

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Color Space Conversion in Hardware for Multimedia Applications

Carlos Rodrigues

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor FEUP: Prof. João Canas Ferreira

Supervisor Synopsys: Eng. João Gonçalves


June 29, 2015

A Dissertação intitulada

“Color Space Conversion in Hardware for Multimedia Applications”

foi aprovada em provas realizadas em 13-07-2015

o júri



Presidente Professor Doutor António José Duarte Araújo
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto



Professor Doutor Manuel Gradim de Oliveira Gericota
Professor Adjunto do Departamento de Engenharia Eletrotécnica do Instituto
Superior de Engenharia do Porto



Professor Doutor João Paulo de Castro Canas Ferreira
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.



Autor - Carlos André Mendonça Rodrigues

Resumo

Em aplicações multimídia de transmissão de vídeo existe por vezes a necessidade de ter em conta as características do recetor, por exemplo negociar os formatos de espaços de cor ou de frequências de vídeo antes de iniciar a transmissão. Consequentemente, em situações em que o conteúdo não está disponível num espaço de cor suportado pelo recetor existe a necessidade de converter esse conteúdo para um espaço que seja suportado, para possibilitar a transmissão.

Esta tese desenvolve um módulo de *hardware* capaz de converter *streams* de vídeo entre diferentes espaços de cor em tempo real. Este módulo é sintetizável para frequências de 600 MHz em tecnologias de 40 nm. Suporta conversões de formatos RGB e YCrCb 4:4:4 e 4:2:2 entre os espaços de cor: ITU-R BT.601, ITU-R BT.709, ITU-R BT.2020, sRGB, opRGB, bg-sRGB, xvYCC601 e xvYCC709. Também suporta repetição de pixel e formatos de vídeo 3D.

O módulo desenvolvido implementa todas as etapas necessárias para converter corretamente entre espaços de cor, desde as conversões matriciais RGB para YCrCb e RGB para RGB à codificação e decodificação *gamma*. As opções de reamostragem dos canais de crominância são implementados por filtros de decimadores e interpoladores em duas configurações, de 30^a e 18^a ordem, entre as quais a primeira é compatível com os requisitos de filtros definidos pelos standards ITU-R BT.601 e ITU-R BT.709.

Abstract

In multimedia applications of video transmission there is usually the need to account for the receiver's capabilities, for example negotiating color formats and video frequencies before initiating the transmission. Consequently, in situations where the content is not available in a color space supported by the receiver, there is the need to convert that content to one color space that is supported.

This thesis develops an hardware module that is capable of performing real-time color space conversion of video streams. This module is synthesizable for 600 MHz frequencies in 40 nm technologies. It supports the RGB and YCrCb 4:4:4 and 4:2:2 conversions between: ITU-R BT.601, ITU-R BT.709, ITU-R BT.2020, sRGB, opRGB, bg-sRGB, xvYCC601 and xvYCC709 color spaces. This module is also compatible with pixel-repetition and 3D video formats.

The developed module implements all the steps required to properly convert between color spaces, from RGB to YCrCb and RGB to RGB conversion matrices to gamma encoding and decoding. The chroma resampling capability is implemented by chroma upsampling and downsampling filters, available in a 30 taps and a 18 taps configurations, from which the first is compliant with the templates defined in the ITU-R BT.601 and ITU-R BT.709 standards.

Acknowledgments

I would like to thank my supervisors at Synopsys and FEUP, Eng. João Gonçalves and Prof. João Canas Ferreira, for all the support and help provided during the development of this thesis.

I would also like to thank the team at Synopsys which welcomed me so well and helped me during the development of this thesis.

I would like to thank my family and my girlfriend for the all the support.

To my friends and Tuna de Engenharia which provided me with so enriching experiences and friendships during my student life.

Carlos Rodrigues

“Color is my day-long obsession, joy and torment.”

Claude Monet

Contents

1	Introduction	1
1.1	Context	1
1.2	Objectives and Contributions	2
1.3	Structure of the Document	3
2	Background Information	5
2.1	An Introduction to Colorimetry	5
2.2	Color Spaces	5
2.3	Gamma correction	7
2.4	Luma/Color-difference encoding	8
2.5	Video Formats and Structure	9
2.6	Color Space's Standards	12
2.6.1	ITU-R BT. 601	12
2.6.2	ITU-R BT. 709	12
2.6.3	ITU-R BT. 2020	13
2.6.4	IEC 61966-2-1/Amendment 1	14
2.6.5	IEC 61966-2-5	14
2.6.6	IEC 61966-2-4	15
2.6.7	Gamut Comparison	15
2.6.8	Summary	15
3	State of the Art of Architectures for Color Space Conversion	17
3.1	Color Space Conversion	17
3.2	Chroma Subsampling	22
3.3	Summary	26
4	Architecture Design	29
4.1	Introduction	29
4.2	Top Level Interface	29
4.3	Implementation Strategy	30
4.4	RGB to RGB Converter	33
4.5	R'G'B'-Y'Cr'Cb' Converters	34
4.6	Gamma Encoder and Decoder	36
4.7	Chroma Resampling Filters	38
4.8	Control Unit and Register Bank	41
4.9	Verification Environment	44
4.10	Verification Plan	45

5	Hardware Implementation and Results	49
5.1	Tools and Development Flow	49
5.2	RGB to RGB Converter	52
5.3	R'G'B'-Y'Cr'Cb' Converters	54
5.4	Gamma Encoder and Decoder	55
5.5	Chroma Resampling Filters	55
5.6	Control Unit and Register Bank	64
5.7	Top Level	64
5.8	Verification Results	65
5.9	Results	68
6	Conclusions and Future Work	77
	References	79

List of Figures

2.1	Color matching experiment (from [1]).	6
2.2	CIE 1931 chromaticity diagram with sRGB color space represented (from [2]).	7
2.3	Generic gamma correction transfer function (from [3]).	8
2.4	Color conversion process.	10
2.5	Video frame structure.	11
3.1	Direct implementation of conversion operation from [4].	20
3.2	Systolic architecture (from [5]).	21
3.3	Conversion matrix (from [5]).	21
3.4	Distributed arithmetic architecture (from [5]).	22
3.5	Comparison of results (from [5]).	22
3.6	Chroma subsampling schemes from [6].	23
3.7	Template for Cr and Cb filtering (from [7]).	24
3.8	Specification for filter 4:4:4 to 4:2:2 color-difference signals (from [8]).	25
3.9	Proposed filters comparison.	28
4.1	Top level interface	30
4.2	System architecture. Table 4.1 relates the top-level signal's naming of figure 4.1 with the higher level description of this figure.	31
4.3	Comparison of the SNR obtained for the RGB to RGB conversion for different coefficients widths for each video bit width.	35
4.4	Comparison of the probability of error obtained for the RGB to RGB conversion for different coefficients widths for each video bit width.	35
4.5	Architectures of the upsampling and downsampling filters.	39
4.6	Magnitude frequency response of the 30 th and 18 th order filters, against the template (dotted line). At the edge of the stopband $0.5\pi\text{rads/sample}$ or $0.25f_s$ the 30 taps and 18 taps filter magnitude responses are approx. -6 dB, being compliant with the template requirement of at least -6 dB.	42
4.7	Detail of the magnitude frequency response of the 30 th and 18 th order filters in the passband. The standards templates require the passband (frequencies up to $0.2f_s$ or $0.4\pi\text{rads/sample}$) attenuation to be $ A \leq 0.05\text{dB}$	43
4.8	Verification Environment architecture.	46
5.1	Comparison of the estimated area and power consumption of the RGB to RGB module with the number of pipeline stages implemented.	53
5.2	Comparison of the estimated area and power consumption of the RGB to YCC module with the number of pipeline stages implemented.	56
5.3	Comparison of the estimated area and power consumption of the YCrCb to RGB module with the number of pipeline stages implemented.	57

5.4	Comparison of the estimated area and power consumption of the gamma encoder module with the number of pipeline stages implemented.	58
5.5	Comparison of the estimated area and power consumption of the gamma decoder module with the number of pipeline stages implemented.	59
5.6	Downsampling filter structure.	60
5.7	Upsampling filters structure.	61
5.8	Register bank interface transfer diagrams.	64
5.9	Coverage results obtained from simulation.	65
5.10	Verification of the downsampling filters impulse response.	66
5.11	Verification of the upsampling filters impulse response.	67
5.12	RGB Rec.2020 non-constant luminance output image, input image read as RGB sRGB image.	71
5.13	Comparison of the Y'Cr'Cb' 4:4:4 Rec.601 image obtained using the developed module and the Matlab <i>rgb2ycbcr</i> function.	72
5.14	Comparison of Y'Cr'Cb 4:2:2 Rec.601 525-lines output images obtained with the two filter configuration, input image read as RGB sRGB image.	73
5.15	Comparison of Y'Cr'Cb 4:2:2 Rec.601 525-lines output images obtained with the two filter configuration and then converted to RGB using Matlab <i>ycbcr2rgb</i> function to improve the legibility of the text. Input image read as RGB sRGB image.	74
5.16	Comparison of the RGB 4:4:4 mandrill's images after being downsampled to YCC 4:2:2 and then upsampled to back to RGB 4:4:4, considering the Rec.601 525-lines color space.	75

List of Tables

2.1	Comparison of the gamut of the different color spaces, as a percentage of the visible colors gamut (from [9])	15
3.1	Comparison on the performances announced in the proposals found	23
4.1	Top level interface signal description.	32
4.2	Comparison of the variation of the RGB to RGB conversion coefficients width and SNR and probability of error measured for 16 bit width video.	34
4.3	Comparison on the length of the segmentation segments and the size of the LUTs required.	38
4.4	Comparison of the hardware costs of 30 th and 18 th order filters.	41
4.5	Verification Plan	47
5.1	Synthesis results of the constant-luminance R'G'B'-Y'Cr'Cb' converters modules.	55
5.2	Synthesis results of the 30 th and 18 th order resampling filters.	63
5.3	Top-level synthesis results of the 30 th and 18 th order filters configurations.	68

Abbreviations and Symbols

3D	Tri-dimensional
BBC	British Broadcasting Corporation
CAD	Computer-Aided Design
CEA	Consumer Electronics Association
CIE	Comission International de l'Éclairage
CRT	Cathode Ray Tube
DUT	Device Under Test
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
HDTV	High Definition Television
IEC	International Electrotechnical Commission
IP	Semiconductor Intellectual Property Core
ITU	International Telecommunication Union
ITU-R BT.	ITU Radiocommunication Sector Broadcasting service (television)
JPEG	Joint Photographic Experts Group
LCD	Liquid Crystal Display
LSB	Least Significant Bit
MSB	Most Significant Bit
NTSC	National Television System Committee
OVM	Open Verification Methodology
PAL	Phase Alternating Line
R&D	Research and Development
RGB	Signal Composed by Red, Green and Blue Components
Rec.	ITU-R BT. Recommendation
SDTV	Standard Definition Television
UHDTV	Ultra High Definition Television
UVM	Universal Verification Methodology
VHS	Video Home System
VMM	Verification Methodology Manual
YCrCb	Signal Composed by Luma and Color Difference Components
YCC	Same as YCrCb

Chapter 1

Introduction

1.1 Context

In multimedia applications of video transmission there is usually the need to account for the receiver's capabilities, for example negotiating color formats and video frequencies before initiating the transmission. Consequently, in situations where the content is not available in a color space supported by the receiver, there is the need to convert that content to one color space that is supported.

The representation of color in digital systems is done using mathematical models called color spaces. Each color space defines a set of coordinates to be used as primaries in the reference CIE XYZ color space, which contains all visible colors. These primaries are usually red, green and blue (RGB) although their exact location varies. The other representable colors are derived from these primaries, whose coordinates in the XYZ color space define the range of colors that each particular color space is capable of representing - its color gamut. Moreover, the use of different color spaces is also related with the capability of the display systems (televisions, computer monitors, projectors, printers, etc.) and their technology (CRT, LCD, LED, plasma, OLED, etc.) to display colors, providing different experiences to the viewers.

Besides the RGB model of representation of color, the luma/color-difference system (YCrCb or YCC) is also broadly used. Luma is a quantity related to the luminance of the picture, which, together with the color difference signals (Cr and Cb), generates a representation of color that can be derived from the RGB signals. The YCrCb model takes advantage from the fact that the human vision is less sensitive to color variations than to luma variations by providing chroma subsampling schemes. In these schemes, the number of chroma samples per luma sample is horizontally and/or vertically reduced, thereby allowing for a bandwidth reduction of the signal without a noticeable loss of image quality.

The development of image systems not only provided new color spaces for data encoding but also demanded the development of transmission and processing systems capable of supporting the increase of image resolutions from standard television to high definition (1920x1080) and ultra high definition television (4k and 8k). In addition, the introduction of 3D broadcasting formats and

the necessity of keeping backward compatibility with legacy contents also imposes new challenges for video systems. These improvements require hardware designs that are compliant with the various systems standards and provide the necessary performance capabilities.

When one of the systems involved in the video transmission (for example between a DVD player and a LCD television) is not compatible with the data encoding format, one possibility is to convert the content to a format that is mutually supported.

This MSc Dissertation has been developed in the context of the Master in Electrical and Computers Engineering at Faculty of Engineering of University of Porto. This thesis was proposed by Synopsys Portugal and has been developed at the company's offices in Maia, Portugal.

Synopsys, Inc. (Nasdaq:SNPS) is an American company headquartered in Mountain View, California, which is a global leader in software for electronic design automation (EDA) and semiconductor intellectual property (IP). Synopsys provides their costumers with solutions that *accelerate innovation*, allowing their clients to address the challenges of designing integrated circuits and bring their products to market faster with lower costs and schedule risks [10].

In its portfolio, Synopsys provides interface IP cores for multimedia systems as HDMI and MHL solutions, JPEG encoders/decoders and MIPI interfaces. For these systems, but also to any others that deal with video transmission or processing, real-time color space conversion of the video stream can be an interesting feature that adds value to the final product and increases its interoperability capabilities.

1.2 Objectives and Contributions

The goal of this work is the development of a hardware module capable of performing real-time color space conversions of video streams. The color spaces which are meant to be supported are defined in the following standards:

- ITU-R BT.601-5 - Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios [8];
- ITU-R BT.709-5 - Parameter values for the HDTV standards for production and international programme exchange [7];
- ITU-R BT.2020 - Parameter values for ultra-high definition television systems for production and international programme exchange [11];
- IEC 61966-2-1/Amendment 1 - Default RGB colour space sRGB and its counterpart sYCC. Also defines two extended gamut encodings: bg-sRGB and bg-sYCC [12];
- IEC 61966-2-5 - Optional RGB colour space - opRGB and its luma/colour-difference counterpart opYCC [13]. Referred in HDMI specs as $Adobe_{RGB}$ and $Adobe_{YCC601}$ [14];
- IEC 61966-2-4 - Extended-gamut YCC colour space for video applications - $xvYCC_{601}$ and $xvYCC_{709}$ [15].

The module should be capable of converting content coded in RGB and in the chroma subsampling formats YCrCb 4:4:4 and YCrCb 4:2:2.

The utilization of this converter in video formats with pixel-repetition and 3D video should also be considered.

This module is meant to be implemented using Verilog hardware description language and should be synthesizable for frequencies over 600 MHz in 40 nm technologies.

This work will contribute to the Synopsys IP portfolio with a module that supports real-time color space conversions of video streams between the main color spaces used currently, notably all the color spaces supported by the HDMI 1.4 and 2.0 specifications. The functionality offered by this module will enable Synopsys video interface IPs to connect with a broader number of devices which have limited color spaces support. This module is synthesizable for the frequencies needed to support all the video formats supported by HDMI specifications and features pixel-repetition and 3D modes compatibility, which enables the compatibility with great part of the video formats supported by those video transmission standards.

Furthermore, this work will contribute to science and engineering with the knowledge and results obtained from the implementation of a computer-intensive task as color space conversion of video streams in an hardware design, which enlarges the number of applications for ASIC and FPGA technologies and can be useful for future works in the area of hardware accelerated video and image processing.

1.3 Structure of the Document

An overview of background information concerning color and video theory are presented in chapter 2, as well as a description of the standards meant to be supported. The proposals found in the state of the art research are discussed and analyzed in chapter 3.

The design of the architecture of the system and the verification methodology are discussed in chapter 4. After a brief presentation of the EDA tools used and their project flow in chapter 5, follows the presentation of the hardware implementation, verification results and synthesis results obtained for the final implementation from which some conclusions are driven in chapter 6, and future developments for this module capabilities are proposed.

Chapter 2

Background Information

2.1 An Introduction to Colorimetry

The representation of color in both analog and digital systems is based on the properties of the human visual system.

The human vision sensation is created in the eye by the incidence of radiation in the cells known as cones and rods. Rods are a type of eye cells that are particularly sensitive to light intensity and provide us the notion of luminosity. There are three types of cone cells, with different responses to light wavelengths. The combination of this tri-stimulus signal in the brain is what gives us the sensation of color. The brain perceives the different wavelengths of light in the visual spectrum as colors.

Moreover, according to Grassman's Third Law, the linear combination of light in different colors (wavelengths) can be used to generate other colors [16]. This is the principle of the additive color model which is observable in the color-matching experience (Figure 2.1) in which a user can match the colors reflected in two targets inside a black box by controlling the intensity of the controllable light sources [17].

Inside the box, one target reflects the light from one reference color source and a separate target reflects the light from three controllable sources of light. Each controllable source has a different fixed wavelength (for example red, green and blue) and the user can match the reflected colors in both targets by adjusting the intensity of these three controllable sources.

2.2 Color Spaces

In 1931 the International Commission on Illumination (CIE - Commission Internationale de l'Éclairage) defined a tri-coordinate system for a Standard Observer in which each color is defined with three primary stimulus X, Y and Z using three sensibility functions [16], similarly to the color matching experience. In this system Y is meant to represent luminance from the scene.

From the normalization of X, Y and Z a chromaticity diagram is obtained (Figure 2.2), where colors are represented for the same amount of luminosity. This chromaticity diagram represents

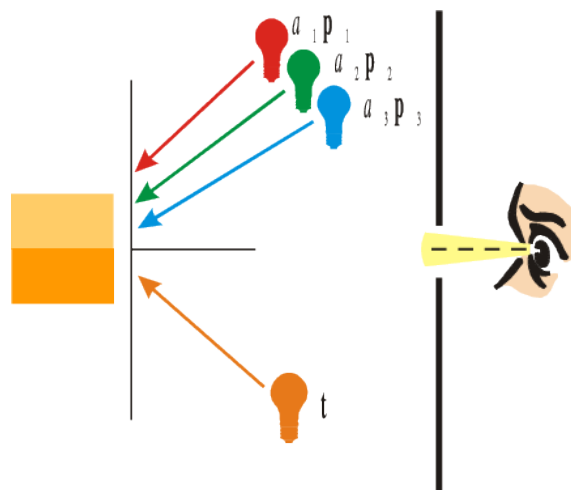


Figure 2.1: Color matching experiment (from [1]).

the complete color gamut, i.e. all colors of the visible spectrum (400-700 nm), with white being approximately at the center of the diagram and the pure wavelengths (monochromatic colors) located in the spectral locus line (borders of the diagram with the wavelengths marked in figure 2.2). The chromaticities from which this diagram is drawn are known as the CIE chromaticities. A set of white points coordinates are also defined by the CIE, for instance the D_{65} white point, which is broadly considered for broadcasting system, because of the type of illumination used in filming studios and the assumptions made about the visualization conditions.

The CIE XYZ color space is not useful for practical implementation because it is impossible to develop a display that would have the XYZ tri-stimulus functions spectral density distribution and properly display colors, as it would require the light source to emit negative light power. Although there are displays that work with the XYZ stimulus, they do not implement their spectral density power.

Consequently, derived from this color space, other color spaces were developed that support a portion of the CIE 1931 color space and are physically realizable. To define a color space one should specify its white point and three primaries between which lie all the colors existing in that color space.

From the shape and color distribution of the chromaticity diagram it is possible to conclude that using red, green and blue as primaries maximizes the surface of the diagram covered by the new color space. Indeed, the majority of the additive color spaces use these same primaries only differentiating from each other in their exact location.

The conversion between the CIE XYZ color space and a defined linear RGB color space can be made by computing the multiplication between the input vector and a matrix of conversion

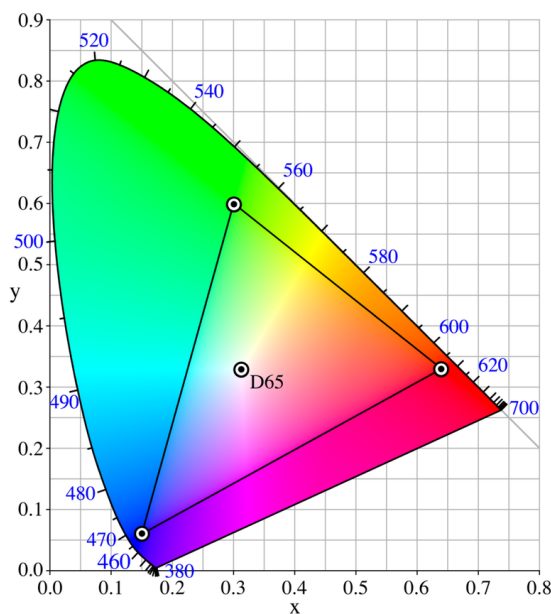


Figure 2.2: CIE 1931 chromaticity diagram with sRGB color space represented (from [2]).

coefficients, derived from the chromaticity coordinates of the color space.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.1)$$

2.3 Gamma correction

The human vision system sensibility to luminance is not linear, it is more sensitive to the same relative variation of luminance in low intensities than in high luminosity intensities. Taking advantage of this property, it is possible to reduce the number of bits used in the intensity codification by expanding the quantization intervals for high luminance values and compressing them for low intensity values, to which we are more sensible so increased precision is desirable. This is done by applying a nonlinear transfer function that implements this behavior - gamma correction (Figure 2.3). Not using gamma encoding would require a larger number of bits than necessary this way to code the luminance for the same perceptible amount of quantization noise.

Generally each color space defines its own transfer function coefficients, but they are all approximately the form of an exponential function that for some color spaces is piece-wise defined for the values near zero where the exponential functions are steeper.

After this transformation applied in the source devices of the transmission chain it would be expected the inverse transformation to be applied in the sink's end, the monitor's decoder. Curiously, the CRT monitors transfer function is also nonlinear and it is approximately the inverse of the gamma correction transfer function. Consequently, to reduce the overall system's complexity

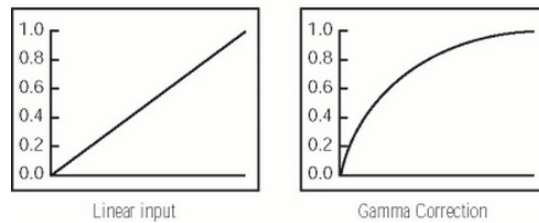


Figure 2.3: Generic gamma correction transfer function (from [3]).

video engineers decided to let the compensation of the gamma correction to be directly applied by the CRT's nonlinear power function. Modern non-CRT monitors (LCDs or plasma, p.e.) are expected to be compliant with this decision, by applying the gamma compensation function adequate to their power transfer response.

When a signal is gamma corrected it should be written with apostrophes like in $R'G'B'$ to indicate that it isn't a linear light signal.

When converting between color spaces one should account for the differences in gamma correction of each color space, which usually means converting the signal to linear RGB before color space conversion and after it re-apply the gamma transformation. Otherwise noise would be being added to the output.

2.4 Luma/Color-difference encoding

Besides the RGB representation of color in multimedia systems, one can take advantage of the human vision inferior sensitivity to color variations than to luminance variations and use the luma/color-difference representation. This representation encodes color in one luma and two color channels, which can be sub-sampled (filtering high frequency chromatic signals) to save bandwidth, due to the less sensitiveness of the human vision to this content. Several luma/color-difference encodings exist both in analog and digital systems. In analog systems we talk of composite coding like YUV or YIQ in NTSC system and in digital systems we talk usually of component coding noted as YCrCb. During the analog broadcasting systems period this encoding allowed backwards compatibility with black and white systems (which supports only the Y channel) and allowed the reduction of the necessary bandwidth, as it does also in the context of digital systems.

All luma/color-difference systems are built on the premise, from color science, that in RGB the luminance is concentrated mainly in the green component. This is implied in the values of the coefficients used in the computation of Y' , where the green channel has a weight of 60 to 70%. Therefore, after deriving the luma value the color components are extracted from the blue and red channels. The chroma channels are coded as the difference between blue and red channels and the luma channel by subtracting the remainder of luma present in R and B components.

$$Y' = x_r R' + x_g G' + x_b B', \quad (2.2)$$

$$Cr' = R' - Y', \quad (2.3)$$

$$Cb' = B' - Y', \quad (2.4)$$

In Y'Cr'Cb' system, luma (Y') is a weighted sum of R', G' and B' parameters which is a quantity related to the luminance of the image. Note that Y' is computed after the RGB signal is gamma corrected, therefore it is built upon a nonlinear signal R'G'B', not after the CIE linear-light luminance, so it should also be written with an apostrophe. Cr' and Cb' are the color-difference signals, respectively R'-Y' and B'-Y', which can be sub-sampled with respect to luma (Y') signal without visually evident loss in quality, due to the less acuity of the human visual system to the color, compared to luminosity. There are several chroma subsampling schemes that are explained in section 3.2.

In Y'Cr'Cb' coding it is possible to represent a wider gamut of colors than in the original R'G'B' signal so it is usually necessary to limit the excursion of the Y'Cr'Cb' to keep it inside the valid R'G'B' values.

R'G'B' and Y' signals swing in real number interval [-1;1] and Cr' and Cb' in [-0.5;+0.5]. Their digital representation is usually in 8-bit codification (0-255) for standard quality although some standards define 10-bit or more. This feature is announced in multimedia products with the name of Deep Color support. These extra bits increase the quality of the picture by reducing the quantization noise, they add precision on sampling and signal processing computations and the downsizing to 8-bit coding can be made by discarding the extra least significant bits.

Moreover, some standards guarantee a headroom and footroom in signal excursion, that is, they limit the signal excursion to a interval smaller than $2^{N_{bits}}$ to guarantee codes over and under the reference signal excursion which allow overshoots and undershoots in the signal caused by signal processing.

The color conversion between different color spaces (not considering gamma encoding and decoding) consists essentially in a matrix transformation applied to the input tri-component signal like the operation represented in 2.1.

This operation requires the input signal to be upsampled to a 4:4:4 representation and down-sampled to the desired chroma subsampling scheme after the matrix operation. Therefore, the color space conversion will be divided in two phases: the chroma subsampling and the 4:4:4 color conversion operations. The path of the color conversion process is illustrated in figure 2.4.

2.5 Video Formats and Structure

In video stream the video images are not the only data transmitted. In fact, the video frame contains one vertical and one horizontal blanking spaces, respectively at the top and at the left of the active video area in the video frame (figure 2.5). This blanking periods are inherited from the era of

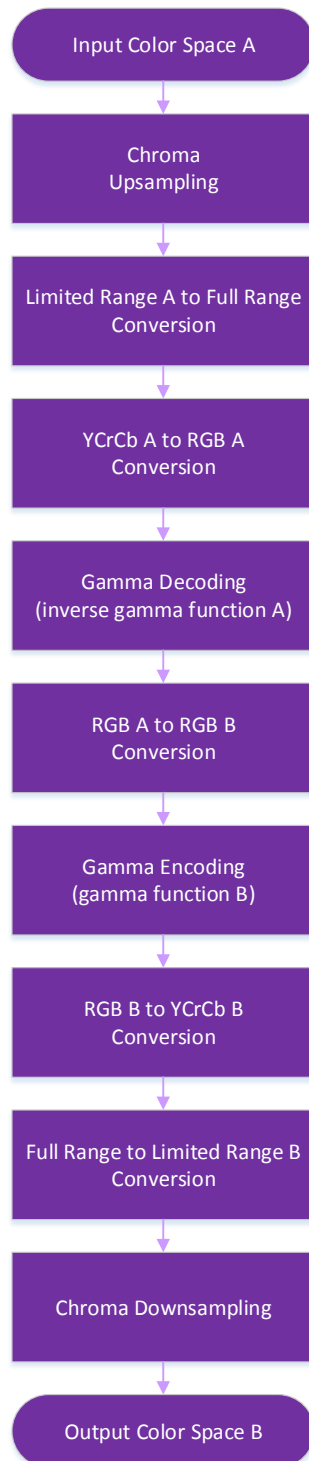


Figure 2.4: Color conversion process.

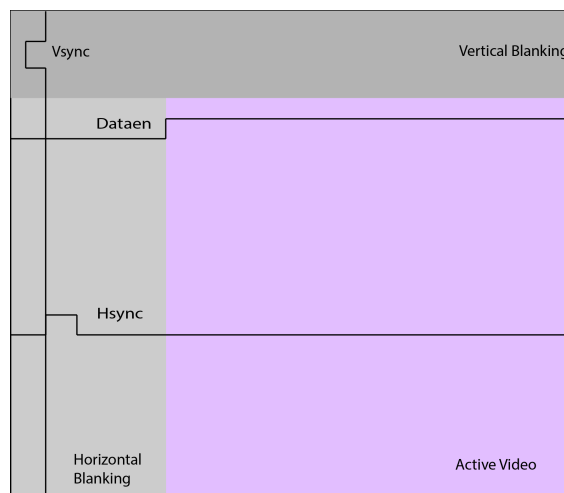


Figure 2.5: Video frame structure.

cathode ray tube television, where they were need to allow the electron gun the time to return to beginning of the line, or to the beginning of the screen. The blanking periods have variable sizes, depending on the video resolutions and pixel clock frequencies, and are used for audio data transmission, as well as other type of data or meta-data, like information about the content being transmitted [16].

Parallel to the video data signals, there are three control signals that ensure the synchronization across the video frame structure. Horizontal sync signal (hsync) is a signal that has a pulse in each line, during the horizontal blank period, marking the beginning of a new line although the pulse doesn't start necessarily at beginning of the line. The vertical sync signal (vsync) is pulsed once in a frame, during the vertical blank period, indicating the beginning of a new frame, although the pulse may start after the first line of the frame and last for several line periods. Finally, active data or data enable signal indicates that the data transmitted in the data channels is valid, that is, we are in a active video period. The duration of this signals is illustrated in figure 2.5.

The video pictures can be transmitted in three formats: progressive, interlaced, or segmented. In progressive format, one complete picture (one field) is transmitted in each active video frame. In interlaced format, in each active frame two different fields are transmitted in alternating lines, one from each field and usually one field is transmitted in two consecutive frames: one carries the odd lines and the next the even lines. Segmented frames are a technique used to split progressive frames so they can be used in interlaced systems.

In 3D video the right and left fields are transmitted in one of three schemes [14]:

- Top-bottom - where the fields are transmitted one after the other in the same frame or in consecutive frames;
- Side-by-side - the two fields are transmitted by dividing half of the horizontal active video area to each one;

- Line alternative - the two fields are transmitted in alternating lines, similar to interlaced video;

Moreover, in HDMI systems, when the video resolution does not have the minimum pixel rate to be transmitted under this systems pixel-repetition is applied to increase the pixel rate 2 to 10 times and allow transmission [14].

2.6 Color Space's Standards

2.6.1 ITU-R BT. 601

This standard [8] was first published in 1982 and defines the studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. The 7th and last revision of this standard dates from 2011.

Recommendation ¹ 601 sets two sampling frequencies families: 13 MHz for 4:3 and 16:9 ratios and 18 MHz for 16:9 aspect ratio. The encoding should be R'G'B' or Y'Cr'Cb' 4:2:2, both co-sited ², and it defines the coefficients for transforming between these formats for gamma corrected signals. It allows for 8-bit or 10-bit encoding, which in the latter case the two extra bits are indicated to represent a fractional part of the signal value.

R'G'B' signals use the full 255-0 signal excursion while Y' is only allowed 220 values with footroom and headroom defined by reference black at level 16 and reference white at level 235. Cr' and Cb' signals may use 225 levels with 0 value referenced at level 128. An optimized derivation and quantization method for the coefficients for R'G'B' to Y'Cr'Cb' conversion that minimizes the quantization error of the Y'Cr'Cb signal is proposed.

The two sampling frequency families define 525-lines, 60 fields/s and 625-lines, 50 fields/s formats, both interlaced. These formats are also referred to by 480i and 576i, respectively.

Since revision 6 of this standard chromaticity coordinates based on CIE 1931 color space are defined, as well as a gamma correction transfer function. Previous versions of the standard didn't define his although the NTSC and PAL standards chromaticities were conventionally used. It is recognized in the recommendation that it is common practice to use HDTV RGB contents re-mapped to SDTV RGB format without performing the proper colorimetry conversion, due to the similarities of both color spaces.

2.6.2 ITU-R BT. 709

Recommendation 709 [7] defines parameter values for the HDTV standards for production and international programme exchange.

The definition of HDTV is carried from Report ITU-R BT.801 [18] where "a high-definition system is a system designed to allow viewing at about three times the picture height, such that

¹ITU-R BT standards are self-referred as Recommendations.

²Co-sited means the three components of the pixel (R, G and B or Y, Cr and Cb) are sampled in the same spatial point of the picture.

the system is virtually, or nearly, transparent to the quality of portrayal that would have been perceived in the original scene or performance by a discerning viewer with normal visual acuity". This standard was first published in 1990 and its last revision (5th) dates from 2002.

The document is divided in two sections, one part that relates HDTV systems to conventional television, defining parameters used in early analogue HDTV systems, and one part for HDTV systems with square pixel common image format (CIF) which aims to specify picture parameters independently of picture rates.

Early days systems define 1125 lines, 60 Hz and 1250 lines, 50 Hz formats and CIF specifies several picture rates: 60 Hz, 50 Hz, 30 Hz, 25 Hz, 24 Hz, 60/1.001 Hz, 30/1.001 Hz and 24/1.001 Hz.

CIF supports progressive (P), interlaced (I) and progressive segmented frames (PsF) scheme. A set of rules to allow progressive images to be transported as segmented frames (and the contrary) is defined.

The second part of this standard sets sampling frequencies at 148.5 MHz, 74.25 MHz and 74.25/1.001 MHz for R'G'B' and Y', half those frequencies for Cr' and Cb'. Chroma subsampling supported schemes are the same as in Rec. 601.

Both parts use the same chromaticity coordinates and gamma correction transfer function. The chromaticity coordinates are different from the ones defined in Rec.601.

As previously stated in 2.6.1, the chromaticity coordinates from Rec. 709 and Rec.601 are not very different. However the luma/color-differences encoding coefficients from this two standards differ significantly and proper conversion must be implemented.

Signal excursion and binary representation rules are the defined for SDTV in Rec. 601.

2.6.3 ITU-R BT. 2020

This recommendation [11] defines parameter values for ultra-high definition television systems for production and international programme exchange. It was first published in 2012 and its last revision (1st) is from 2014.

Ultra-high definition television (UHDTV) enhances viewers experience by expanding the size of the screen size, both for home and public places. This aims to improve the observer sense of *being there*.

Rec. 2020 specifies its chromaticity coordinates and gamma correction transfer function. It defines different gamma correction parameters for 10-bit and 12-bit coding.

Besides R'G'B' and Y'Cr'Cb' formats it also defines a $Y'_C Cr'_C Cb'_C$ which is constant luminance luma/color-difference coding. $Y'_C Cr'_C Cb'_C$ has different conversion coefficients from Y'Cr'Cb' and its luma is derivated from linear RGB and gamma correction is applied already in luma format.

This standard supports three chroma subsampling schemes: 4:4:4, 4:2:2 and 4:2:0, all co-sited. Signals are quantized in 10-bit or 12-bit coding and it also defines headroom and footroom intervals for every signal.

The need for constant luminance $Y'_C Cr'_C Cb'_C$ coding arises from the constant luminance issue stated in Rep. ITU-R BT.2246-3 [19]. In fact, when computing luma Y' from $R'G'B'$ gamma-corrected signals in the matrix operation some of the energy in luminance signal is leaked to chroma components. After chroma subsampling, errors generated in chroma signals in the re-sampling operations will appear also in the luminosity of the picture after reconversion to $R'G'B'$ format. Due to the inversion of the gamma encoding and RGB to YCrCb conversion, constant luminance encoding fixes this issue.

2.6.4 IEC 61966-2-1/Amendment 1

This standard [12] was first published in 1999 and defines the sRGB color space. The amendment dates from 2003 and it adds the specifications for sYCC encoding, as well as bg-sRGB and bg-sYCC.

sRGB color space is meant to be compatible with Rec. 709 and it aims to further improve the standardization achieved by Rec.709 by defining a reference display, reference viewing conditions and a reference observer. The main difference from Rec. 709 is the gamma correction transfer function specified.

This amendment adds the sRGB color space for a luma/color-difference space sYCC. Both sRGB and sYCC spaces use 8-bit quantization by default, although longer code words are allowed, notably 16-bit.

An extended gamut version of this color spaces is also presented: bg-sRGB and bg-sYCC. The default encoding bit depth of this spaces is 10-bit and when converting from sRGB, input negative signal values are allowed, as well as values greater than unity.

The sYCC color space defined in this standard is referred in HDMI specifications as $sYCC_{601}$ [14], where the 601 indicates that the $R'G'B'$ to $Y'Cr'Cb'$ conversion is done using the coefficients of Rec.601.

2.6.5 IEC 61966-2-5

This standard [13] was published in 2007 and specifies a color space based on sRGB but with wider color gamut: opRGB and its luma/color-difference version opYCC.

It presents reference image display system characteristics, a reference observer and viewing conditions. The chromaticity coordinates here defined are different from Rec.709 and sRGB, as they provide a wider gamut while having a default bit depth of 8-bit. Signal quantization with more than 8-bit is allowed.

The opRGB and opYCC color spaces are referred in HDMI specifications as $Adobe_{RGB}$ and $Adobe_{YCC601}$ [14]. Except for some differences in image display system characteristics, reference observer and viewing conditions this standard parameters are also the same as defined in $Adobe_{RGB}_{1998}$ standard [20].

2.6.6 IEC 61966-2-4

This standard [15] extends the Rec.709 [7] color space by allowing a wider signal excursion in the Y'Cr'Cb encoded signals. It was published in 2006.

The Y'Cr'Cb' signals in standard Rec.709[7] are limited between the reference white and black points (235 and 16, for 8-bit encoding), although it is stated that overshoot signals can go beyond this limits, guaranteeing that values 0 and 254 are reserved for synchronization purposes.

This standard removes this ambiguity in the limitation of the video signals, allowing them to use the full excursion, except the synchronization values. This extension provides the Rec.709 color space almost the same gamut area that sYCC and sRGB (sYCC and sRGB don't have reserved codes).

2.6.7 Gamut Comparison

The following table presents a comparison on the gamut area of each standard, compared to the area of the full gamut (all visible colors) [9][21]³.

The extended gamut color spaces(bg-sRGB, xvYCC) and luma/color-difference encoded colors spaces have wider gamuts than the colors spaces to which they relate, however no data was found to do the comparison.

Color Space	% Visible Colors
Rec.601 625 lines	35.7 %
Rec.601 525 lines	31.9 %
Rec.709 / sRGB	35.0 %
opRGB	50.6 %
Rec. 2020	75.8 %

Table 2.1: Comparison of the gamut of the different color spaces, as a percentage of the visible colors gamut (from [9])

2.6.8 Summary

From the analysis of each standard, some requisites and implementation limitations are herein summarized:

- The standards considered define 7 RGB color spaces: Rec.601 525-lines, Rec.601 625-lines, Rec.709 CIF, Rec.2020, sRGB, bg-sRGB and opRGB. This results in 20 unique RGB to RGB conversion matrices.
- From the above RGB color spaces, 11 YCrCb color spaces are derived: Rec.601 525-lines, Rec.601 625-lines, Rec.709 1125-lines, Rec.709 1250-lines, Rec.709 CIF, Rec.2020, sYCC,

³Due to differences in naming these values are inferred from the color spaces compared in the references [9][21] by comparison of their chromatic primaries and white point with the ones from the color spaces considered here. The two sources considered don't agree in the values for some of the color spaces, but the differences are in the order of units and don't interfere with the relative sizes between different color spaces.

bg-sYCC, opYCC, $xvYCC_{601}$ and $xvYCC_{709}$. This results in 6 RGB to YCC and 6 YCC to RGB unique conversion matrices.

- These color spaces use three different gamma functions: one for Rec.609, Rec.701 and Rec.2020 color spaces, one for opRGB color space and one for sRGB color space.
- Rec.2020 also defines a constant luminance Y'Cr'Cb' encoding. This encoding requires significantly architectural changes compared to the non-constant luminance encodings because of the differences in the order of operations concerning the gamma encoding and conversion from RGB to Y'Cr'Cb', which require a separate datapath to support this encoding.
- The standards considered define 6 different allowed signal ranges, from full range RGB and YCC 0-255⁴, to limited ranges 16-240 for RGB and Y' signals and 16-235 for Cr'Cb' signals, or extended ranges 1-254 for RGB and Y'Cr'Cb signals in extended gamut color spaces. Additionally, bg-sRGB and bg-sYCC color spaces define particular ranges and scaling for their signals.

⁴Represented here for 8-bit width signals, for simplification.

Chapter 3

State of the Art of Architectures for Color Space Conversion

In this chapter, we will divide our study in two sections: first the pixel to pixel operations: RGB to RGB conversion, RGB to YCrCb and YCrCb to RGB conversions, gamma encoding and decoding, and then the filtering operations of chroma upsampling and downsampling.

3.1 Color Space Conversion

Conversion between two tri-component color spaces consists in a 3x3 matrix operation applied to an input vector of size 3 as exemplified in the matrix of Section 2.1. This applies both to RGB to RGB conversion and RGB to YCrCb conversion. This operation can also be described in the form of three equations:

$$X = x_r R + x_g G + x_b B \quad (3.1)$$

$$Y = y_r R + y_g G + y_b B \quad (3.2)$$

$$Z = z_r R + z_g G + z_b B \quad (3.3)$$

The example is given for RGB to XYZ color space conversion without loss of generality.

The matrix operation consists of 9 multiplications and 3 sums, where x_i, y_i, z_i are fixed coefficients in each type of conversion. After this, it is necessary to round the result which may also have to be clamped or clipped, to ensure it is between the allowed excursion ranges. Usually the result must be added to a constant value, notably if the color space ensures signal footroom. The most efficient way to round the value to a integer value, for example, is add 0.5 to it and then truncate it.

However, as it has been previously stated in Section 2.3 when converting between different color spaces the gamma correction of the signals must be considered to properly apply the conversion. Only when converting between RGB and YCrCb formats in the same color space this can be discarded, for example between $R'G'B'_{Rec.709}$ and $Y'Cr'Cb'_{Rec.709}$.

Lets consider a practical example: the conversion from the sYCC 4:2:2 color space to the Rec.601 525-lines YCC 4:2:2 color space considering 8-bit depth data¹. First of all, it is necessary to upsample the input data to a 4:4:4 signal by applying an appropriate interpolation filter. After that, the data is normalized to $[0; +1]$ and $[-0.5; +0.5]$ intervals:

$$Y'_{sYCC} = Y'_{sYCC(8)}/255, \quad (3.4)$$

$$Cr'_{sYCC} = (Cr'_{sYCC(8)} - 128)/255, \quad (3.5)$$

$$Cb'_{sYCC} = (Cb'_{sYCC(8)} - 128)/255. \quad (3.6)$$

The sR'G'B' signal is obtained from:

$$\begin{bmatrix} R'_{sRGB} \\ G'_{sRGB} \\ B'_{sRGB} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} Y'_{sYCC} \\ Cr'_{sYCC} \\ Cb'_{sYCC} \end{bmatrix}, \quad (3.7)$$

where a_{ij} are the coefficients defined in [12]. Before converting to the Rec.601 525-lines RGB color space it is necessary to decode the gamma correction of the signal, to obtain the linear sRGB. In this particular case for sRGB color space, the gamma correction decoding function is defined in branches:

$$x = \begin{cases} -\left[\frac{-x'+0.055}{1.055}\right]^{2.4} & \text{if } x' < -0.04045 \\ \frac{x'}{12.92} & \text{if } -0.04045 \leq x' \leq 0.04045 \\ \left[\frac{x'+0.055}{1.055}\right]^{2.4} & \text{if } x' \geq 0.04045 \end{cases} \quad (3.8)$$

where x represents the obtained linear signal and x' the input gamma encoded signal.

The standards may define the coefficients for converting their particular color space to and from CIE XYZ color space or define the coordinates of their chromaticity primaries from which it is possible to obtain the coefficients. The conversion from the input color space to CIE XYZ color space and then to the output color space is done in a linear domain so this double step operation can be merged in one by multiplying the two intermediate conversion matrices and obtaining matrix B_{ij} :

$$\begin{bmatrix} R_{Rec601525} \\ G_{Rec601525} \\ B_{Rec601525} \end{bmatrix} = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix}. \quad (3.9)$$

Before converting to Rec.601 525-lines Y'Cr'Cb the content should be gamma encoded with the Rec.601 525-lines gamma transfer function, which is also defined in branches:

$$x' = \begin{cases} 1.099x^{0.45} - 0.099 & \text{if } 1.00 \geq x \geq 0.018 \\ 4.500x & \text{if } 0.018 \geq x \geq 0 \end{cases} \quad (3.10)$$

¹Despite the notation used (YCC) both this spaces are non-linear (Y'C'C).

and only then converted to the luma/color-difference format where C_{ij} coefficients are defined in [8] :

$$\begin{bmatrix} Y'_{Rec601525} \\ Cr'_{Rec601525} \\ Cb'_{Rec601525} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} R'_{Rec601525} \\ G'_{Rec601525} \\ B'_{Rec601525} \end{bmatrix}. \quad (3.11)$$

Finally, chroma subsampling can be done by applying the decimator filter and the results can be coded in 8-bit depth, obtaining opYCC 4:2:2 format.

$$Y'_{Rec601525(8)} = \text{round}[(Y'_{Rec601525} \times 219) + 16], \quad (3.12)$$

$$Cr'_{Rec601525(8)} = \text{round}[(Cr'_{Rec601525} \times 224) + 128], \quad (3.13)$$

$$Cb'_{Rec601525(8)} = \text{round}[(Cb'_{Rec601525} \times 224) + 128]. \quad (3.14)$$

In Rec.601 525-lines Y'Cr'Cb the signals should be limited as following:

$$235 \geq Y'_{Rec601525(8)} \geq 16 \quad (3.15)$$

$$240 \geq Cr'_{Rec601525(8)} \geq 16 \quad (3.16)$$

$$240 \geq Cb'_{Rec601525(8)} \geq 16 \quad (3.17)$$

A research on available solutions for color space conversion and on scientific publications addressing this issue was made and the conclusions are presented next. It wasn't possible to find solutions that implement all the operations concerned in this module, that is, proposals found only consider the RGB to YCrCb and YCrCb to RGB conversion or chroma resampling. It was also researched proposals for the implementation of the gamma encoding and decoding modules.

Altera has two Color Space Converter cores in its Video and Image Processing Suite [22] with different features that allow conversion between RGB and YCrCb for SDTV and HDTV color spaces, with components encoded in 4 to 20 unsigned bits. This IPs allow the configuration of the coefficients (from the set available or custom) to use in runtime, as well as the methods for rounding and scaling the data. Each coefficient is represented using fixed-point with 0 to 16 integer bits and 0 to 34 fractional part bits.

Xilinx LogiCORE IP family provides two cores for color conversion: RGB to YCrCb Color-Space Converter [4] (Figure 3.1) and YCrCb to RGB Color-Space Converter [23]. Both this cores support SD and HD resolutions in Rec.601 and Rec.709 color spaces with 8,10,12 and 16-bit depth and the conversion implementation is optimized by pre-computing all the arithmetic between fixed parameters.

Lattice Semiconductor also offers a Color Space Converter IP [24] RGB to YCrCb in its LatticeCORE products, supporting SD and HDTV conversions, 8 to 16 bits data and parametrized coefficients from 9 to 18 bits.

iWave Systems Technologies has a core[25] that converts from YCbCr to RGB using a pipeline implementation of the matrix operations that runs for 8-bit video data.

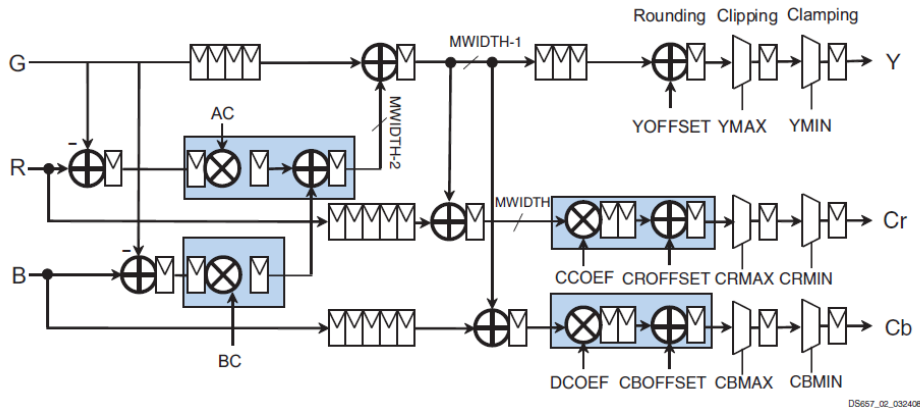


Figure 3.1: Direct implementation of conversion operation from [4].

Several papers concerning the implementation of color space conversion in FPGAs were found, although some concern conversions to color spaces not considered here (HSV, Lab, CMYK), others simply implement the conversion without further optimization, considerations or studies. A selection of relevant papers is presented next.

Bensaali et al.[5] proposes two alternative approaches to the implementation of the color space conversion, one using distributed arithmetic and one using a systolic design. These architectures aim to improve the performance of the color conversion cores for FPGA applications. Despite these alternatives are presented here, in this project it was chosen to optimize the core through parallelization and pipeline techniques, while conserving the architecture of the operations.

The systolic design consists in dividing the system in a network of processing elements (PE) that compute and pass data across the system as described in figure 3.2. To help understanding the system the correspondent conversion matrix is in figure 3.3. This matrix differs from the one presented in 2.1 because it includes the summation of constant values A_{i3} for setting an offset in each component. This design is also presented in an alternative architecture which requires less PE elements in exchange for longer computation times, but the same principles apply.

Distributed arithmetic is a technique that decomposes multiplications into bit level operations that can be precomputed. This allows for the use of ROM tables to store the precomputed data and surpass the need for multipliers, which usually need more area and longer clock cycles.

An example for R'G'B' to Y'Cr'Cb' conversion is presented in the paper. Using data encoded with 8-bit depth it would require 3 ROMs (one for each matrix row) with $2^N = 2^4 = 16$ entries. N is the number of columns in the matrix presented in figure 3.3.

A parallel implementation of this system is illustrated in figure 3.4 that allows the computation of the 8 bits of each component in a cycle, after a initial latency of 8 cycles. A serial version of this system is also presented in the paper.

The author compares his proposals with other existing cores for FPGA applications and presents the results in the table of figure 3.5. It concludes that the distributed arithmetic approach effec-

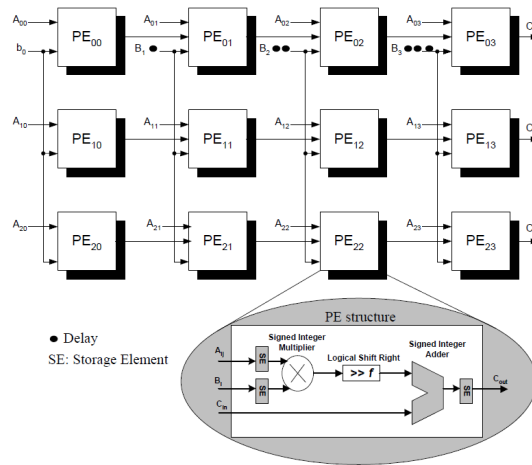


Figure 3.2: Systolic architecture (from [5]).

tively improves performance in terms of area and running frequency when compared with the other systems.

A "fast method" [26] for 8-bit video RGB to YCrCb conversion is proposed substituting the multiplications by lookup tables (similar to the Distributed Arithmetic concept) and pipelining the add operations, which allows this architecture to be implemented in FPGA's without embedded multipliers, as synthesizing multipliers directly in an FPGA is very inefficient in terms of area and performance.

Several constant matrix multiplications algorithms for hardware implementation are discussed in [27]. The authors compare their hand-optimized implementation results with the ones presented in 3.5 from [5], which achieve 229 MHz with full pipelining, but using only 140 slices, or 105 Mhz and 74 slices without pipelining, in a not specified Xilinx FPGA.

The following table 3.1 summarizes the performance results collected from the references studied. However, it doesn't compare the neither the area and power statistics from this proposals neither different levels of features in each one of them. The observations column indicates the FPGA boards used to implement the modules, when indicated by its authors, for better comparison. The frequency values achieved may be dependent on the bit width configured or other options.

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \end{pmatrix} \times \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ 1 \end{pmatrix} \quad (3)$$

Figure 3.3: Conversion matrix (from [5]).

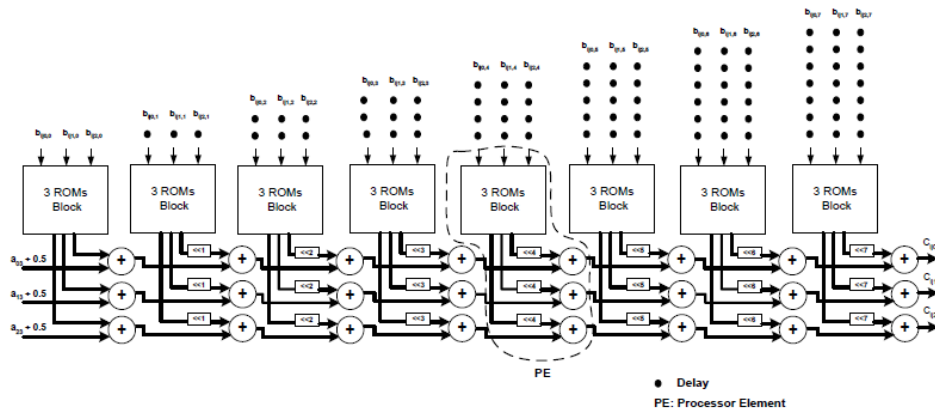


Figure 3.4: Distributed arithmetic architecture (from [5]).

3.2 Chroma Subsampling

It was referred in 2.4 that one of the main objectives in representing images in color-difference formats is to allow for chroma subsampling, providing a reduction in data bitrate with little degradation of picture quality due to the less sensitivity of the human vision to color variations when compared to sensitivity to luminosity variations.

The notation used to refer to the scheme where all samples are present is 4:4:4. This notation applies both to R'G'B' systems where no subsampling is allowed and to Y'Cr'Cb' with all samples. The chroma sampling schemes are always defined with respect to the luma sampling rates.

Chroma samples can be subsampled both horizontally or vertically. Horizontal subsampling provides the 4:2:2 format, where in each line only the odd numbered samples of luma are accompanied by Cr' and Cb' samples. If vertical subsampling is also applied, then Cr' and Cb' samples will only be present once for each square formed by four luma samples in consecutive lines - 4:2:0. Other sampling schemes exist, for example 4:1:1 used in DV video format where Cr' and Cb' samples don't appear together like in 4:2:0 but each in alternate horizontal luma samples with chroma presence. Nevertheless, this project is only meant to support 4:4:4 and 4:2:2 chroma subsampling schemes.

Design Parameters	Slices	Speed (MHz)
Proposed SA architecture (1)	305	68
Proposed SA architecture (2)	1022	72
Proposed DA architecture (1)	70	128
Proposed DA architecture (2)	193	234
CAST.Inc [4]	222	112
ALMA. Tech [5]	222	105
Amphion Ltd [3]	204	90

Figure 3.5: Comparison of results (from [5]).

Source	Max. Frequency	Observations
Xilinx [23] [4]	226 to 234 MHz	Virtex 7 and Zynq-7000.
Altera [22]	148.5 or 100 MHz	Arria V or Cyclone V, respectively.
Lattice Semiconductor [24]	196 to 262 MHz	Lattice ECP3 LFE3-150EA-6FN1156C.
iWave [25]	78 MHz	ProASIC3
"Fast method" [26]	358.2 MHz	Virtex 4 XC4VLX15-10
"Hand-optimized (full pipeline)" [27]	229 MHz	Xilinx FPGA
"Distributed Arithmetic" [5]	234 MHz	Virtex E XCV50E-8

Table 3.1: Comparison on the performances announced in the proposals found

Chroma subsampled schemes can be achieved from 4:4:4 by discarding chroma samples where necessary. In signal processing this operation is known as decimation. Although simple discarding of samples (nearest-neighbor method) can be used by simpler systems, higher quality results require the use of low-pass filters to prevent the apparition of alias and other artifacts in the decimated signal.

The filters used in the decimation should not only block the undesired frequencies of the signal but should also account to the sampling positions imposed by the reference standards. That is, although the Cr' and Cb' samples are transmitted aligned with the luma samples, their value may not correspond to that sampling point of the image, for example, they can represent a sampling point between two luma samples. When luma and chroma sampling points are the same they are said to be co-sited.

The upsampling of the chroma signals to restore 4:4:4 or 4:2:2 is achieved by interpolation of the subsampled values. The simplest form of interpolation is repeating the previous chroma sample - nearest-neighbor method. Better upsampling results can be obtained if a weighted sum of neighboring samples is used to compute the new sample.

ITU-R BT.601 [8] and ITU-R BT.709 [7] suggest low-pass filters for $R'G'B'$, Y' and $Cr'Cb'$

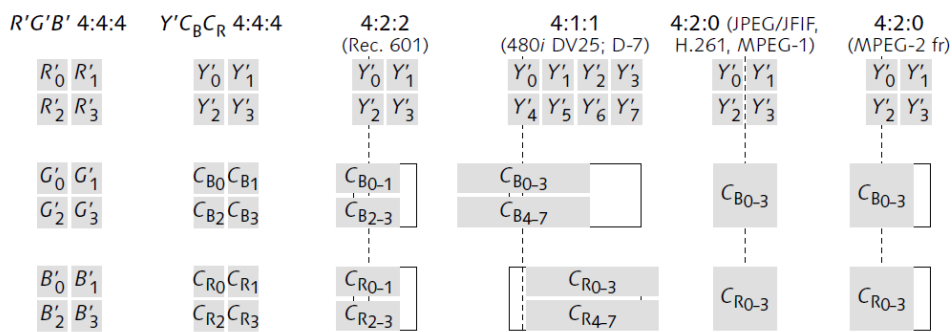


Figure 3.6: Chroma subsampling schemes (from [6]). Format 4:2:2 from Rec. 601 is co-sited while 4:2:0 from JPEG is sited interstitially.

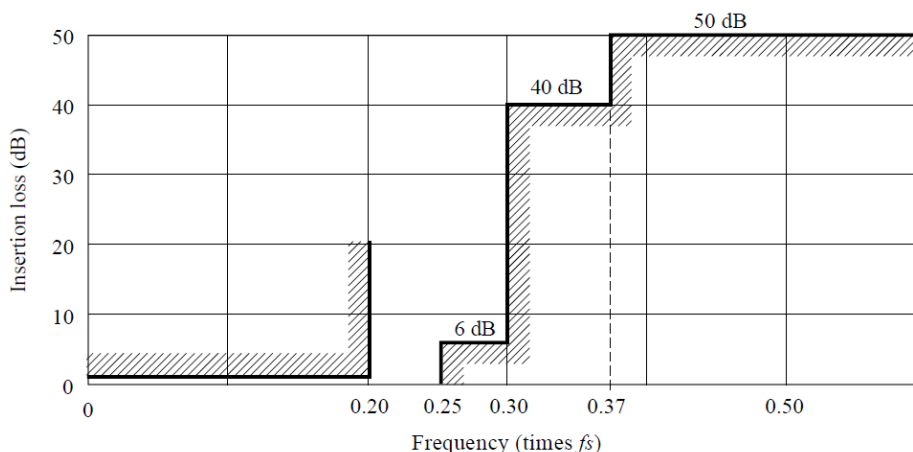


Figure 3.7: Template from Rec.709 for Cr and Cb filtering (from [7]). f_s is the sampling frequency of the R,G,B and Y signals.

signals for filtering in analog-digital and digital-analog conversions that could be applied when resampling chroma - figures 3.8 and 3.7. However, this filter templates are very exigent and preliminary studies made using *Matlab* indicate they would require a heavy hardware cost.

A single-rate lowpass equiripple FIR filter compliant with Rec.601 and Rec.709 templates would need to be of order 27, which typically would require 27 state registers or taps, 28 multipliers and 27 adders to implement in a direct structure, or only 14 multiplications in a symmetric structure taking advantage of the symmetry of the coefficients.

Alternatively, a single-rate lowpass half-band FIR filter topology would require a filter of order 30, but because almost half of the coefficients of an half-band filter are zero, only 30 state registers, 17 multipliers and 16 adders are needed to implement a direct FIR structure.

The actual implementation structure and hardware cost details for the filters designed are presented in section 5.5.

An analysis of the available solutions for chroma resampling systems in the market and in scientific documentation has been made and the collected proposals are presented next.

In [16] Poynton presents simple averaging filters (2 to 3 taps) which main concern is to respect the chroma sampling positions defined in the different standards. In other note, [28], an alternative filter to the templates defined in Rec.601 with 13 taps is suggested, which Poynton considers to be a good compromise between performance and complexity while being less demanding than the Rec.601 templates.

Finally, in [29] concerning chroma interpolation Poynton claims that for VHS video quality nearest-neighbor interpolation is sufficient and that for better results linear interpolation could be used. For higher quality a multi-tap FIR filter is required and an example of a suitable filter is presented. These same filters are cited in a Xilinx application note [30] where a 24-tap filter for 4:2:2 to 4:4:4 interpolation respecting Rec.601 templates is also proposed.

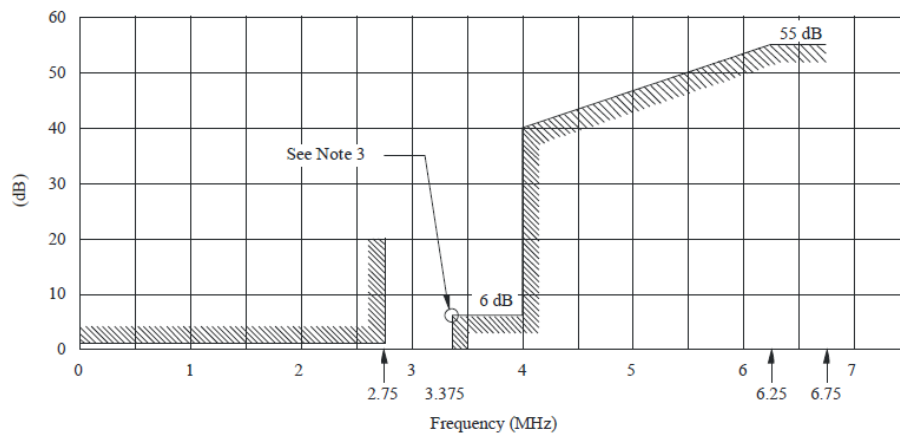


Figure 3.8: Specification from Rec.601 for a digital filter for sampling-rate conversion from 4:4:4 to 4:2:2 color-difference signals (from [8]). The frequencies indicated are for the 13.5 MHz family, with sampling frequencies of 13.5 MHz and 6.75 MHz for the luminance and chroma-difference signals, respectively. The same normalized template applies for the 18 MHz family.

Keith Jack [31] also refers that Rec. 601 filter templates are very complex to apply in interpolation from 4:2:2 to 4:4:4 and consequently most NTSC and PAL video decoders implement linear interpolation.

In Akrammullah [32] one horizontal low-pass filter is proposed to downsample from 4:4:4 to 4:2:2 and a vertical low-pass filter to further downsample to 4:2:0 encoding, both with 11-taps.

A research has been conducted by the R&D department of BBC in order to investigate re-sampling filters that would allow to convert from 4:2:2 to 4:4:4 or 4:2:0 and then back to 4:2:2 while minimizing conversion losses. These filters are meant to apply in situations where sections of the broadcasting chain are not compatible with the content chroma format (usually 4:2:2) and so chroma resampling is needed before and after these sections. The results from this research are published in a white paper [33] where one linear and one nonlinear filter are proposed for this problem. Both these filters are reversible and achieve lossless results in the end of the two resampling operations. The linear filter proposed is a 5-tap FIR filter with increased precision relative to content bit depth that could be interesting for this thesis problem. The nonlinear filter consists in a nearest-neighbor interpolation reversible by a weighted-sum average filter which doesn't require increased precision.

A comparison between different chroma subsampling schemes is done by Glenn Chan [34] considering the usual visual problems caused by downsampling: blurry images due to imperfect frequency response, spurious image detail caused by aliasing and the apparition of ringing artifacts around high-contrast edges. From the comparison of combinations of linear, box, multi-tap FIR and nearest-neighbor downsampling methods with linear and box upsampling methods the author claims that the best subjective results are obtained with multi-tap FIR and linear/tent filters, although it concedes that chroma subsampling artifacts are rarely noticed with any method in the majority of situations.

An analysis of the out-of-gamut output and the non-constant luminance problems while converting from Y'Cr'Cb' to R'G'B' is also presented and possible solutions addressed. However, the standards considered in this thesis already define the expected responses in this situations.

Keith Jack [35] emphasizes the importance of keeping the filters passband as flat as possible in exchange for reducing the stopband cutoff rate when trying to optimize the filters design and implementation.

In Bartkowiak [36] an improved interpolation method for 4:2:0 to 4:4:4 conversion based on the correlation between Y' and Cr'Cb' signals is proposed. Despite this proposal being out of the scope of this thesis the comparison of this method's results is made with a 7-tap low-pass FIR filter which is of interest to this work.

Xilinx LogiCore IP products offer a Chroma Resampler core described in [37]. This core supports chroma resampling between 4:4:4, 4:2:2 and 4:2:0 formats for progressive and interlaced video with bit depths of 8, 10 and 12-bits per component.

Simple nearest-neighbor decimation and interpolation (by dropping or replicating samples) are possible but FIR filters are also provided for all the conversions in two different implementations: predefined fixed power-of-two coefficients to simplify multiplications to shifts and additions and a implementation with programmable coefficients and number of taps.

The default filters are 2 and 3-taps polyphase implementations and computations are done with full precision by extending input and coefficients bit widths in intermediate arithmetic. Coefficients are 16-bit with 1 sign bit, 1 integer part bit and 14 fractional bits.

Altera also offers a Chroma Resampler core in its Video and Image Processing Suite [22] which supports resampling between 4:4:4, 4:2:2 and 4:2:0 by nearest-neighboring or filtering (only for horizontal resampling). FIR filters are implemented with fixed power-of-two coefficients with 4-tap or 9-tap (upsampling and downsampling, respectively) based on Lanczos-2 function and its quantized form known as the Turkowski Decimator. The Lanczos-2 function is a two-lobed windowed sinc function.

Turkowski [38] compares a series of different filters and their quantized versions for decimation and interpolation of image data and conclude that Lanczos functions were one of the best compromises in reducing aliasing, sharpness and ringing, as well as having one of the best passband and cut off frequency responses.

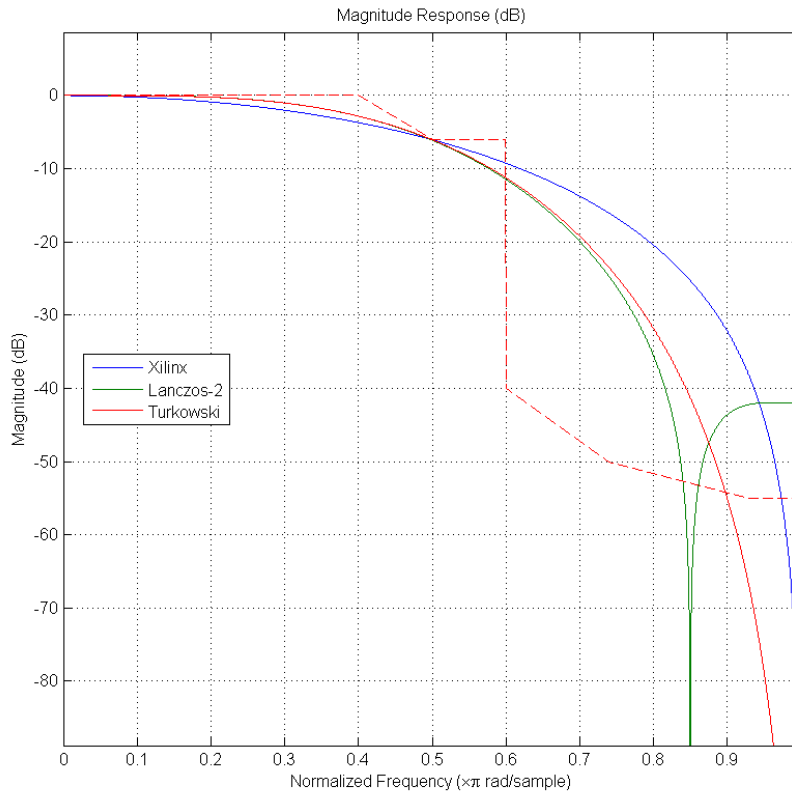
3.3 Summary

The comparison with the IPs available in the industry [4][22][23][24][25] allowed to evaluate the list of features proposed for this module. The proposals of color space converters studied only consider RGB-YCC conversions. In this work, we aim to support not only these conversions but also conversions between different RGB color spaces.

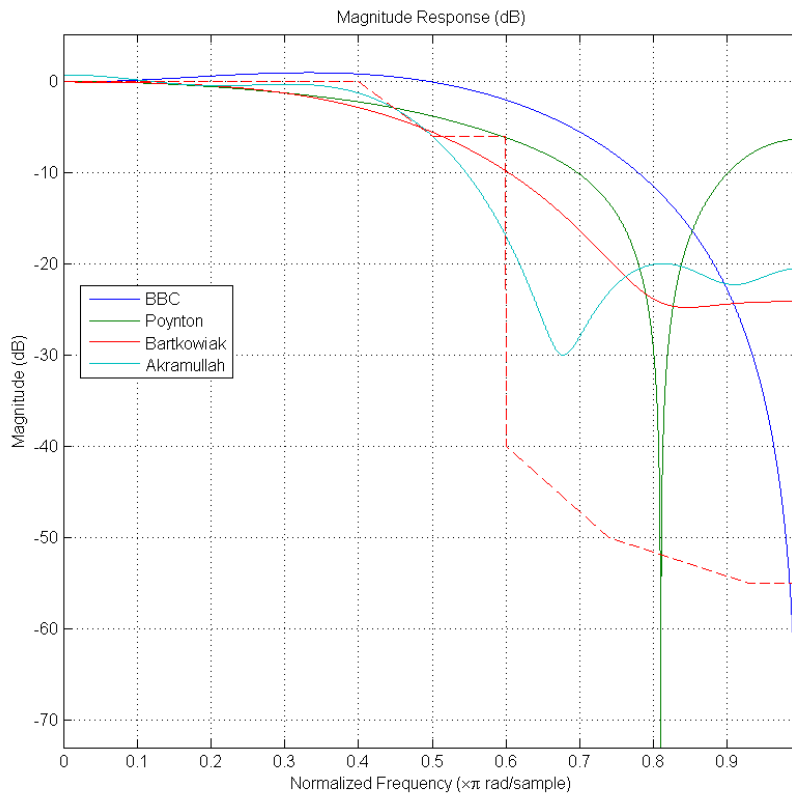
The architecture that has been developed in this thesis is a direct implementation of the algorithmic structures required by the conversion process, optimized at the performance level by dividing it into pipeline stages. From the study of the state of the art of color space converters, two

interesting architectural alternatives using distributed arithmetic[5][26] and systolic structures[5] have been presented. These alternatives claim to improve performance and area metrics when compared to direct implementations. However, the results have been obtained from implementations in FPGAs so we cannot extrapolate the metrics obtained to the 40nm technology.

The responses of the chroma resampling filters proposed in the references are compared in figure 3.9 against the templates defined in Rec.709 and 601. Most of the proposed filters in section 3.2 don't respect the templates defined in Rec. 601 and Rec. 709. However, the majority of these filters have a common 6 dB attenuation at half the sampling frequency, which is a characteristic that was considered in the design of the smaller chroma resampling filters (section 5.5), together with the advise from [35] to keep the pass-band as flat as possible when optimizing a filter's architecture, lowering the expectations for the stop-band's attenuation instead.



(a) Xilinx [37], Lanczos-2 [22], Turkowski [38].



(b) BBC [33], Poynton [28], Bartkowiak [36], Akramullah [32].

Figure 3.9: Comparison of the filters proposed in the references against a template (dashed line) that is compliant with both Rec.601 (figure 3.8) and Rec.701 templates (figure 3.7).

Chapter 4

Architecture Design

4.1 Introduction

The main design goal of this project is to develop a module that is synthesizable for 600 MHz frequencies at 40 nm technologies. This performance goal is driven from the specification of 4K video formats in the Consumer Electronics Association (CEA) standard of digital television profiles for uncompressed high speed digital interfaces [39].

These 4K progressive formats have active video resolutions of 3840x2160 and 4096x2160 at both 50 Hz and 60 Hz frame rates. Considering the full frame sizes (with vertical and horizontal blanking) of 5280x2250 and 4400x2250 it results in 594 million pixels per second to be transmitted, which requires a pixel clock of 594 MHz. Considering a parallel interface, where the three video channels are transmitted in parallel, these formats originate a video data rate of 28,51 Gbps for RGB or YCrCb 4:4:4 16-bits video.

This module implements a parallel video interface with the three video channels and the three synchronization signals in parallel due to compatibility reasons with other modules, as this is the most direct interface to be implemented.

The architecture design for this module is heavily driven by the color space conversion algorithm (figure 2.4) necessary to support the conversions between the proposed color spaces. This algorithm has been divided in indivisible steps, corresponding to the operations that may need to be executed or bypassed for each conversion configuration.

4.2 Top Level Interface

The top level interface (figure 4.1) of the developed module requires the support of the video interface input and output signals: video data, hsync and vsync. The video data signals are concatenated in a 48 bits bus with the most significant 16 bits corresponding to the channel 1 (representing R or Y), then channel 2 (G or Cr) and finally the least significant 16 bits correspond to channel 3 (B or Cb signals). A register bank was implemented in the module for configuration of its functioning mode, so an interface to access the registers is offered, composed of the following signals:

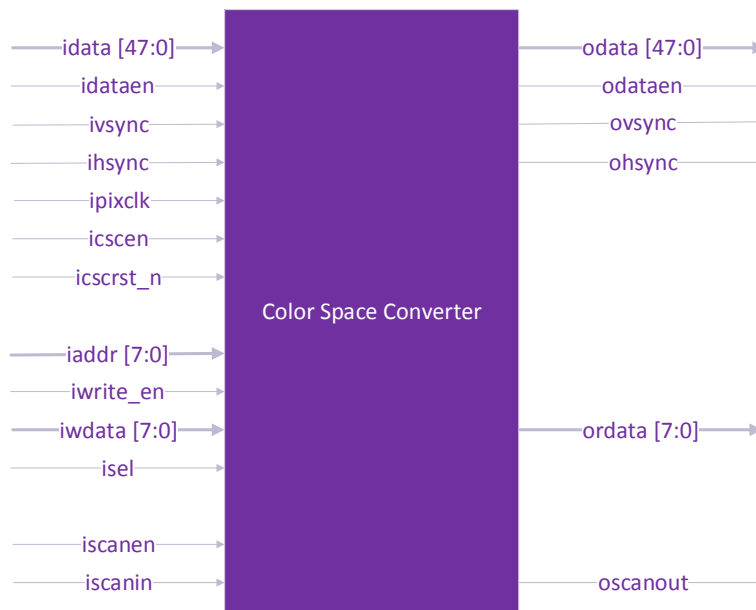


Figure 4.1: Top level interface

address, write enable, write data, read data and select signal. Finally, the top level has the input ports for the module's enable, reset and pixel clock signals. The top level interface signals are further described in table 4.1.

4.3 Implementation Strategy

The project was divided in different sub-modules corresponding to the tasks performed in the color space conversion - figure 4.2. This hierarchical division not only allows for a divide-and-conquer problem solving approach as it allows each module to be tested and validated independently, which facilitates the verification task. Furthermore, because depending on the running configuration not all tasks are needed to the required conversion process, some modules can be simply set to bypass mode.

The color space conversion module was divided in 9 sub-modules: a register bank for configuration purposes, a control unit module which controls the datapath configuration based on the register bank data, two chroma horizontal resampling filters, one RGB to RGB color space conversion, one gamma encoder and a gamma decoder, one RGB to YCC and an YCC to RGB converters which also perform the limited/full range data conversion.

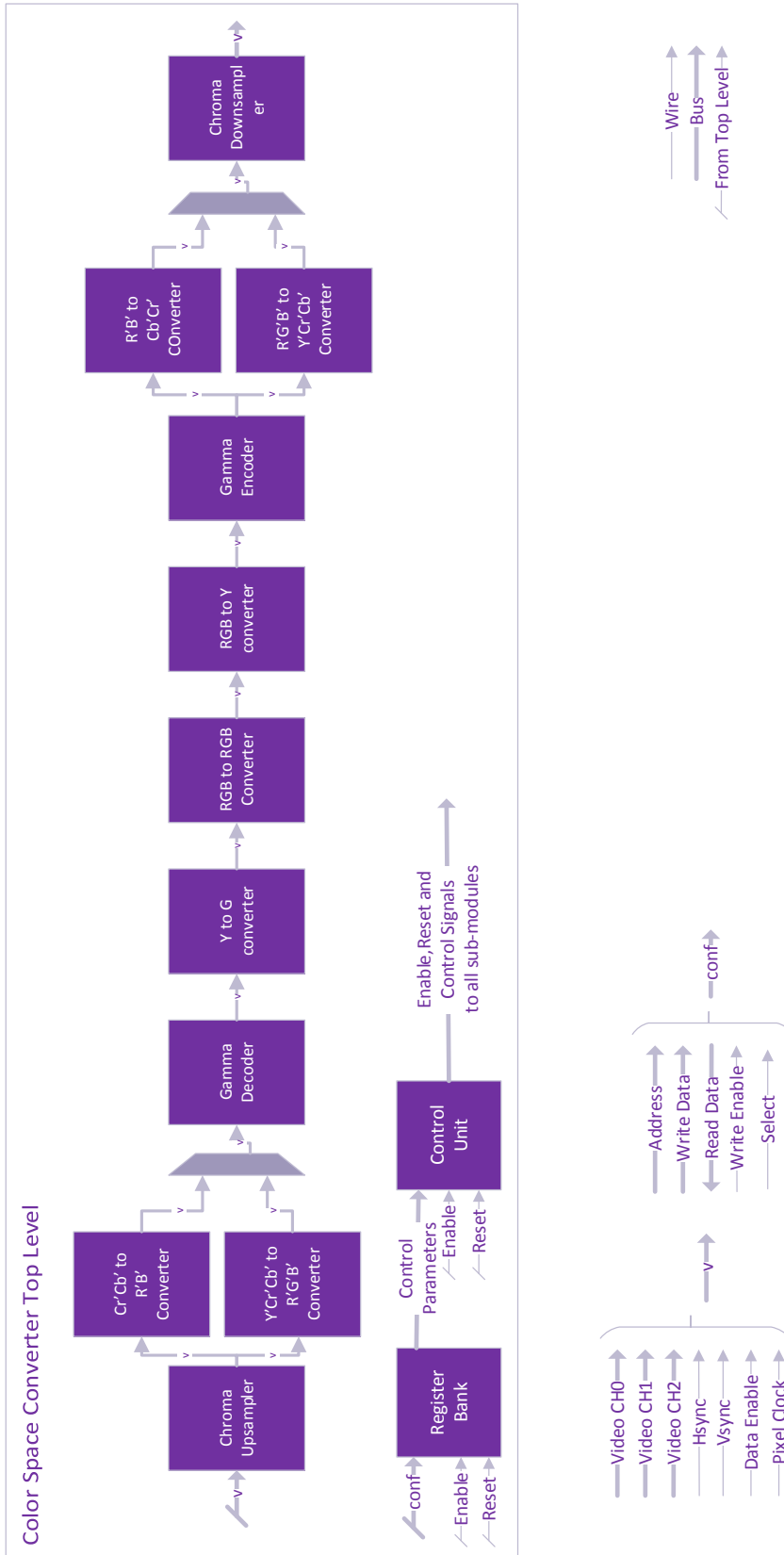


Figure 4.2: System architecture. Table 4.1 relates the top-level signal's naming of figure 4.1 with the higher level description of this figure.

Signal	Bit Width	Description
idata	48	Input video data, contains the three 16 bits video channels: video ch0, ch1 and ch3.
idataen	1	Input data enable signal. High when idata contains active video data.
ivsync	1	Input vertical synchronizations signal.
ihsync	1	Input horizontal synchronizations signal.
ipixclk	1	Input pixel clock.
icscen	1	Input enable signal. Active high.
icsrst_n	1	Input asynchronous reset signal. Active low.
iaddr	8	Input register bank access address.
iwrite_en	1	Input write enable for register bank access.
iwdata	8	Input write data for register bank write operation.
isel	1	Input select signal for register bank access.
iscanen	1	Input scan enable for DfT. ¹
iscanin	1	Input scan chain port.
odata	48	Output video data, contains the three 16 bits video channels: video ch0, ch1 and ch3.
odataen	1	Output data enable signal. High when idata contains active video data.
ovsync	1	Output vertical synchronizations signal.
ohsync	1	Output horizontal synchronizations signal.
owdata	8	Output read data for register bank read operation.
oscanin	1	Output scan chain port.

Table 4.1: Top level interface signal description.

The implementation process of each module started with the evaluation of the requirements imposed by the standards and the color space conversion process over each task. After this, a software model in Matlab or C was implemented using floating point arithmetic. These models were then converted to fixed point arithmetic and their response compared to the floating point references, which allowed to make some design decisions as the bit width of the coefficients and computations. Finally, the RTL module was described in Verilog and verified against the fixed point software model.

The strategy followed in the synthesis process was to synthesize each sub-module individually in order to facilitate the iterative process over different optimizations. For this task, each module was implemented with registered inputs and outputs and the synthesis constraints were applied with the objective of providing a good working margin for a future physical implementation process. The synthesis constraints set are described in the chapter 5.9.

¹These scan ports were added to the module to perform the synthesis with scan chain insertion, because the insertion of the scan chain affects the flip-flop cells used, which affects the performance obtained. This project isn't meant to consider design for testability strategies and implementation.

4.4 RGB to RGB Converter

This module implements the matrix operation presented in equation 3.9 for a particular example. Each standard defines its color primaries points from which it is possible to extract the coefficients to convert from and to each color space and the CIE XYZ color space. By multiplying the coefficients from the conversion RGB-XYZ of the input color space and the XYZ-RGB coefficients of the output color space, one can obtain the coefficients for the RGB-RGB conversion considered. The number of color spaces this project is required to support would require 36 matrices to be computed and somehow made available for conversion. However, due to the overlap between the gamuts of different color spaces one can eliminate the redundancy between the matrices and consider only 14 different sets of coefficients.

The process of obtaining the matrix coefficients from the standard's primaries is presented next, as described in [16]. The chromaticities of the RGB primaries x , y and z are directly obtained from each standard, and can be represented in a matrix:

$$C = \begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{bmatrix} \quad (4.1)$$

Usually the values for the z_i chromaticities are omitted, but can be driven from $z_i = 1 - x_i - y_i$. The RGB to RGB conversion matrix R is obtained from

$$R = C \times \begin{bmatrix} J_r & 0 & 0 \\ 0 & J_g & 0 \\ 0 & 0 & J_b \end{bmatrix} \quad (4.2)$$

where J is computed with the chromaticities matrix C and the coordinates w_i of the white point w used by the standard concerned:

$$\begin{bmatrix} J_r \\ J_g \\ J_b \end{bmatrix} = C^{-1} \times \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \times \frac{1}{w_y} \quad (4.3)$$

There is a set of standard white-points defined by the CIE, which the majority of standards supported in this thesis use the D65, with coordinates $W_{D65} = [0.3127, 0.3290, 0.3583]$.

This process is validated against the coefficients presented in examples described in some standards. However, there are some differences between values for the same coefficients presented in different standards. In these cases, the values from the most recent standard (by date of publication) has been used, which are coherent with the ones obtained by the mathematical derivation.

The RGB to RGB conversion algorithm has been implemented in C using both floating point arithmetic and fixed-point arithmetic. The coefficients have been quantized by multiplying them to $2^{N_{bits}}$ and rounding. In the standard IEC 61966-2-1-Am1 [12] it is indicated that for 16 bits video the coefficients should have 7 decimal values (6 bits for RGB to YCrCb conversion), which

could be represented with full precision by 24 bits (20 bits for RGB to YCrCb conversion), so the floating point values to be quantized are computed respecting this indication.

A comparison was made between the SNR and probability of error obtained from using different number of bits for storing the coefficients for each video data input width (8 to 16 bits), for random input pixel values. This comparison was made by taking as reference the results obtained with floating point arithmetic, quantized with the same method used with the coefficients. The results obtained for 16-bit video and coefficients width varying between 16 and 30-bits are presented in table 4.2. The results for 8 to 16 bit data widths are plotted in figures 4.3 and 4.4. The 8 extra bits when comparing 8-bit to 16-bit video can be understood as increased precision, so it was decided to implement all tasks in the color conversion process for 16 bit video, which can be reduced to a smaller width at the output (and expanded in the input) without additional loss of precision.

These results, together with the ones obtained for the RGB-YCC and YCC-RGB conversions presented in section 5.3, show that little advantage is obtained from using more than 24-bit for the coefficients, as the SNR values tend to stabilize and seem to vary based on the variability of the input vectors instead of added precision. This behavior is expected and coherent with the use of 7 decimal values floating point coefficients, which require 24 bits binary representation for full precision. It was decided to implement the modules with 24-bit coefficients and to consider also 24-bit video for reducing the quantization error accumulated in each operation. This decision could be reverted in a later stage if necessary for the achievement of the performance goals.

Coefficients Width	SNR (dB)	Probability of Error (%)
16 bits	104.2	11.568
18 bits	110.8	2.5048
20 bits	116.3	0.7208
22 bits	121.3	0.2305
24 bits	124.5	0.1135
26 bits	124.8	0.1045
28 bits	124.7	0.1083
30 bits	124.9	0.1015

Table 4.2: Comparison of the variation of the RGB to RGB conversion coefficients width and SNR and probability of error measured for 16 bit width video.

4.5 R'G'B'-Y'Cr'Cb' Converters

These modules implement not only the conversion between R'G'B' and Y'Cr'Cb' color spaces but they also perform the scaling and offsetting of the video data values - exemplified in equations 3.6, 3.7, 3.11, 3.14 and 3.17. Despite the scaling operations, this is the same mathematical operation as the RGB to RGB converter.

The coefficients for R'G'B'-Y'Cr'Cb conversion are directly defined in the respective standards. Based on the analysis made for the RGB to RGB module and the similarities between these

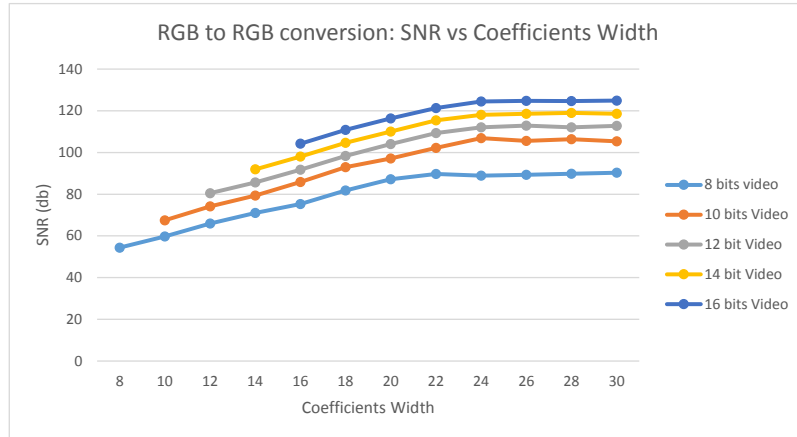


Figure 4.3: Comparison of the SNR obtained for the RGB to RGB conversion for different coefficients widths for each video bit width.

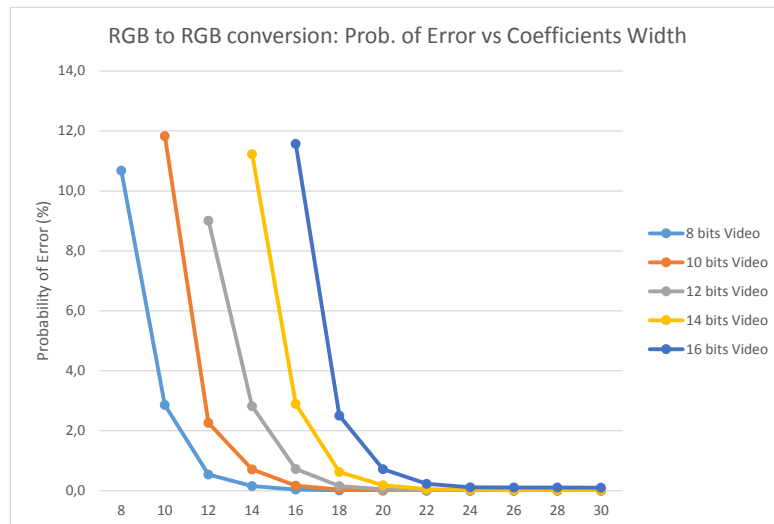


Figure 4.4: Comparison of the probability of error obtained for the RGB to RGB conversion for different coefficients widths for each video bit width.

modules arithmetic operations, the coefficients used by this module were also quantized after its floating point 7 decimal values representation by multiplying them to 2^{24} and rounding, despite some of the standards define the coefficients with lower precision.

The first step in the Y'Cr'Cb' to R'G'B' module is to scale the signals from the input range, if it is a limited range, to full range and adjust their offsets accordingly. The natural range of the chroma signals Cr and Cb is [-0.5;+0.5] which requires these signals to be treated as signed.

After this, we can perform the conversion to the R'G'B' color space using the coefficients matrix defined in the concerned color space standard. The Y'Cr'Cb' color space has a wider gamut than its equivalent RGB color space which may result in out of gamut values after this conversion so the RGB values must be limited to the interval $[0;2^{24} - 1]$.

The R'G'B' to Y'Cr'Cb' module implements the inverse operation, first converting from R'G'B' to Y'Cr'Cb' using the matrix multiplication and then scaling the signals to the output ranges, limiting them to the range limits.

Besides the non-constant luminance R'G'B'-Y'Cr'Cb' conversion modules shared between all color spaces conversions, the constant luminance conversion defined by Rec.2020 requires the use of different modules, because of the algorithmic differences.

In R'G'B' to Y'Cr'Cb' constant-luminance conversion, the linear Y signal is obtained from the linear RGB signals, and only then gamma encoded into Y'. The Cr' and Cb' values are computed from the gamma encoded Y', R' and B' signals.

Although the computation of Y is a weighted sum of the RGB channels, the coefficients used in the scaling of Y'-R' and Y'-B' that originate Cr' and Cb' respectively depend if these differences have positive or negative values.

In the Y'Cr'Cb' to R'G'B' constant-luminance conversion the inverse algorithm is applied.

Due to the significant differences between non-constant luminance and constant-luminance conversions these were implemented in independent modules in the datapath - figure 4.2. When converting in Rec.2020 constant-luminance mode the data flows through the Cr'Cb'-R'B' converters and Y-G converters. Otherwise, it flows through Y'Cr'Cb'-R'G'B' converters and the Y-G are set to bypass mode.

4.6 Gamma Encoder and Decoder

The gamma function is a non-linear function described by an exponentiation of the input value. Moreover, the gamma functions defined in some of the concerned standards are defined in branches, where in the section near zero the function is linearized, due to the high derivative of the exponential function in that section - see equations 3.8 and 3.10.

It isn't possible to implement directly a non-linear function in hardware, because only linear arithmetic operators are available (adders, multipliers, dividers). Therefore, some strategy must be set.

One usual approach to this situations is to implement a lookup table which relates a set of input values with output values. This method allows us to approximate the expected output value

with a precision that grows with the size of the lookup table implemented. It is also possible to interpolate the output value from the nearest table values, instead of simple substitution. There are other possibilities of approximation of exponential functions, as the Newton-Raphson algorithm. However, these algorithms are recursive, which imposes some difficulties in their implementation due to the real time nature of this module. It was chosen to develop a lookup table based solution, depending on the results obtained by this approach.

The specifications for this module require the implementation of 3 pairs of gamma encoding and decoding functions. These functions were coded in Matlab with the objective of linearize them by segments. The strategy followed was to divide the function in segments and in each segment the approximation would be obtained by the computation of a first order linear function $y = mx + b$, where the m and b constants for each segment would be stored in a LUT.

Because the highest the precision the larger the LUTs, it was decided to keep the data width as 16 bits, to constraint the number of segments required, and a allowable maximum error of 1 LSB.

Several attempts of linearization of the 6 functions with this requisites were made: growing number of segments with the same length, varying the length of each segment across the input range to better fit the derivatives of the functions, dynamically defining the length of each segment based on the output error generated. None of these strategies conducted to a solution which fulfilled the error constraint without an excessively large number of segments. At some point, the number of segments required started to compare to the size of the LUTs required to relate every possible input, that is, $2^{16} = 65536$ entries. The errors didn't appear in a specific pattern which could indicate that some section of a function just required a little extra precision, as they were scattered around the full range, and after a certain level of reducing the probability of error, further advances required an exponential increase of the effort.

The analysis of the distribution probability of the errors suggested an alternative solution which was developed using segments of constant length. The dynamic range of the function was divided in segments with lengths power of 2, which allows a vector of the most significant bits of the input to be the LUTs address. On top of this, an additional third LUT (besides the gain m and offset b LUTs) registers the pre-computed outputs for a set of vectors for which the method doesn't offer the 1 LSB maximum of error. A comparison was made between the total number of LUT entries that would be required for different segments lengths, and the minimizing solution was chosen - table 4.3.

The increase of the segment length reduces the number of segments needed, which reduces the number of entries in the linear LUTs that store the gain and offset values. However, the increase of the distance between the interpolation points increases the number of errors and therefore the size of the additional LUT. The total number of entries required for the overall of the encoding and decoding modules is computed by adding six times the number of linear LUTs entries (which includes the entries for the twin LUTs that store the gain and offset values) plus the number of entries in the error LUTs (the value includes the number of entries for the 6 functions, which is variable by function).

The encoding and decoding functions were linearized in segments of 64 bits for 16-bit input/output values, which means that the most significant 10 bits of the input are the address to the LUTs where the gain m and offset b of that segment are stored. The additional LUT for special values receives the full 16-bit input and if there is a value stored for the current input, then the output of this LUT is preferred to the output of the segmented linearization. This strategy allows us to use only 4.92% of the LUT entries compared with mapping the 2^{16} values for the 6 functions.

Segment length	Total Entries	Linear LUT entries	Error LUT entries	% Full mapping entries
8	99 273	16 384	969	25.25
16	50 873	8 192	1 721	12.94
32	27 829	4 096	3 253	7.08
64	19 316	2 048	7 028	4.92
128	25 368	1 024	19 224	6.45
256	84 888	512	81 816	21.59
512	226 782	256	225 246	57.68

Table 4.3: Comparison on the length of the segmentation segments and the size of the LUTs required.

4.7 Chroma Resampling Filters

In section 3.2 we have discussed some existing alternatives to perform the 4:4:4 to 4:2:2 conversion. Because in this project the compliance with the industry standards and the assurance of the highest quality of results is a necessity, it was decided to implement resampling filters according to the indicated in the standards. From the filter response guidelines presented in Rec. 601 for 4:4:4 to 4:2:2 filtering and in Rec. 709 for Cb and Cr signals a filter specification that respects both these indications was elaborated.

A Finite Impulse Response (FIR) was the choice for this application, because of its linear phase response and simple polyphase decomposition. Moreover, FIR filters are less sensitive to quantization effects due to the lack of feedback, which in IIR filters can originate increased quantization effects.

Using the Matlab Filter Design Tool it was possible to obtain a 30th order lowpass half-band filter with these characteristics, computed using the Parks-McClellan algorithm. In a half-band filter, almost all odd numbered coefficients are zero, except for one which is 0.5, which allows efficient hardware implementations. The coefficients obtained from the tool were quantized to 24 bits integer coefficients by multiplying them by 2^{24} and rounding.

The chroma upsampling filter was implemented using a polyphase decomposition of the original filter. Because this filter is an interpolator of ratio 2, the filter has been decomposed in two phases. Each of the phases is implemented in a direct FIR structure, which requires a total of 17 multipliers (16 for phase 0 and 1 for phase 1), 15 adders and 15 registers - structure exemplified in figure 4.5 for a generic smaller order filter. Obviously, although the figure 4.5 presents all the

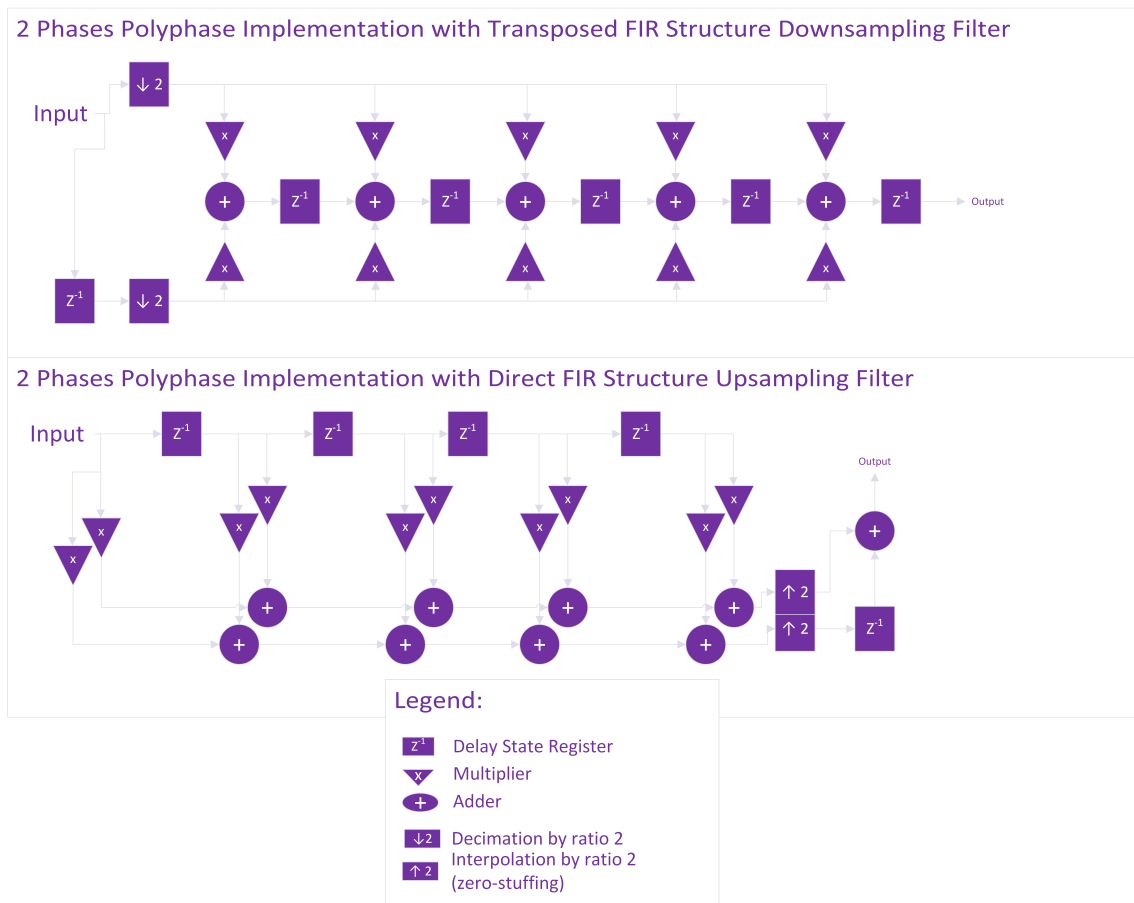


Figure 4.5: Architectures of the upsampling and downsampling filters.

multipliers for better comprehension of the polyphase structure, the multiplications by zero coefficients are discarded. Taking advantage of the noble identities [40], the polyphase implementation allows the upsampling to be done after filtering each phase, instead of before filtering to prevent aliasing, as usual. At the end of the adders chain, each of the outputs is interpolated with zero-values and the phase 1 is delayed one cycle. The output of the filter is obtained in alternating the outputs from the phase 0 and the delayed phase 1.

The advantage of the polyphase decomposition by ratio of 2 is that it allows each of the phases to be working at half the cadence of the original signal, i.e. for each input value (or pair of output values) each phase will only compute a single value, which reduces the timing constraints for the adders and multipliers logic by half. Otherwise, if using a normal filter, we would have to be computing samples at twice the input's frequency.

The downsampling filter was designed using the same method described above for the upsampling filter, but it was implemented in a polyphase transposed FIR filter structure - exemplified in figure 4.5. The transposed structure in a decimator reduces the number of state registers required, compared to a direct structure implementation. In this case, the decimation is performed before the filtering in each phase, taking advantage once more of the noble identities [40]. This way, each phase filtering is also computed at half the rate of the input signal.

The phase 0 of each of this filters has a symmetric response, and in the downsampling filter all the multiplications are computed using the same input value - figure 4.5. This allows to implement a symmetric structure in the phase 0 that further reduces the number of phase 0 multipliers in half.

Besides having a filter with the wanted frequency response, it is necessary to control its behavior in the borders of the image, when there are no available inputs to fill the filter taps. In other image processing applications, it is usual to extend the image borders to accommodate this need. This extension can be made by padding the image with zero-valued pixels, replicating the pixel on the border or mirroring the image by the margin. There are also the possibilities of adapting the filter's order near the margin or crop the output image to consider only pixels computed with the available data.

Padding the image with zero-valued pixels is the most simple approach. However, it usually results in a output image with black margins, due to the injected zero-values. Replicating the border pixels is a good compromise between the quality of the outputs (the border pixel values are over-weighted in the margins of the output image) without adding too much complexity to the hardware implementation, as the other mentioned strategies.

The impulse response $y[n]$ of these linear FIR filters is non-causal, that is, it depends on the values of past and future pixels relatively to the current filter ($n < 0$ and $n > 0$), and its response is symmetrical in respect to the instant $n = 0$, which results in output values also at the right and left of the current pixel (past and future pixels). Therefore, it is necessary to set the values of the filter registers at the borders of the image, padding the image by replication, and adjust the latency of its response so that only the half of the response inside of the image is propagated during the active video (the filter becomes causal). During the blanking periods (data enable is not active) the data channels are required to stay constant.

Additionally, when in the left-right 3D mode, the two fields are sent in the same frame side-by-side so at the edge between the two images it is necessary to perform the same padding process. The implementation of this feature required the duplication of the filter logic to avoid interference from one image to the other, and the output is multiplexed from the filter correspondent to the current image.

These FIR filters are the largest sub-modules of this project. Because of this, it was implemented two smaller filters of order 18, that aim to be a compromise between the quality of results and the hardware implementation cost. The compromise on the filter response was achieved by relaxing the stop-band response and the pass-band response, although maintaining the pass-band as flat as possible, as indicated in [35]. Both the half-band filters have an approximately -6 dB attenuation at the stop frequency, which is a requirement from the templates and a common characteristic of the alternative proposals presented in 3.2.

The comparison on the hardware requirements of the 30th and 18th orders filters considering the already mentioned optimizations is presented in table 4.4, and their magnitude response is compared in figures 4.6 and 4.7 against the template used, based on the requisites of both Rec.601 and Rec.709 filter templates.

Filter	Upsampling		Downsampling	
	30 taps	18 taps	30 taps	18 taps
Adders	15	9	15	9
Multipliers	17	11	9	6
Registers	15	9	15	9

Table 4.4: Comparison of the hardware costs of 30th and 18th order filters.

4.8 Control Unit and Register Bank

This sub-modules work together in managing the control signals to each of the datapath sub-modules according to the configurations defined in the register bank. The control unit also forwards the enable and reset signals from the top-level inputs to the other sub-modules.

The register bank allows the configuration of the system to a desired functioning mode. To each register of the bank corresponds an output of its module, wired to the control unit.

The generic interface implemented for the register bank can be easily coupled to a reusable bridge that adapts the interface to one of the standard bus protocols used in the industry, for instance the Advanced Microcontroller Bus Architecture (AMBA) introduced by ARM in 1996.

The configuration codes were built upon the ones used in a ANSI/CEA standard [39] for AVI infoframes and other metadata packets used in video transmission to transmit information about the content transmitted. This way the integration of this core in a higher level video interface core can be facilitated.

The register bank provides the configuration of the following parameters:

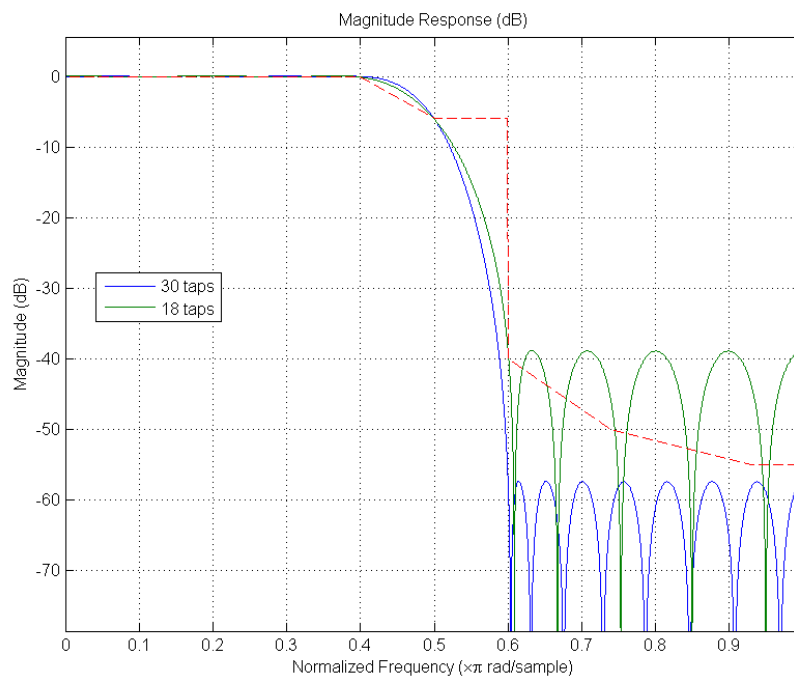


Figure 4.6: Magnitude frequency response of the 30th and 18th order filters, against the template (dotted line). At the edge of the stopband 0.5π rad/sample or $0.25f_s$ the 30 taps and 18 taps filter magnitude responses are approx. -6 dB, being compliant with the template requirement of at least -6 dB.

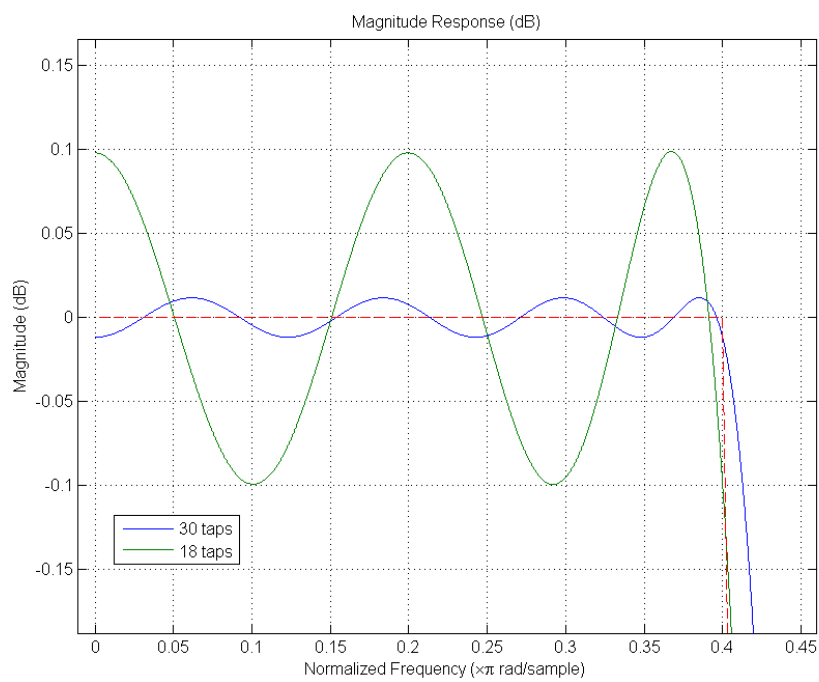


Figure 4.7: Detail of the magnitude frequency response of the 30th and 18th order filters in the pass-band. The standards templates require the passband (frequencies up to $0.2f_s$ or 0.4π rads/sample) attenuation to be $|A| \leq |0.05|dB$.

- `range_in` - defines the range of the input data: limited or full;
- `range_out` - defines the range of the output data: limited or full;
- `cspace_in` - defines the color space of the input data;
- `cspace_out` - defines the color space of the input data;
- `chroma_in` - defines the chroma format of the input data: RGB 4:4:4, YCrCb 4:4:4 or YCrCb 4:2:2;
- `chroma_out` - defines the chroma format of the output data: RGB 4:4:4, YCrCb 4:4:4 or YCrCb 4:2:2;
- `width_in` - defines the bit width of the input data: 8, 10, 12, 14 or 16 bits;
- `width_out` - defines the bit width of the output data: 8, 10, 12, 14 or 16 bits;
- `px_rep` - defines the number times an active video pixel is repeated : 0 to 9 repetitions (plus the original pixel).
- `half_hactive` - defines the horizontal width in pixels of the L/R fields in side-by-side 3D modes, or half the horizontal width of the active video. Configuration required even in 2D modes.
- `3D_structure` - defines the 3D structure if in 3D mode: side-by-side, top-bottom, etc.
- `3D_enable` - enables the 3D mode.

4.9 Verification Environment

The verification environment is presented in figure 4.8. The testbench module of the verification environment allows the user to configure the test scenario by setting a set of parameters. These configurations are directly loaded to the instantiated modules using the respective interfaces and protocols.

The reference model is constituted by C functions which implement the conversion operations. To each RTL module there is a C model to which it relates. The C functional models are encapsulated in SystemVerilog modules that replicate the architecture and latency from the RTL modules.

The fact that the reference model replicates the structure of the DUT, allows the output lines monitor to compare not only the outputs of the top levels, but also the signals between internal modules (grey box approach), which facilitates the debugging process and enables the detection of errors that could be masked by the following operations of the datapath. The video comparator contains a FIFO where it records a set of recent input stimulus applied, which it prints to a file together with a message indicating the running configuration when an error is found, for later

replication of the test scenario. This allows longer simulations to be performed, without the need to save the waveform databases. Saving the waveform databases not only slows down the simulation but the size of the database can also reach the dimension of several gigabytes, which becomes impracticable to save on disk and load into a waveform viewer like DVE. The results interface consists not only in this log printed to file but also in the logs printed to the terminal and error signals in the testbench for each internal and input/output net.

The video generator is a re-used module which generates video data that covers the input signals range, for any video format structure: frame sizes, 3D modes, pixel repetition, YCrCb modes. A simpler testbench without the video generator was also built to test with random and directed input values, although it doesn't aim to implement correctly the different video formats. This simpler testbench enables the creation of directed tests, for instance for testing the impulse response of the filters. Each sub-module was also tested in dedicated testbenches during the development phase.

The downsampling and upsampling filters impulse responses were manually compared with the responses computed of the floating-point filter models in Matlab.

4.10 Verification Plan

The verification plan was developed based on the requirements each operation in the color space conversion imposes. It contains verification tasks for each module of the RTL Color Space Converter individually, by configuring the other modules to bypass mode, and for the global system - table 4.5.

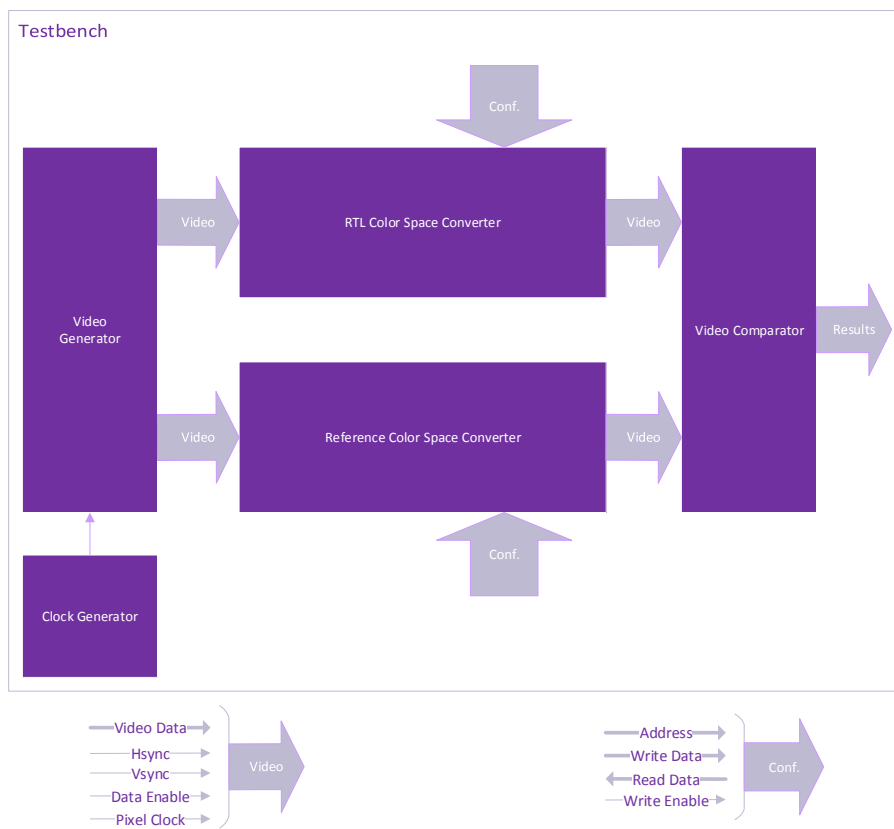


Figure 4.8: Verification Environment architecture.

Module	Task
RGB-RGB Converter	Correctness 36 types Overflow and underflow processing Sync propagation of hsync,vsync,data enable Bypass mode
RGB-YCrCb onverter	Correctness 14 types Overflow and underflow processing Sync propagation of hsync,vsync,data enable Limited to full range correctness Full to limited range correctness Bypass mode
Gamma Encoder	Correctness 3 encoding types Overflow and underflow processing Sync propagation of hsync,vsync,data enable Bypass mode
Gamma Decoder	Correctness 3 decoding types Overflow and underflow processing Sync propagation of hsync,vsync,data enable Bypass mode
Upsampling Filter	Horizontal interpolation Pixel Repetition correctness 3D side-by-side correctness Image borders processing Step/Impulse response compared with Matlab Correct behavior in blanking periods Sync propagation of hsync,vsync,data enable Sync propagation of Y channel Bypass mode
Downsampling Filter	Horizontal decimation Pixel Repetition correctness 3D side-by-side correctness Image borders processing Step/Impulse response compared with Matlab Correct behavior in blanking periods Sync propagation of hsync,vsync,data enable Sync propagation of Y channel Bypass mode
Control Unit	Correct RGB-RGB types Correct RGB-YCC types Correct Gamma types Correct Upsampling types Correct Downsampling types
Register Bank	Read correctly from all registers Write correctly to all registers Correct valid flag bit behavior Correct write flag bit behavior

Table 4.5: Verification Plan

Chapter 5

Hardware Implementation and Results

5.1 Tools and Development Flow

The design of a digital integrated circuit usually follows a development flow that is common to the majority of projects. The first step in this flow, is the definition or analysis of the specification and requirements of the design. It is fundamental to have a precise specification of the design before developing the architecture or even the RTL, because otherwise during the development of the design, some of the features or design choices may be driven by the implementation and not the project's specification, which may originate a final product that doesn't correspond to the stakeholders¹ interests.

After the design specifications and requisites are well defined, the architecture of the system is designed. The system may be divided in smaller sub-modules (divide-and-conquer approach) and the top-level and internal interfaces are defined, as well as block level specification describing the functional specifications of each component. During this phase, a functional model of the project may be developed to help the definition of the architecture. This was the strategy followed in this work, with the C and Matlab models of the blocks being developed prior to the RTL implementation. Finally, the RTL (register-transfer level) description of the system can be written using a hardware description language as Verilog, SystemVerilog or VHDL, using any text editor software.

After the design is implemented, it is necessary to verify its logical and functional correctness. The verification is usually done by simulation, developing a testbench where the device under test (DUT) is built on and stimulated to verify if it behaves accordingly. This verification can be done by comparing the DUT's output with the outputs from a functional model of the system, which is the case in this work as explained in section 4.9. The simulation is executed using a HDL simulator, which in this work was used the Synopsys tool VCS and its waveform viewer DVE to debug the modules.

¹The stakeholders of a project are all the parts interested in the project. Not only the project's clients (internal or external to the company), but also the engineers that develop the project, managers, company's owners and investors, suppliers, creditors, etc.

The verification of the design is a task usually done by a verification team, independent from the implementation team, which allows the functional models and the RTL models to be implemented on different interpretations of the same specification, which helps on identifying structural errors. Unfortunately, by this point of view, the verification environment, functional model and RTL implementation of this project were done by the same person.

Two levels of functional verification by simulation can be done: one at the block level, when each sub-module is verified using its dedicated testbench, and another at the top-level, where the functional correctness of the global system is evaluated. In this level, the complexity of the tests to fully stimulate the system increase significantly. Techniques like the use of constrained random variables, which stimulate the DUT randomly but from a set of intended configurations, the use of assertions and object-oriented frameworks in the construction of the verification environment are now common practice, in contrast with the use of directed tests to evaluate specific parts of the system. Moreover, standard verification methodologies as VMM, UVM and OVM have been developed which aim to standardize the verification strategies across teams and projects.

The quality of a testbench can be assessed from its coverage metrics, which measure if the DUT has been completely stimulated and tested during the simulation. There are several metrics usually considered: line coverage measures the number of lines of the DUT's RTL code evaluated during the simulation; toggle coverage measures the transitions (1 to 0 and 0 to 1) of each bit in the simulated system; branch and condition coverage measure the decision and evaluation of the conditional statements in the code; FSM coverage measures if the state machines of the module reach all possible states. Finally, the verification engineers define a set of coverage points that should be stimulated in order to verify the functional features of the system. The execution of this coverage points is measured in the functional coverage.

The goal of the design of a verification environment is to achieve 100% of coverage in all these metrics. However, achieving full coverage doesn't mean the design is completely tested, specially concerning the functional coverage where fault scenarios that weren't considered during the verification plan elaboration may be found using randomized tests.

The quality of the RTL code can also be evaluated using linting tools, like LEDA from Synopsys, which check if the code follows the norms defined by the language standards and coding styles meant to be followed. They also check if the code contains constructs that may arise problems during the development flow, as unintended latches, bad modeled state machines and problematic assignments between variables of different bit-width.

The synthesis of the design is an automatic process in which the RTL description of the design is synthesized into a gate-level netlist of cells of the target library. The target library of this project is a 40nm technology library and the synthesis tool used was Design Compiler from Synopsys.

The synthesis tool analyses the RTL code to extract standard constructs used like arithmetic operations, finite state machines, conditional evaluations, etc. and builds a gate level description using cells from a generic technology-independent library. This description is then converted into the target technology library. During these steps, several optimizations are performed concerning the goals of performance, area and power consumption of the resulting circuit. The designer

should define the constraints that guide this process before running the tool, using the GUI of the tool or scripts in TCL.

The set of constraints to be defined include: the target technology library; target clock frequency of the system; clock non-ideal characteristics as jitter and clock skew; delays in the paths between the inputs and outputs from the synthesized design and the environment where it will be integrated; loads at the input and output ports of the design and finally the driving cells to be considered at the input ports of the design. These constraints help the tool evaluate the results of the synthesis process considering the electrical conditions of the environment where the design will be used.

Design Compiler also has several switches and configurations that may be used to optimize the synthesis process [41]. In this project, it has been taken advantage of the clock gating switch, that enables the automatic implementation of clock gated cells that help reducing the power consumption of the system and also its area. The clock gating technique consists essentially in turning off the clock to flip-flops that aren't needed at a particular moment, reducing transitions and saving dynamic power. The use of clock gated cells allows the tool to reduce parts of mux logic, which results in a reduced circuit area. It has also been used the pipeline re-timing switch that enables the automatic optimization of the location of pipeline stages, and the switch that enables the replication of registers in order to overcome large fanout problems. The synthesis of this project was done in a bottom-up approach, synthesizing the sub-modules first and then the top-level with the output results of the already synthesized sub-modules. The pipeline re-timing wasn't applied to the top-level neither the filter's modules, because the register's re-timing could interfere with the filter's response.

The synthesis of the upsampling and downsampling filters have also taken advantage of the definition of multi-cycle paths in the arithmetic paths where data changes at half the input/output cadence due to the polyphase structure and also of the definition of false paths in the paths concerning configuration settings which are *quasi-static*. This options untie the timing constraints in these paths without changing their functionality, allowing the tool to achieve a solution more quickly, and using smaller logic. Without this strategies, this modules wouldn't be synthesizable for the proposed goal, at least with this architecture.

During the synthesis process the tool may also insert the scan chains for automatic tests of the fabricated chips, that help identifying problems generated during the production of the chips, like short circuits and open circuits (stuck-at faults). After synthesis, the test vectors to this scan tests can be automatically generated using tools like TetraMax from Synopsys.

After the synthesis process, it is important to verify if the resultant gate-level netlist is logically equivalent to the RTL description, because the synthesis process may originate errors if it hasn't run correctly. This verification equivalence may be done using the tool Formality from Synopsys. This tool identifies a set of comparison points between the two descriptions and elaborates a formal logical comparison between them.

The synthesized circuit performance should be evaluated using a static timing analysis tool like PrimeTime from Synopsys, which performs an analysis of the paths in order to find timing

violations, as setup and hold violations. Design Compiler also performs a similar analysis during the synthesis process, although it uses different assumptions, oriented to the synthesis process, which originate different results.

This flow of tasks since the beginning of the project is known as the front-end flow. After this, follows the back-end flow that drives the design from gate-level netlist to its final physical implementation, ready to be fabricated.

This project objectives don't consider the last steps of the front-end flow. It was performed the insertion of the scan chains during synthesis because it is a step that affects the synthesis results of the design, but it weren't generated any automatic test patterns. The static time analysis was conducted considering only the synthesis reports.

5.2 RGB to RGB Converter

The RTL description of this module implements the arithmetic operations concerned in signed combinational logic, considering input and output data widths of 24 bits per channel. The values of the coefficients are implemented in 26 bits constant signed wired logic and selected depending on the configuration (input control signal) from a parallel case with its outputs registered. The value of the video channels data is saturated between zeros and $2^{24} - 1$ before being asserted to outputs. The widths of the intermediate results vary depending on the operations performed due to the width growth each operation originates. The values are right-shifted only after all arithmetic operations to limit the quantization noise added by rounding, which is made before the right-shift by adding 0.5 and truncating.

The fact that the outputs from the multiplications between the constant coefficients quantized to 24 bits, so that 1.0 is encoded by 2^{24} , and the 24 bits video data are right-shifted by 24 bits, same as dividing it by 2^{24} , results that in fact the coefficients stored in integer representation originate the same results as if they were coded in fixed-point representation with integer and decimal part.

The synthesis of this module with registered inputs and outputs didn't match the performance goal. Therefore, it was optimized into a pipeline architecture taking advantage of the register retiming capabilities of the Design Compiler for pipelines. This consists in writing the pipeline registers concentrated at the input or output of the module so this way Design Compiler can relocate them freely, achieving better results than if the pipeline registers were pre-located across the module's datapath [41]. At least 4 pipeline stages are needed to fulfill the requirements and a comparison was made between the number of stages implemented and the area² and power statistics³ reported for the synthesis.

This data indicates that by using a larger number of pipeline stages and therefore reducing each stage timing constraints the synthesis tool is able to use combinational logic with smaller area, although the sequential area increases. The estimated power consumption of the module

²The area values reported by the synthesis tool were divided by the area of a 2x1 NAND gate of the technology library used to obtain the presented values in number of gates.

³The power statistics were reported by the synthesis tool with clock gating enabled.

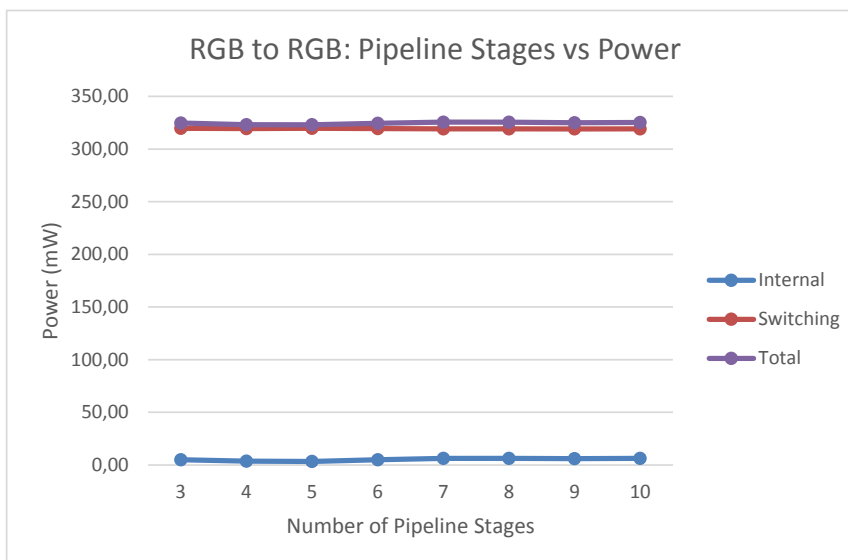
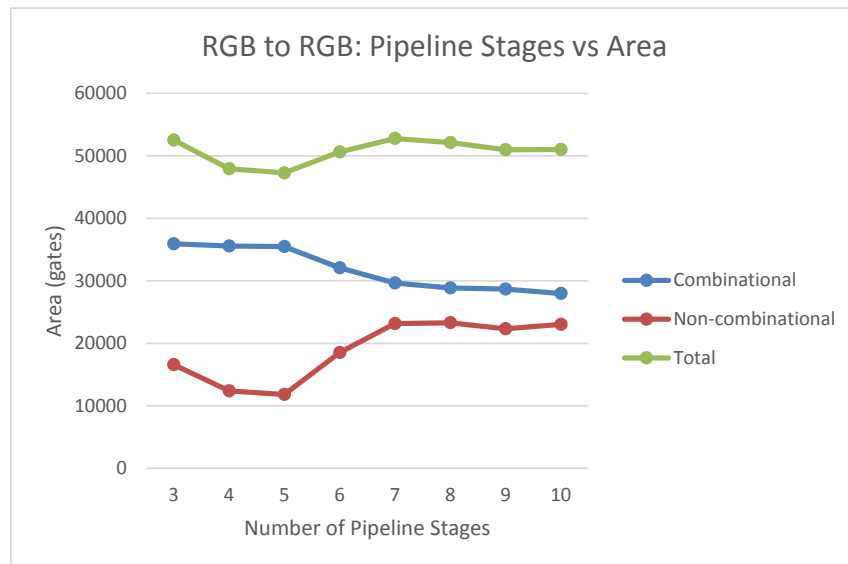


Figure 5.1: Comparison of the estimated area and power consumption of the RGB to RGB module with the number of pipeline stages implemented.

doesn't change significantly as a function of its number of pipeline stages. Because the latency of the module isn't a project requirement and the latency added by the increasing of pipeline stages (one cycle per stage) tends to be irrelevant when the system is running compared to possible area savings, it was decided to use 5 pipeline stages, which is the optimum number of stages indicated by this area and power data.

The statistics reported by the synthesis tool are based in estimations that at this stage of the design - RTL code - have a large uncertainty. One way this predictions can be improved is to provide the synthesis tool with databases of simulation results, which allow the tool to compute the number of transitions in each node in a more accurate way. Performing area and power estimations after the physical implementation of the module also provides more accurate results.

However, one would expect the global area/power behavior as a function of the number of pipeline stages to be similar in a physical design estimation, or even in the fabricated chip.

It was attempted to synthesize this module with the coefficients declared as input ports instead of local constants - this would allow the coefficients to be dynamically programmed in the register bank, for instance - but it didn't met the performance goals this way, even when adding more pipeline stages. This may suggest that the synthesis tool is parallelizing the arithmetic logic of the 14 sets of coefficients and replacing the multiplications by constants by shifts and adds operations, or, at least, eliminating a significant part of the combinational logic that implements the multipliers which is, in this case, only propagating constant values.

5.3 R'G'B'-Y'Cr'Cb' Converters

These modules were implemented using the same strategies and HDL techniques as the RGB to RGB module, as it performs very similar operations. The datapath is in signed encoding and the coefficients values from both the scaling and matrix operations are implemented in constant wires and selected from parallel cases.

The main differences are that in this modules there are two control signals, one for the matrix conversion and one for the data range conversion, and two rounding operations are done to limit the growth of intermediate values. The data is shifted-right and rounded at the end of both the scaling and the matrix operations. Moreover, in the R'G'B' to Y'Cr'Cb' converter the limiting values for the clipping and clamping operations are also driven from a parallel case because they depend on the output color space ranges.

The analysis of the number of pipeline stages to use in each of this modules is presented in figures 5.2 and 5.3, from which it was decided to implement 9 pipeline stages in the R'G'B' to Y'Cr'Cb' module and 7 in the Y'Cr'Cb' to R'G'B' module. Interestingly, the difference in the order of the scaling and matrix operations allows the synthesis tool to achieve different levels of datapath optimization which result in significant differences in the area of the synthesized modules.

The modules for constant-luminance conversion were implemented using the same principles. They are significantly smaller than the other modules of the system so it wasn't performed any comparison on the influence of the number of pipeline stages, as it would have relatively smaller

effects in the global area. They were synthesized with the fewer number of pipeline stages possible. Their synthesis results are presented in table 5.1.

In the Cr'Cb'-R'B' conversion modules the coefficients used depend on the signal of the $Y' - Cr'$, $Y' - Cb'$, $Y' - R'$ and $Y' - B'$ differences, respectively. To improve the timing performance of this conversion, this operations were parallelized so the all the computations (for negative and positive differences) are done in parallel with the evaluation of the difference values and the selection of the output value is done at the end.

Module	Cr'Cb' to R'B'	Y to G	RGB to Y	R'B' to Cr'Cb'
Pipeline Stages	2	5	2	4
Combinational Area (gates)	12 853	8 020	6 097	12 366
Sequential Area (gates)	7 396	4 803	3 619	7 067
Total Area (gates)	20 250	12 822	9 716	19 432
Internal Power (mW)	0.410	1.89	1.36	2.10
Switching Power (mW)	358.99	1.13	0.98	326.65
Total Power (mW)	359.40	3.03	2.35	328.76

Table 5.1: Synthesis results of the constant-luminance R'G'B'-Y'Cr'Cb' converters modules.

5.4 Gamma Encoder and Decoder

The encoding and decoding functions were divided in two independent modules which were successfully synthesized once more taking advantage of pipeline optimizations with the automatic re-timing option- figures 5.4 and 5.5. Based on the comparisons about the number of pipeline stages to use it was decided to implement 2 pipeline stages in both modules. Due to the differences of the encoding and decoding functions, the encoding module has approximately the double of the area since there are a larger number of entries that the additional LUT requires, compared to the decoding function (i.e. the nature of the encoding functions originates more errors using the linear segmentation method).

The lookup tables are implemented in parallel cases, which reduces the propagating delay compared to the use of nested conditional logic. The arithmetic computations required to interpolate the output value from the gain and offset values stored in the LUTS don't require the use of signed arithmetic, so they are implemented in unsigned logic.

Finally, the computed output values are limited to the range $[0; 2^{16} - 1]$.

5.5 Chroma Resampling Filters

The RTL implementation of the filters follows directly from their polyphase structure which is exemplified for shorter filters in figures 5.6 and 5.7. They are implemented using signed arithmetic logic and the multiplication coefficients are described as constant wired logic. The multiplications

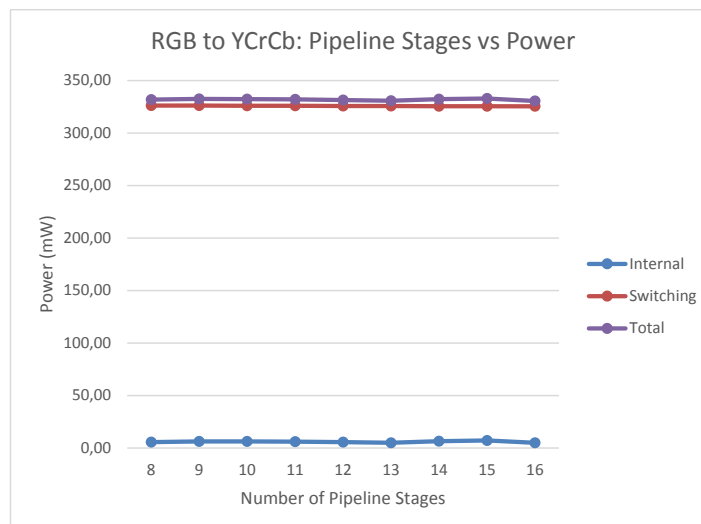
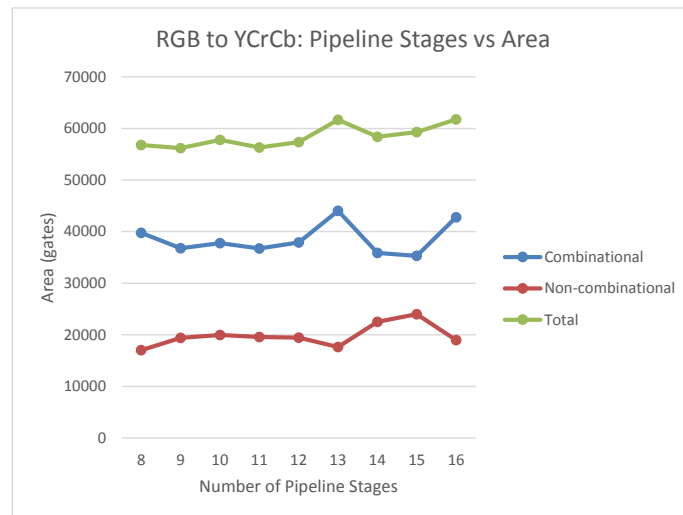


Figure 5.2: Comparison of the estimated area and power consumption of the RGB to YCC module with the number of pipeline stages implemented.

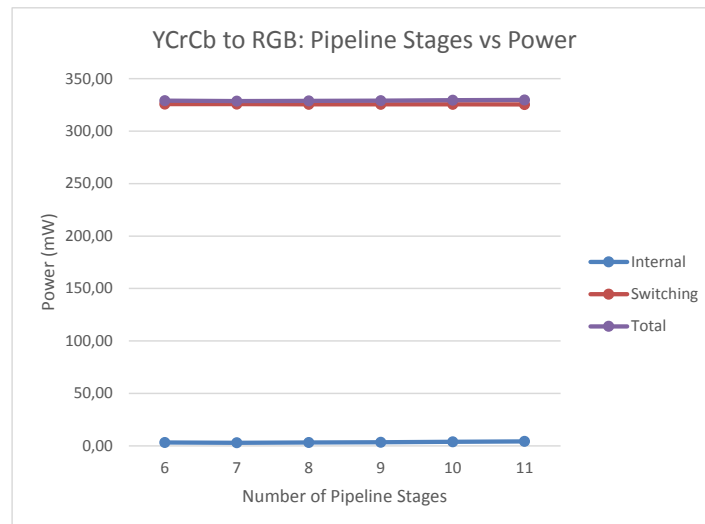
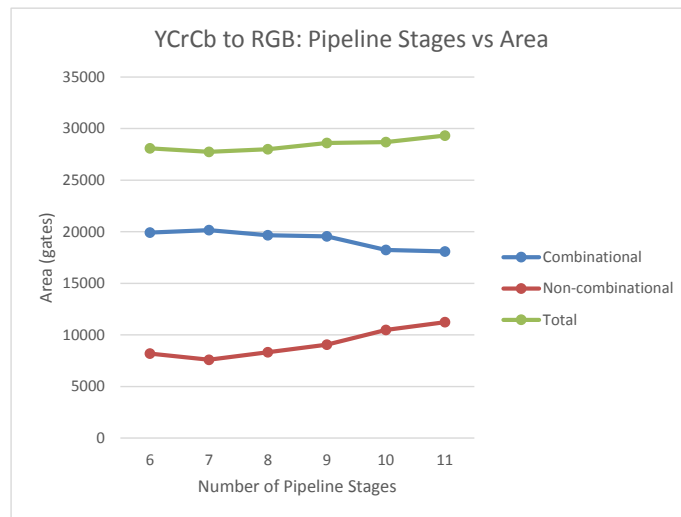


Figure 5.3: Comparison of the estimated area and power consumption of the YCrCb to RGB module with the number of pipeline stages implemented.

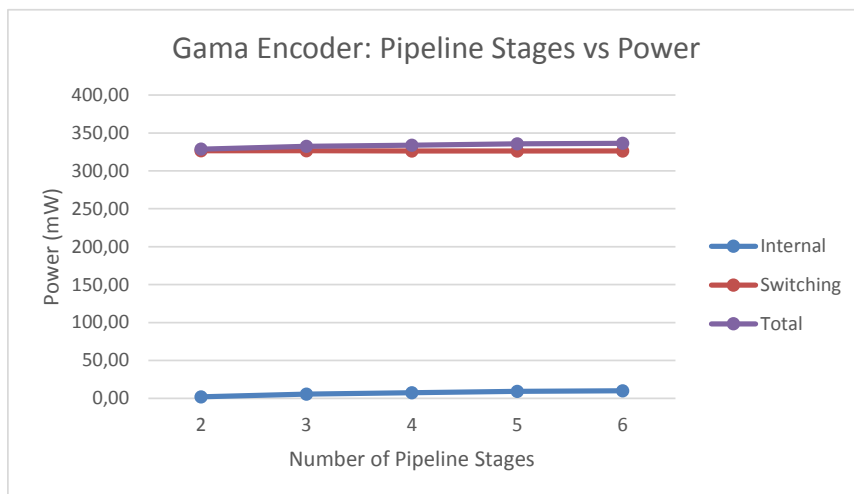
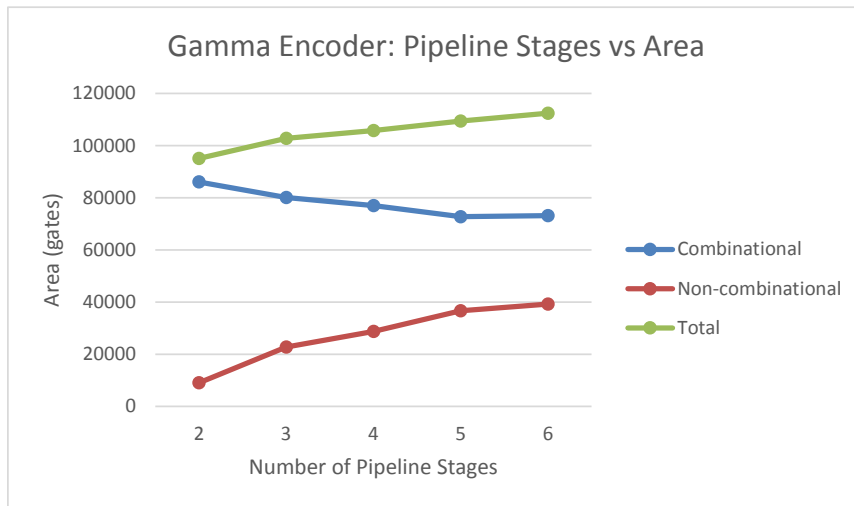


Figure 5.4: Comparison of the estimated area and power consumption of the gamma encoder module with the number of pipeline stages implemented.

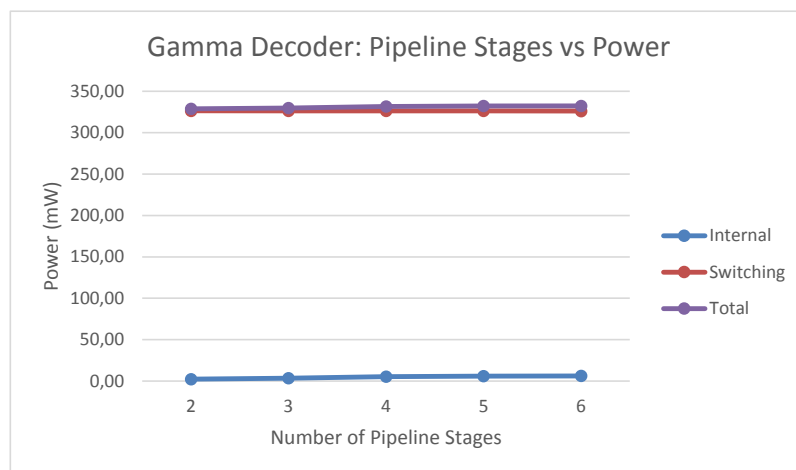
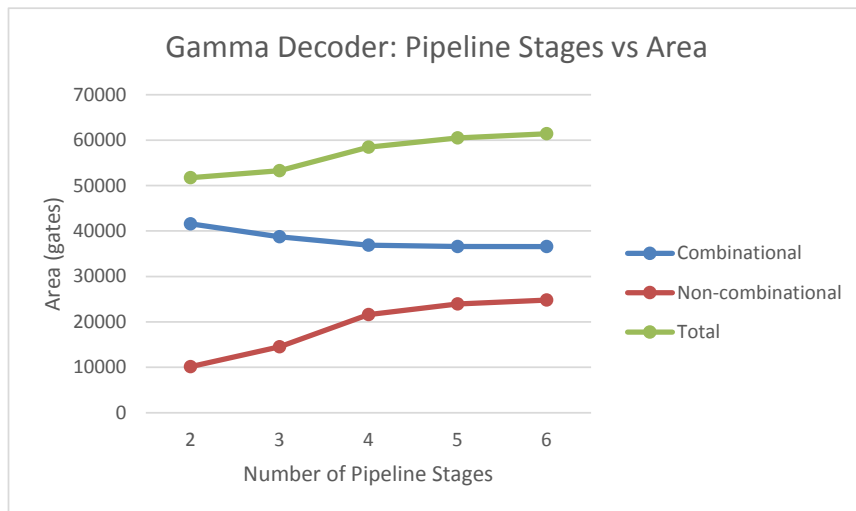


Figure 5.5: Comparison of the estimated area and power consumption of the gamma decoder module with the number of pipeline stages implemented.

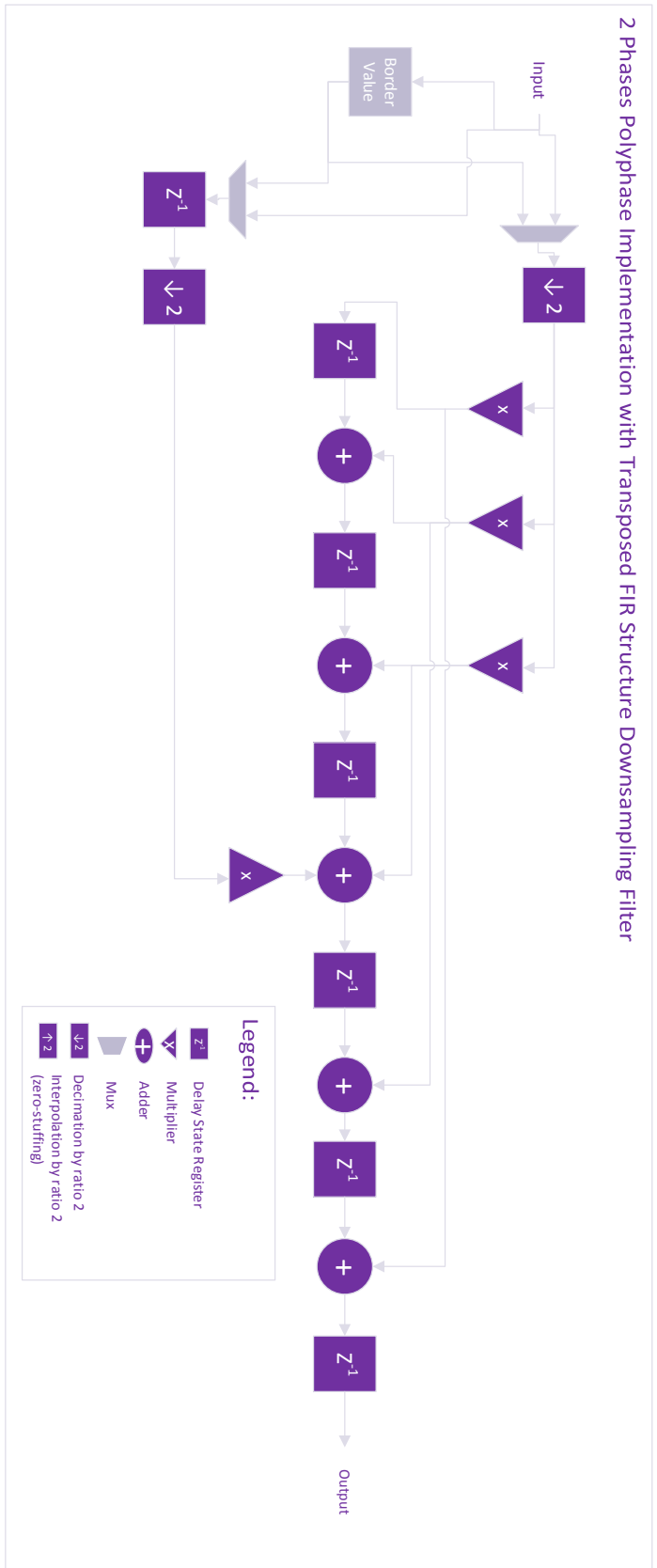


Figure 5.6: Downsampling filter structure.

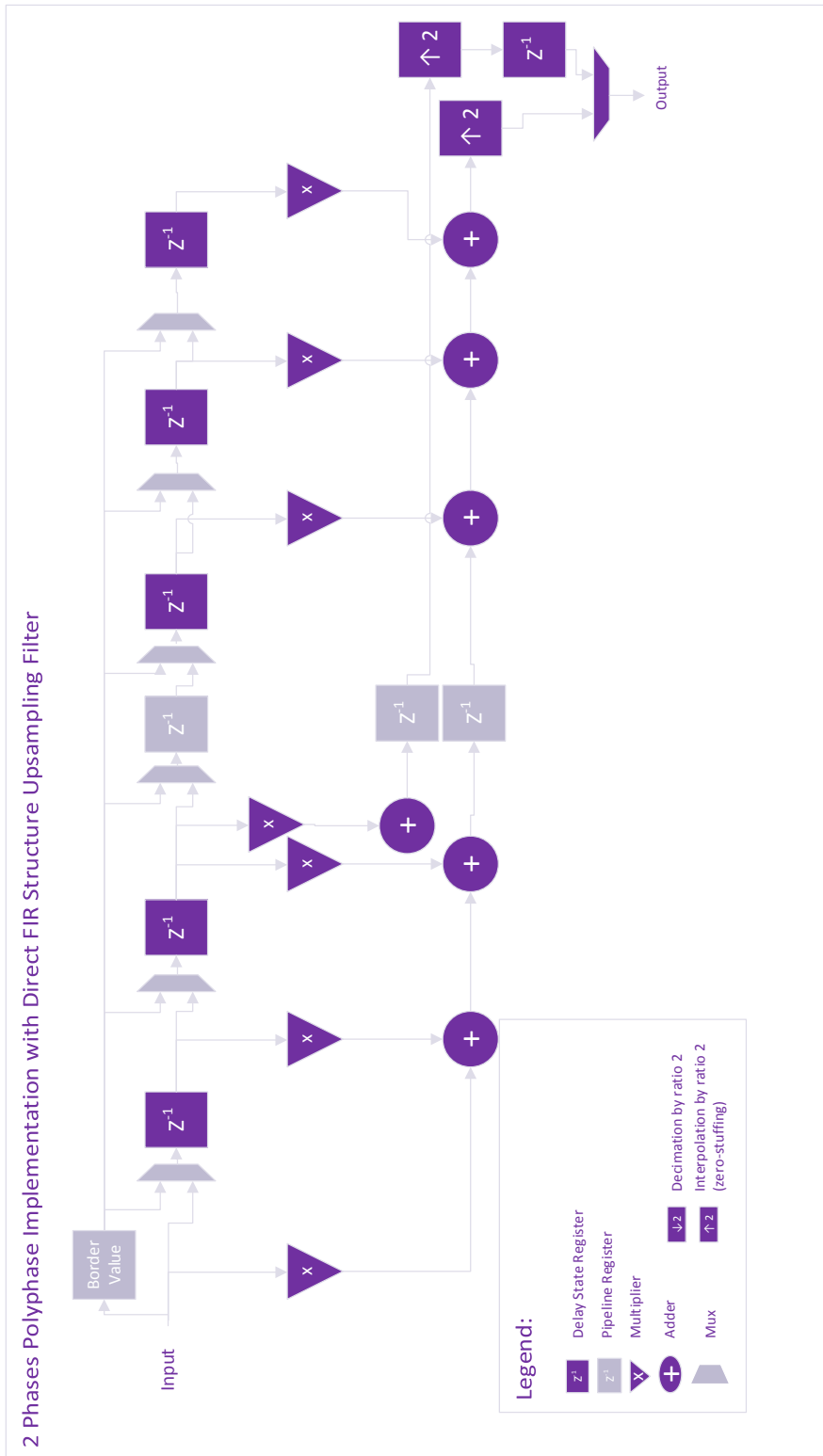


Figure 5.7: Upsampling filters structure.

by zero-coefficients are discarded and in the downsampling filter it is taking advantage of the symmetry of the phase 0 coefficients to reduce the number of multipliers implemented.

The polyphase decomposition by ratio of 2 reduces the timing constraints for the adders and multipliers logic by half, by allowing it to work at half the input/output cadence. Otherwise, if using a normal filter, we would have to be computing samples at twice the input's frequency.

However, concerning the upsampling filter, the long chain of subsequent additions for each phase results in a critical path which isn't synthesizable for the target frequency, even with the polyphase decomposition. Because of this issue, the filter was divided in three pipeline stages, cutting the critical path in smaller sections.

For a proper implementation of pipelining in a FIR filter without modifying its frequency response, one must guarantee that the data flow in the datapath is cut completely transversely. For that purpose, it is necessary to add a pipeline register both in the state's register chain and in the adders chain, in the same chain position - lighter color registers in figure 5.7.

The synthesis of this module was successfully achieved using 3 pipeline stages and setting each stage as multicycle paths which dispose of 2 clock cycles for its computations.

A fundamental concept of linear and time-invariant (LTI) systems is that its behavior can be completely characterized by its impulse or step response. Therefore, the step and impulse responses of the filter were computed using the Matlab model and quantized using the same process as the coefficients. This way, it was possible to validate the frequency response of the implemented filter. The results from this validation are presented in section 5.8.

The resampling filter are the only modules in the system that don't process in a pixel-to-pixel basis. Therefore, they need to be adapted to support pixel repetition formats, in which we're not interested in filtering the copies of each pixel. This feature was implemented by adding controlled enable signals to the flip-flops that implement the filters states so that they only update their stored value on the first sample of the same pixel. This control enables signals are generated by counters that are synchronized with the data enable signal positive edge, because the repetition of pixels only applies to video data signals during active video periods. During blanking periods the video channels are expected to stay still.

Furthermore, the padding of the image borders requires the implementation of input buffers in both filters and the saving of the border values in registers. This way, in the upsampling filter, when the first pixel of the video active period arrives at the input of the filter, and the border value register, all the state registers are set to the border value, effectively padding the image with border's value. At the end of the video active period, instead of setting all the states to the border value, it should only be fed to the input of the filter, as if the image continued to be propagated.

The same strategy is implemented in the downsampling filter. However, because in this filter the state registers accumulate the results of the multiplications, it isn't possible to set them all to the border's value at the beginning of the video active period. This requires the input buffer of the filter to have the size of the filter's latency in order to execute the same process as at the end of the video active period.

In 3D side-by-side mode, the filtering of the two images must be independent. For that reason, two filters for each chroma channel were implemented, one for the left image and one for the right image. This imposes a hardware cost for these modules that is double that predicted in the table 4.4.

In the middle of the active video line, at the left and right images division, there are no synchronization signals that could be used to control the padding of the two images and the selection of the output value to be propagated. An additional counter was implemented that counts the number of pixels during each active video period. The horizontal width of the two fields (images) is fed to the module as an input, together with control inputs that define the pixel repetition mode and the 3D or 2D modes. Depending on the number of pixel repetitions (1 to 10 samples of the same pixel), a set of flag values are computed from constant values and the width of the fields, which are compared to the state of the pixel counter and activate control signals that manage the padding and filter swap in 3D mode.

In 2D modes or 3D non side-by-side modes only the left image filters are used. In both modes, the luma channel and the control signals are propagated through buffer chains. The luma channel registers are controlled with the same enable signals as the chroma states registers because of the pixel repetition support. However, pixel repetition doesn't apply to hsync, vsync and data enable signals which requires longer buffer chains to which the output registers are connected to different positions depending on the pixel repetition mode. The luma buffers have the same length as the chroma chain of registers.

At the output of the filters the chroma values are limited to the full range $[0; 2^{24} - 1]$ in the upsampling filter or to the range required by the output color space in the downsampling filter, which requires the output range control signal which is fed to the R'G'B' to Y'Cr'Cb module to be wired also to this module.

The synthesis results of the 30 taps and 18 taps filters implemented are presented in table 5.2. The 18 taps filters represent a decrease of 44% and 34% in the area of the upsampling and downsampling filters, respectively.

Filter	Upsampling		Downsampling	
	30 taps	18 taps	30 taps	18 taps
Combinational Area (gates)	139 494	74 400	65 819	39 836
Sequential Area (gates)	39 550	25 242	37 924	28 344
Total Area (gates)	179 045	99 642	103 742	68 181
Internal Power (mW)	3.12	2.20	1.61	1.37
Switching Power (mW)	35.82	11.18	349.50.68	358.32
Total Power (mW)	39.03	13.42	351.32	359.71

Table 5.2: Synthesis results of the 30th and 18th order resampling filters.

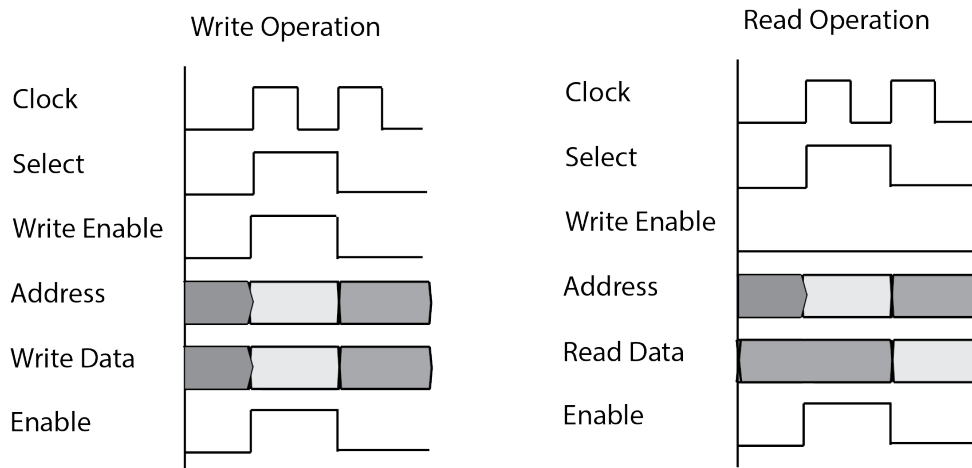


Figure 5.8: Register bank interface transfer diagrams.

5.6 Control Unit and Register Bank

The implementation of the control unit consists in a set of parallel cases that select the values of the control signals wired to the other modules, depending on the parameters configured in the register bank.

The memory of the register bank module is implemented using flip-flops directly connected to this module outputs which are wired to the control unit inputs. The reading and writing operations to this module are managed by a block of conditional logic, which controls the selection of the addressed positions and its accessibility. The transfer diagrams of the register bank interface are presented in figure 5.8, which describes the expected sequence of input signal values to perform read and write operations. The lighter grey blocks in the bus ports are the periods during which the data in that port is valid to be written or read.

5.7 Top Level

The top level module of the color space converter not only instantiates its internal modules but it also performs the video data shifts from and to the input and output data widths to the 24 bits width used in the internal datapath. All the modules of this project have been developed using parameterized data and coefficient widths but the values of the coefficients needed in each module must be manually replaced. However, the matrix operations and the LUT tables coefficients are defined in independent Verilog files that are automatically generated for different bit widths by Matlab scripts developed in this project.

SCORE	LINE	COND	TOGGLE	BRANCH	
100.00	100.00	100.00	100.00	100.00	u_dut_csc
SCORE	LINE	COND	TOGGLE	BRANCH	
100.00	100.00	100.00	100.00	100.00	u_csc_chroma_downsampler
100.00	100.00	100.00	100.00	100.00	u_csc_chroma_upsampler
100.00	100.00	100.00	100.00	100.00	u_csc_control
100.00	100.00	100.00	100.00	100.00	u_csc_gamma_dec
100.00	100.00	100.00	100.00	100.00	u_csc_gamma_enc
100.00	100.00	100.00	100.00	100.00	u_csc_regbank
100.00	100.00		100.00	100.00	u_csc_rgb2rgb
100.00	100.00		100.00	100.00	u_csc_rgb2ycc_range
100.00	100.00		100.00	100.00	u_csc_rgb2ycc_range_cl_crcb
100.00	100.00		100.00	100.00	u_csc_rgb2ycc_range_cl_y
100.00	100.00		100.00	100.00	u_csc_ycc2rgb_range
100.00	100.00		100.00	100.00	u_csc_ycc2rgb_range_cl_crcb
100.00	100.00		100.00	100.00	u_csc_ycc2rgb_range_cl_y

Figure 5.9: Coverage results obtained from simulation.

5.8 Verification Results

The verification of the module was conducted considering the verification plan presented in section 4.10. It were developed testcases that simulate the tasks described, and some of them were also verified manually because they have a higher probability of having structural errors in both the DUT and the reference model, as it is the case of the verification of the propagation of the control signals, synchronized with the flow of the video data.

The coverage results obtained from the testbench developed are presented in figure 5.9. These results demonstrate that the testbench developed fully exercises the DUT. To achieve this results, the coverage metrics of non-controllable nodes have been excluded, for instance toggle coverage in constant coefficients nets or in control signal nets, which can only have a predefined set of values.

The impulse response of the resampling filters have also been compared with the impulse response of the filter models in Matlab, which response has been quantized the same way as the implemented filters coefficients.

In the figures 5.10 and 5.11 the output values of the luma and data enable signals are also plotted, which allows to evaluate also the behavior of the filter response at the border of the active video period. To facilitate this comparison the impulse response of the reference filter model is only stimulated for the impulses at the right. The impulse response from the reference filters haven't be clamped to only unsigned values, which allows to verify that the implemented filters are correctly limiting the output chroma signals to values greater or equal to zero, as expected.

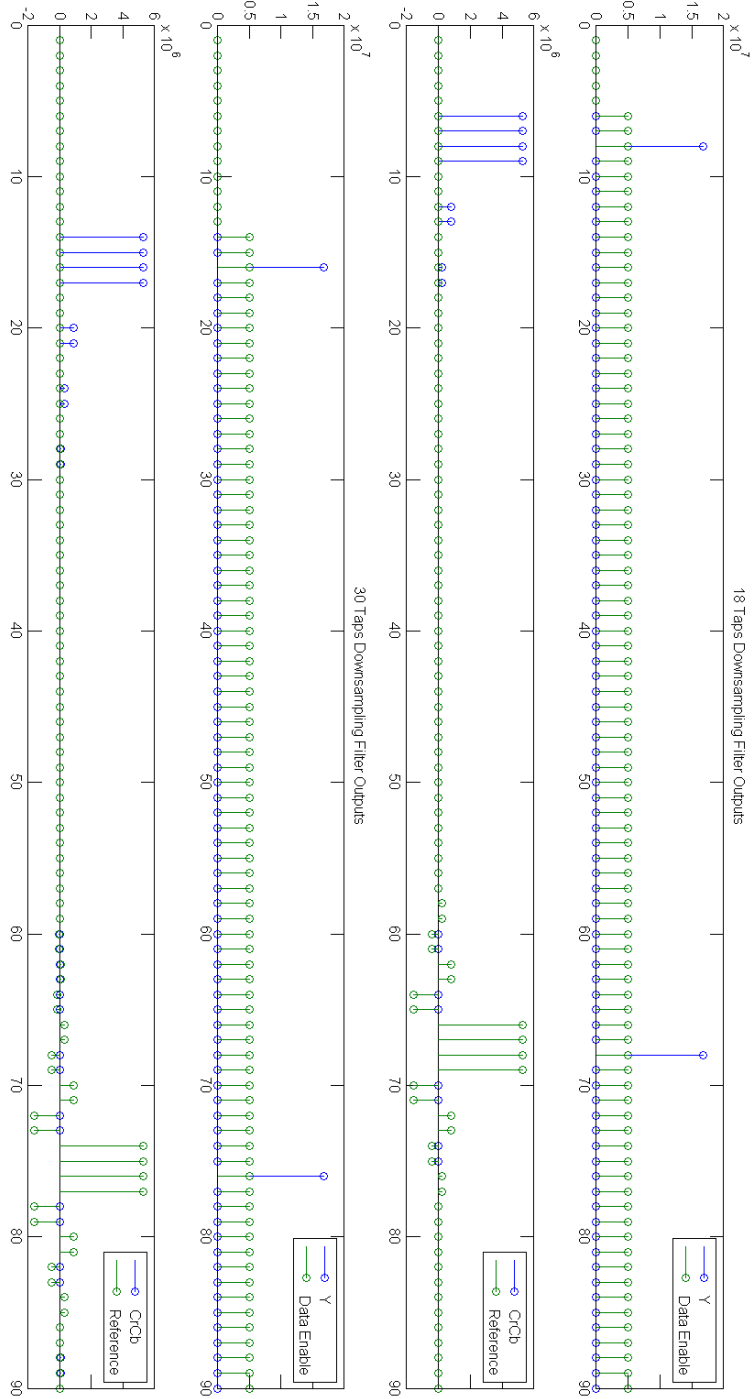


Figure 5.10: Verification of the downsampling filters impulse response.

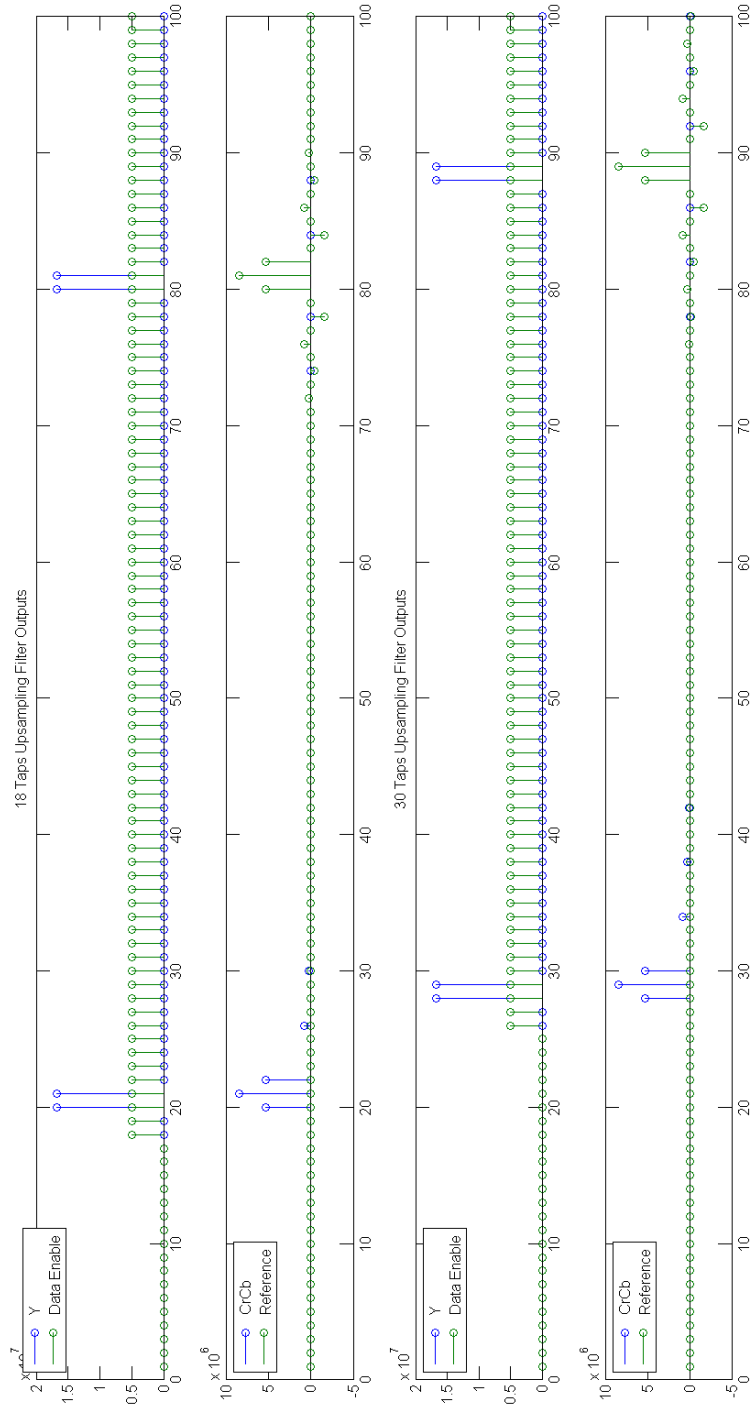


Figure 5.11: Verification of the upsampling filters impulse response.

5.9 Results

All sub-modules and the top level module were synthesized with the target clock frequency set to 660 MHz (10% margin over the required) and the clock uncertainty (variations in the clock skew) was defined to approximately 5% of the clock period. It was also defined that in the input and output paths 40% of the clock period was scheduled to hypothetical combinational paths outside of the module. The input pins driving cells were defined to inverter cells with its load capacity increased 10 times. The output pins were loaded with the cells with highest load capacity from the technology library used. The narrowing of the synthesis constraints beyond the goals defined allow us to have better confidence on the physical feasibility of this module with the requirements proposed.

The color space conversion module was successfully synthesized with the constraints described above for the two configurations considered: with 30th order filter and 18th order filters - table 5.3. The 18 taps configuration is 18,7% smaller than the 30 taps configuration, which also results in less consumed power.

The verification of equivalence tool was executed after the synthesis process of each module, but for some reason that wasn't possible to identify, all the verification equivalence tests failed. The execution reports from Design Compiler and Formality were extensively analyzed, and the linting reports obtained from LEDA also, but the root cause of this problem remained unidentified. To verify if the synthesized gate-level netlist was functionally correct the verification environment developed was simulated with the gate-level netlists of the DUT obtained from the synthesis tool. This verification validated the functional equivalence between the RTL description of the module and its gate-level netlist.

Configuration	30 taps	18 taps
Combinational Area (gates)	461 082	369 422
Sequential Area (gates)	152 503	129 575
Total Area (gates)	613 585	498 996
Internal Power (mW)	17.39	16.34
Switching Power (mW)	2002.00	1692.10
Total Power (mW)	2019.60	1708.60

Table 5.3: Top-level synthesis results of the 30th and 18th order filters configurations.

Despite the verification tasks that have been conducted it was also achieved to simulate the module with real images as inputs so the output results could be evaluated at a higher abstraction level.

For this task, it was used one of the images usually used for image processing tests, known as *mandrill*. This image has the advantage that besides having varied color content it has segments with high frequency content in the mandrill's fur. The image matrix was transformed in Matlab into input vectors that are loaded into memories in the SystemVerilog testbench. The output of the module is written to a file that can be loaded to Matlab to reconstruct the image.

The evaluation of the correct colors appearance in each output color space isn't possible because it would require the operating system of the visualization system and the monitors used to be capable of supporting those color spaces. Moreover, the environment's visualization conditions would also need to be controlled. Nevertheless, the methodology applied allows to evaluate if the color content of the image is being modified according to the expectations.

In a RGB to RGB conversion, it is expected that the colors of the image will change slightly, because of the differences between the gamuts. These changes are expected to be more visible in saturated colors, that are located at the boundaries of the gamuts of each color space - figure 2.2. The less saturated colors (nearer the white point) are expect to change less because they're visible in all color spaces and the considered color spaces use all the same white point - D65.

Moreover, if we open an image coded in a particular color space and read it as if it is coded in other color space with a wider gamut (without converting the color values), it is expected the saturation of the colors to increase, because the image's color map is being expanded so the colors shift towards the boundaries of the CIE 1931 color space. So when we convert an image to a wider gamut color space but read it as if it is in the original color space it is expected the image to appear less saturated, because the conversion changes the color values (in this situation, decreases them) in order to keep the appearance in the wider color space. This happens considering that the wider color space encodes a wider range of values with the same number of codes.

This behavior is observable in the figure 5.12, where the image was converted from a smaller gamut to a wider gamut color space (sRGB to Rec.2020 non-constant luminance - see section 2.6.7), but the Matlab image viewer interprets the input and output images as being in the same color space so the output image appears less vivid, specially in the areas of more saturated colors like the nose of the mandrill.

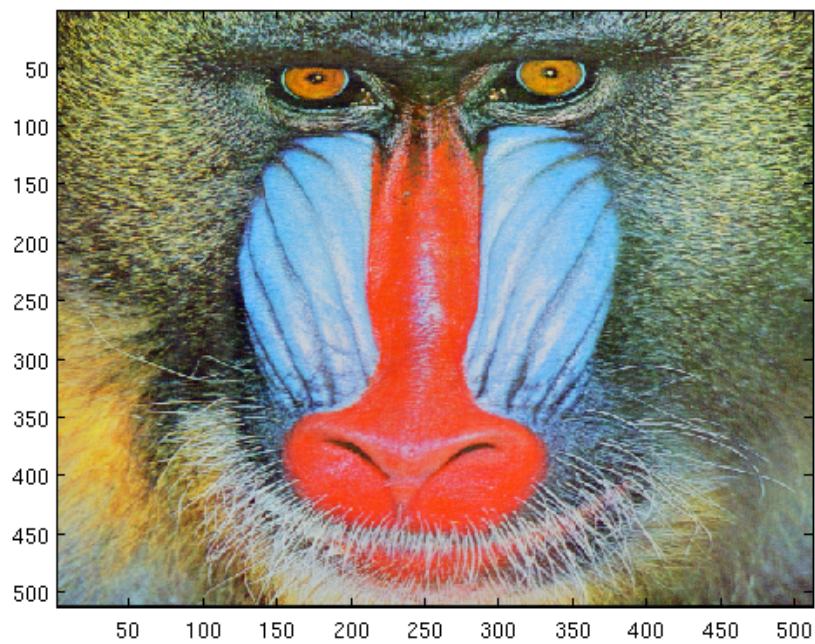
Reading images encoded in Y'Cr'Cb' as if they are encoded in RGB originates the results presented in figure 5.13, where the output image obtained from the module is compared with the output from a Matlab function that performs the same conversion.

The effect of the chroma downsampling using the two implemented filters is presented in figures 5.14 and 5.15. Instead of the mandrill figure it was used an image with textual content in different colors, referred to as specifically designed to identify chroma subsampling schemes in televisions and monitors [42]. From this very subjective comparison, it seems that the use of the smaller filters which reduce the size of the global module in 18.7% when compared with the standards compliant ones doesn't affect significantly the perceptible quality of the image, as this distortion suffered by the text appears to be approximately the same, except in the last two lines of text where the 30 taps filter performs significantly better. Nevertheless, it has to be considered that this image is an extreme situation, with the purpose of detecting if chroma sub-sampling schemes have been applied or not.

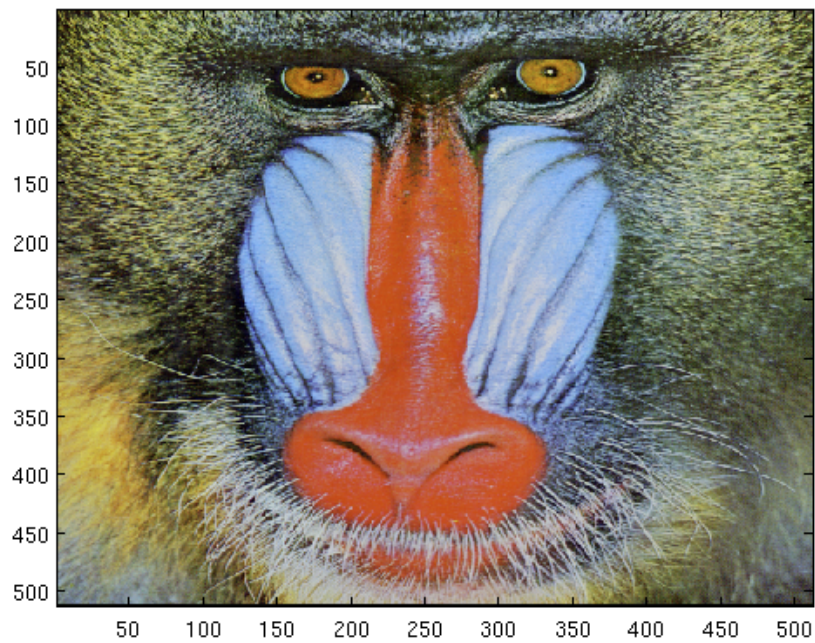
In figure 5.16 the output results of downsampling the mandrill's image from RGB 4:4:4 to YCC 4:2:2 and then back to RGB 4:4:4 are compared for the two filter configurations, considering the Rec.601 525-lines color space. There are no perceptual differences between these two pictures and in fact, computing the PSNR between these images and the original image it is obtained a

small difference of 25.33 dB for the 30 taps configuration and 25.19 dB for the 18 taps configuration. Once more, this seems to indicate that the smaller filters would be sufficient for video content, where the images aren't static, with a similar performance as the filters compliant with the standards.

However, it would be needed further tests with methods of evaluation of perceptual quality of images and video to be able to driven further conclusions.

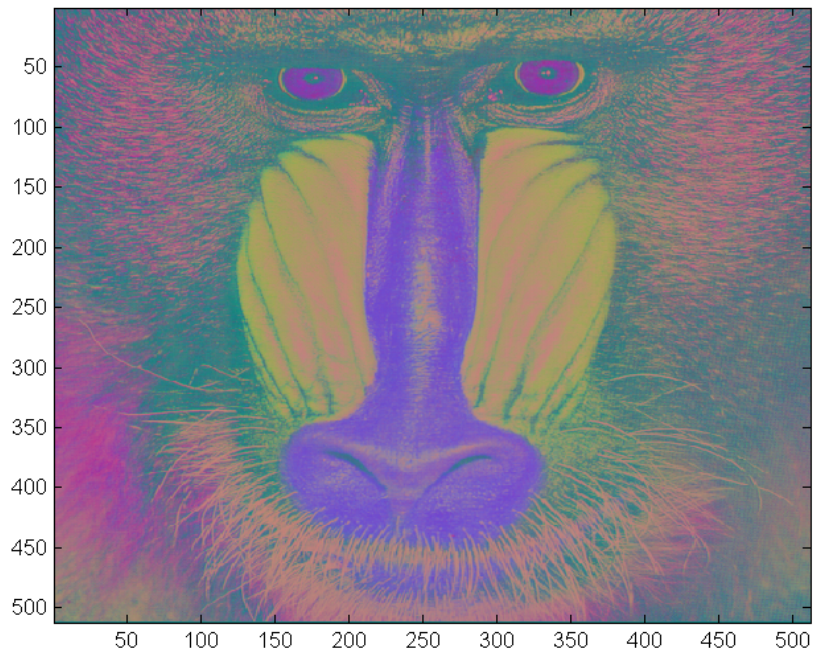


(a) Original mandrill image.

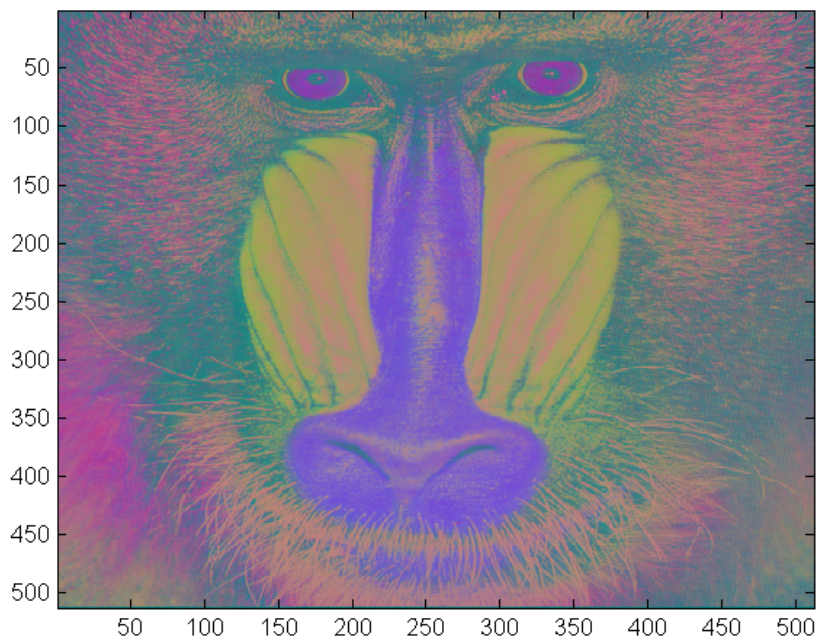


(b) Output mandrill image.

Figure 5.12: RGB Rec.2020 non-constant luminance output image, input image read as RGB sRGB image.

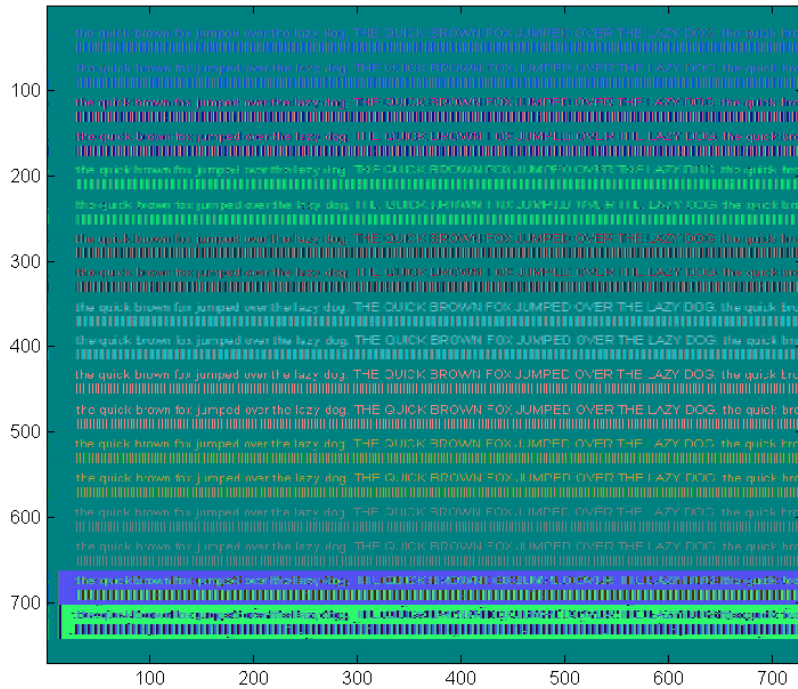


(a) Y'Cr'Cb' 4:4:4 Rec.601 525-lines output image, input image read as RGB Rec.601 525-lines image.



(b) Y'Cr'Cb' 4:4:4 Rec.601 image obtained using Matlab *rgb2ycbcr* function which converts RGB images to YCC using Rec.601's coefficients matrix.

Figure 5.13: Comparison of the Y'Cr'Cb' 4:4:4 Rec.601 image obtained using the developed module and the Matlab *rgb2ycbcr* function.



(a) 18 taps filter configuration.



(b) 30 taps filter configuration.

Figure 5.14: Comparison of Y'Cr'Cb 4:2:2 Rec.601 525-lines output images obtained with the two filter configuration, input image read as RGB sRGB image.

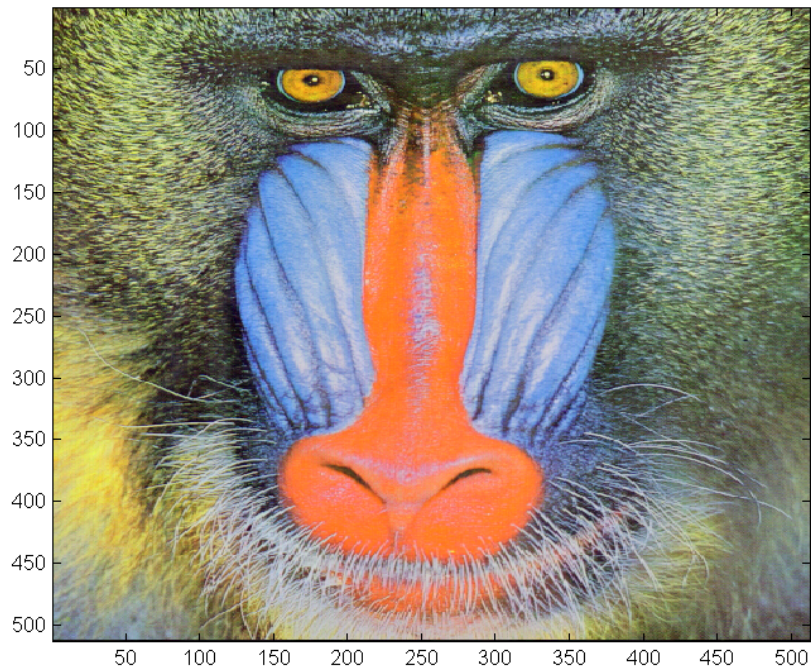


(a) 18 taps filter configuration.

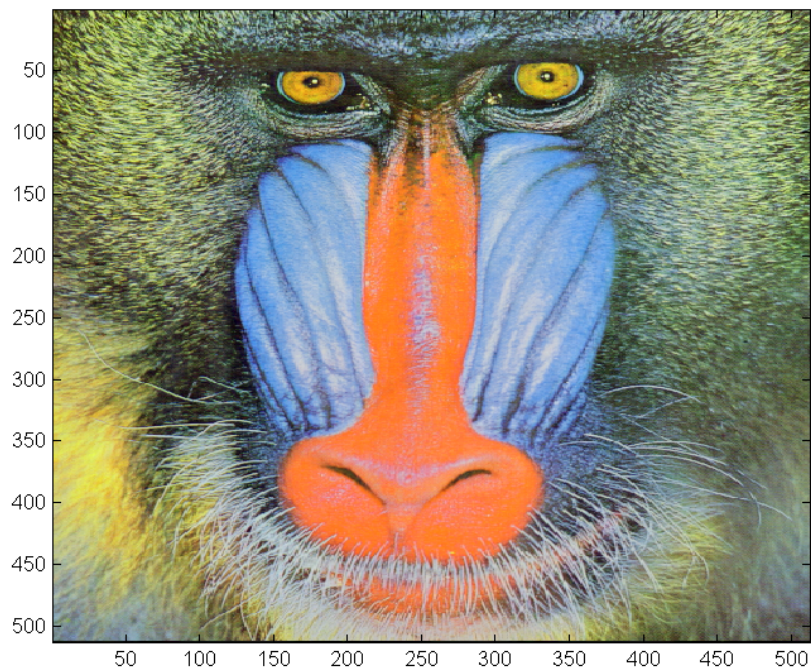


(b) 30 taps filter configuration.

Figure 5.15: Comparison of Y'Cr'Cb 4:2:2 Rec.601 525-lines output images obtained with the two filter configuration and then converted to RGB using Matlab *ycbcr2rgb* function to improve the legibility of the text. Input image read as RGB sRGB image.



(a) 18 taps filter configuration.



(b) 30 taps filter configuration.

Figure 5.16: Comparison of the RGB 4:4:4 mandrill's images after being downsampled to YCC 4:2:2 and then upsampled to back to RGB 4:4:4, considering the Rec.601 525-lines color space.

Chapter 6

Conclusions and Future Work

The development of this thesis achieved the proposed goals: it was developed an hardware module capable of performing real time color space conversion of video streams. This module supports all the color spaces presented in the standards proposed, performs downsampling and upsampling to Y'Cr'Cb 4:2:2 formats with two different FIR filter implementations and supports pixel repetition and 3D video structures. The two FIR filter implementations provide configurations of the module with different image quality/area and power compromises, which may be a competitive strength of this module. It was verified by comparing with a software model implementing its expected behavior and it was successfully synthesized to a 40nm technology, achieving the goal frequency of 600 MHz.

The module developed is innovative in the sense that the market and bibliographic research didn't provide results of other cores that implement the full color space conversion chain, from any RGB/YCrCb color space to any RGB/YCrCb color space. The majority of the available solutions only consider RGB-YCrCb conversion.

Therefore, this module offers a real-time hardware accelerated solution for a task very computationally intense, which effort is growing with the increase of video frame resolutions, following the industry tendency of moving imaging computational exigent tasks to dedicated hardware with the arising of embedded image and vision processing solutions from several IP vendors.

Concerning future work, the capability of upsampling and downsampling to YCrCb 4:2:0 formats would be an interesting addition to the features list of this module. It could be easily done using the nearest neighbor method, where it would be needed to have buffers the size of half a line in progressive mode (in 4:2:2 the number of Cb or Cr samples is half of the Y samples), or two half lines in interlaced mode. An implementation of this feature has been already done, although this feature was discarded from the features list due to the increased area it would require: the largest horizontal video active width is 4096 pixels, which means at least 2048 flip-flops in progressive mode, if implementing the buffer with a chain of registers. Using a multi-tap FIR filter to re-sample from and to 4:2:0 would impose an even larger area.

For improving the quality of the IP, the implementation of the module could be reviewed having in consideration power and area implications, which was not a goal during the development

of this thesis.

Finally, one could integrate this module in a higher level design or follow the backend flow on this module, developing the place-and-route, parasitic extraction, static timing analysis and verification of the physical design, obtaining an IP ready to be produced into a physical chip.

References

- [1] Rein van den Boomgaard. Image processing and computer vision, 18/10/2013 2013. URL: <https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV/IP/Color/colorimetry.html>.
- [2] PAR. CIE 1931 xy chromaticity diagram, 2005. URL: <http://commons.wikimedia.org/wiki/File:CIExy1931.png>.
- [3] David Katz and Rick Gentile. Fundamentals of embedded video, part 2, 2007. URL: http://www.eetimes.com/document.asp?doc_id=1275460.
- [4] Xilinx. LogiCORE IP Product Guide - RGB to YCrCb Color-Space Converter v7.1 , 2014.
- [5] F Bensaali, A Amira, and A Bouridane. Design and implementation of efficient architectures for color space conversion. *International Journal on Graphics, Vision and Image Processing*, 5(1):37–47, 2004.
- [6] Charles Poynton. Merging computing with studio video: Converting between R'G'B' and 4:2:2. 2004.
- [7] ITU. Recommendation ITU-R BT.709-5, 2002.
- [8] ITU. Recommendation ITU-R BT.601-7, 2011.
- [9] Bruce Lindbloom. RGB Working Space Information, 31/1/2014 2014. URL: http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html.
- [10] Inc. Synopsys. About Synopsys. URL: <http://www.synopsys.com/Company/AboutSynopsys/Pages/About.aspx>.
- [11] ITU. Recommendation ITU-R BT.2020-1, 2014.
- [12] IEC. International Standard IEC 61966-2-1 Amendment 1 Multimedia systems and equipment - Colour measurement and management, 2003.
- [13] IEC. International Standard IEC 61966-2-5 Multimedia systems and equipment - Colour measurement and management, 2007.
- [14] Hitachi, Ltd., Panasonic Corporation, Philips Consumer Electronics International, B.V., Silicon Image, Inc., Sony Corporation, Technicolor, S.A., Toshiba Corporation. High-Definition Multimedia Interface Specification Version 1.4a, 2010.
- [15] IEC. International Standard IEC 61966-2-4 multimedia systems and equipment - colour measurement and management.

- [16] Charles Poynton. *Digital video and HDTV algorithms and interfaces*, pages XLII, 692 p.–XLII, 692 p. Morgan Kaufmann Publishers, Amsterdam [etc.], 2003. eng.
- [17] Aníbal João de Sousa Ferreira. *Comunicações audiovisuais tecnologias, normas e aplicações*. Ensino da Ciência e da Tecnologia. IST Press, Lisboa, 2009. por.
- [18] ITU. Report ITU-R BT.801-4 - The present state of high-definition television. Report, 1990.
- [19] ITU. Report ITU-R BT.2246-3 - The present state of ultra-high definition television. Report, 2014.
- [20] Adobe Systems Incorporated. Adobe RGB (1998) Color Image Encoding, 2005.
- [21] Kenichiro Masaoka Kohei Ohmura Masaki Emoto Yukihiro Nishida Masayuki Sugawara Takayuki Yamashita, Hiroyasu Masuda. Super hi-vision as next-generation television and its video parameters, 2012. URL: <http://informationdisplay.org/IDArchive/2012/NovemberDecember/FrontlineTechnologySuperHiVisionasNextGen.aspx>.
- [22] Altera. *Video and Image Processing Suite - User Guide*. 2014.
- [23] Xilinx. LogiCORE IP Product Guide - YCrCb to RGB Color-Space Converter v7.1. 2014.
- [24] Lattice Semiconductor. LatticeCore Color Space Converter IP Core User Guide, 2015.
- [25] iWave Systems Technologies. iW-Color Space Converter Datasheet.
- [26] Jiang Hongxu, Li Hanqing, Liu Tingshan, Zhang Ping, and Lu Jinyuan. A fast method for RGB to YCrCb conversion based on FPGA.
- [27] K. Holm and O. Gustafsson. Low-complexity and low-power color space conversion for digital video. In *Norchip Conference, 2006. 24th*, pages 179–182.
- [28] Charles Poynton. ITU-R BT.601-4 Digital Filters, 1998. URL: http://www.poynton.com/notes/short_subjects/video/ITU-R_Rec_601_digital_filter.
- [29] Charles Poynton. Converting Y'CbCr to R'G'B', 2005 1998. URL: http://www.poynton.com/notes/short_subjects/video/YCbCr_to_RGB.
- [30] Gregg Hawkes. XAPP294 - digital component video conversion 4:2:2 to 4:4:4. 2001.
- [31] Keith Jack. YCbCr to RGB Considerations, March 1997 1997.
- [32] S. Akramullah. *Digital Video Concepts, Methods, and Metrics: Quality, Compression, Performance, and Power Trade-off Analysis*. Apress, 2014.
- [33] A. Gabriellini and M. Mrak. On coding and resampling of video in 4:2:2 chroma format for cascaded coding applications. In *2013 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4.
- [34] Glenn Chan. Toward better chroma subsampling: Recipient of the 2007 SMPTE student paper award. *SMPTE Motion Imaging Journal*, 117(4):39–45, 2008.
- [35] K. Jack. *Video Demystified: A Handbook for the Digital Engineer*. Elsevier Science, 2011.

- [36] Maciej Bartkowiak. *Improved Interpolation of 4:2:0 Colour Images to 4:4:4 Format Exploiting Intercomponent Correlation*. na, 2004.
- [37] Xilinx. *Chroma Resampler v4.0*. 2014.
- [38] Ken Turkowski. *Filters for common resampling tasks*. Academic Press Professional, Inc., 1990.
- [39] ANSI/CEA. *A DTV Profile for Uncompressed High Speed Digital Interfaces ANSI/CEA-861-F*, 2013.
- [40] P.P. Vaidyanathan. *Multirate Systems And Filter Banks*. Electrical engineering. Electronic and digital design. Dorling Kindersley, 1993.
- [41] Synopsys. *Design Compiler User Guide - Version J-2014.09*, September 2014, September 2014.
- [42] Geeks 3D. *How to Quickly Check the Chroma Subsampling used by your 4K UHD TV*.

