



XAPP930 (v1.0.1) August 27, 2007

## Color-Space Converter: RGB to YCrCb

Author: Gabor Szedo

### Summary

This application note describes the implementation of an RGB color space to a YCbCr color space conversion circuit necessary in many video designs. The reference design files include RTL VHDL code defining an optimized structure using only four multipliers to implement the RGB to YCrCb transformation. Source files in compilation order are:

1. GenXlib\_util.vhd,
2. GenXlib\_arch.vhd,
3. color\_space\_pkg.vhd,
4. Xil\_RGB2YCrCb.vhd

A System Generator token encapsulating the HDL code is also available for System Generator users. A System Generator testbench is also provided to visually inspect output results. The code is parameterizable for the input/output precision (8 bit or 10 bit), internal word-length, and coefficient precision (8 to 18 bits have been defined). Typical scaling, offset, clipping, and clamping parameters are supplied for many standards.

### Introduction

The reference design has a fully synchronous interface through the CE, CLK and SCLR ports. Ports R,G, and B are the RGB color-space inputs; Y, Cr, and Cb are the YCrCb color-space outputs. See [Figure 1](#).

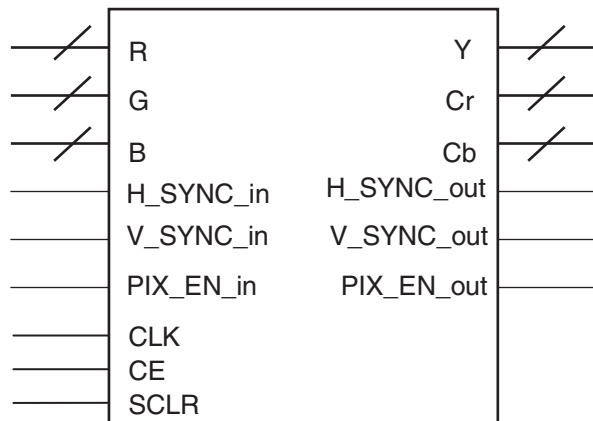


Figure 1: RGB to YCrCb Pinout

To facilitate easy insertion to practical video systems, the reference design takes up to three stream control signals (H\_SYNC, V\_SYNC, and PIX\_EN) and delays them appropriately, so that control signals can be easily synchronized with the output stream. The reference design does not use the control signals; therefore, connecting these signals is optional.

## Parameterization

The design input parameters are listed in [Table 1](#).

**Table 1: Design Parameters**

| Design parameter | Type    | Range                           | Notes  |
|------------------|---------|---------------------------------|--|
| FAMILY_HAS_MAC   | Integer | 0, 1                            | 1, if target family has DSP48 like MAC unit <sup>(1)</sup>               |
| FABRIC_ADDS      | Integer | 0,1                             | 1, if adders/subtractors should be implemented in fabric. <sup>(1)</sup> |
| IWIDTH           | Integer | 8,10                            | Input (RGB) data width   |
| CWIDTH           | Integer | 8 to 18                         | Coefficient data width   |
| MWIDTH           | Integer | $IWIDTH+1$ to 18                | Embedded multiplier width <sup>(2)</sup>                                 |
| OWIDTH           | Integer | 8, 10                           | Output (YCrCb) data width  |
| YMAX             | Integer | 0 to $2^{OWIDTH-1}$             | Clipping value for the luma (Y) output                                   |
| YMIN             | Integer | 0 to $2^{OWIDTH-1}$             | Clamping value for the luma (Y) output                                   |
| CMAX             | Integer | 0 to $2^{OWIDTH-1}$             | Clipping value for the chroma (Cr, Cb) outputs                           |
| CMIN             | Integer | 0 to $2^{OWIDTH-1}$             | Clamping value for the chroma (Cr, Cb) outputs                           |
| YOFFSET          | Integer | 0 to $2^{OWIDTH-1}$             | Offset value for the luma (Y) output                                     |
| COFFSET          | Integer | 0 to $2^{OWIDTH-1}$             | Offset value for the chroma (Cr, Cb) outputs                             |
| ACOEf            | Integer | $-2^{CWIDTH}$ to $2^{CWIDTH-1}$ | Coefficient A value <sup>(3)</sup>                                       |
| BCOEf            | Integer |                                 | Coefficient B value <sup>(3)</sup>                                       |
| CCOEf            | Integer |                                 | Coefficient C value <sup>(3)</sup>                                       |
| DCOEf            | Integer |                                 | Coefficient D value <sup>(3)</sup>                                       |
| HAS_CLIP         | Integer | 0,1                             | 1 if outputs have clipping logic <sup>(4)</sup>                          |
| HAS_CLAMP        | Integer | 0,1                             | 1 if outputs have clamping logic <sup>(4)</sup>                          |

### Notes:

1. Refer to [Figure 3](#) for more information.
2. Refer to section ““Error Analysis”” for more information.
3. Refer to section ““Assigning Values to Design Parameters”” for more information.
4. Refer to section ““Output Clipping Noise”” for more information.

## Detailed Operation

### Color Spaces

A color space is a mathematical representation of a set of colors. The three most popular color models are:

- RGB (used in computer graphics), R'G'B' (gamma corrected RGB)
- YIQ, YUV, YCrCb used in video systems
- CMYK (used in color printing).

However, none of these color spaces are directly related to the intuitive notions of hue, saturation, and brightness.

All color spaces can be derived from the RGB information supplied by devices such as cameras and scanners.

Different color spaces have historically evolved for different applications. In each case, a color space was chosen for application specific reasons. A certain choice was better because it required less storage, bandwidth, or computation in analog or digital domains.

Whatever historical reasons caused color space choices in the past, the convergence of computers, the Internet, and a wide variety of video devices, all using different color representations, is forcing the digital designer today to convert between them. The objective is to have all inputs converted to a common color space before algorithms and processes are executed. Converters are useful for a number of markets, including image and video processing. This application note describes one such conversion.

## RGB Color Space

The red, green, and blue (RGB) color space is widely used throughout computer graphics. Red, green, and blue are three primary additive colors: individual components are added together to form a desired color and are represented by a three dimensional, Cartesian coordinate system [Figure 2](#).

[Table 2](#) contains the RGB values for 100% amplitude, 100% saturated color bars, a common video test signal [\[Ref 1\]](#). The RGB color space is the most prevalent choice for computer graphics, because color displays use red, green, and blue to create the desired color. Therefore, the choice of the RGB color-space simplifies the architecture and design of the system. Also, a system that is designed using the RGB color space can take advantage of a large number of existing software algorithms, since this color space has been around for a number of years.

However, RGB is not very efficient when dealing with real-world images. All three components need to be of equal bandwidth to generate any color within the RGB color cube. Also, processing an image in the RGB color space is usually not the most efficient method. For example, to modify the intensity or color of a given pixel, all three RGB values must be read, modified and written back to the frame buffer. If the system had access to the image stored in the intensity and color format, these processing steps would be faster.

*Table 2: 100% RGB Color Bars*

|   | Nominal Range | White | Yellow | Cyan | Green | Magenta | Red | Blue | Black |
|---|---------------|-------|--------|------|-------|---------|-----|------|-------|
| R | 0 to 255      | 255   | 255    | 0    | 0     | 255     | 255 | 0    | 0     |
| G | 0 to 255      | 255   | 255    | 255  | 255   | 0       | 0   | 0    | 0     |
| B | 0 to 255      | 255   | 0      | 255  | 0     | 255     | 0   | 255  | 0     |

## R' G' B' Color Space

While the RGB color-space is ideal to represent computer graphics, 8-bit linear-light coding performs poorly for images to be viewed [\[Ref 2\]](#). 12 or 14 bits per component are necessary to achieve excellent quality. The best perceptual use is made of a limited number of bits by using nonlinear coding that mimics the nonlinear lightness response of human vision. In video JPEG, MPEG, computing and digital still photography, a nonlinear transfer function is applied to the RGB signals to give nonlinearly coded gamma-corrected components, denoted with symbols R'G'B'. Excellent image quality can be obtained with 10-bit nonlinear coding with a transfer function similar to that of Rec. 709 [\[Ref 4\]](#) or sRGB.

## YUV Color Space

The YUV color space is used by the PAL, NTSC, and SECAM color video/TV standards. Black-and-white systems used only the luma (Y) information. Chroma information (U and V) was added in such a way that a black-and-white receiver would still display a normal black-and-white picture.

## YCrCb (or YCbCr) Color Space

The YCbCr color space was developed as part of the ITU-R BT.601 [Ref 3] during the development of a world-wide digital component video standard. YCbCr is a scaled and offset version of the YUV color space. Y is defined to have a nominal range of 16-235; Cb and Cr are defined to have a nominal range of 16-240. There are several YCbCr sampling formats, such as 4:4:4, 4:2:2, and 4:2:0.

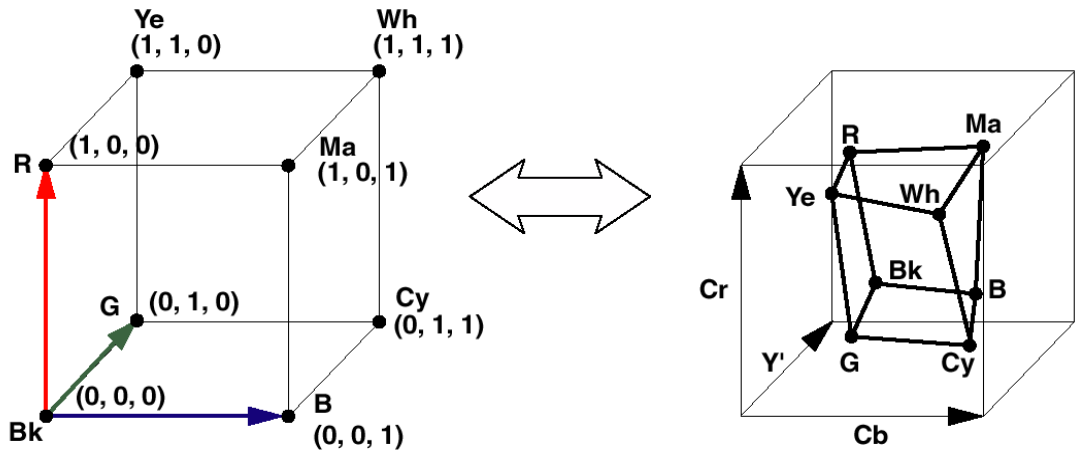


Figure 2: RGB and YCrCb Color Representations

## Conversion Equations

### Derivation of Conversion Equations

To generate the Luma (Y, or gray value) component, biometric experiments were employed to measure how the human eye perceives the intensities of the Red, Green, and Blue colors. Based on these experiments, optimal values for coefficients CA and CB were determined such that:

$$Y = CA * R + (1 - CA - CB) * G + CB * \tag{Equation 1}$$

Actual values for CA and CB differ slightly in different standards.

Conversion from the RGB color space to Luma and Chroma (differential color components) could be described with the following equations:

$$\begin{bmatrix} Y \\ B - Y \\ R - Y \end{bmatrix} = \begin{bmatrix} CA & 1 - CA - CB & CB \\ -CA & CA + CB - 1 & 1 - CB \\ 1 - CA & CA + CB - 1 & -CB \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{Equation 2}$$

Coefficients CA, CB, and 1-CA-CB are chosen between 0 and 1, which guarantees that the range of Y is constrained between RGB<sub>min</sub> and RGB<sub>max</sub>.

However, the minimum and maximum values of B-Y is:

$$\min_{B-Y} = RGB_{min} - (CA * RGB_{max} + (1 - CA - CB) * RGB_{max} + CB * RGB_{min}) = -(1 - CB) * (RGB_{max} - RGB_{min})$$

$$\max_{B-Y} = RGB_{max} - (CA * RGB_{min} + (1 - CA - CB) * RGB_{min} + CB * RGB_{max}) = (1 - CB) * (RGB_{max} - RGB_{min}),$$

Thus, the range of B-Y is 2(1-CB) (RGB<sub>max</sub> - RGB<sub>min</sub>).

Similarly, the minimum and maximum values of R-Y is:

$$\min_{R-Y} = RGB_{min} - (CA * RGB_{min} + (1 - CA - CB) * RGB_{max} + CB * RGB_{max}) = -(1 - CA) * (RGB_{max} - RGB_{min})$$

$$\max_{R-Y} = RGB_{max} - (CA * RGB_{max} + (1 - CA - CB) * RGB_{min} + CB * RGB_{min}) = (1 - CA) * (RGB_{max} - RGB_{min}),$$

Thus, the range of R-Y is 2(1-CA) (RGB<sub>max</sub> - RGB<sub>min</sub>).

In a practical implementation, the range of the luma and chroma components should be equal. There are two ways to accomplish this. The chroma components ( $B-Y$  and  $R-Y$ ) can be normalized (compressed and offset compensated), or values above and below the luma range can be clipped.

Both clipping and dynamic range compression and requantization results in loss of information. However, the effects are different. To leverage differences in the input (RGB) range, different standards choose different tradeoffs between clipping and normalization.

The RGB to YCrCb application supports only the conversions that fits the following general form:

$$\begin{bmatrix} Y \\ C_R \\ C_B \end{bmatrix} = \begin{bmatrix} CA & 1 - CA - CB & CB \\ CC(-CA) & CC(CA + CB - 1) & CC(1 - CB) \\ CD(1 - CA) & CD(CA + CB - 1) & CD(-CB) \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} O_Y \\ O_C \\ O_C \end{bmatrix} \quad \text{Equation 3}$$

where  $CC$  and  $CD$  allows dynamic range compression for  $B-Y$  and  $R-Y$ , and constants  $O_Y$  and  $O_C$  facilitate offset compensation for the resulting  $C_B$  and  $C_R$ . To constrain resulting chroma components ( $C_B$  and  $C_R$ ) into the  $[0..1]$  range,  $OC = 0.5$ ,

$$CC = \frac{1}{2(1 - CB) * (RGB_{MAX} - RGB_{MIN})} \quad \text{Equation 4}$$

$$CD = \frac{1}{2(1 - CA) * (RGB_{MAX} - RGB_{MIN})} \quad \text{Equation 5}$$

When  $RGB$  values are also in the  $[0..1]$  range,  $OC = 0.5$ ,

$$CC = \frac{1}{2(1 - CB)} \quad CD = \frac{1}{2(1 - CA)} \quad \text{Equation 6}$$

avoids arithmetic under- and overflows.

## Assigning Values to Design Parameters

The following sections specify parameter values for some widely used standards. Most parameter values, except for  $COEFA$ ,  $COEFB$ ,  $COEFC$  and  $COEFD$  can be assigned from the table directly. Real-valued coefficients  $CA$ ,  $CB$ ,  $CC$  and  $CD$ , however first have to be scaled to the precision specified by  $CWIDTH$ , using the following equations:

$$COEFA = [CA * 2^{CWIDTH-1}]$$

$$COEFB = [CB * 2^{CWIDTH-1}]$$

$$COEFC = [CC * 2^{CWIDTH-1}]$$

$$COEFD = [CD * 2^{CWIDTH-1}]$$

Where  $[ ]$  denotes rounding to nearest integer.

## ITU 601 (SD) and 709 - 1125/60 (NTSC)

Table 3: Parameterization Values for the 601 and NTSC 709 Standards

| Coefficient/<br>Parameter | Range                |                      |                      |
|---------------------------|----------------------|----------------------|----------------------|
|                           | 16-240               | 16-235               | 0-255                |
| CA                        | 0.299                |                      | 0.2568               |
| CB                        | 0.114                |                      | 0.0979               |
| CC                        | 0.564                | 0.5772               |                      |
| CD                        | 0.713                | 0.7295               | 0.5910               |
| YOFFSET                   | $2^{OWIDTH-4}$       |                      |                      |
| COFFSET                   | $2^{OWIDTH-1}$       |                      |                      |
| HAS_CLIP                  | 1                    |                      | 0                    |
| HAS_CLAMP                 | 1                    |                      | 0                    |
| YMAX                      | $240 * 2^{OWIDTH-8}$ | $235 * 2^{OWIDTH-8}$ | $255 * 2^{OWIDTH-8}$ |
| CMAX                      | $240 * 2^{OWIDTH-8}$ | $235 * 2^{OWIDTH-8}$ | $255 * 2^{OWIDTH-8}$ |
| YMIN                      | $16 * 2^{OWIDTH-8}$  |                      | 0                    |
| CMIN                      | $16 * 2^{OWIDTH-8}$  |                      | 0                    |

## Standard ITU 709 (HD) 1250/50 (PAL)

Table 4: Parameterization Values for the PAL 709 Standard

| Coefficient/<br>Parameter | Input range          |                      |                      |
|---------------------------|----------------------|----------------------|----------------------|
|                           | 16-240               | 16-235               | 0-255                |
| CA                        | 0.2126               |                      | 0.1819               |
| CB                        | 0.0722               |                      | 0.0618               |
| CC                        | 0.5389               | 0.5512               |                      |
| CD                        | 0.6350               | 0.6495               | 0.6495               |
| YOFFSET                   | $2^{OWIDTH-4}$       |                      |                      |
| COFFSET                   | $2^{OWIDTH-1}$       |                      |                      |
| HAS_CLIP                  | 1                    |                      | 0                    |
| HAS_CLAMP                 | 1                    |                      | 0                    |
| YMAX                      | $240 * 2^{OWIDTH-8}$ | $235 * 2^{OWIDTH-8}$ | $255 * 2^{OWIDTH-8}$ |
| CMAX                      | $240 * 2^{OWIDTH-8}$ | $235 * 2^{OWIDTH-8}$ | $255 * 2^{OWIDTH-8}$ |
| YMIN                      | $16 * 2^{OWIDTH-8}$  |                      | 0                    |
| CMIN                      | $16 * 2^{OWIDTH-8}$  |                      | 0                    |

## YUV Standard

Table 5: Parameterization Values for the YUV Standard

| Coefficient/Parameter | Value                |
|-----------------------|----------------------|
| CA                    | 0.299                |
| CB                    | 0.114                |
| CC                    | 0.492111             |
| CD                    | 0.877283             |
| YOFFSET               | $2^{OWIDTH-4}$       |
| COFFSET               | $2^{OWIDTH-1}$       |
| HAS_CLIP              | 1                    |
| HAS_CLAMP             | 1                    |
| YMAX                  | $240 * 2^{OWIDTH-8}$ |
| CMAX                  | $240 * 2^{OWIDTH-8}$ |
| YMIN                  | $16 * 2^{OWIDTH-8}$  |
| CMIN                  | $16 * 2^{OWIDTH-8}$  |

## Application Schematic

The color-space transformation equations

$$Y = ACOEF * (R - G) + G + BCOEFF * (B - G) + YOFFSET$$

$$Cb = CCOEF * (B - Y) + COFFSET$$

$$Cr = DCOEF * (R - Y) + COFFSET$$

can be directly mapped to the following architecture shown in Figure 3.

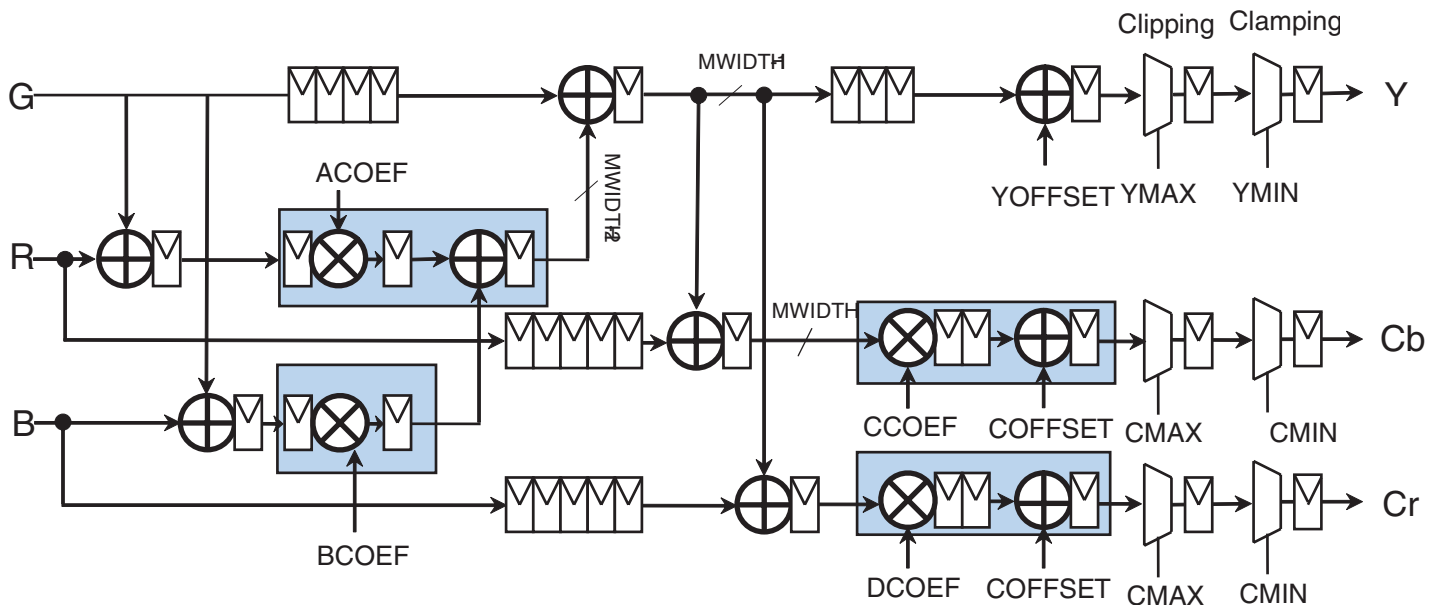


Figure 3: Application Schematic

The blue boxes present logic blocks, which are always implemented using DSP blocks, if DSP blocks are available in the target device. When targeting Virtex™-4 devices, set parameter *FAMILY\_HAS\_MAC* to 1. Setting parameter *FABRIC\_ADDS=1* also forces other arithmetic components in the design into DSP Blocks.

## Error Analysis

The following analysis, based on DSP fundamentals [Ref 5], presents mean-square-error (MSE) calculations for RGB to YCrCb, assuming  $IWIDTH$  bit RGB input data,  $OWIDTH$  bit wide YCrCb output data, and  $CWIDTH$  bits for coefficient precision. [Ref 6] arrives to similar results for fixed coefficient values and input and output representations.

Taking rounding/quantization into account, the structure illustrated on Figure 3 implements the following equations:

$$Y_{RAW} = [ACOEFF \cdot (R - G) + BCOEFF \cdot (B - G)]_{MWIDTH-2} + G \quad \text{Equation 7}$$

$$Y = [Y_{RAW}]_{OWIDTH} + YOFFSET \quad \text{Equation 8}$$

$$Cb = [CCOE \cdot (B - Y_{RAW})]_{OWIDTH} + COFFSET \quad \text{Equation 9}$$

$$Cr = [DCOE \cdot (R - Y_{RAW})]_{OWIDTH} + COFFSET \quad \text{Equation 10}$$

where  $[ ]_k$  denotes rounding to  $k$  bits. The architecture contains three possible operators that might introduce noise. Quantization noise is inserted when data is rounded.

1. Data is rounded to  $MWIDTH-2$  bits after calculating  $Y_{raw}$ .
2. Data is rounded to  $OWIDTH$  bits at the output.
3. If  $CCOE$  and  $DCOE$  are chosen such, that  $Cb$  and  $Cr$  may over, or underflow, clipping noise gets inserted to the signal flow.

Before analyzing the effects of these noise sources, first look at the input Signal to Quantization Noise Ratio (SQNR). Assuming uniformly distributed quantization error,

$$SQNR_{RGB} = 10 \log \frac{P_x}{P_N} = 10 \log \frac{\int_{RGBMIN}^{RGBMAX} x^2 dx}{\int_{-\Delta/2}^{\Delta/2} e^2 dx} \quad \text{Equation 11}$$

Substituting  $LSB = 2^{-INBITS}$ , where  $INBITS$  is the input (RGB) precision,  $SQNR_{RGB}$  becomes a function of the input dynamic range. In the next three calculations, when calculating  $SQNR_{RGB}$  for the typical dynamic ranges,  $INBITS = 8$  for all three cases.

When RGB values are in the **(0, 255) range**:

$$SQNR_{RGB} = 10 \log \frac{\frac{1}{255} \int_0^{255} x^2 dx}{\int_{-1/2}^{1/2} x^2 dx} = 10 \log \frac{\frac{1}{3 \cdot 255} [255^3]}{\frac{1}{12}} = 54.15 \text{ dB} \quad \text{Equation 12}$$

when RGB values are in the **(16, 240) range**:

$$SQNR_{RGB} = 10 \log \frac{\frac{1}{224} \int_{16}^{240} x^2 dx}{\int_{-1/2}^{1/2} x^2 dx} = 53.92 \text{ dB} \quad \text{Equation 13}$$

and when RGB values are in the **(16, 235)** range:

$$SQNR_{RGB} = 10 \log \frac{\frac{1}{219} \int_{16}^{235} x^2 dx}{\int_{(-1)/2}^{1/2} x^2 dx} = 53.74 \text{ dB} \quad \text{Equation 14}$$

The first rounding noise source can be practically eliminated by the careful choice of *MWIDTH*. Approximating SQNR by  $6.02 \text{ MWIDTH}$  [dB], intuitively the rounding noise can be reduced by increasing *MWIDTH*. However, *MWIDTH* affects the resource usage and carry chain length in the design (thereby, affecting maximum speed). Choosing *MWIDTH* > 18 significantly increase the dedicated multiplier count of the design.

Therefore, optimal *MWIDTH* values, in the *IWIDTH*+4 to 18 range, do not significantly increase resource counts but assure that quantization noise inserted is negligible (at least 20 dB less than the input noise).

## Output Quantization Noise

Coefficients *CC* and *CD* in Equation 3 allow standard 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 *CC* and *CD* are larger than the maximum values calculated in Equation 4 and Equation 5, output values may clip, introducing clipping noise. However, the lower *CC* and *CD* values are chosen, the worse *Cb* and *Cr* values will utilize 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 *CC* and *CD* values knowing the statistics of *Cb* and *Cr* values. For instance, when probabilities of extreme chrominance values are very small, it can be beneficial to increase *CC* and *CD* values as the extra noise inserted by occasional clipping is less than the gain in average signal power (and thus SQNR).

Though a quantitative noise analysis of the signal flow graph based on Figure 3 is possible by replacing quantizers with appropriate AWGN sources, the complexity of the derivation of a final noise formula, which addresses clipping noise as well, is beyond the scope of this document. Instead, Table 6 illustrates noise figures for some typical (Table 3) parameter combinations.

Table 6: *Input and Output SNR measurement results [dB] for ITU-REC 601 (SD)*

| SNR                        | <i>IWIDTH</i> = <i>OWIDTH</i> =<br>8 Bits | <i>IWIDTH</i> = <i>OWIDTH</i> =<br>10 Bits | Input Range                                  |
|----------------------------|---|--|--|
| SNR <sub>RGB</sub> (input) | 54.1                                      | 66.2                                       | [0..255] (8bit)<br>Or<br>[0..1023] (10 bit)  |
| SNR <sub>Y</sub>           | 51.9                                      | 64.0                                       |  |
| SNR <sub>Cr</sub>          | 47.0                                      | 58.9                                       |  |
| SNR <sub>Cb</sub>          | 47.0                                      | 58.9                                       |  |
| SNR <sub>RGB</sub> (input) | 54.0                                      | 65.9                                       | [16..240] (8bit)<br>Or<br>[64..960] (10 bit) |
| SNR <sub>Y</sub>           | 51.8                                      | 63.9                                       |  |
| SNR <sub>Cr</sub>          | 46.9                                      | 58.8                                       |  |
| SNR <sub>Cb</sub>          | 46.9                                      | 58.8                                       |  |
| SNR <sub>RGB</sub> (input) | 53.8                                      | 65.8                                       | [16..235] (8bit)<br>Or<br>[64..920] (10 bit) |
| SNR <sub>Y</sub>           | 51.5                                      | 63.6                                       |  |
| SNR <sub>Cr</sub>          | 46.9                                      | 58.8                                       |  |
| SNR <sub>Cb</sub>          | 46.9                                      | 58.8                                       |  |

## Output Clipping Noise

If coefficients  $CC$  and  $CD$  in Equation 3 are larger than the maximum values calculated in Equation 4 and Equation 5,  $C_r$  and  $C_b$  output values may get larger (overflow) than the maximum or smaller (underflow) than minimum value the output representation can carry. If overflow occurs and the design does not have clipping logic ( $HAS\_CLIPPING=0$ ), binary values wraparound inserting substantial noise to the output. If  $HAS\_CLIPPING=1$ , output values saturate, introducing less noise (Figure 4).

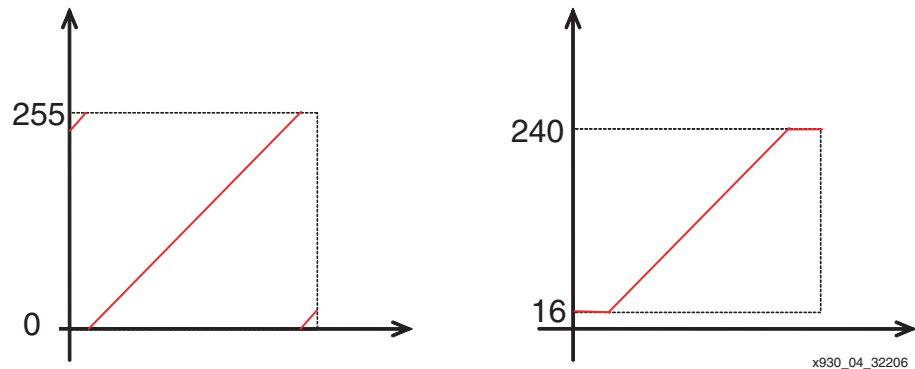


Figure 4: Wraparound and Saturation

Similarly, clamping logic is included in the design, if  $HAS\_CLAMPING=1$ . Use of clipping and clamping increases slice count of the design by approximately  $6OWIDTH$  slices.

If a targeted standard limits output of values to a predefined range other than those of binary representation, such as ITU-R BT.601-5 [Ref 3], use of clipping and clamping logic facilitates constraining output values to the predefined range by setting  $YMAX$  and  $YMIN$  values (constraining luminance), as well as  $CMAX$  and  $CMIN$  values (constraining chrominance) values according to the standard specifications.

## Performance, Latency, and Resource Estimation

The design was tested using ISE 8.1 tools with default options for characterization data. For *xst*, optimization goal was set to area. For map, the global optimization and timing driven packing options were turned on.

For Virtex-4 testing, an XC4VSX35 part with -10 speed grade and FF668 packaging were selected.

Table 7: Performance and Resource Estimation for Virtex-4 Devices

|                             |         |
|-----------------------------|---------|
| Maximum Operating Frequency | 264 MHz |
| Number of Slice Flip Flops  | 226     |
| Number of 4 Input LUTs      | 121     |
| Number of Occupied Slices   | 189     |
| Number of DSP48s            | 4       |

For Spartan™-3 testing, an XC3S1000 part with -4 speed grade and FG320 packaging were selected.

*Table 8: Performance and Resource Estimation for Spartan-3 Devices*

|                                    |         |
|------------------------------------|---------|
| <b>Maximum Operating Frequency</b> | 185 MHz |
| <b>Number of Slice Flip Flops</b>  | 338     |
| <b>Number of 4 Input LUTs</b>      | 139     |
| <b>Number of Occupied Slices</b>   | 194     |
| <b>Number of MULT18x18s</b>        | 4       |

To calculate the exact latency of the module, a support function:

```

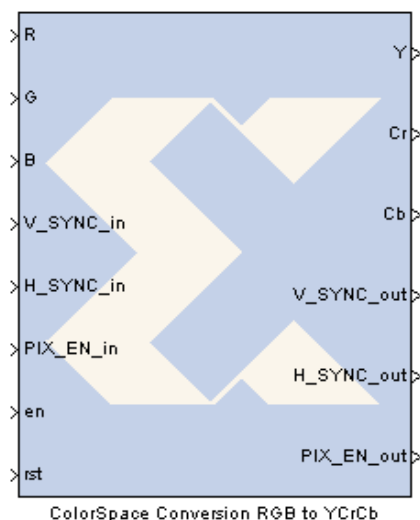
RGB2YCrCb_LATENCY(FAMILY_HAS_MAC, FABRIC_ADDS, HAS_CLIP, HAS_CLAMP:
integer)
return integer

```

is included in `color_space_pkg.vhd`.

## System Generator Token

To facilitate easy integration of the RGB to YCrCb design into a complex system developed using System Generator, a token encapsulating the VHDL code is supplied (Figure 5).



*Figure 5: System Generator Token*

Files necessary for the system generator token to work correctly must be copied to the MATLAB working directory are:

- `Xil_RGB2YCrCb_config.m`
- `rgb2ycrcb_action.m`
- `rgb2ycrcb_enablement.m`
- `Xil_RGB2YCrCb_GUI.xml`

The system generator instance can be parameterized through a Graphical User Interface (GUI) activated by double clicking the token. If the RGB to YCrCb conversion implements one of the widespread standards described in section “Assigning Values to Design Parameters,” then picking the standard on the Basic tab of the GUI (Figure 6) and setting the required input/output precision is all the configuration that needs to be done.

If the user wants to implement a custom designed converter or a standard, which is not listed in the drop-down box, choosing Custom as a standard allows editing contents of the Advanced tab (Figure 7). Conversion Matrix controls allow entering parameters for  $CA$ ,  $CB$ ,  $CC$ , and  $CD$  introduced in section “Conversion Equations” directly in floating point format (range [0..1]).

Offset compensation, clipping, and clamping settings are identical to those of the VHDL parameters, discussed in detail in section “Parameterization.”

Values in the Advanced panel are initialized with values corresponding to the standard selected before modifying standard selection to Custom. After custom configuration options are set, the GUI keeps the user-defined values, even if the standard-selection is changed back and forth between a predefined and custom standard.

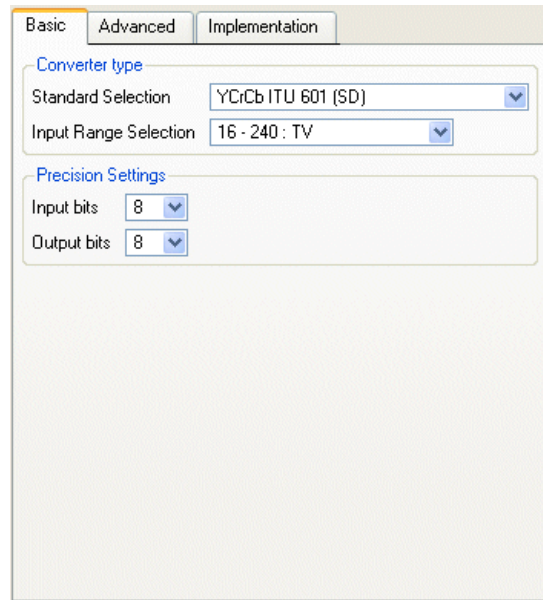


Figure 6: System Generator GUI, Basic Tab

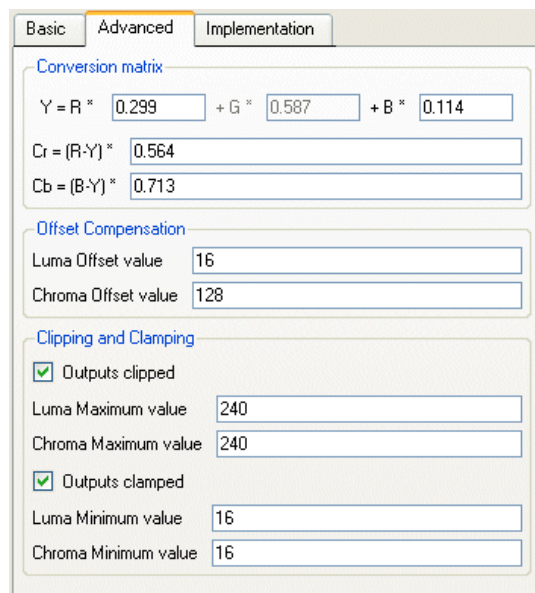


Figure 7: System Generator GUI, Advanced Tab

The third tab, Implementation (Figure 8), allows setting specific options governing the layout of the design. Both coefficient width and multiplier input bits affect the resource usage and noise properties of the design. Predefined values offer maximum SNR without increasing the DSP48 count of the design. Lowering multiplier input bit width results in savings in slice count.

When the design is implemented in a target chip that contains DSP48s, the four multipliers and three adders immediately following them, as well as the pipeline registers, are implemented using the DSP48s. Logic that gets implemented in DSP48s when DSP48s are available are marked with light-blue background in Figure 3.

The *Use fabric for adders* checkbox, corresponding to the *FABRIC\_ADDS* VHDL parameter, controls whether other adders in the design get implemented in logic fabric (using slice-based logic) or in DSP48s.

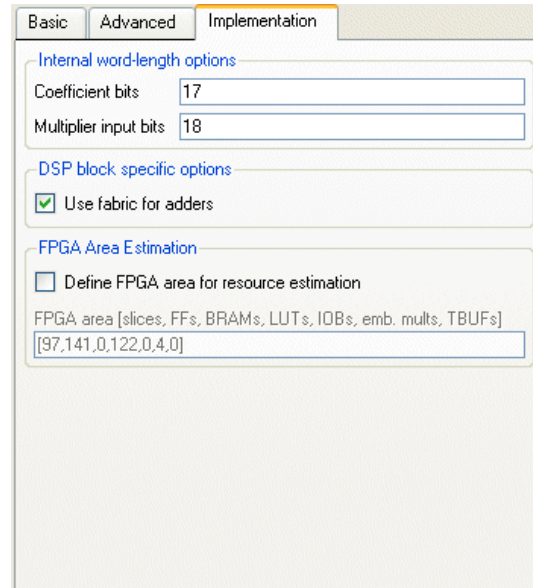


Figure 8: System Generator GUI, Implementation Tab

## System Generator Testbench

To help prototyping, testing, and verification of the RGB to YCrCb subsystem, a System Generator testbench is included with the reference design. Testbench files are under the `/sysgen` directory. To open the testbench, change the MATLAB directory to `/sysgen`, and load `Xil_RGB2YCrCb_tb.mdl` (Figure 9).

During initialization of the model, `Xil_RGB2YCrCb_init.mdl.m` is executed, which initializes the stimulus workspace variables `input_image`, `input_image_r`, `input_image_g` and `input_image_b` with color bars (Figure 10). The `init` function creates golden reference for the output `matlab_y`, `matlab_cr`, and `matlab_cb`, using the function `double_rgb2ycrcb`.

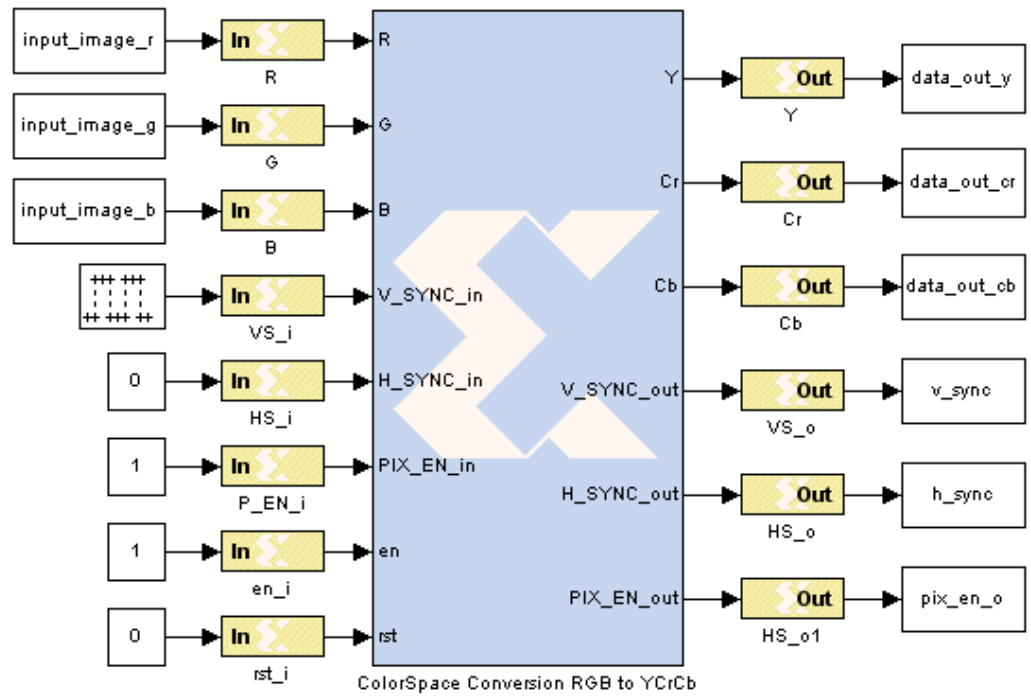


Figure 9: System Generator Testbench

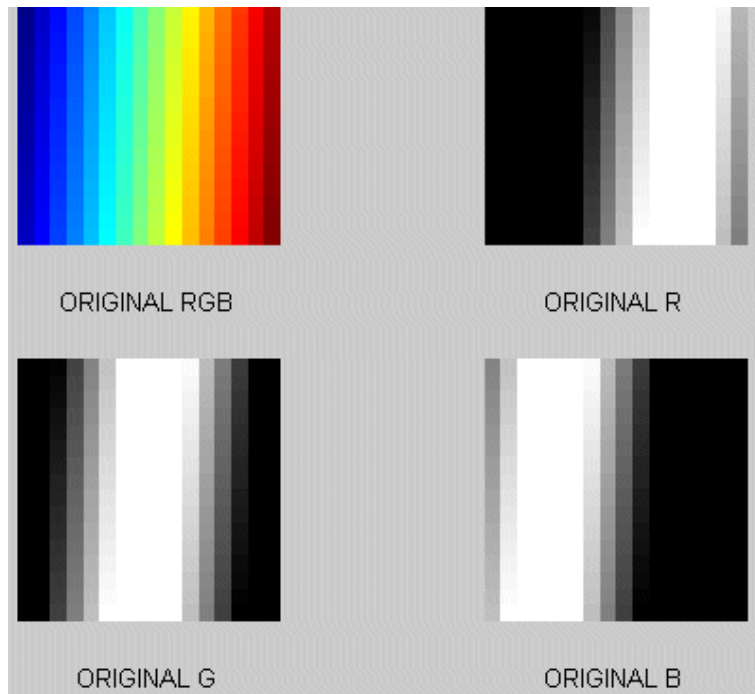


Figure 10: R,G,B Stimulus

## Running the Testbench

**Note:** The testbench is set up to run the token with default parameterization. If parameters IWIDTH or OWIDTH were changed, *number of bits* and *binary point* settings must be changed accordingly in Gateway In modules for R,G, and B.

The testbench can be executed using ISE simulator ISIM, external simulator ModelSim, or using hardware co-simulation. Refer to the System Generator documentation for more information on hardware co-simulation. By default, the testbench uses ModelSim, taking advantage of the option to leave the ModelSim simulation window open after the simulation has completed.

1. To switch between simulators, right click on the RGB2YCrCb token, and select *Look under Mask* from the context menu.
2. Double click on the *Colorspace* token, and select the Simulation Mode of your choice in the block properties dialog box displayed. ModelSim specific options can be set by double-clicking the ModelSim token. An important feature is the ability to load a macro file before the simulation starts, which enables displaying additional (internal) VHDL signals during simulation. This is an excellent tool for debugging black-box designs.
3. To specify the name of the macro file, select the Advanced tab on the ModelSim block properties dialog box, and enter the name of a `.do` file into field *Script to run after vsim*. By default, `wave_add_ycrCb.do` is loaded, which displays some key signals into the waveform window.
4. Click on the Start simulation (-) icon to run the simulation. After the simulation is finished, function `Xil_YCrCb2RGB_post_proc` is called, which displays VHDL output for visual verification (Figure 11).

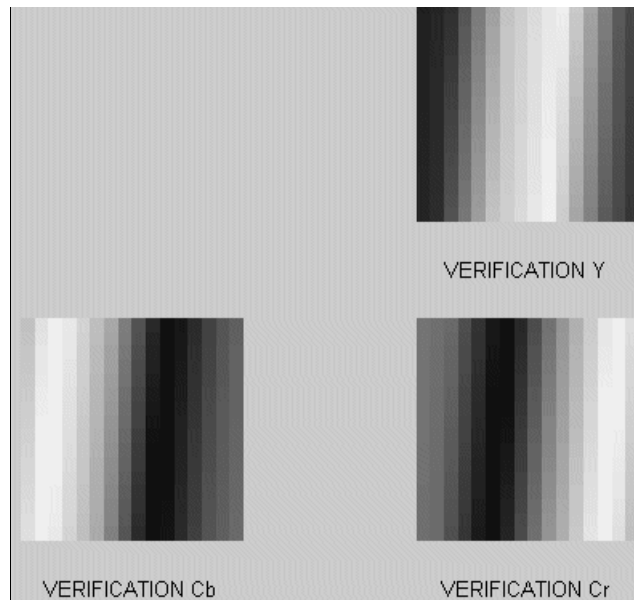


Figure 11: Y,Cb,Cr Output

## Reference Design Files

The post-processing function contains templates for calculating some key error statistics between the VHDL output and the double-precision MATLAB model. Also, a fixed-point MATLAB model (`Xil_RGB2YCrCb_fi_model.m`) is included in the bundle to facilitate bit-true verification of VHDL results.

The reference design files can be downloaded from the Xilinx website at:

<http://www.xilinx.com/bvdocs/apnotes/xapp930.zip>

## References

1. *Keith Jack*, Video Demystified, 4<sup>th</sup> Edition, ISBN 0-7506-7822-4, pp 15-19.
2. *Charles Poynton*, Digital Video and HDTV, ISBN 1-55860-792-7, pp 302 – 321.
3. Recommendation ITU-R BT.601-5 standard definition: <http://www.itu.int>.
4. Recommendation ITU-R BT.709-5 standard definition: <http://www.itu.int>.
5. *John G. Proakis, Dimitris G. Manolakis*, Digital Signal Processing (3<sup>rd</sup> edition), ISBN 0-13-373762-4, pp 755-756.
6. *Gary Sullivan*, “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.)

## Revision History

The following table shows the revision history for this document.

| Date     | Version | Revision  |
|----------|---------|---|
| 05/09/06 | 1.0     | Initial Xilinx release.   |
| 08/27/07 | 1.0.1   | Changed filename in “ <a href="#">System Generator Testbench</a> ” section (page 13). Updated Copyright notice on page 1. Added new Notice of Disclaimer. |

## Notice of Disclaimer

Xilinx is disclosing this Application Note to you “AS-IS” with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.