A design of the universal color-space converter IP-cores based on field - programmalle gate array

Van Nghia Tran*

Moscow Institute of Physics and Technology

Received 1 December 2016; accepted 22 January 2017

Abstract:

This paper presents the implementation of RGB↔YCbCr color space conversion circuits (IP-cores) necessary for many video designs. These IP-cores offer simplified 3x3 matrix multipliers used to convert three input color samples (RGB or YCbCr) to three output samples (YCbCr or RGB). The optimized structure of IP-cores uses only four multipliers (or XtremeDSP slices) to implement the RGB↔YCbCr transformation by taking advantage of the dependencies between coefficients in the conversion matrix. Parameterizable IP-cores are used to confirm compliance to multi-standards and optimize hardware implementation. IP-cores have been verified successfully using xilinx vivado system generator 2016.2 for visually inspected output results and for verification against known models in the MATLAB/Simulink software environment.

Keywords: *color space conversion, field-programmalle gate array, IP-Core, RGB, Vivado System Generator, YCbCr.*

Classification number: *2.3*

Introduction

A "color space" is a mathematical representation of a set of colors. There are two popular color models: (1) RGB (red, green, and blue) or R'G'B' (gamma corrected RGB) which is often used in computer graphics, and (2) YCrCb (Y - luminance or brightness, Cr - chrominance of red and Cb - chrominance of blue) which is often used in video systems. These color spaces are directly related to the intuitive notions of hue, saturation and brightness. The convergence of computers, the Internet, and a wide variety of video devices, all using different color representations, is forcing the digital designers of today to convert between the two. Converters are useful for a number of purposes, including video image compression and transmission, such as with video/image compression JPEG and MPEG [1, 2] and in NTSC/PAL/SECAM television standards.

As far as we know, there are currently no publications about color space converters with multi-standard and adjustable parameters. There are, however, several methods for YCbCr↔RGB color space conversion [3-6]. There are also methods in the works [3, 4] implemented with the fixed ITU-R standard. In practice, image and video processing systems require the integration of many color space conversion standards and adjustable parameters which can be set by users, such as H.264 and H.265 encoders [1, 2]. Xilinx IP-cores [5, 6] provide four common format conversions as well as a custom mode for entering user-defined conversion matrices. Users can configure these using the GUI (Graphical user interface) and instantiate the core from the Vivado design tool. However, after configuring the cores and downloading bit files on the target device, the field-programmalle gate array (FPGA) operates with a fixed set of parameters, including a fixed ITU-R standard. This work aims at developing the universal color-space conversion IP-cores based on FPGA for a product with selectable parameters.

Color Spaces

RGB color space is used by computer displays. RGB corresponds most closely to the behavior of the human eye. RGB uses three numerical components to represent a color. This color space can be thought of as a three-dimensional coordinate system whose axes correspond to the three components, R (red), G (green), and B (blue). RGB is an additive color system. The three primary colors (red, green, and blue) are added together to form the desired color. Each component has a range of values from 0 to 255, with three '0' values producing black and three '255' values producing white.

Since all three components are highly correlated, RGB is not very efficient when dealing with real-world images. Therefore, compression standards employ the YCbCr color space, because human eyes are more sensitive to differences in brightness than differences in color, the reduction in bandwidth requirement seemed a valid trade for little or no visual difference.

The most frequently used color transform is defined by ITU-R recommendation BT.601 [7], BT.709 [8] and BT.2020

*Email: nghiamosmipt@gmail.com

[9] for SD, HD and UHD standards, respectively. Conversion between the YCbCr and RGB formats can be accomplished using the following equations [7-9]:

$$\begin{bmatrix} EY \\ ECr \\ ECb \end{bmatrix} = \begin{bmatrix} k_a & 1-k_a-k_b & k_b \\ k_c(1-k_a) & k_c(k_a+k_b-1) & -k_c k_b \\ -k_d k_a & k_d(k_a+k_b-1) & k_d(1-k_b) \end{bmatrix} \begin{bmatrix} ER \\ EG \\ EB \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} ER \\ EG \\ EB \end{bmatrix} = \begin{bmatrix} 1 & 1/k_c & 0 \\ 1 & \dfrac{-k_a}{k_c(1-k_a-k_b)} & \dfrac{-k_b}{k_d(1-k_a-k_b)} \\ 1 & 0 & 1/k_d \end{bmatrix} \begin{bmatrix} EY \\ ECr \\ ECb \end{bmatrix} \quad (2)$$

where, the signal values *ER, EG* and *EB* are normalized signals *R, G,* and *B* in the [0,1] range; coefficients k_a and k_b are chosen between 0 and 1, actual values for them differ slightly in different standards, as shown in Table 1; constants $k_c = 1/2(1-k_a)$ and $k_d = 1/2(1-k_b)$ allow dynamic range compression of chroma-components (Cb and Cr).

Both RGB and YCrCb color space are represented by a three dimensional, Cartesian coordinate system, as shown in Fig. 1.

Table 1. Parameterization Values for ITU-R Standards.

Standards	ITU-R BT.601	ITU-R BT.709	ITU-R BT.2020
k_a	0.299	0.2126	0.2627
k_b	0.114	0.0722	0.0593

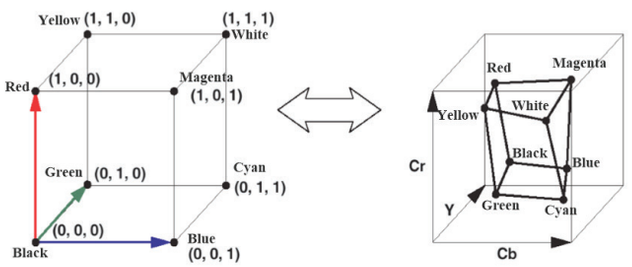

Fig. 1. RGB and YCbCr color representations.

Hardware Implementation

To map the hardware architecture, the color space transformation equations (1) and (2) can be expressed as:

$$\begin{cases} EY = k_a(ER-EG)+EG+k_b(EB-EG) \\ ECr = k_c(ER-EY) \\ ECb = k_d(EB-EY) \end{cases} \quad (3)$$

$$\begin{cases} ER = EY + 1/k_c * ECr \\ EG = EY + \dfrac{-k_a ECr}{k_c(1-k_a-k_b)} + \dfrac{-k_b ECb}{k_d(1-k_a-k_b)} \\ EB = EY + 1/k_d\, ECb \end{cases} \quad (4)$$

The decimal value of the quantized Y, Cr and Cb signals is:

$$\begin{cases} Y = \text{int}\left[(219EY+16).2^{n-8}\right] \\ Cr = \text{int}\left[(224ECr+128).2^{n-8}\right] \\ Cb = \text{int}\left[(224ECb+128).2^{n-8}\right] \end{cases} \quad (5)$$

$$\begin{cases} EY = \dfrac{Y-16.2^{n-8}}{219.2^{n-8}} \\ ECr = \dfrac{Cr-128.2^{n-8}}{224.2^{n-8}} \\ ECb = \dfrac{Cb-128.2^{n-8}}{224.2^{n-8}} \end{cases} \quad (6)$$

where, "*n*" denotes the bit length of the quantized signal. The operator int(.) returns the value of 0 for fractional parts in the range of 0 to 0.4999 and +1 for fractional parts in the range 0.5 to 0.999, i.e. it rounds to nearest integer.

In the case of a uniformly-quantized 8-bit binary encoding, Y is defined to have a nominal range of 16-235; Cb and Cr are defined to have a nominal range of 16-240.

The decimal value of the quantized R, G and B signals is:

$$R\,|\,G\,|\,B = \text{int}\left[\left(2^n-1\right)*ER\,|\,EG\,|\,EB\right] \quad (7)$$

Architecture of the color-space converters shows in Fig. 2. ROM memory stores coefficients k_a and k_b in different ITU-R BT standards. Signal "stand" selects stored coefficients k_a and k_b in ROM memory or user-defined coefficients. This architecture allows users to substitute values from any previous standard and new standard, which usually specify k_a and k_b values. This solution only requires four multipliers to replace the prior 3x3 matrix multiplier.

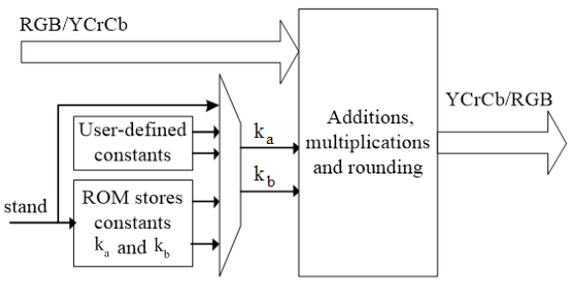

Fig. 2. Architecture of color-space conversion IP-cores.

Error Analysis

Errors are expected to occur from these calculations because of the nature of the quantification, rounding and fixed-point operators used. Thus, the results must be further analyzed to verify correct implementation for the conversion processes. The error analysis [10] allows for mean-square-error (MSE) calculations for the conversion from RGB to YCbCr and back to RGB. We assume that RGB data width is n_{RGB} bits, YCrCb data width is n_{YCC} bits, and coefficient precision is n_{Const} bits.

Practically, the YCbCr sample values are computed using the same number of bits ($n_{RGB} = n_{YCC} = n_b$) of precision with fixed-point RGB data.

IP-cores architecture contains three possible operators that might introduce noise.

1. The outputs of multipliers are truncated down to n_M bits to reduce the size and the carry-chain length of adders. Noise attributed to this truncation is inserted.

2. Word-length is further reduced to n_b bits at the output. The quantization noise is introduced.

3. Since the quantification, Cb and Cr may over - or under-flow, clipping noise gets inserted to the signal flow.

The first rounding noise source can be practically eliminated by the careful choice of n_M. Approximating SQNR by 6.02*n_M [dB], intuitively the rounding noise can be reduced by increasing n_M. However, n_M affects the resource usage and carry chain length in the design, thereby, affecting maximum speed.

Coefficients k_c and k_d in above equations allow designers to trade-off output quantization and clipping noise. Actual noise inserted depends on the probability statistics of the Cb and Cr variables, but in general if k_c and k_d are larger than the actual values in the standard, Cr and Cb output values may get larger (overflow) than the maximum or smaller (underflow) than the minimum values, which result in corrupted output values. If overflow occurs and the design does not have clipping, binary values wrap around and insert substantial noise into the output. Similarly, clamping is included in the design for underflow. If clamping/clipping is used, output values saturate and less noise is introduced, as shown in Fig. 3.

However, the lower k_c and k_d values are chosen, the worse the Cb and Cr values will use for the available dynamic range, thus introducing more quantization noise. Therefore, the designer's task is to equalize output quantization and clipping noise insertion by carefully choosing k_c and k_d values knowing the statistics of Cb and Cr values.

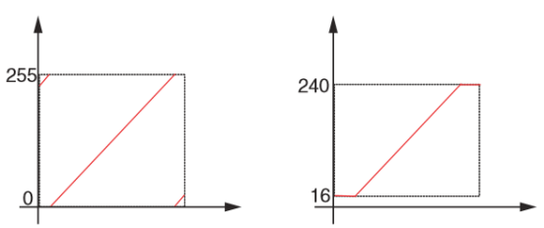

Fig. 3. Wraparound and Saturation.

The rounding operations at the output of converters introduce an approximate MSE of 1/12 due to uniform rounding quantization error [10]. When RGB values are in the (0, 255) range. For sake of simplicity, choose $n_b = 8$. For the cascade of the two conversion processes, MSEs for red (MSEr), green (MSEg) and blue (MSEb) signals are defined as:

$$\begin{cases} MSEr = \frac{1}{12}\left[1+\left(\frac{2*255(1-k_a)}{224}\right)^2+\left(\frac{255}{219}\right)^2\right] \\ MSEb = \frac{1}{12}\left[1+\left(\frac{2*255(1-k_b)}{224}\right)^2+\left(\frac{255}{219}\right)^2\right] \\ MSEg = \frac{1}{12}\left[1+\left(\frac{2*255k_a(1-k_a)}{224(1-k_a-k_b)}\right)^2+\left(\frac{2*255k_b(1-k_b)}{224(1-k_a-k_b)}\right)^2+\left(\frac{255}{219}\right)^2\right] \end{cases} \quad (8)$$

Peak signal-to-noise ratio (PSNR) is defined via MSE as:

$$PSNR = 10\log_{10}\left(\frac{(2^{n_b}-1)^2}{MSE}\right) \quad (9)$$

Using the ITU-R Rec. BT.709 standard, $k_a = 0.2126$, $k_b = 0.0722$. We obtain:

$$\begin{cases} MSEr = 0.464,\ PSNRr = 51.4\ dB \\ MSEb = 0.568,\ PSNRb = 50.6\ dB \\ MSEg = 0.223,\ PSNRg = 54.6\ dB \end{cases}$$

Note that the Green component, which is the most perceptually important component, the PSNR, is significantly higher than for the Red and Green components. Also note that all three PSNRs exceed 50 dB. Value of the error of Blue signal is the most maximum.

Experimental results

FPGA Implementation of the color-space conversion IP-cores presented in Fig. 4 is composed of two IP-cores: RGB2YCrCb IP-core converts the RGB inputs to YCrCb, and YCrCb2RGB IP-core converts YCbCr signals back to RGB. The original and converted images and resulting conversion errors are plotted based on MATLAB/Simulink tools.

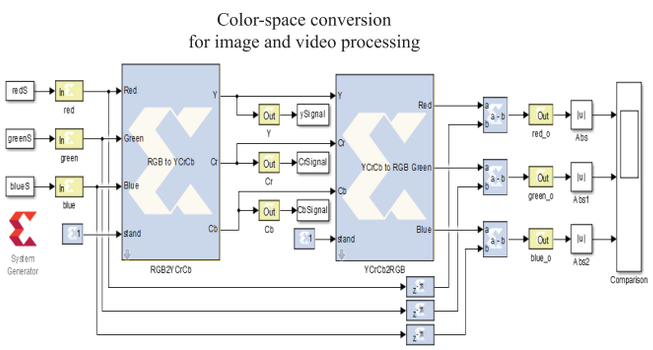

Fig. 4. System Generator Test bench of color-space conversion IP-cores.

The system generator instance can be parameterized through a Graphical User Interface (GUI) activated by double clicking the IP-cores as illustrated in Fig. 5. The designer sets the required input/output precision, enters parameters for k_a and k_b to implement a custom designed converter standard and sets the value for "stand" port to select between a predefined standard and custom.

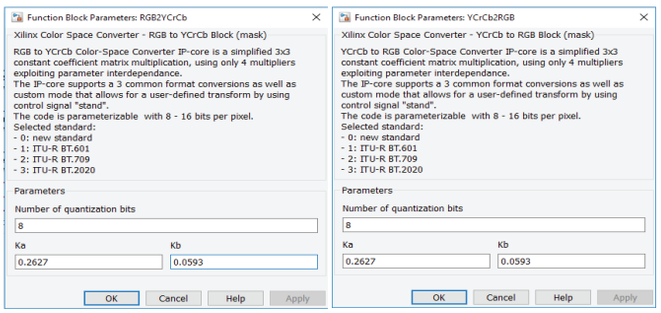

Fig. 5. Setting the configuration parameters for IP-cores.

Original RGB images Lena and Xilinx Logo are used for the inputs to test and evaluate the efficiency of the IP-cores. A digital image consists of a two-dimensional array of color pixels. Each pixel is a sample in time composed of R, G, and B components. RGB data of the original images and YCrCb samples after converting are displayed by using MATLAB/ Simulink tools as plotted in Fig. 6. Total errors of the two conversion processes for R, G, B samples are also calculated, as seen on Fig. 7.

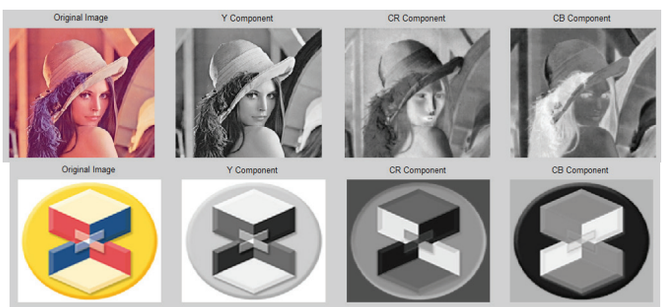

Fig. 6. Original images and Y, Cr, Cb outputs.

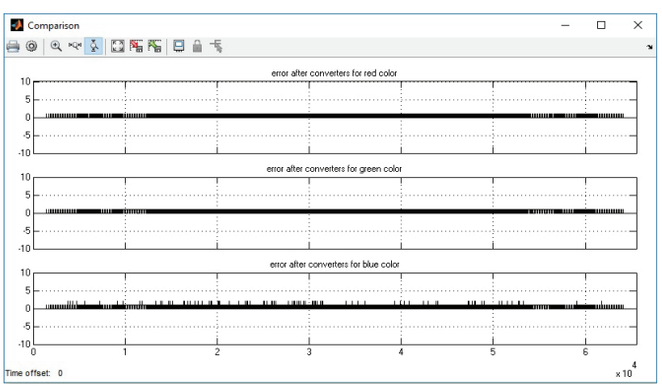

Fig. 7. Error statistics of the R, G, B samples.

According to the experimental results, the maximal error of both conversion processes for the Red and Green samples is only 1 ($2*MSEr = 2*0.464 < 1$ and $2*MSEg = 2*0.223 < 1$), and for the Blue samples is 2 ($2*MSEb = 2*0.568 < 2$). These results are correct in the according to the above error analysis. Therefore, the conversion precision of the proposed IP-cores is high enough for the video/imaging applications.

Conclusions

FPGA is widely used in video image processing applications, and it's important to speed up the RGB↔YCrCb color space conversion on FPGA. In this paper, the novel color space conversion IP-cores were presented and demonstrated by using the Vivado System Generator. FPGA hardware with these parameterizable IP-cores can allow users to change the parameters and can adapt between a predefined standard and a custom designed converter standard. Experimental results show that the conversion precision of the proposed IP-cores is high enough for the video/imaging applications.

In future work, we intend to consider a conversion between chroma sub-sampling formats (Y:Cr:Cb) of 4:4:4, 4:2:2, and 4:2:0. Since the human visual system is more sensitive to the luminance component than chrominance ones, we can subsample Cr and Cb to reduce the data amount without sacrificing video quality to provide a simple and effective video compression to reduce storage and transmission costs.

REFERENCES

[1] ITU (2016), *Advanced video coding for generic audiovisual services,* Recommendation ITU-T H.264.

[2] ITU (2016), *High efficiency video coding,* Recommendation ITU-T H.265.

[3] Juan Xue, Xudong Cao (2012), "Color Space Conversion Based on FPGA", *IEEE International Conference on Computer Science and Automation Engineering*, **2**, pp.422-425.

[4] Hongxu Jiang, Hanqing Li, Tingshan Liu, Ping zhang, Jinyuan Lu (2013), "A Fast Method for RGB to YCrCb Conversion Based on FPGA", *IEEE International Conference on Computer Science and Network Technology*, **2**, pp.588-591.

[5] Xilinx (2015), *RGB to YCrCb Color-Space Converter v7.1*, LogiCORE IP Product Guide.

[6] Xilinx (2015), *YCrCb to RGB Color-Space Converter v7.1*, LogiCORE IP Product Guide.

[7] ITU (2011), *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*, Recommendation ITU-R BT.601-7.

[8] ITU (2015), *Parameter values for the HDTV standards for production and international program exchange*, Recommendation ITU-R BT.709-6.

[9] ITU (2015), *Parameter values for ultra-high definition television systems for production and international program exchange*, Recommendation ITU-R BT.2020-2.

[10] Gary Sullivan (2005), *Approximate theoretical analysis of RGB to YCbCr to RGB conversion error*, in Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6).