XAPP931 (v1.2) December 2, 2009

Color-Space Converter: YCrCb to RGB

Author: Gabor Szedo

Summary

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

1. `GenXlib_util.vhd`
2. `GenXlib_arch.vhd`
3. `color_space_pkg.vhd`
4. `Xil_YCrCb2RGB.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 Y, Cr, and Cb are the YCrCb color-space inputs. R, G, and B are the RGB color-space outputs. See Figure 1.

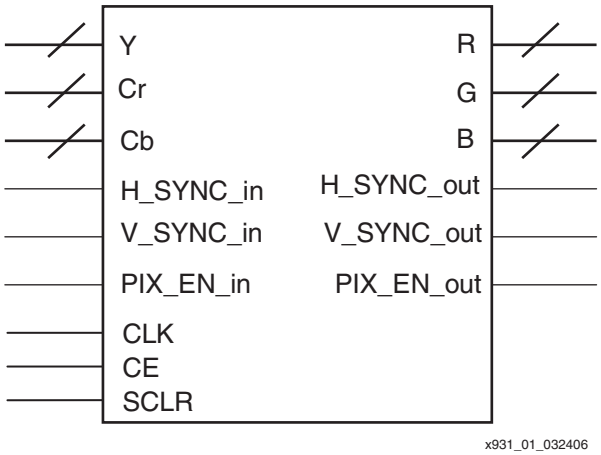

Figure 1: **YCrCb to RGB Pinout**

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 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(1)
FABRIC_ADDS	Integer	0,1	1, if adders/subtractors should be implemented in fabric(1)
IWIDTH	Integer	8,10	Input (RGB) data width
CWIDTH	Integer	8 to 18	Coefficient data width
MWIDTH	Integer	Between *IWIDTH* and *IWIDTH+CWIDTH*	Embedded multiplier width(2)
OWIDTH	Integer	8, 10	Output (YCrCb) data width
RGBMAX	Integer	0 to 2^{OWIDTH}-1	Clipping value for the R, G, B outputs
RGBMIN	Integer	0 to 2^{OWIDTH}-1	Clamping value for the R, G, B outputs
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 (3)
BCOEF	Integer		Coefficient B value (3)
CCOEF	Integer		Coefficient C value (3)
DCOEF	Integer		Coefficient D value(3)
ROFFSET	Integer	0 to 2^{OWIDTH}-1	Offset value for the R output
GOFFSET	Integer	0 to 2^{OWIDTH}-1	Offset value for the G output
BOFFSET	Integer	0 to 2^{OWIDTH}-1	Offset value for the B output
HAS_CLIP	Integer	0,1	1, if outputs have clipping logic(4)
HAS_CLAMP	Integer	0,1	1, if outputs have clamping logic(4)

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 [Ref 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 user 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 and in many other domains 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.

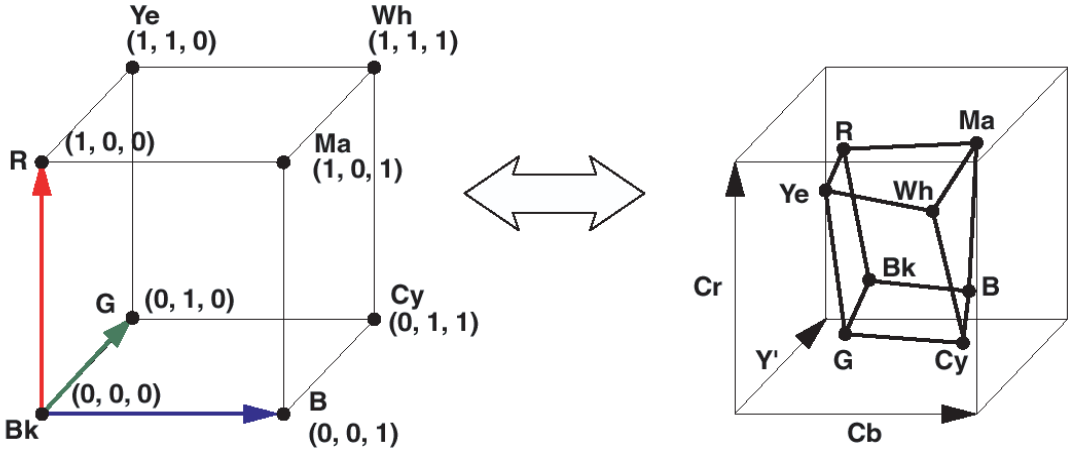

Figure 2: **RGB and YCrCb Color Representations**

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:

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

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:

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

Coefficients *CA*,*CB,* and 1-*CA*-*CB* are chosen between 0 and 1, which guarantees that the range of *Y* is constrained between RGB_{min} and RGB_{max}.

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

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

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

Thus, the range of *B*-*Y* is $2(1-CB)\ (RGB_{max} - RGB_{min})$.

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

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

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

Thus, the range of R-*Y* is $2(1-CA)\ (RGB_{max} - RGB_{min})$.

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

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

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. Coefficients *CC* and *CD* allow reducing the dynamic range of chroma components (C_B and C_R). When *RGB* values are also in the [0...1[range,

$$CC = \frac{1}{2(1-CB)} \qquad CD = \frac{1}{2(1-CA)} \quad \text{Equation 4}$$

avoids arithmetic under- and overflows.

Conversely, by inverting the transformation matrix in Equation 5, the transformation from the YCrCb color space to the RGB color space can be defined as:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1/\ CD \\ 1 & \dfrac{-CB}{CC(1-CA-CB)} & \dfrac{-CA}{CD(1-CA-CB)} \\ 1 & 1/\ CC & 0 \end{bmatrix} \begin{bmatrix} Y\ -O_Y \\ C_B\ -O_C \\ C_R\ -O_C \end{bmatrix} \quad \text{Equation 5}$$

Assigning Values to Design Parameters

There are only four non-trivial constant coefficient multiplications in Equation 5. The quantized coefficients used are:

$$ACOEF = \left[2^{CWIDTH-2}\ \frac{1}{CD}\right]$$

$$BCOEF = \left[2^{CWIDTH-2}\ \frac{-CB}{CC(1-CA-CB)}\right]$$

$$CCOEF = \left[2^{CWIDTH-2}\ \frac{-CA}{CD(1-CA-CB)}\right]$$

$$DCOEF = \left[2^{CWIDTH-2}\ \frac{1}{CC}\right]$$

Where [ ] denotes rounding to nearest integer.

In Equation 5, offset compensation happens before the matrix multiplication. To take advantage of the adders in the DSP48, calculation is reordered such that offset compensation takes place after the multiplication. Offset values have to get adjusted:

$$ROFFSET = RND\_CONST - ACOEF \cdot COFFSET + YOFFSET \cdot 2^{CDWIDTH-2} \quad \text{Equation 6}$$

$$GOFFSET = RND\_CONST - (BCOEF + CCOEF) \cdot COFFSET + YOFFSET \cdot 2^{CDWIDTH-2} \quad \text{Equation 7}$$

$$BOFFSET = RND\_CONST - DCOEF \cdot COFFSET + YOFFSET \cdot 2^{CDWIDTH-2} \quad \text{Equation 8}$$

And the YCrCb to RGB color-space transformation equations are:

$$R = [(ACOEF * Cr + ROFFSET)/ 2^{CWIDTH-2}]_{OWIDTH} + Y \quad \text{Equation 9}$$

$$G = [(BCOEF * Cb + CCOEF * Cr + GOFFSET)/2^{CWIDTH-2}]_{OWIDTH} + Y \quad \text{Equation 10}$$

$$B = [(DCOEF * Cb + BOFFSET)/2^{CWIDTH-2}]_{OWIDTH} + Y \quad \text{Equation 11}$$

where $[\,]_k$ denotes rounding to k bits. Rounding is implemented by adding ½ LSB and truncating the data. This rounding constant , $RND\_CONST = 2^{CWIDTH-3}$, is added to *ROFFSET*, *GOFFSET*, and *BOFFSET.*

Figure 3 presents the architecture implementing the above equations.

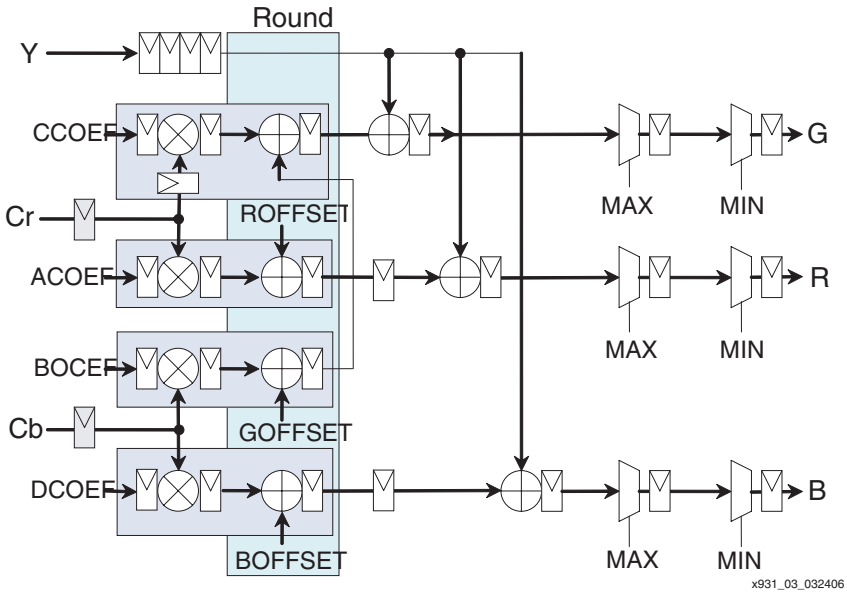

Figure 3: **Application Schematic**

Equation 6 through Equation 11 correspond to cases where the width of the adders is unconstrained. If fabric adders are used, the width of the adders, which determines the carry-chain length, may be constrained to trade off operating speeds with quantization noise. When (FAMILY_HAS_MAC=0) and MWIDTH<CWIDTH+IWIDTH, only MWIDTH bits are carried forward from the multiplication results and OFFSETs have to be requantized to MWIDTH bits:

$$RND\_CONST = 2^{MWIDTH - OWIDTH - 2 - FAMILY\_HAS\_MAC}$$

$$SCALE\_M = 2^{IWIDTH + CWIDTH - MWIDTH + FAMILY\_HAS\_MAC}$$

$$ROFFSET = RND\_CONST - [(ACOEF * COFFSET + YOFFSET\ 2^{CWIDTH-2}) / SCALE\_M] \quad \text{Equation 12}$$

$$GOFFSET = RND\_CONST - [(BCOEF + CCOEF) * COFFSET + YOFFSET\ 2^{CWIDTH-2} / SCALE\_M] \quad \text{Equation 13}$$

$$BOFFSET = RND\_CONST - [(DCOEF * COFFSET + YOFFSET\ 2^{CWIDTH-2}) / SCALE\_M] \quad \text{Equation 14}$$

Although calculation of VHDL parameter values seem cumbersome, it allows users to substitute values from any standard, which usually specify *COEF* and *OFFSET* values, rather than the values that need to configure four multiplier solutions (Figure 3).

Contents of Table 3, Table 4, and Table 5 were based on data found in [Ref 1], [Ref 2], [Ref 3], and [Ref 4].

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

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

Coefficient/ 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/ 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	$2^{OWIDTH-4}$

Table 5: **Parameterization Values for the YUV Standard *(Continued)***

Coefficient/ Parameter	Value
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	$16*2^{OWIDTH-8}$

Error Analysis

The gray boxes present logic blocks, which are implemented using DSP blocks whenever DSP blocks are available in the target device. When targeting Virtex™-4 or Virtex-5 devices, set parameter FAMILY_HAS_MAC to 1. Setting parameter FABRIC_ADDS=0 also forces other arithmetic components in the design into DSP Blocks.

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

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

$$R = \left[Cr \cdot ACOEF + ROFFSET\right]_{OWIDTH} + Y \qquad \text{Equation 15}$$

$$G = \left[Cb \cdot BCOEF + Cr \cdot CCOEF + ROFFSET\right]_{OWIDTH} + Y \qquad \text{Equation 16}$$

$$B = \left[Cb \cdot DCOEF + BOFFSET\right]_{OWIDTH} + Y \qquad \text{Equation 17}$$

where $[\,]_k$ denotes rounding to k bits. When the design is implemented on a target device which does not contain DSP48 blocks, the outputs of multipliers are truncated down to *MWIDTH* before additions, to reduce the size and the carry-chain length of fabric-based adders. Noise attributed to this truncation can be set to an order of magnitude less than the quantization noise inserted in later stages when word-length is further reduced to *OWIDTH*. Intuitively, by approximating SQNR $\cong$ 6.02 *MWIDTH* [dB], the rounding noise can be reduced by increasing *MWIDTH.* However, *MWIDTH* affects the resource usage and the maximum carry chain length in the design, thereby, affecting maximum speed. Therefore, optimal *MWIDTH* values in the [*IWIDTH*+4, *IWIDTH*+8] range do not significantly increase resource counts, but assure that quantization noise inserted is negligible (at least 20 dB less than the input noise)**.**Therefore, optimal *MWIDTH* values in the [*IWIDTH*+4, 18] range do not significantly increase resource counts, but assures that quantization noise inserted is negligible (at least 20 dB less than the input noise).

The design may introduce noise quantization and clipping noise to the output. Quantization noise is inserted when data is rounded to *OWIDTH* bits at the output. Clipping noise may be inserted; *COEF* and *OFFSET* values are chosen such that resulting R,G,B values might become larger (overflow), or smaller (underflow), than the representation capabilities of the output format [0...1).

Before analyzing the effects of these noise sources, first look at the input Signal-to-Quantization Noise Ratio (SQNR). Assuming uniformly distributed quantization error, Equation 18 illustrates the calculation of SQNR for the Y input channel:

$$SQNR_{YCrCb} = 10\log\frac{P_x}{P_N} = 10\log\frac{\int_{YMIN}^{YMAX}(x-m(x)^2)}{\frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2}x^2dx} \quad \text{Equation 18}$$

Substituting 1 LSB=$2^{-INBITS}$, where *INBITS* is the input (RGB) precision, $SQNR_{RGB}$ becomes a function of the input dynamic range. For sake of simplicity, choose *INBITS* =8, and instead of using YCrCb in the x =[0,1) range, inspect some of following standard conventions.

When YCrCb values are in the **(0, 255) range** (Equation 19):

$$SQNR_{YCrCb} = 10\log\frac{\frac{1}{255}\int_{0}^{255}x^2dx}{\int_{-1/2}^{1/2}x^2dx} = 10\log\frac{\frac{1}{3\cdot 255}[255^3]}{\frac{1}{12}} = 54.15\,dB \quad \text{Equation 19}$$

when YCrCb values are in the **(16, 240) range** (Equation 20):

$$SQNR_{YCrCb} = 10\log\frac{\frac{1}{224}\int_{16}^{240}x^2dx}{\int_{-1/2}^{1/2}x^2dx} = 53.92\,dB \quad \text{Equation 20}$$

and when RGB values are in the **(16, 235) range** (Equation 21):

$$SQNR_{YCrCb} = 10\log\frac{\frac{1}{219}\int_{16}^{235}x^2dx}{\int_{-1/2}^{1/2}x^2dx} = 53.47\,dB \quad \text{Equation 21}$$

Output Quantization Noise

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 formulas which addresses clipping noise as well is beyond the scope of this document.

Table 6 illustrates noise figures for some typical parameter combinations [Ref 3].

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

SNR	*IWIDTH = OWIDTH =* 8 Bits	*IWIDTH = OWIDTH =* 10 Bits	Input Range
SNR_{YCrCb} (input)	53.5	65.6	[0...255] (8bit) Or [0...1023] (10 bit)
SNR_R	47.7	59.7	
SNR_G	42.4	54.4	
SNR_B	45.7	57.9	
SNR_{YCrCb} (input)	53.4	65.3	[16...240] (8bit) Or [64...960] (10 bit)
SNR_R	47.5	59.4	
SNR_G	42.1	54.2	
SNR_B	45.5	57.5	
SNR_{YCrCb} (input)	53.3	65.2	[16...235] (8bit) Or [64...920] (10 bit)
SNR_R	47.2	59.3	
SNR_G	42.0	54.1	
SNR_B	45.3	57.3	

Output Clipping Noise

Certain input (R,G,B) and/or parameter (*COEF + OFFSET*) combinations can lead to output values extending beyond the capabilities of the output representation [0..1). Output values can get larger (overflow) than the maximum or smaller (underflow) than the minimum value, which result in corrupted output values. If overflow or underflow occurs and the design does not have clipping logic (*HAS_CLIPPING*=0), binary values wrap around, inserting substantial noise to the output. If HAS_CLIPPING=1, output values saturate, introducing less noise (Figure 4).

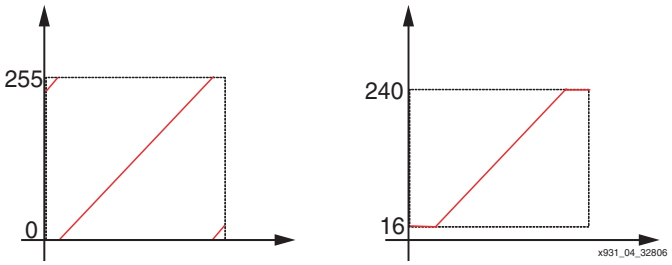

Figure 4: **Wraparound and Saturation**

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 6*OWIDTH* 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 constrains output values to the predefined range by setting *YMAX* and *YMIN* values (constraining luminance), as well as *CMAX* and *CMIN* values (constraining chrominance) 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 (Table 7), 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	288.76 MHz
Number of Slice Flip Flops	141
Number of 4 Input LUTs	122
Number of Occupied Slices	97
Number of DSP48s	4

For Spartan™-3 testing (Table 8), an XC3S1000 part with -4 speed grade and FG320 packaging were selected.

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

Maximum Operating Frequency	158.17 MHz
Number of Slice Flip Flops	206
Number of 4 Input LUTs	122
Number of Occupied Slices	131
Number of MULT18x18s	4

To calculate the exact latency of the core, a support function

YCrCb2RGB_LATENCY(FAMILY_HAS_MAC, FABRIC_ADDS, HAS_CLIP, HAS_CLAMP: integer) return integer

is included in `color_space_pkg.vhd`, which returns the latency of the YCrCb2RGB module.

System Generator Token

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

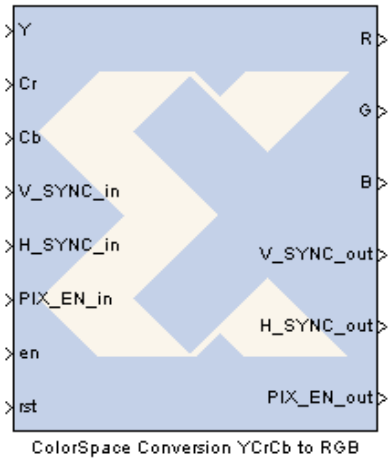

Figure 5: **System Generator Token**

The system generator instance can be parameterized through a Graphical User Interface (GUI) activated by double clicking the token. If the YCrCb to RGB 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.

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

- `Xil_YCrCb2RGB_config.m`
- `ycrcb2rgb_action.m`
- `ycrcb2rgb_enablement.m`
- `Xil_YCrCb2RGB_GUI.xml`

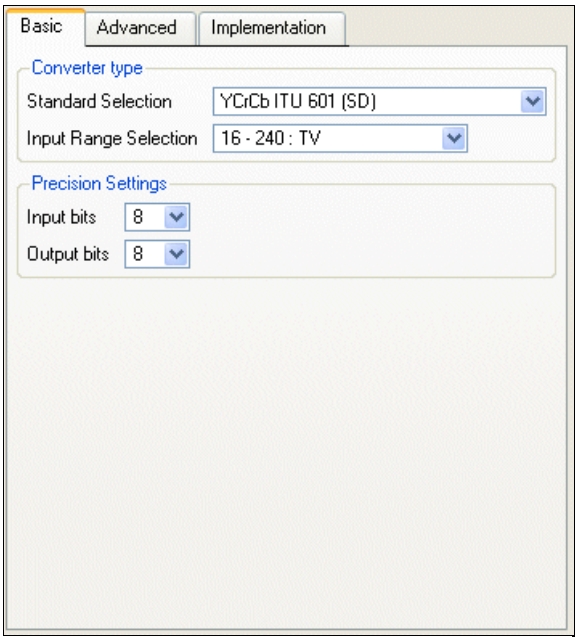

Figure 6: **System Generator GUI, Basic Tab**

If the user wants to implement a custom designed converter or a standard that 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]). The Advanced tab was designed to be very similar to those of the RGB to YCrCb reference design. Entering the same parameters to both GUIs result in a matching transformation pair that apart from computation noise returns the original pixel data.

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 or modified, the GUI keeps the user-defined values even if the standard-selection is changed back and forth between a predefined standard and custom.

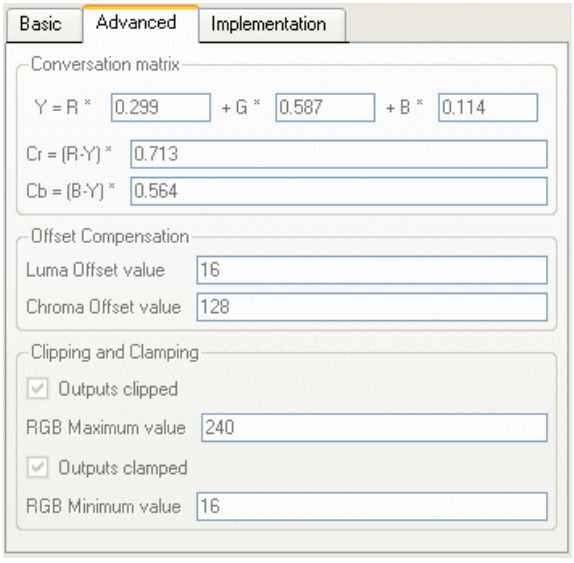

Figure 7: **Advanced Tab**

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 to 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 gray background on 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.

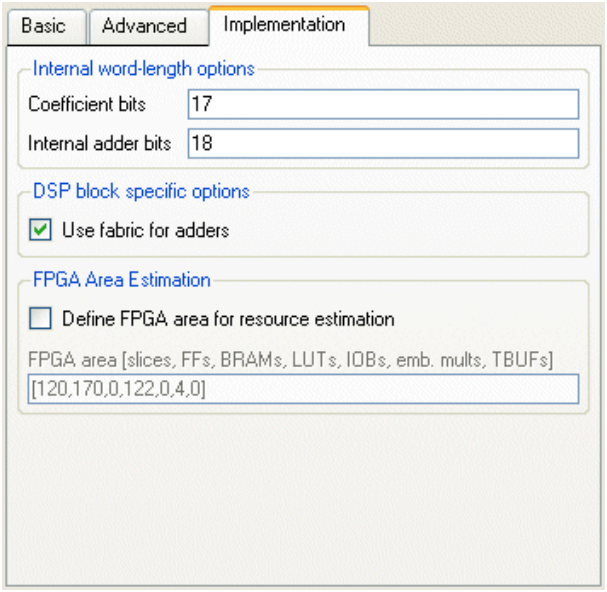

Figure 8: **Implementation Tab**

System Generator Testbench

To help prototyping, testing, and verification of the YCrCb to RGB subsystem, a system generator testbench is included with the reference design. Testbench files are under the `/sysgen/testbench` directory. To open the testbench, change your MATLAB directory to `/sysgen/testbench`, and load `Xil_YCrCb2RGB_tb.mdl`. See Figure 9.

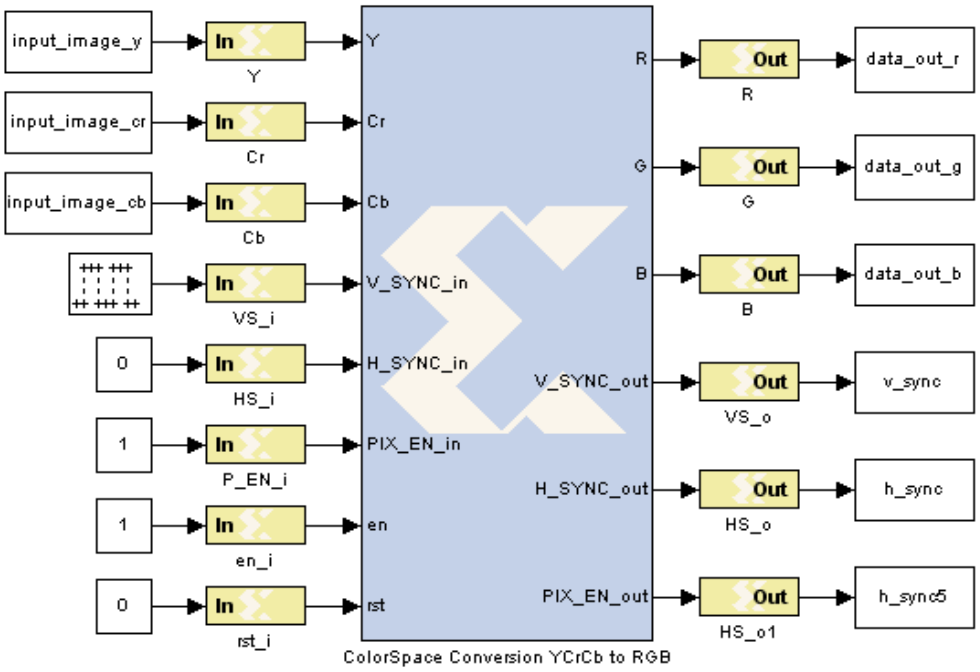

Figure 9: **System Generator Testbench**

During initialization of the model, `Xil_YCrCb2RGB_preload_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). Also, the preload function creates golden reference for the output matlab_y, matlab_cr, matlab_cb, using the function double_ycrcb2rgb.

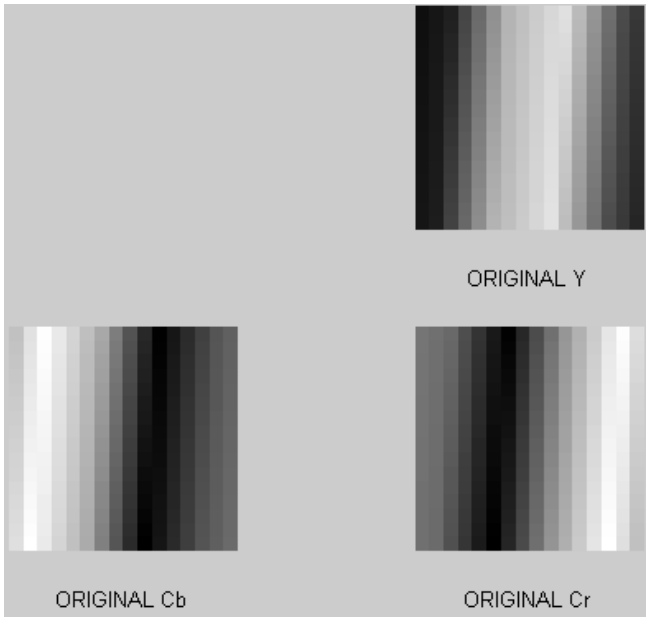

Figure 10: **R, G, B Stimulus**

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 YCrCb2RGB 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_rgb.do` is loaded, which displays some key signals into the waveform window.
4. Click on 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).

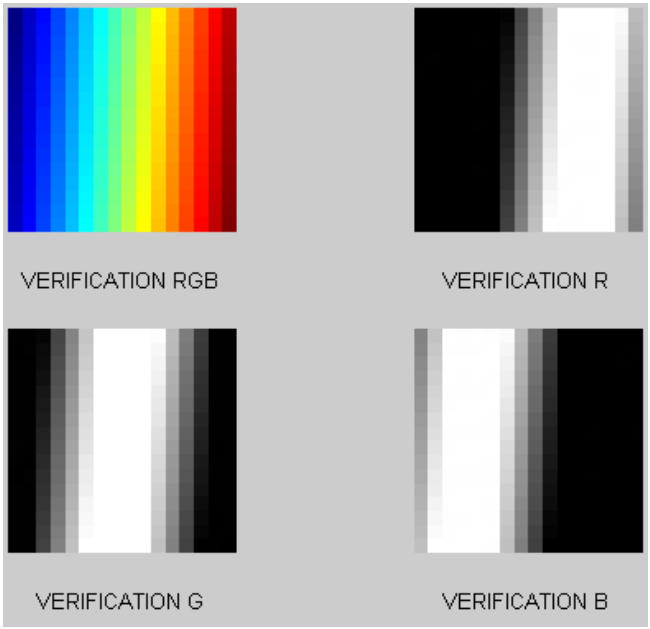

Figure 11: **Y, Cb, Cr Output**

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_YCrCb2RGB_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/appnotes/xapp931.zip

References

1. *Keith Jack*, Video Demystified, 4th 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. *Gary Sullivan*, "A pproximate theoretical analysis of RGB to YCbCr to RGB conversion error," in Joint Video Team (JVT) of ISO/IEC MPEG and 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.
10/13/06	1.1	Minor edits and changes to equations on pages 5 and 6.
10/02/09	1.2	Corrected Equations 1, 5 and 8 on pages 4, 5 and 6.