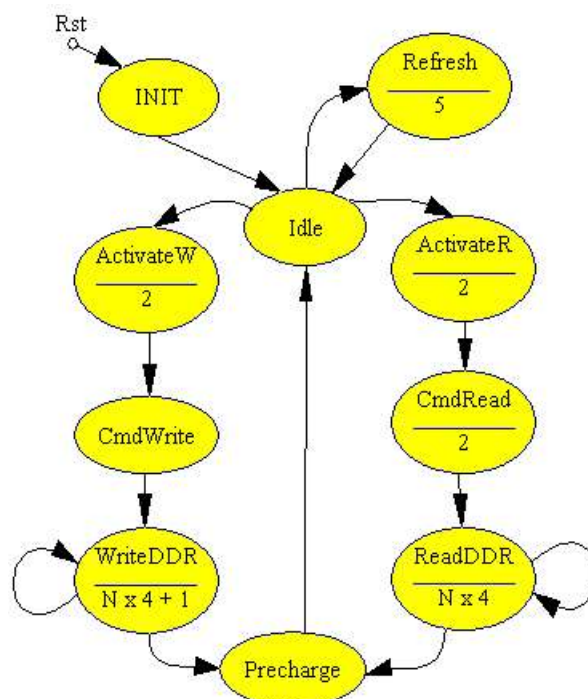


Obr. 24 - DDR IO

Asas

Řídící stavový automat (FSM)

Základem celého kontroléru je stavový automat (jeho orientační schéma je na Obr. 25). Ten nejprve po resetu paměť inicializuje (Příloha 2), a poté přijímá požadavky ze strany FPGA, podle nich řídí komunikaci s pamětí a předává data dovnitř FPGA.



Obr. 25 Zjednodušený stavový automat DDR kontroléru

Čísla ve stavech automatu značí kolik period hodinového signálu v nich stavový automat “zůstává”. Dále přechod po každé hraně zabere jednu periodu. “N” značí počet multiburstů. Z automatu je jasné, že čím větší bude počet multiburstů, tím bude práce s pamětí efektivnější.

Příklad:

Při zápisu multiburstu délky 1 bude průchod ze stavu Idle zpět do stavu Idle trvat 13 hodinových cyklů. Z toho jsou pro zápis použity pouze čtyři, to znamená že efektivita je asi 31 %

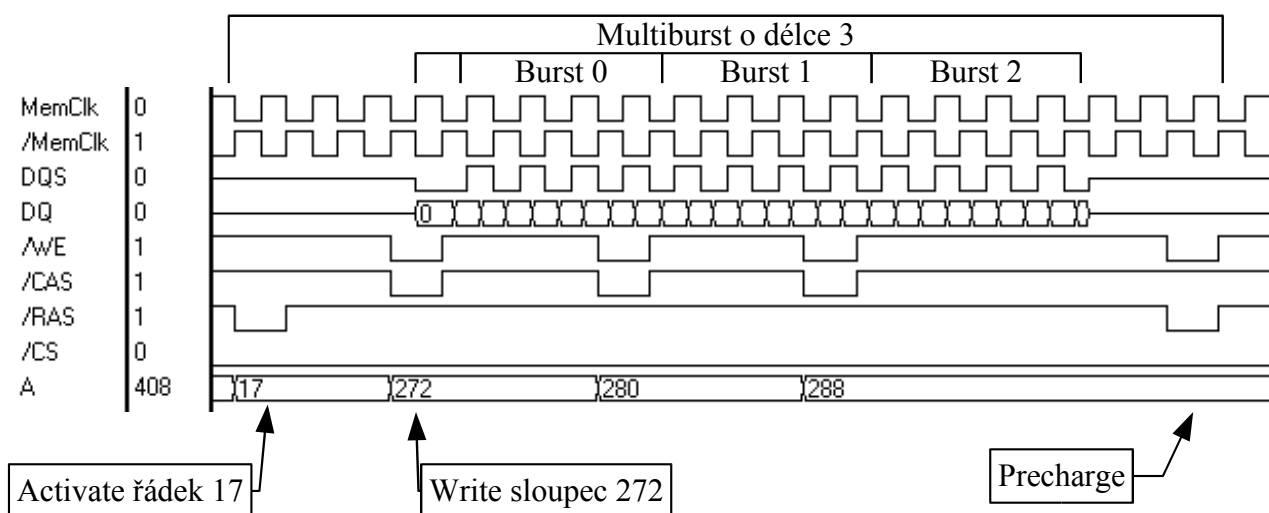
Naproti tomu při zápisu multiburstu o délce 64 trvá průchod 265 cyklů, z toho jich je pro zápis použito 256. Efektivita je v tomto přes 96%. V případě čtení je situace obdobná.

Z uvedeného příkladu jasně vyplývá že čím je multiburst delší, tím je přístup do paměti efektivnější.

Vyrovnění

Pro správnou funkci je potřeba některé signály pozdržet oproti ostatním, proto jsou v DDR kontroléru některé signály vedeny skrz pipeline. Nejsou tvořeny speciálním posuvným registrem [18] (str.1761), ale klopnými obvody řazenými za sebou. To usnadní práci routeru, který může signály vedené na velkou vzdálenost po čipu, rozdělit na několik kratších.

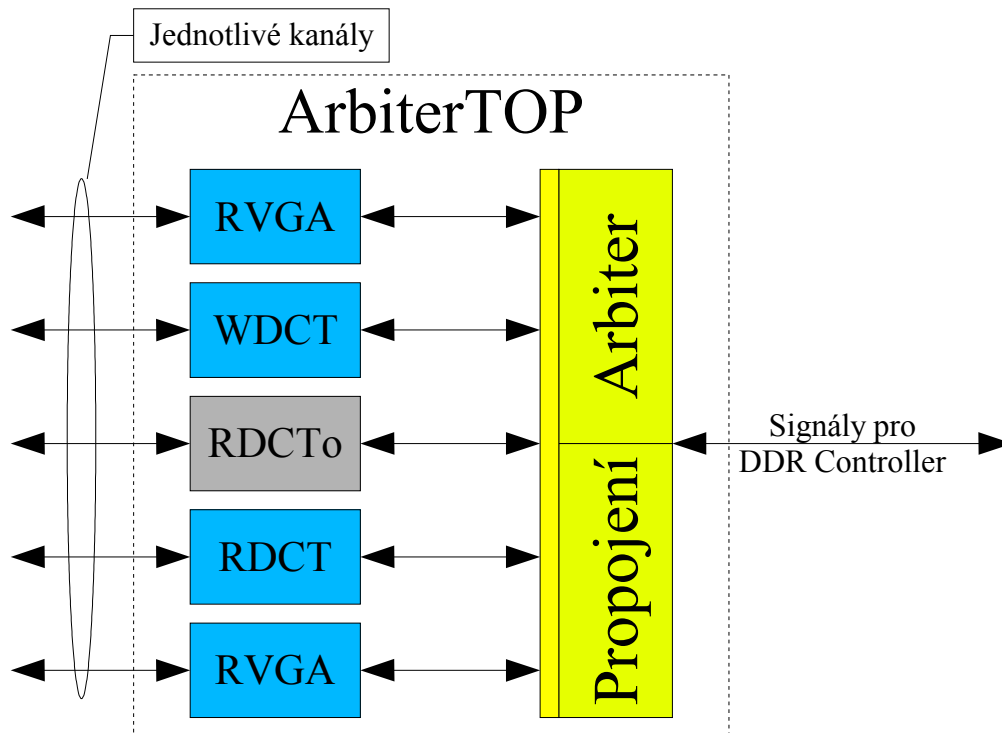
Při návrhu DDR kontroléru jsem vycházel z [16]. Simulace funkce bloku je na Obr. 26.



Obr. 26 - Simulace DDR kontroléru

Memory Arbiter

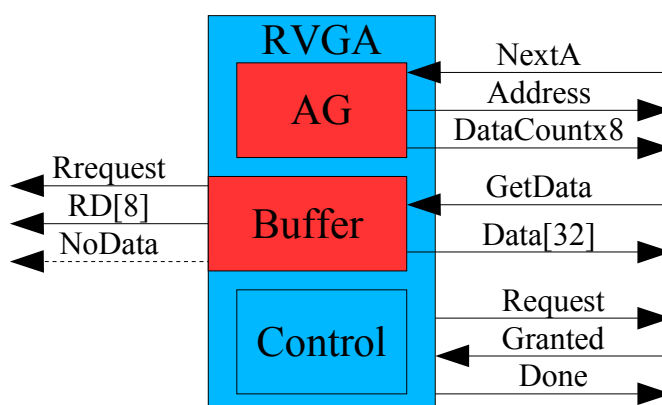
Tento blok slouží k maskování jedné fyzické paměti jako dvou logických s čtyřmi (resp. pěti) nezávislými branami. Aby by byl přístup jednotlivých kanálů do paměti transparentní, obsahují bloky RVGA, WDCT... vyrovnávací paměť.



Obr. 27 - Blokové schéma arbitru

Blok označený jako arbiter je stavový automat, který podle žádostí jednotlivých bloků přiděluje přístup k fyzické paměti. Blok propojení se podle příkazů Arbitru stará o propojení právě aktivního bloku s DDR kontrolérem a vice versa.

Jak již bylo řečeno, nejdůležitějším prvkem vyrovnávacích bloků je buffer – vyrovnávací paměť. Ta je koncipována buď jako jednoduché FIFO, nebo jako FIFO s přerovnáním (viz níže).



Obr. 28 - Vyrovnávací a adresovací blok

Dalším prvkem je generátor adres (AG). Ten určuje fyzické čtecí respektive zápisové adresy podle daného adresového schématu.

Propojení mezi vyrovnávacími bloky a arbiterem je popsáno v Tab. 17.

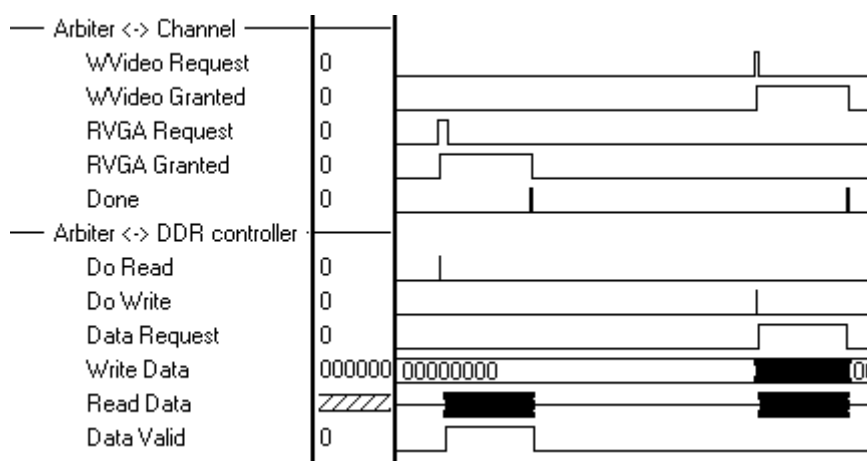
<i>Signál</i>	<i>Funkce</i>
Request	Požadavek na obslužení
Granted	Přístup udělen
Done	Obsloužení hotovo
GetData	Požadavek na další data
Data	16+16 bitů dat které mají být zapsány do paměti
NextA	Zvyš adresu
Address	Adresa
DataCountx8	Délka multiburstu-1

Tab. 17 Popis signálů vyrovnávacího bloku

Jako formát dat v paměti DDR jsem pro jednoduchost zvolil stejnou jako ve videosignálu, tj. na jednu řádkovou adresu v paměti je zapsána jedna řádka videosignálu. Paměťové prostory jednotlivých snímků se liší bázovou adresou.

Stavový automat arbiteru je zřejmý, proto zde jen slovně popíšu jeho funkci. Postupně sleduje jednotlivé thready a jakmile některý z nich nastaví požadavek (request), oznámí mu přidělení “přístupu” (granted) a zároveň s tím řízení propojí odpovídající signály s kontrolérem paměti. Dále se stará o periodické opakování čtecích/zápisových cyklů paměťového kontroléru. Thread tak provádí zápis/čtení do/z fyzické paměti. V případě požadavku jiného threadu v této době je tento ignorován. Teprve po řádném ukončení přístupu prvního threadu (Done) začne arbiter sledovat další thready, přitom pokračuje tam kde skončil.

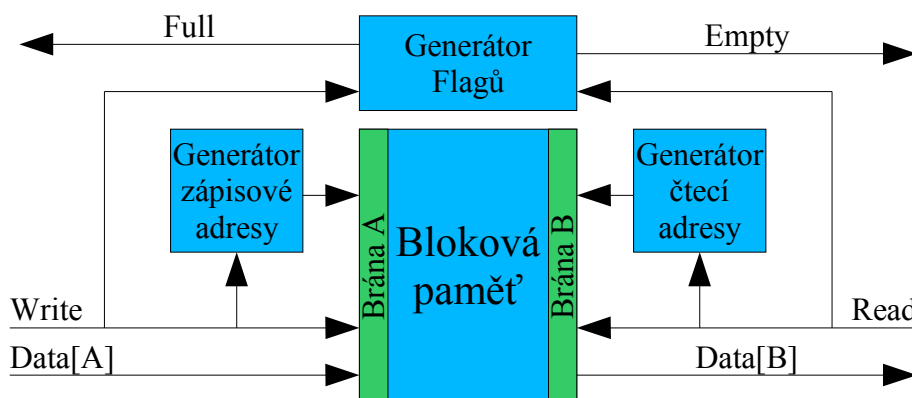
Na Obr. 29 je uveden příklad komunikace jednak mezi arbiterem a bloky RVGA a WVID (Arbiter <-> Channel) a tomu odpovídající komunikace mezi arbiterem a DDR kontrolérem. Signál Done je logickým součtem Done jednotlivých bloků.



Obr. 29 Komunikace arbiteru s ostatními bloky

Všechny buffery jsou implementovány s použitím blokových pamětí. Ty jsou pro tento účel ideální, dají se konfigurovat jako paměti s dvěma úplně nezávislými bránami. Obecná struktura bufferu je na Obr. 30. Je dokonce možné specifikovat rozdílné šířky jednotlivých bran, což umožňuje konverzi šířky dat. Například při šířce brány A=8 a brány B=16 jsou vstupní data A,B,C,D,E,F na bráně B k dispozici jako AB,CD,EF

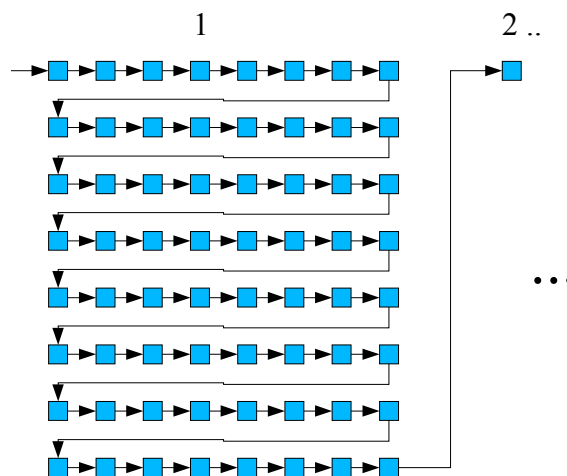
Při stejném(lineárním) generování zápisové a čtecí adresy se jedná o klasické FIFO.V systému jsou také potřeba převodníky z řádkového formátu na formát blokový.



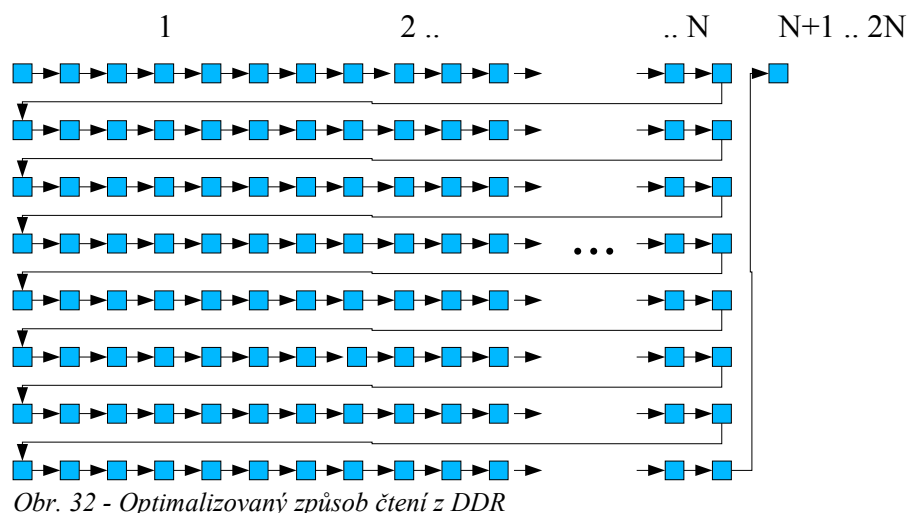
Obr. 30 - Možnosti využití blokové paměti jako vyrovnávací

Principiálně by se dal převodník řádky-bloky implementovat pouze čtením odpovídajících pozic z paměti, jako bychom četli rovnou bloky. To je ale neefektivní pro paměť DDR, která by tak během čtení jednoho bloku musela osmkrát otevřít a zavřít řádek(Obr. 31). Viz příklad rychlosti stavového automatu DDR kontroléru.

Výhodnější je sloučit několik bloků dohromady a přečíst první řádek pro všechny, poté druhý pro všechny atd. Opět budeme předadresovávat řádek osmkrát, ale tentokrát už budeme mít načteno N bloků(Obr. 32). Tímto maximalizujeme délku multiburstu a umožníme efektivní využívání paměti. Samotné čtení jednotlivých bloků bude pak provedeno rozdílnou adresací čtecí brány paměti blokové paměti, která již nemá žádné omezení týkající se adresování.

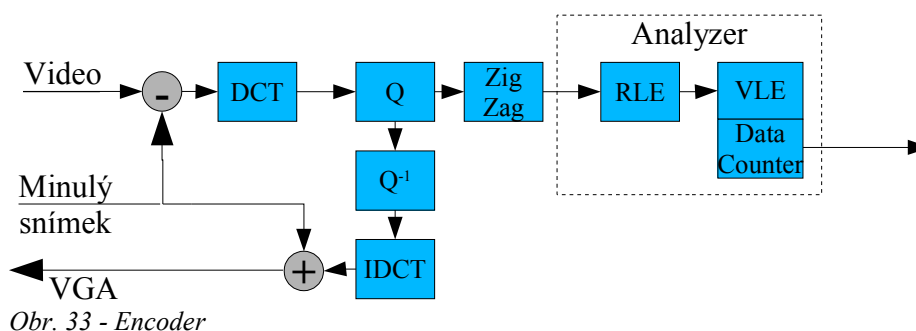


Obr. 31 - Neoptimalizovaný způsob čtení z DDR

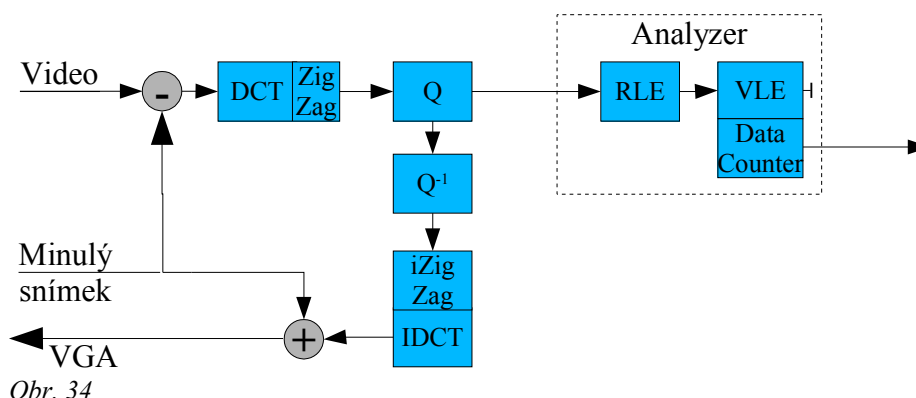


JPEG kodér/dekodér

Tento blok slouží ke kompresi podle algoritmu JPEG, popřípadě podle jeho rozšíření (na Obr. 33 šedivé bloky). Načítá data videosignálu z paměti, ty zkomprimuje a zároveň do paměti rozkomprimovaný degradovaný. Blok Analyzer vyhodnotí datový tok, vlastní výstupní data nejsou v této verzi systému poskytována.



Pro efektivnější implementaci bylo zaměněno pořadí některých kroků a dále byly některé kroky sloučeny. Výsledné schéma je na Obr. 34.



V upraveném schématu vydává DCT koeficienty již v posloupnosti Zig-Zag. Místo kvantizační matice je použit kvantizační vektor, který dostaneme čtením kvantizační matice v Zig-Zag posloupnosti.

DCT

Dvourozměrná Diskrétní Cosinová Transformace je jádrem celé komprese. Od začátku bylo naprosto zřejmé, že díky její separovatelnosti bude v provedena jako dvě jednorozměrné transformace. Bylo však potřeba zvolit konkrétní implementaci.

1. IP (Intellectual Property) jádro
2. Design dostupný na webových stránkách [6][17]
3. Vlastní

V první fázi vývoje byla k dispozici pouze vývojová deska XESS s první generací FPGA Virtex. Obvod obsahoval konkrétně 3000 slices. To je příliš málo pro použití IP jádra distribuovaného společně s návrhovým systémem, kde by jedna 2D transformace zabrala více než polovinu prostředků. Ani mezi soft jádry nebyla nalezena vhodná implementace. Pro tento obvod měla být DCT založena na její aproximaci vycházející z [19], tak jak bylo prezentováno v [1].

S nákupem vývojové desky V2LC1000 se značně rozšířily možnosti, jelikož obsahuje obvod druhé generace Virtex a to ještě s větší kapacitou než deska XSV-300. Možnosti pro implementaci DCT shrnu v tabulce Tab. 18.

<i>Jádro</i>	<i>Počet jednotek slice</i>
Xilinx Logicore 2D DCT [6]	1652
Xilinx Application Note 610 [6]	558
Beznásobičková aproximace DCT [19]	-
Pipelined Fast 2D DCT [20]	4792 logical cells ⁶

Tab. 18 Vlastnosti jednotlivých implementací DCT

Výrazný rozdíl mezi prvními dvěmi uvedenými možnostmi je dán tím, že jádro Logicore ještě nevyužívá vestavěných násobiček. I přes lákavý počet slices při implmentaci podle AN610 jsem se nakonec rozhodl pro vlastní implementaci. Důvodem bylo jednak získání dalších zkušeností a také snaha ještě trochu snížit počet potřebných jednotek slice.

Aby mohla celá komprese probíhat v reálném čase musí mít transformace datovou propustnost 10,368 MSps. Já jsem zvolil propustnost 25 MSps, aby byl systém případně schopen zpracovat barevný signál. V případě hodin 100MHz to znamená čtyři hodinové cykly na jeden vzorek.

Většina optimalizovaných algoritmů se snažila o minimalizaci počtu násobení, protože jde o časově/prostorově náročnou funkci (v je porovnána prostorová náročnost se sčítáním). Většinou se jedná o různé “motýlkové” struktury - ty sice sniží počet násobení, ale je to vykoupeno náročnějším řízením celé transformace.

⁶ Implementační platformou nebyl obvod od fy. Xilinx, ale obvod fy. Altera

Tab. 19 Porovnání prostorové náročnosti sčítání

Funkce	Počet jednotek slice
Násobička signed 18x18 bitů	175
Sčítačka signed 18+18 bitů	10

Přítomnost vestavěných násobiček tuto situaci mění – v FPGA jsou přítomny tak jako tak a jejich alternativní využití pro jinou funkci je omezené, ne-li žádné.

Odstranění stejnosměrné složky (převedení dat z reprezentace 0 až 255 na -128 až 127) je provedeno na vstupu do bloku DCT a je jednoduše provedeno inverzí bitu s nejvyšší vahou (MSB).

Jednorozměrná DCT transformace může být provedena jako násobení vektoru transformační matice. Já jsem zvolil tento způsob, protože díky přítomnosti vestavěných násobiček v obvodu Virtex II může být i tento přístup efektivním zpracováním.

Na Obr. 35 je znázorněno matematické vyjádření jednorozměrné DCT transformace.

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} & C_{03} & C_{04} & C_{05} & C_{06} & C_{07} \\ C_{10} & C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} & C_{17} \\ C_{20} & C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} & C_{27} \\ C_{30} & C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} & C_{37} \\ C_{40} & C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} & C_{47} \\ C_{50} & C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} & C_{57} \\ C_{60} & C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} & C_{67} \\ C_{70} & C_{71} & C_{72} & C_{73} & C_{74} & C_{75} & C_{76} & C_{77} \end{bmatrix} \cdot \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix}$$

Obr. 35 Transformace obecně

Elementy transformovaného vektoru jsou, dle definice násobení matice vektorem, počítány následovně

$$Y_0 = C_{00} \cdot X_0 + C_{01} \cdot X_1 + C_{02} \cdot X_2 + C_{03} \cdot X_3 + C_{04} \cdot X_4 + C_{05} \cdot X_5 + C_{06} \cdot X_6 + C_{07} \cdot X_7$$

$$Y_1 = C_{10} \cdot X_0 + C_{11} \cdot X_1 + \dots$$

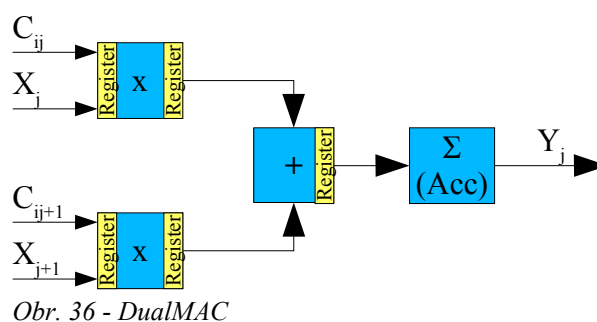
.....
Rovnice 5

Přestože se jedná o naprosto ekvivalentní zápis přepíšu do formy, která nejlépe přiblíží algoritmus implementace

$$Y_0 = \sum_{j=0,2,4,6} C_{0j} \cdot X_j + C_{0j+1} \cdot X_{j+1}$$

Rovnice 6

Na Obr. 36 je zobrazen grafický “přepis” Rovnice 6. Jde o ideové znázornění hlavního výpočetního bloku – DualMAC (jedná se vlastně o dvojitý Multiply Accumulate).

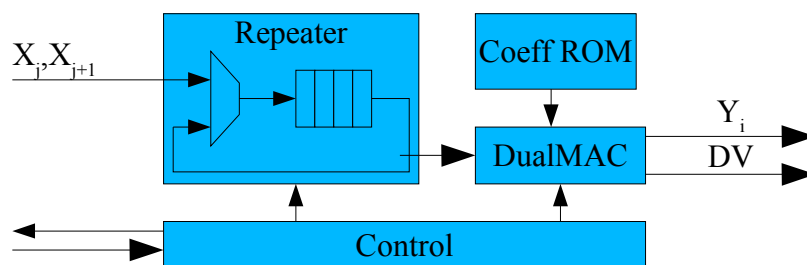


Obr. 36 - DualMAC

Použité vestavěné násobičky jsou schopné vynásobit vstupy během jednoho hodinového cyklu. Je ale nutné registrovat jejich vstupní i výstupní signály, jinak se jejich parametry vlivem zpoždění na datových cestách zhorší. Tyto registry jsou na Obr. 36 zvýrazněny.

Blok DualMAC v celém projektu použit několikrát, pokaždé s jiným formátem dat na sběrnicích. Ty se liší za prvé šířkou a za druhé pozicí desetinné čárky. Tady jsem využil vlastnosti jazyka VHDL, který umožňuje přiřadit jednotlivým komponentům parametry (tzv. generic) na jejichž základě vygeneruje syntetizér odpovídající variantu.

Pro vypočtení všech hodnot Y_0 až Y_7 je nutné přivést na vstup osmkrát za sebou vstupní vektor. Na Obr. 37 je finální uspořádání celé jednorozměrné DCT transformace. Blok repeater je, podle nastavení multiplexeru buď plněn vstupními hodnotami nebo stále opakuje ty načtené. Celý proces transformace je spuštěn až v okamžiku, kdy vyrovnávací buffer z bloku memory arbiter oznámí signálem Block Ready že má k dispozici jeden celý blok 8×8 k provedení transformace. Platnost dat na výstupu je určena signálem DataValid (DV)- Převod celého bloku (64 vzorků) trvá 256 hodinových cyklů.



Obr. 37 - DCT A

Výstupem této transformace je soubor vektorů transformovaných jednorozměrně. Pro transformaci v druhém rozměru je použit další blok, na jeho výstupu jsou data již v Zig-Zag posloupnosti. Jeho výpočetním jádrem je opět DualMAC, ale jinak se liší. Je použita bloková paměť s rozdílnou 16ti bitovou šířkou zápisové brány oproti A 32 bitovou šířkou čtecí brány. Tím jsou data sdružena do dvojic, tak jak je zpracovává DualMAC.

Zápis probíhá na adresy po "sloupcích", čímž je provedena transpozice matice. Jde o jednoduché prohození bitů – pokud vezmeme jednoduchý lineární čítač, pak je prohození bitů stejné jako je znázorněno v Tab. 20.

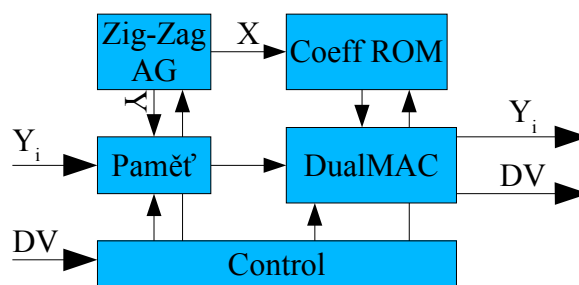
Lineární čítač	9	8	7	6	5	4	3	2	1	0
Adresa	9	8	7	6	2	1	0	5	4	3

Tab. 20 - Prohození bitů lineárního čítače

Bity 9 až 6 jsou vlastně adresou bloku, proto jsou v obou případech na stejném řádkovém místě. Výsledné posloupnosti jsou zobrazeny v Tab. 21.

Čítač	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	..	62	63	64	65	66	..
Adresa	0	8	16	24	32	40	48	56	1	9	17	25	33	41	49	..	55	63	64	72	80	..

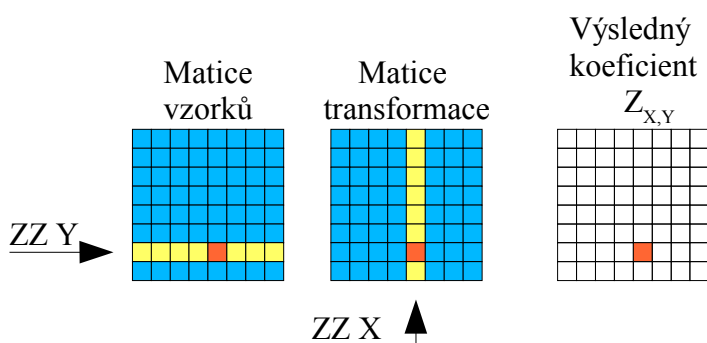
Tab. 21



Obr. 38 DCT - B

Samotná transformace je spuštěna po naplnění paměti jedním blokem 8x8. Výpočet transformace probíhá již podle zig-zag posloupnosti.

Blok Zig-Zag generuje jednotlivé souřadnice aktuálního požadovaného vzorku. Adresa Y vybere vektor který se má transformovat a adresa X vybere koeficienty odpovídající požadované “frekvenční” složce. Z paměti jsou vybrány 4 dvojité vzorky a po zpracování dualMACem získáme výstupní hodnotu $Z_{X,Y}$.



Obr. 39

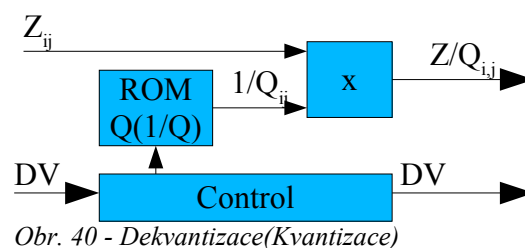
Vlastní koeficienty celé transformace jsou uloženy v paměti ROM. Ta je naplněna podle souboru DCT_coeff.vhd, a v souboru functions.vhd je možné nastavit reprezentaci koeficientů (šířku a pozici desetinné čárky).

IDCT

IDCT je velmi podobná DCT. Je také rozdělena na dvě jednorozměrné transformace – IDCTA a IDCTB. Vstupní data přijímá v Zig-Zag posloupnosti a po naplnění bufferu minimálně 64mi vzorky je provedena vlastní transformace. IDCTB poté provede transpozici a transformaci v druhém rozměru a na jejím výstupu jsou již bloky inverzně transformovaných dat. Její podobnost s dopřednou transformací je velká, nebudu to zde proto detailněji rozebírat. Šířka a pozice desetinné čárky vstupních dat, mezivýsledků i konečných výsledků je možné nastavit parametry (generic).

Kvantizace

V tomto bloku by se jednotlivé koeficienty měly vydělit odpovídajícími kvantizačními koeficienty. Opět zde lze s úspěchem použít vestavěnou násobičku. Využil jsem jednoduchého vztahu $\frac{Z_{i,j}}{Q_{i,j}} = Z_{i,j} \cdot \left[\frac{1}{Q_{i,j}} \right]$. V paměti ROM jsou tedy uloženy převrácené hodnoty kvantizačních koeficientů a místo děličky je použita násobička.



Obr. 40 - Dekvantizace(Kvantizace)

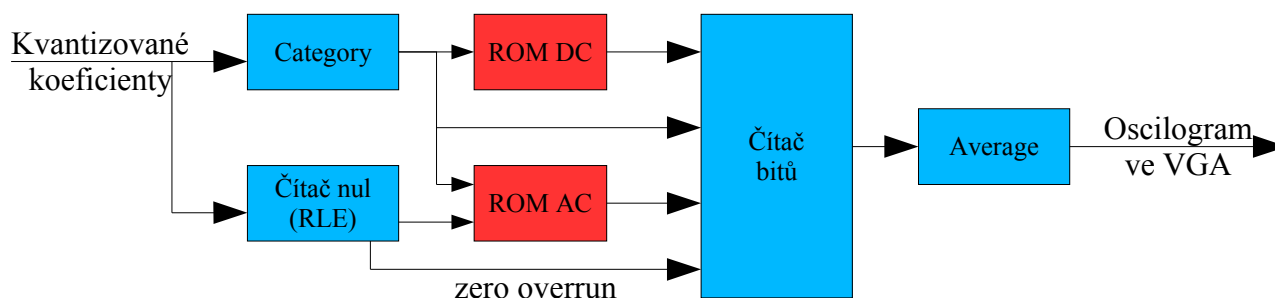
Hodnoty v paměti ROM jsou určeny v souboru DCT_coeff.vhd, a v souboru functions.vhd je možné nastavit reprezentaci koeficientů(šířku a pozici desetinné čárky).

Dekvantizace

Dekvantizace je identická s blokem kvantizace, pouze s tím rozdílem že v paměti jsou uloženy přímo hodnoty z kvantizační tabulky. Struktura tohoto bloku je zobrazena na Obr. 40.

Analyzer

Analyzer provádí kompresi RLE. Neprovádí fyzickou kompresi VLE, ale pouze extrahuje počet bitů potřebný na zakomprimování celého bloku.



Obr. 41 - Analyzer

Koeficienty jsou buď AC nebo DC. Pro DC je určena jejich kategorie a z paměti ROM DC je vyčten potřebný počet bitů pro reprezentaci DC koeficientu.

Pro AC koeficient záleží potřebná délka na jeho reprezentaci nejen na kategorii vlastního koeficientu, ale také na počtu předcházejících nul. Z paměti ROM AC je tedy na základě kategorie a počtu nul vyčtena odpovídající délka pro reprezentaci RLE zakódovaného AC koeficientu.

Čítač bitů slouží k čítání jednak délek potřebných pro reprezentaci AC a DC koeficientů ale také automaticky připočítává bity potřebné pro reprezentaci značek EOB (konec bloku) a zero overrun. Čítač bitů počítá bity nutné k reprezentaci jednoho snímku. Výsledek předá bloku Average.

Blok average má za úkol vyhladit křivku datového toku. To je provedeno algoritmem klouzavého průměru jehož definice je v Rovnice 7. Počet vzorků na kterých je klouzavý průměr počítán je 32. Zprůměrovaný datový tok je předán bloku oscilogram v bloku VGA, který zajistí jeho zobrazení na monitoru.

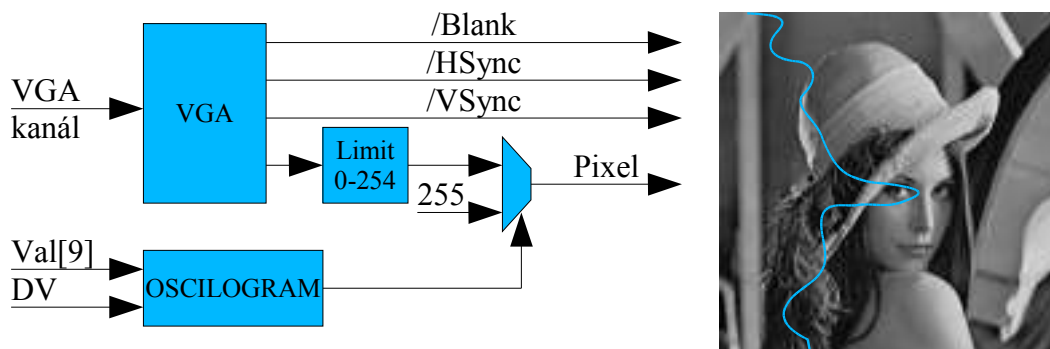
$$y_n = \frac{\sum_{i=0}^{N-1} x_{n-i}}{N}$$

Rovnice 7

VGA

Slouží k vykreslení obrazu na VGA monitoru. Protože deska Insight neobsahuje potřebné periférie, jsou všechny signály po externí sběrnici přeneseny skrz desku XESS přímo na VideoDAC.

Navíc obsahuje blok oscilogram, který v grafické formě zobrazuje posloupnost vstupních dat Val, jejichž platnost určuje DV. Nová data jsou zobrazena na nejnižším řádku a všechna stará jsou posunuta o jeden řádek výše.



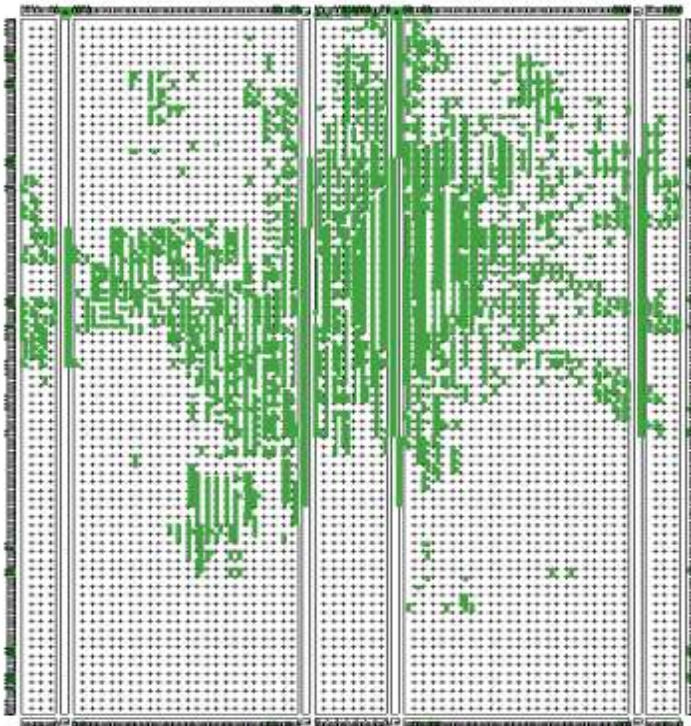
Obr. 42 -Způsob vykreslování obrazu na monitoru VGA

Aby křivka nesplývala s obrazem, je DA převodník naprogramován stupni šedi pouze 0-254 (viz. kapitola DACProg) a na pozici 255 je naprogramován ke zobrazení modré barvy. Proto je nutné omezit rozsah obrazového signálu, hodnota 255 je limitována na 254, všechny ostatní úrovně jsou nezměněny.

Ideální pro tuto funkci by bylo využití funkce overlay kterou nabízí DA převodník, ale na vývojové desce bohužel nejsou tyto piny dostupné.

6.4 Zhodnocení realizace

Funkce XSV-300 jsou pouze podpůrné a proto se v hodnocení budu soustředit na klíčovou část systému, která je implementována v FPGA Virtex II (XC2V1000-4fg256) na vývojové desce V2LC1000.



Obr. 43 - Implementovaný návrh – Virtex II

Na Obr. 43 je vidět implementovaný návrh tak jak je zobrazen v programu Floorplanner. Není možné jednoznačně odlišit jednotlivé bloky, ty vzájemně splývají. Automatizovaná implementace měla za cíl dosažení maximální operační frekvence - tomu je podřízeno umísťování jednotlivých elementárních funkcí po čipu bez ohledu na rozdělení do hierarchických bloků.

Operační frekvence FPGA je 100MHz a je omezena především rychlostí DDR kontroléru a rychlostí vestavěných násobiček. Použitá varianta FPGA je ta nejpomalejší možná (označení -4 v názvu součástky), ale i přesto je naprosto dostačující pro uvedenou aplikaci.

V Tab. 22 je shrnuto využití jednotlivých „prvků“ v FPGA. Jak je vidět, je využití téměř rovnoměrné. Asi nejlépe reprezentuje zaplnění FPGA počet jednotek slice⁷ potřebný pro jejich implementaci.

⁷ To neplatí stoprocentně - pro určité funkce může být omezující například blokových pamětí nebo násobiček.

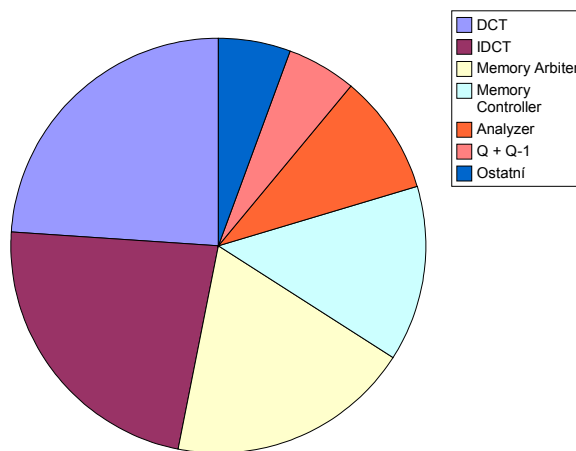
<i>Element</i>	<i>K dispozici</i>	<i>Využito</i>	
IO piny	172	72	41%
Násobičky 18x18	40	10	25%
Blokové paměti	40	8	20%
Jednotky slice	5120	1696	33%
Jednotky DCM	8	2	25%

Tab. 22 - Celkové využití prostředků FPGA

V Tab. 23 (a v Grafu 1) je procentuální využití jednotek slice jednotlivými bloky návrhu..

<i>Blok</i>	<i>Jednotky slice [%]</i>
DCT	24
IDCT	23
Memory Arbiter	19
Memory Controller	13,5
Analyzer	9,5
Q + Q ⁻¹	5,5
Ostatní	5,5

Tab. 23 Relativní využití jednotek slice jednotlivými bloky.



Graf 1 – Relativní využití jednotek slice jednotlivými bloky

Na optimalizaci návrhu jsem začal pracovat teprve nedávno a předpokládám, že počet jednotek slice je možné snížit ještě asi o 10 – 20 %, například sdílením koeficientů transformace mezi jednotlivými bloky DCT (IDCT).

7 Závěr

Seznámil jsem se s architekturou FPGA řady Virtex a Virtex II a metodikou jejich popisu v jazyce VHASIC HDL.

Prostudoval jsem metody komprese videosignálu a jako nejvhodnější pro danou aplikaci jsem zvolil algoritmus Motion-JPEG, především z důvodu jeho snadné rozšiřitelnosti na algoritmy s vyšším kompresním výkonem.

Pro implementaci tohoto algoritmu jsem použil vývojové desky XSV-300 a V2LC1000 s FPGA Virtex respektive Virtex II. Realizace pomocí dvou desek byla v dané chvíli vhodným přístupem, protože jsem mohl využít potřebné periferie na desce XSV-300 a výpočetní možnosti desky V2LC1000.

Celý systém je popsán jazykem VHASIC HDL a je navržen v návrhovém systému Xilinx ISE 6.1i.

Zrealizoval jsem všechny potřebné bloky pro digitalizaci videosignálu a téměř všechny kroky algoritmu M-JPEG, s výjimkou posledního - Vlastní výstupní datový tok není generován - je pouze analyzován.

Navíc jsem oproti zadání zrealizoval základní bloky pro dekódování formátu JPEG a blok pro zobrazení na standardním monitoru VGA. To podstatně usnadnilo testovatelnost celého systému a zároveň usnadnilo možnost přechodu k výkonnějším kompresním algoritmům.

Z důvodů integrace výše uvedených bloků do systému jsem taktéž zrealizoval kontrolér paměti DDR a blok který umožnil její sdílení různými částmi systému.

Systém digitalizuje videosignál z kamery. V reálném čase jej komprimuje za použití algoritmu M-JPEG a kvantifikuje výstupní datový tok. Komprimovaný signál je ve stejném systému opět dekomprimován a obraz degradovaný ztrátou informace je zobrazen na připojeném VGA monitoru. Zároveň je na VGA monitoru zobrazen aktuální datový tok.

Na příloženém CD je kompletní návrh společně s vybranými katalogovými listy součástek a dalšími dokumenty, které byly použity při návrhu

7.1 Dosažené výsledky

Celý systém pracuje v reálném čase a jeho latence je menší než 80 ms. Kompresní poměr a tudíž i výsledný datový tok je určen převážně kvantizační tabulkou, kterou je možné nastavit. Dále je tento kompresní poměr ovlivněn charakterem snímaného obrazu. Při kompresi obrazu typického pro bezpečnostní kamery (uzavřená místnost, lidé na chodbě) se dosažený datový tok, s použitím standardní JPEG kvantizační tabulky, pohyboval okolo 6 MBit/s. Úpravou kvantizační tabulky se dá, za cenu mírného snížení kvality signálu, dosáhnout požadovaného datového toku.

7.2 Doporučení pro další vývoj

System byl koncipován jako část řetězce pro přenos videesignálu po klasické počítačové síti. Proto by dalším krokem měl být vývoj rozhraní pro komunikaci po této síti. Použitý obvod FPGA je zaplněn pouze ze 33% a je velmi pravděpodobné, že potřebné bloky pro komunikaci po síti bude možné realizovat v tom samém FPGA.

Další možností je vylepšení kompresního poměru rozšířením způsobu komprese a další algoritmičké kroky, jako například rozdílový snímek.

Literatura

- [1] Typl, P., Tuček, P.: Real Time Video Compression Using FPGA, Proceedings of the 6th International Student Conference on Electrical Engineering, POSTER, 2003, EI37
- [2] David Solomon: Data Compression, Springer, 1997
- [3] Watson, A.B., Hu, J., McGowan, J. F.: Digital Video Quality Metrics Based on Human Vision, Journal of Electronic Imaging, vol 10, 2001
- [4] [online] <http://www.chiariglione.org/mpeg/>
- [5] JPEG File Interchange Format v 1.02, International Telecommunication Union, revision v1.02, September 1992
- [6] [online] <http://www.xilinx.com>
- [7] Virtex II Platform FPGAs: Complete Datasheet, katalogový list, revize v3.3, Xilinx, June 2004
- [8] [online] <http://www.xess.com>
- [9] [online] <http://www.insight.com>
- [10] [online] <http://www.eda.org/vhdl-200x/>
- [11] [online] <http://www.eda.org/comp.lang.vhdl/>
- [12] A. Rushton: VHDL for Logic Synthesis, J. Wiley, 1999
- [13] SAA7113H - 9 bit Video Input Processor, katalogový list, revize v1.0, Philips, July 1999
- [14] Bt481 RAMDAC, katalogový list, revize v1.0c, Broketeer, November 1996
- [15] The I2C - Bus specifications, revize v2.1, Philips semiconductors, January 2000
- [16] Double Data Rate (DDR) SDRAM Specification, revize C, JEDEC, January 2004
- [17] [online] <http://www.opencores.org>
- [18] Libraries Guide, revize v6.1i, Xilinx, Inc., 2003
- [19] Jie Liang, Trac D. Tran: Fast Multiplierless Approximation of the DCT with the Lifting Scheme, IEEE Transaction on Signal Processing, vol 12, December 2001
- [20] Agostini, L. V.: Pipelined Fast 2-D DCT Architecture for JPEG Image Compression, 2000

Přílohy

Propojení mezi deskou XSV-300 a V2LC1000

Name	WireColor	XESS	Board SS	INSIGHT pin	Insight conn
VGA link					
P7	Green	108	A10	L1	41
P6	White-Or	103	A12	L2	45
P5	Green	102	A13	L4	47
P4	White-BI	100	A15	M2	51
P3	Blue	99	A16	L5	53
P2	Red	96	A18	T5	55
P1	White-BI	95	/OE	P6	57
P0	Yellow	93	D14	N7	59
PixClk	Blue	107	A11	L3	46
nVSync	Yellow	101	A14	M1	48
nHSync	White	97	A17	M6	52
nBlanc	Red	94	D15	T6	54
GND	Green	--			
GND	Green	--			
Video link					
FIELD			A4	C5	
FE	Red	87	D13	T11	
DV	Red	85	D11	M11	
DATA7	Red	84	D10	P10	
DATA6	Red	81	D8	M10	
DATA5	Red	65	A2	R13	
DATA4	Red	64	A3	T12	
DATA3	Red	57	A5	P11	
DATA2	Red	56	A6	N11	
DATA1	Red	54	A8	N10	
DATA0	Red	53	A9	T14	

DDR POWER UP SEQUENCE

1. Apply power and attempt to CKE at a low state (≤ 0.2 V). (all other inputs may be undefined)
2. Apply VDD before or at the same time as VDDQ.
3. Apply VDDQ before or at the same time as VTT and VREF.
4. Start Clock and maintain stable condition for 200 μ s (min).
5. After stable power and clock, apply NOP and take CKE high.
6. Issue EMRS-enable DLL and establish Output Driver Type.
7. Issue MRS-reset DLL and set device to idle with bit A8.
8. (an additional 200 cycles (min) of clock are required for DLL Lock)
9. Issue precharge command for all banks of the device.
10. Issue two or more Auto Refresh commands.
11. Issue MRS-Initialize device operation.

(If device operation mode is set at sequence 5, sequence 8 can be skipped.)

EMRS: Extended Mode Register Set

MRS: Mode Register Set