Image Processing Pipeline v1.0

DS723 April 24, 2009 **Product Specification**

Introduction

The Xilinx LogiCORE™ IP Image Processing Pipeline core provides an optimized hardware block to pre-process images captured by a color image sensor fitted with a Bayer Color Filter Array (CFA). The Image Processing Pipeline core provides an efficient and low-footprint solution to correct defective pixels, interpolate the missing color components for every pixel, correct colors to adjust to lighting conditions, and set gamma to compensate for the intensity distortion of different display devices.

Features

- RGB and CMY Bayer image sensor support
- Low-footprint, high quality 5x5 CFA interpolation
- Defective Pixel Correction
- Color Correction Matrix
- Gamma Correction
- Optional RGB to YCrCb conversion
- Selectable processor interface
  - EDK pCore
  - General Purpose Processor
  - Constant Interface
- 8, 10, and, 12 bit input and output precision
- Automatic detection of timing parameters and timing signal polarities
- Support for Virtex®-5 and Spartan®-3A DSP
- For use with Xilinx CORE Generator™ 11.1 or later

Applications

- Pre-processing Block for Image Sensors
- Consumer Digital Still Camera
- Video Surveillance
- Medical Imaging
- Industrial Imaging
- Video Conferencing
- Machine Vision

Overview

The Xilinx Image Processing Pipeline core is a cascade of five major blocks (Figure 1):

- Defective Pixel Correction
- Color Filter Array Interpolation
- Color Correction Matrix
- Gamma Correction
- Color-Space Conversion

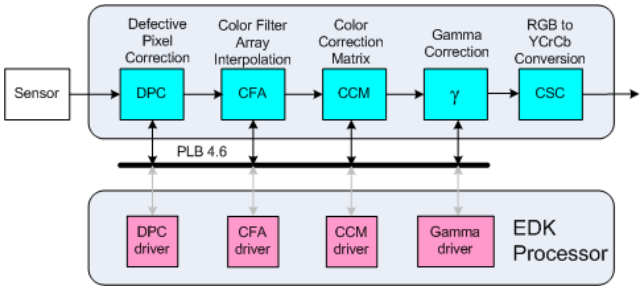

Figure 1: **Image Processing Pipeline**

Defective Pixel Correction

An image sensor may have a certain number of defective pixels that can result from manufacturing faults or failures during operation. Defective pixels can be static (always present) or dynamic (function of exposure or temperature). The Defective Pixel Correction block can correct static or dynamic defective pixels that are:

- dead (always low)
- hot (always high)
- stuck (to a certain value)

Based on the assumption that defective samples significantly stand out from their adjacent neighbors, defective samples can be identified dynamically, without the use of a frame buffer.

Each sample in the sensor array is compared to its adjacent samples in the same scan-line. If the differences are greater than a specified value, then the defective sample is replaced. To illustrate the correction process for the typical Bayer array, consider five adjacent samples $[G_{n-1}, R_{n-1}, G_n, R_{n+1}, G_{n+1}]$, where G_n is the location to

be evaluated for correction. If the differences between same color [G_{n-1}, G_n, G_{n+1}] samples and nearest neighbor samples [R_{n-1}, G_n, R_{n+1}] are beyond predefined, user-programmable thresholds, then G_n is replaced by the median value of [G_{n-1}, G_n, G_{n+1}]. The programmable thresholds should be defined to reflect the correlation between similar color samples. The same process is performed for all samples in each line of the array. This method can correct, at most, every third pixel.

Extremely noisy conditions, like low light or long exposure times may cause the comparisons between samples to vary slightly above or below the thresholds. This condition may cause pixels to flicker. This effect can be reduced by discarding the least significant bits defined by the kerning bits during the comparison operation. This method helps reduce the potential for small changes in the neighbors pixel values to cause unnecessary sample replacements.

For additional details, see the Xilinx LogiCORE IP Defective Pixel Correction Data Sheet (www.xilinx.com/products/ipcenter/EF-DI-DEF-PIX-CORR.htm) [Ref 1].

Color Filter Array Interpolation

CMOS and CCD image sensors used in color imaging applications are typically fitted with a color filter array (CFA). The color filter contains a set of colored micro-lenses which focus the photons and allow only one principle color to pass at each phototransistor site. The most common CFA patterns are the RGB and CMY Bayer patterns shown in Figure 2.

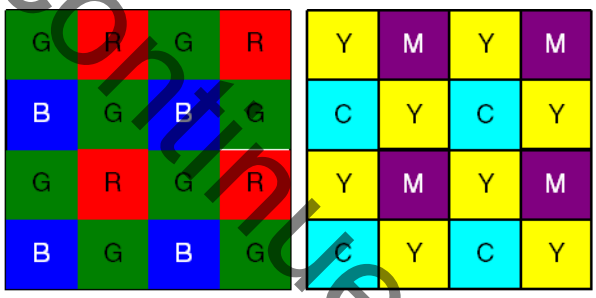

Figure 2: **RGB and CMY Bayer Patterns**

In order to reconstruct a color image, all three primary colors are needed at each phototransistor (pixel) location. There is no exact method to fully recover the missing information, as color channels have been physically sub-sampled by the CFA which can lead to possible aliasing between color channels. The Color Filter Array Interpolation block was designed to efficiently provide high quality color image reconstruction for data from a sensor with a RGB or CMY Bayer CFA without the use of an external frame buffer. The Color Filter Array Interpolation block suppresses interpolation artifacts, such as the zipper and color aliasing effects, by minimizing Chrominance Variances in a 5x5 neighborhood.

For additional details, see the Xilinx LogiCORE IP Color Filter Array Interpolation data sheet (www.xilinx.com/products/ipcenter/EF-DI-CFA.htm) [Ref 2].

Color Correction Matrix

Color correction is used for adjusting white balance, color cast, brightness, and/or contrast in an RGB or CMY image. A common use for this function is white balance adjustment which can correct for lighting source variations such as daylight, fluorescent, or tungsten. The Color Correction Matrix block is a 3x3 programmable coefficient matrix multiplier with offset compensation (Equation 1) and can be used to correct image data for these variations.

Equation 1

$$\begin{bmatrix} R_c \\ G_c \\ B_c \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

As seen from the matrix operation, the input pixels are transformed to a set of corrected output pixels. The matrix multiplication and offset compensation is implemented solely with XtremeDSP™ slices. The coefficients and offsets can be dynamically adapted to changing lighting conditions by programming their values on a frame-by-frame basis through software drivers.

Wide coefficient and offset compensation ranges enable this block to also serve as a general purpose color-space converter between the RGB, YCrCb, YUV and CMY color-spaces.

For additional details see the Xilinx LogiCORE IP Color Correction Matrix data sheet (www.xilinx.com/products/ipcenter/EF-DI-CCM.htm) [Ref 3].

Gamma Correction

Gamma correction, also known as gamma compression or encoding, is used to encode linear luminance or RGB values into signals that match the non-linear characteristics of display devices. Gamma correction helps to map data into a more perceptually uniform domain in order to optimize perceptual performance.

Gamma correction is, in the simplest cases, defined by $V_{out} = V_{in}^{\gamma}$, where the input and output values are between 0 and 1. The case γ <1 is often called gamma compression and γ >1 is called gamma expansion (Figure 3).

The Gamma Correction block is implemented as a Look-Up Table (LUT) and supports single or triple color channel tables and can be re-programmed frame-by-frame dynamically. As a general purpose intensity map, the Look-Up-Tables can be reprogrammed with arbitrary functions, supporting a wide range of applications, such as intensity correction, feature enhancement, lin-log, log-lin conversion and thresholding.

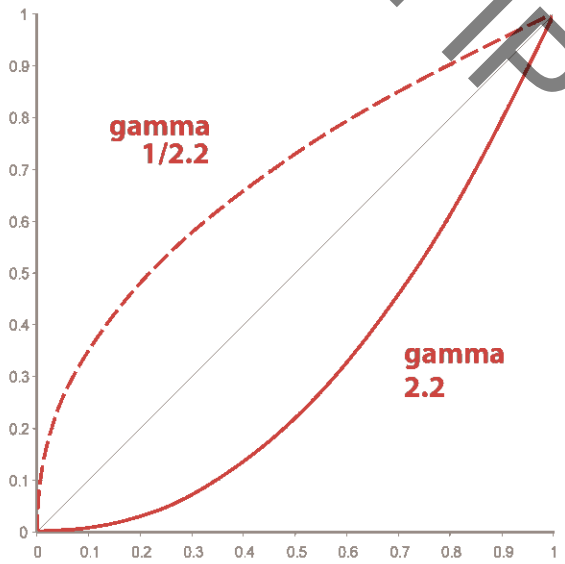

Figure 3: **Gamma Correction**

The Gamma Correction block offers various configuration options for a designer to optimize the block RAM footprint required by the core. Options include using a single correction curve for three channels (R, G, B) (8-, or 10-bit input data) and optional LUT interpolation (12-bit input data).

For additional details, see the Xilinx Gamma Correction LogiCORE data sheet (www.xilinx.com/products/ipcenter/EF-DI-GAMMA.htm) [Ref 4].

Color-Space Conversion

The YCrCb color space is commonly used in imaging applications because it allows the chroma components (Cr and Cb) to be bandwidth-reduced, subsampled, compressed, or otherwise treated separately for improved system efficiency. The final module in the Image Processing Pipeline core is an optional RGB to YCrCb color-space conversion.

The RGB to YCrCb Color-Space Conversion block is a simplified 3x3 constant-coefficient matrix multiplier converting three input color samples to three output samples in a single clock cycle. This module can be optimized through configuration to use only four XtremeDSP slices. The Color-Space Conversion block supports conversion from the RGB color-space to the following color spaces:

- YCrCb ITU 601 (SD) [Ref 6]
- YCrCb ITU 709 (HD) 1125/60 (PAL) [Ref 7]
- YCrCb ITU 709 (HD) 1250/50 (NTSC)
- YUV

For more information, see the Xilinx RGB to YCrCb Color-Space Conversion LogiCORE Data Sheet (www.xilinx.com/products/ipcenter/RGB_to_YCrCb.htm) [Ref 5].

Timing Parameters

A built-in timing detector module in the Color Filter Array Interpolation block (Figure 4) measures timing parameters of the input video stream, such as the total number of rows and columns, blank rows and columns, and makes the measurement results accessible through an EDK pCore Interface or General Purpose Processor Interface. A built-in, programmable timing generator module can create blanking and active video signals, based on user-provided parameters, and use these signals to re-frame the input video data-stream. This module enables the user to change the position of blanked regions as well as to crop the active area. However, the Color Filter Array Interpolation block cannot change the input/output image sizes, nor can it change the input and output pixel clock rates or total image sizes.

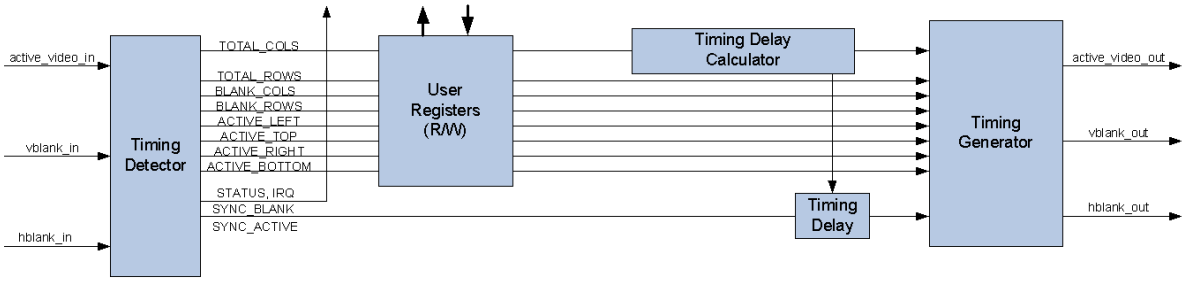

Figure 4: **Block Diagram of the Xilinx Timing Detector Module**

Blanking Signal Polarities

Typical constituents of a video stream, blanking signals provide framing and blanking information that complements and formats image data provided via the video_data_in port. However, some image sensors provide this data by the use of horizontal (line) and vertical (frame) valid signals [Ref 9], while others use blanking signals.

Valid signals mark the non-blank, active area of the input frame by logic high, logic low corresponding to blanked or inactive areas. Conversely, blanking signals mark the blanked, inactive area by logic high.

The Color Filter Array Interpolation block is equipped with automatic detection of blanking signal polarity, based on the phase relations between the blanking and active signals. The active_video signals are always active high. If active_video_in is high during the logic high period of a blanking signal, that blanking signal is considered active high (valid signaling). If active_video_in is high during the logic low period of a blanking signal, the blanking signal is considered active low (blank signaling). The high portion of active_video_in should not extend across edges of either blanking signals.

Note: The following definition of timing parameters assumes the hblank_in and vblank_in are driven by blanking signals, with logic high corresponding to blanked areas and logic low corresponding to non-blanked areas.

Definition of Timing Parameters

From the edge transitions of the three input timing signals, the timing circuitry can measure:

- Blanking signal polarities
- Overall (total) frame dimensions
- The size and position of the non-blank area
- The size and position of the active area

The total, non-blank, and active areas of video frames can be visualized as a set of rectangles. The total area of the frame contains the non-blank area, which in turn contains the active area (Figure 5).

The top-left corner of the frame is defined by a falling edge of the vblank_in signal. A rising edge transition on vblank_in signals the start of the vertical blanking area. Cyclic repetition of vblank_in defines the frame boundaries within the video stream. Similarly, a rising edge on hblank_in signals the beginning of the horizontal blanking period. The falling edge transition signals the end of the blanking period.

The number of clock cycles between two falling edge transitions of hblank_in defines the total number of columns in a frame (TOTAL_COLS). Dividing the number of clock cycles between two falling vblank_in edges by TOTAL_COLS defines the total number of rows in a frame. The number of blank rows (BLANK_ROWS) is defined as the number of clock cycles between the rising and falling edges of vblank_in, divided by TOTAL_COLS.

The number of clock cycles between the falling edge of vblank_in and next falling edge on hblank_in defines the number of blank columns on the left side of the non-blank area (BLANK_LEFT). Similarly, BLANK_RIGHT is defined as the number of clock cycles after the falling edge of vblank_in and before the immediate next rising edge on hblank_in.

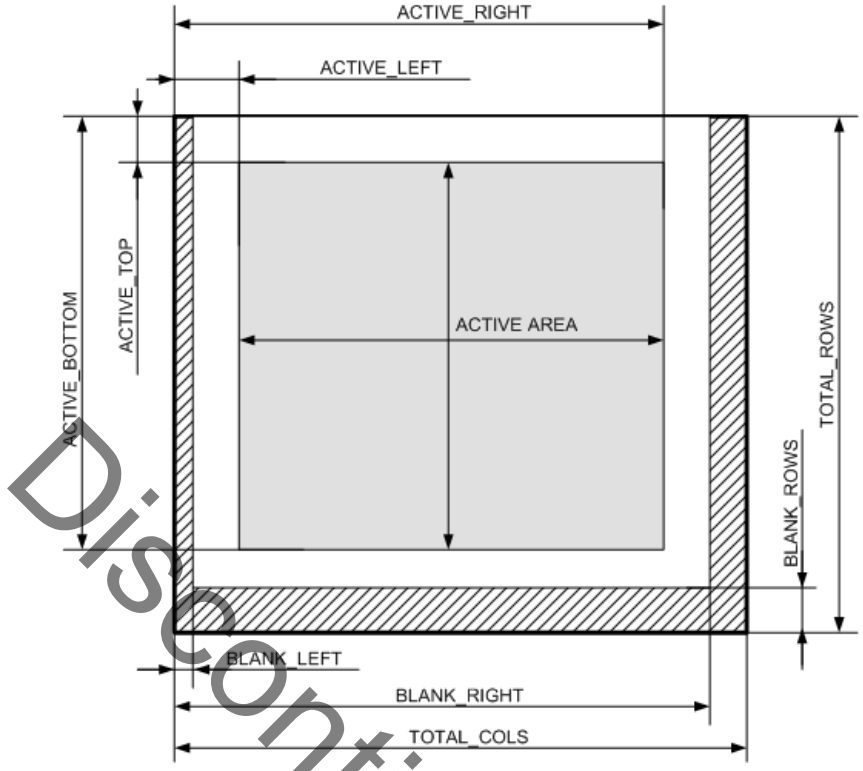

Figure 5: **Timing Parameters**

The high state of input signal `active_video_in` can mark any sample of `video_data_in` as valid. Although this signal can be used to mark an arbitrary region of the frame active, typical image sensors use this signal to designate a rectangle within the non-blank area as active/valid area. The top-left corner of the active area is defined by `ACTIVE_TOP` and `ACTIVE_LEFT`, which are the coordinates of the first sample marked active by `active_video_in` in the coordinate system defined by the blanking input signals. Likewise, `ACTIVE_BOTTOM` and `ACTIVE_RIGHT` are the coordinates of the last sample marked active by `active_video_in`. An example of horizontal timing and corresponding timing parameters is provided in Figure 16.

Processor Interfaces

The Image Processing Pipeline core supports three processor interface options:

- EDK pCore Interface
- General Purpose Processor Interface
- Constant Interface

The processor interfaces allow access to:

- Thresholds of the Defective Pixel Correction block
- Input timing information measured by the Color Filter Array Interpolation block
- Coefficients and offsets for the Color Correction Matrix block
- Gamma Correction block LUT contents

EDK pCore Interface

Many imaging applications have an embedded processor which can dynamically control the parameters within the core. The user can select an EDK pCore Interface, which creates a pCore that can be added to an EDK project as a hardware peripheral. This pCore provides a memory mapped interface for the programmable registers within the core, which are described in Tables 1 through 4.

Tables 1 through 4 describe the programmable registers grouped by functional blocks. The address offsets are not in sequential order.

Table 1: **Defective Pixel Correction Register Descriptions**

Address Offset (hex)	Register Name	Access Type	Default Value (hex)	Description	
BASEADDR + 0x878	spc_reg00_control	R/W	0x00000001	Bit 0	Software enable • 0 – Not enabled • 1 – Enabled
				Bit 1	Host processor write done semaphore • 0 – Host processor actively updating registers • 1 – Register update completed by host processor
BASEADDR + 0x87C	spc_reg01_reset	R/W	0x00000000	Bit 0	Software reset • 0 – Not reset • 1 – Reset
BASEADDR + 0x880	spc_reg04_thresh1	R/W	from GUI	Threshold for comparing against nearest neighbors	
BASEADDR + 0x884	spc_reg05_thresh2	R/W	from GUI	Threshold for comparing against same color pixels	

Table 2: **Color Filter Array Interpolation Register Descriptions**

Address Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
BASEADDR + 0x838	cfa_reg00_control	R/W	0x00000001	General control register
BASEADDR + 0x83C	cfa_reg01_sw_reset	R/W	0x00000000	Software reset register
BASEADDR + 0x840	cfa_reg03_interrupt	R/W	from GUI	Interrupt control register
BASEADDR + 0x844	cfa_reg04_active_left	R/W	from GUI	User defined value for ACTIVE_LEFT(1)
BASEADDR + 0x848	cfa_reg05_active_right	R/W	from GUI	User defined value for ACTIVE_RIGHT(1)
BASEADDR + 0x84C	cfa_reg06_active_top	R/W	from GUI	User defined value for ACTIVE_TOP(1)
BASEADDR + 0x850	cfa_reg07_active_bottom	R/W	from GUI	User defined value for ACTIVE_BOTTOM(1)

Table 2: **Color Filter Array Interpolation Register Descriptions *(Cont'd)***

BASEADDR + 0x854	cfa_reg08_total_rows	R/W	from GUI	User defined value for TOTAL_ROWS(2)
BASEADDR + 0x858	cfa_reg09_total_cols	R/W	from GUI	User defined value for TOTAL_COLS(2)
BASEADDR + 0x85C	cfa_reg10_blank_rows	R/W	from GUI	User defined value for BLANK_ROWS(1)
BASEADDR + 0x860	cfa_reg11_blank_left	R/W	from GUI	User defined value for BLANK_LEFT(1)
BASEADDR + 0x864	cfa_reg12_blank_right	R/W	from GUI	User defined value for BLANK_RIGHT(1)
BASEADDR + 0x868	cfa_reg13_blank_polarity	R/W	from GUI	User defined polarity values for Vertical (Bit 1) and Horizontal (Bit 0) Blanking. 0: indicates blanking (active low) signal 1: indicates valid video (active high) signal
BASEADDR + 0x86C	cfa_reg14_bayer_phase	R/W	from GUI	User defined register to specify the Bayer grid. Bits 0 (bayer_phase_x), and 1 (bayer_phase_y) specify whether the top-left corner of the Bayer sampling grid starts with a Green, Red or Blue pixel.
BASEADDR + 0x888	cfa_reg02_status	R		General status register
BASEADDR + 0x88C	cfa_reg15_active_left_r	R		Value used by the core for ACTIVE_LEFT(1)
BASEADDR + 0x890	cfa_reg16_active_right_r	R		Value used by the core for ACTIVE_RIGHT(1)
BASEADDR + 0x894	cfa_reg17_active_top_r	R		Value used by the core for ACTIVE_TOP(1)
BASEADDR + 0x898	cfa_reg18_active_bottom_r	R		Value used by the core for ACTIVE_BOTTOM(1)
BASEADDR + 0x89C	cfa_reg19_total_rows_r	R		Value used by the core for TOTAL_ROWS(2)
BASEADDR + 0x8A0	cfa_reg20_total_cols_r	R		Value used by the core for TOTAL_COLS(2)
BASEADDR + 0x8A4	cfa_reg21_blank_rows_r	R		Value used by the core for BLANK_ROWS(1)
BASEADDR + 0x8A8	cfa_reg22_blank_left_r	R		Value used by the core for BLANK_LEFT(1)
BASEADDR + 0x8AC	cfa_reg23_blank_right_r	R		Value used by the core for BLANK_RIGHT(1)
BASEADDR + 0x8B0	cfa_reg24_blank_polarity_r	R		Measured polarity values for Vertical (Bit 1) and Horizontal (Bit 0) Blanking. 0: indicates blanking (active low) signal 1: indicates valid video (active high) signal

1. Counting of rows and columns start from 0. For example, if the first pixel of the first line is active, both ACTIVE_LEFT and ACTIVE_TOP will be equal to 0.
2. Counting of total rows and columns starts from 1. For example, if rows 0 - 499 are non-blank, and 500-599 are blank, there are TOTAL_ROWS = 600 lines in the frame.

Table 3: **Color Correction Matrix Register Descriptions**

Address Offset (hex)	Register Name	Access Type	Default Value (hex)	Description	
BASEADDR + 0x800	ccm_reg00_control	R/W	0x00000001	Bit 0	Software enable • 0 – Not enabled • 1 – Enabled
				Bit 1	Host processor write done semaphore • 0 – Host processor actively updating registers • 1 – Register update completed by host processor
BASEADDR + 0x804	ccm_reg01_reset	R/W	0x00000000	Bit 0	Software reset • 0 – Not reset • 1 – Reset
BASEADDR + 0x808	ccm_reg04_K11	R/W	from GUI	Matrix coefficient K1,1 (1)	
BASEADDR + 0x80C	ccm_reg05_K12	R/W	from GUI	Matrix coefficient K1, (1)	
BASEADDR + 0x810	ccm_reg06_K13	R/W	from GUI	Matrix coefficient K1,3 (1)	
BASEADDR + 0x814	ccm_reg07_K21	R/W	from GUI	Matrix coefficient K2,1 (1)	
BASEADDR + 0x818	ccm_reg08_K22	R/W	from GUI	Matrix coefficient K2, (1)	
BASEADDR + 0x81C	ccm_reg09_K23	R/W	from GUI	Matrix coefficient K2, (1)	
BASEADDR + 0x820	ccm_reg10_K31	R/W	from GUI	Matrix coefficient K3, (1)	
BASEADDR + 0x824	ccm_reg11_K32	R/W	from GUI	Matrix coefficient K3,2 (1)	
BASEADDR + 0x828	ccm_reg12_K33	R/W	from GUI	Matrix coefficient K3, (1)	
BASEADDR + 0x82C	ccm_reg13_ROFFSET	R/W	from GUI	Red offset	
BASEADDR + 0x830	ccm_reg14_GOFFSET	R/W	from GUI	Green offset	
BASEADDR + 0x834	ccm_reg15_BOFFSET	R/W	from GUI	Blue offset	

1. Matrix coefficients are represented as 18.15 fixed point numbers. For example the floating point value of 1.5 is represented as 18'b011000000000000000 or 0x00018000.

Table 4: **Gamma Correction Register Descriptions**

Address Offset (hex)	Register Name	Access Type	Default Value	Description	
BASEADDR + 0x400	gamma_reg3_data	R/W	0x00000000		Values for gamma LUT
BASEADDR + 0x404	gamma_reg4_start_addr	R/W	0x00000000	Bits [15-0] Bits [31-16]	Start address Number of words to be written
BASEADDR + 0x870	gamma_reg0_control	R/W	0x00000001	Bit 0	Software enable • 0 – Not enabled • 1 – Enabled
BASEADDR + 0x874	gamma_reg1_reset	R/W	0x00000000	Bit 0	Software reset • 0 – Not reset • 1 – Reset
BASEADDR + 0x8B4	gamma_reg2_status	R	0x00000000		Number of data words received to be written to active buffer

All of the Write registers are also readable, enabling the user to verify writes or read back current values. Default values of registers are defined in the "CORE Generator - Graphical User Interface" section unless indicated otherwise above.

Control and Reset Registers

All four blocks with run-time programmable settings (Defective Pixel Correction, Color Filter Array Interpolation, Color Correction Matrix, Gamma Correction) have control and reset registers which share the same layout (Tables 5 and 6).

Table 5: **Control Register Bit Definitions**

Bit	Name	Function
0	SW_ENABLE	Software enable '0' effectively disables the core halting further operations, which blocks the propagation of all video signals. The default value of SW_ENABLE is 1 (enabled).
1	REG_UPDATE	Host processor write done semaphore '1' indicates the host processor has finished updating timing registers, which are ready to be copied over at the next rising edge of vblank_in. (See "General EDK pCore Programming Guidelines.") Applicable in the Color Correction Matrix, Color Filter Array Interpolation, Defective Pixel Correction blocks.
2	CLEAR_STATUS	'1' clears flags in the status registers (clears interrupt source) Applicable only for the Color Filter Array Interpolation block.

The core has a feature that allows the core to be enabled or disabled. This halts the operation of the core by blocking the propagation of all video signals. This function is controlled by setting Software Enable to 0; the default value of Software Enable is 1 (enabled). Software enabling and disabling is coupled to the rising edge of `vblank_in`, preventing enabling/disabling the block mid-frame.

Table 6: **Reset Register Bit Definitions**

Bit	Name	Function
0	SW_RESET	Software reset The default value of SW_RESET is 0.

Each block can be effectively reset in-system by asserting the Software Reset (bit 0), which returns the registers to their default values, specified through the Graphical User Interface when the core is instantiated. The core outputs are also forced to 0 or their default value until SW_RESET is deasserted.

Status Register

Table 7 contains the Status Register bit descriptions for the Color Filter Array Interpolation block.

Table 7: **cfa_reg02_status Bit Definitions**

Bit	Name	Function
0-7	-	Reserved
8	VSYNC_DET	Vertical Sync detected
9	VSYNC_ERR	Vertical Sync error (TOTAL_ROWS larger than MAX_ROWS parameter)
10	HSYNC_ERR	Horizontal Sync error (TOTAL_COLS larger than MAX_COLS parameter)
11	VBLANK_CHG	VBLANK POLARITY change detected
12	HBLANK_CHG	HBLANK POLARITY change detected
13	TROWS_CHG	TOTAL_ROWS change detected
14	TCOLS_CHG	TOTAL_COLS change detected
15	BROWS_CHG	BLANK_ROWS change detected
16	BCOLS_CHG	BLANK_COLS change detected

Interrupt Control Register

Table 8 contains the Interrupt Control Register bit descriptions for the Color Filter Array Interpolation block.

Table 8: **cfa_reg03_interrupt Bit Definitions**

Bit	Name	Function
0	INT_EN	Enable / disable interrupts
1	CLR_SRC	Clear interrupt sources
2-7	-	Reserved
8	VSYNC_DET_INT	'1' enables rising VSYNC_DET to request interrupt
9	VSYNC_ERR_INT	'1' enables rising VSYNC_ERR to request interrupt
10	HSYNC_ERR_INT	'1' enables rising HSYNC_ERR to request interrupt
11	VBLANK_CHG_INT	'1' enables rising VBLANK_CHG to request interrupt
12	HBLANK_CHG_INT	'1' enables rising HBLANK_CHG to request interrupt
13	TROWS_CHG_INT	'1' enables rising TROWS_CHG to request interrupt
14	TCOLS_CHG_INT	'1' enables rising TCOLS_CHG to request interrupt
15	BROWS_CHG_INT	'1' enables rising BROWS_CHG to request interrupt
16	BCOLS_CHG_INT	'1' enables rising BCOLS_CHG to request interrupt

If multiple bits of the Interrupt Control Register are set to 1, an interrupt service routine would be required to determine the source of the interrupt by polling the Status Register. To facilitate subsequent interrupts by the same event, the interrupt service routine has to clear the interrupt source in the Status Register.

Timing Registers

Registers `cfa_reg04_active_left`, `cfa_reg05_active_right`, `cfa_reg09_total_cols`, `cfa_reg11_blank_left`, and `cfa_reg12_blank_right` take unsigned integers smaller than generic core variable `MAX_COL`. For example, if `MAX_COLS` is defined as 1024, then the registers accept 10-bit unsigned integers.

Registers `cfa_reg06_active_top`, `cfa_reg07_active_bottom`, `cfa_reg08_total_rows`, and `cfa_reg10_blank_rows` take unsigned integers smaller than generic core variable `MAX_ROWS`. For example, if `MAX_ROWS` is defined as 1024, then the registers accept 10-bit unsigned integers.

Bayer Phase Register

Bits 0 (`bayer_phase_x`), and 1 (`bayer_phase_y`) specify whether the top-left corner of the Bayer sampling grid starts with Green, Red, or Blue Pixel, according to Figure 6, which displays top-left corner of the image sample matrix along with the `cfa_reg14_bayer_phase` register value combinations.

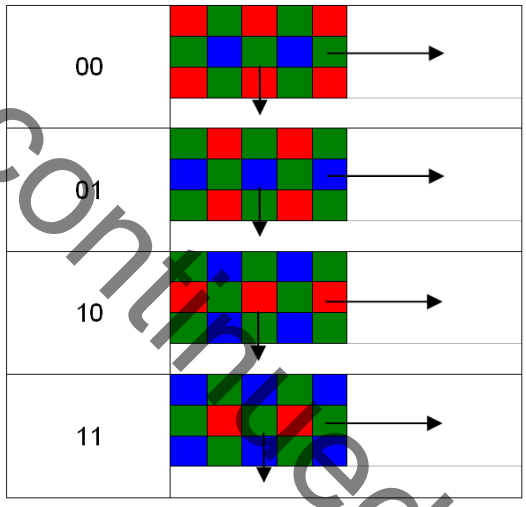

Figure 6: **Bayer Phase Register Combination Definitions**

General EDK pCore Programming Guidelines

All registers are double buffered to ensure no image tearing happens if values are modified in the active area of a frame. Exceptions are:

- All control registers
- All software reset registers
- Status and interrupt control registers
- Gamma LUT start address and data registers

Updated values for double-buffered registers are first latched into shadow registers immediately after writing. Shadow register values are copied to working registers at the rising edge of `vblank_in` when bit 1 of the corresponding block's Control Register is set to 1. Double-buffering decouples register updates from the blanking period allowing software a much larger window to update the parameter values without tearing. For more information about writing double-buffered registers, see "Writing User Registers."

More information about programming this core is provided in the API documentation available in the generated pCore directory in the following location:

```
doc/html/api/index.html
```

See the "EDK pCore Interface" section of "Core Symbol and Port Descriptions."

Writing User Registers

Figure 7 provides a software flow diagram for updating registers during the operation of the core.

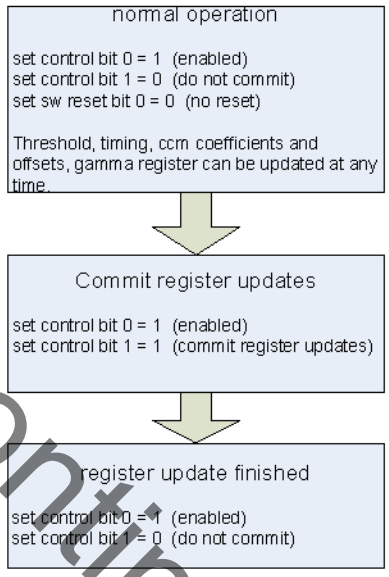

Figure 7: **Image Processing Pipeline Programming Flow Chart**

By default, the Color Filter Array Interpolation timing register working values are equal to those measured by the timing detector circuitry. The first time the Color Filter Array Interpolation shadow register values are copied to the working registers (at the rising edge of `vblank_in` when bit 1 of `cfa_reg00_control` is set to '1'), the cores will start using user-provided values. Resetting the block returns the block to work from values measured by the timing detector circuitry.

Updating the Gamma Look-Up-Tables

To implement the Gamma Correction block, the Xilinx Gamma Correction core is used with Double Buffered EDK pCore Interface selection. Therefore, gamma tables in the Image Processing Pipeline core are also double-buffered to ensure no image tearing happens while the gamma tables are updated by the system processor. One buffer, the active buffer, is used actively by the core, while the other (shadow) buffer can be updated by the processor. Double-buffering decouples updating the gamma tables from the video signal blanking period, allowing software to update the gamma tables anytime within the video frame.

The gamma register interface supports full or partial updating of gamma tables. For a full LUT update, the number of words to be written is:

- $3\text{-}2^{IWIDTH}$, when no optimization is used
- 2^{IWIDTH}, when the *Table Values are Identical for All Channels* optimization is used
- $3\text{-}2^{IWIDTH-2}$, when the *Use Interpolation* optimization is selected

When three independent LUTs are used, the Red, Green and Blue LUT contents are represented as a continuous address with the Red gamma table starting at address 0, followed by the Green and Blue tables. In order to initiate a LUT update, the starting address and the number of words to be written are needed. Bits [15-0] of register `gamma_reg4_start_addr` are designated as the start address, while bits [31-16] represent the number of words to be written into the gamma LUT(s).

Writing the `gamma_reg4_start_addr` register initiates a table update similar to a DMA command. After writing the `gamma_reg4_start_addr` register, sufficient number of data words have to be written into the shadow gamma LUT by subsequent writes to the `gamma_reg3_data` register. This may take place by a DMA command on the host processor side. A single register write to the `gamma_reg4_start_addr` register and a DMA setup command in the user application can initiate updating all three LUTs.

Data words written to the `gamma_reg3_data` register are stored in a shadow FIFO until all data words necessary for the LUT update (as prescribed by `gamma_reg4_start_addr`) are downloaded. During this download process `gamma_reg2_status` presents 5-bit information about the empty/full status of the gamma shadow buffer, which may be useful for the user application to monitor the downloading process. Once the required number of words has been received, the Gamma Correction block will wait until the next rising edge on `vblank_in` to copy the downloaded table values from the shadow buffer to the active gamma LUT. This copying process is guaranteed to finish during the vertical blanking period as long as the number of video clock cycles during the vertical blanking period is greater than the number of gamma initialization words to be copied. This condition can be met easily in typical video systems, resulting to the elimination of tearing. The copying process from the shadow to the active buffer results in the value of `gamma_reg2_status` returning to 0.

The `gamma_reg4_start_addr` register is implemented as 16 word deep FIFO in hardware, allowing the user application to initiate subsequent updates before actually supplying all the data for the previous LUT update. This functionality guarantees that no write commands are lost. Packing the start address and the number of words written into a single register allows initiating a gamma LUT update in a single register write cycle, avoiding confusion about which address and length pairs belong together when multiple write commands were issued.

Using the Interrupt Subsystem

The Image Processing Pipeline core can signal several exceptional events to the host processor using the `irq` output. Any or all of the conditions monitored by the Status Register can request an interrupt, if the interrupt control bit corresponding to the particular status bit is set to '1'.

For example, if `TOTAL_COLS`, established by the timing detector circuitry or entered dynamically through a processor interface, gets larger than `MAX_COLS`, bit 10 of the status register is set to '1'. If bit 10 of the Interrupt Control register is also set (='1') and the general interrupt enable flag (Interrupt Control Register, bit 0) is also set (='1'), then the event sets the `irq` output to '1' as well.

For the complete list of interrupt events, see the section "Status Register."

Once the interrupt is serviced by the host processor, the processor should identify the interrupt source by polling the status register, then pulsing the clear-status flag (bit 1of the Color Filter Array Interpolation Interrupt Control Register). Individual interrupt sources can be masked using the Interrupt Control Register.

General Purpose Processor Interface

The General Purpose Processor Interface exposes the timing registers as ports. This option is very useful for users designing a system with a user-defined bus interface (decoding and register banks) to an arbitrary processor. The function of the registers is identical to corresponding register functions described in Tables 1 through 4.

Double-buffering is also supported by the General Purpose Processor Interface; however the first set of registers (shadow register bank) must be supplied by the decoding logic of the user-defined bus interface. Values from external registers are copied over to the internal registers at the rising edge of `vblank_in` when bit 1 of the Control Register is set to '1'. Similar to the EDK pCore Interface, the following registers are not buffered internally (changes on these ports take immediate effect):

- All Control Registers
- All Reset Registers
- Status and Interrupt Control Registers
- Gamma LUT start address and data registers

See also the "General Purpose Processor Interface" section of "Core Symbol and Port Descriptions."

Constant Interface

The Constant Interface caters to the user who wants to interface to a particular image sensor with known, stationary timing parameters and bayer phase. For Constant Interface implementations of the Image Processing Pipeline core, the Color Filter Array Interpolation block is implemented with **Transparent Interface** selected on the Xilinx Color Filter Array Interpolation core. This selection enables automatic detection of timing parameters. The processor interface and double-buffering modules are trimmed from the design, leading to savings in FPGA resources. As there is no processor interface of any kind generated, and the core is not programmable but can be reset, enabled/disabled using the `SCLR` and `CE` ports.

See also the "Constant Interface" section of "Core Symbol and Port Descriptions."

CORE Generator - Graphical User Interface

The Image Processing Pipeline core is easily configured to meet the user's specific needs through the CORE Generator GUI. This section provides a quick reference to parameters that can be configured at generation time. Figure 8 shows the main Image Processing Pipeline screen.

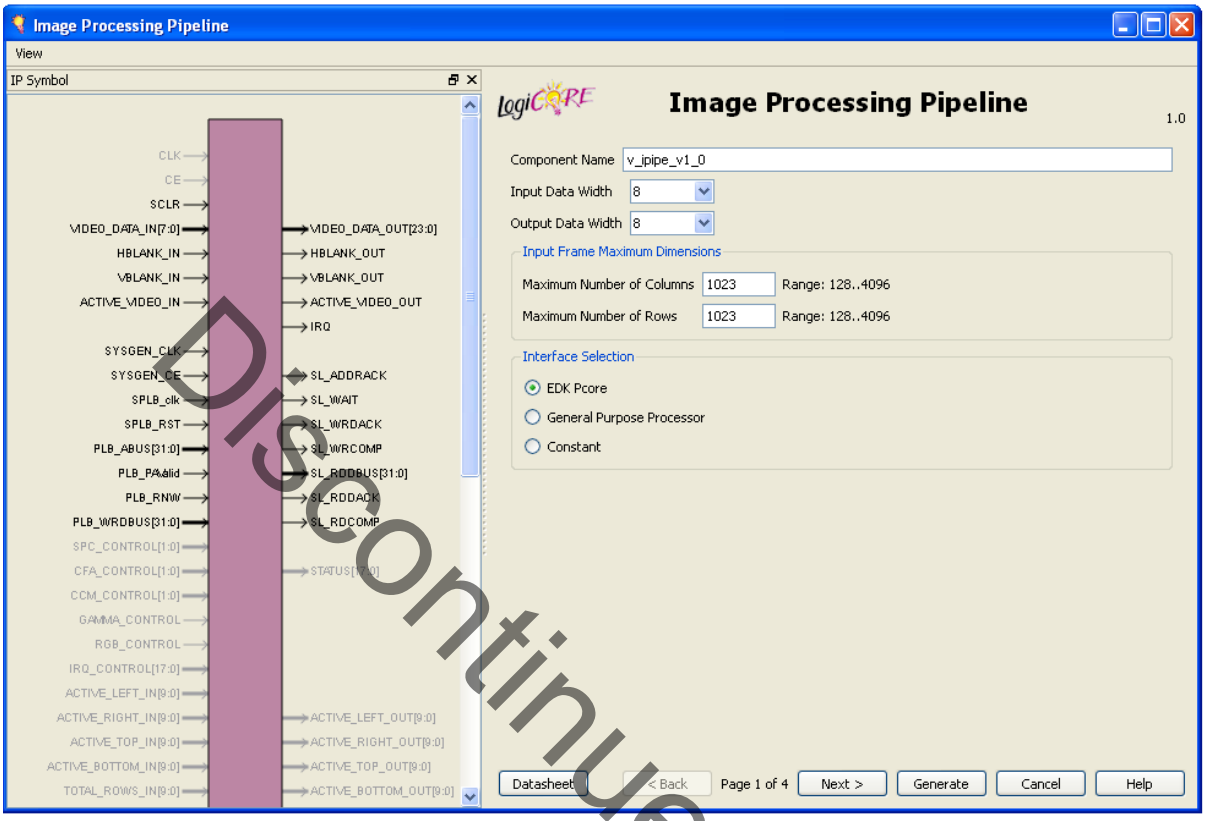

Figure 8: **Image Processing Pipeline Main Screen**

The main screen displays a representation of the IP symbol on the left side, and the parameter assignments on the right side, which are described below:

- **Component Name**: The component name is used as the base name of output files generated for the module. Names must begin with a letter and must be composed from characters: a to z, 0 to 9 and "_".
- **Input Data Width (IWIDTH)**: Specifies the bit width of input samples. Permitted values are 8, 10 and 12 bits.
- **Output Data Width (OWIDTH)**: Specifies the bit width of the output color channel for each component. Permitted values are 8, 10 and 12 bits.
- **Maximum Number of Columns (MAX_COLS)**: Specifies the maximum number of columns that can be processed by the core. Permitted values are from 128 to 2048. Specifying this value is necessary to establish the internal widths of counters and control-logic components as well as the depth of line buffers. Using a tight upper-bound on possible values of `TOTAL_COLS` results in optimal block RAM usage. However, feeding the configured Color Filter Array Interpolation instance timing signals which violate the `MAX_COLS` constraint will lead to data and output timing signal corruption and is flagged by the status register.

- **Maximum Number of Rows (MAX_ROWS)**: Specifies the maximum number of rows that can be processed by the core. Permitted values are from 128 to 2048. Specifying this value is necessary to establish the internal widths of counters and control-logic components. Feeding the configured Color Filter Array Interpolation instance timing signals which violate the `MAX_ROWS` constraint will lead to data and output timing signal corruption and is flagged by the status register.
- **Interface Selection**: This option allows selection from the three core interface options described previously in the "Processor Interfaces" section.
  - **EDK pCore Interface**: CORE Generator will generate a pCore which can be easily imported into an EDK project as a hardware peripheral. For more information see the "EDK pCore Interface" section of "Processor Interfaces."
  - **General Purpose Processor Interface**: CORE Generator will generate a set of ports to be used to program the core. For more information see the "General Purpose Processor Interface" section of "Processor Interfaces"
  - **Constant Interface**: Parameters provided on screens 2, 3, and 4 of the GUI are constant, and therefore no programming is necessary. The processor interfaces and double-buffering modules are trimmed from the design leading to savings in FPGA resources.

Color Filter Array Interpolation Default Values

Default Values: The second screen of the GUI (Figure 9) allows the definition of default timing, polarity, Bayer Phase and interrupt control values. For the Constant Interface, Interrupt Control and Bayer Phase, values are permanent for the generated Color Filter Array Interpolation instance, while Timing Initialization values are discarded once valid input timing signals are detected.

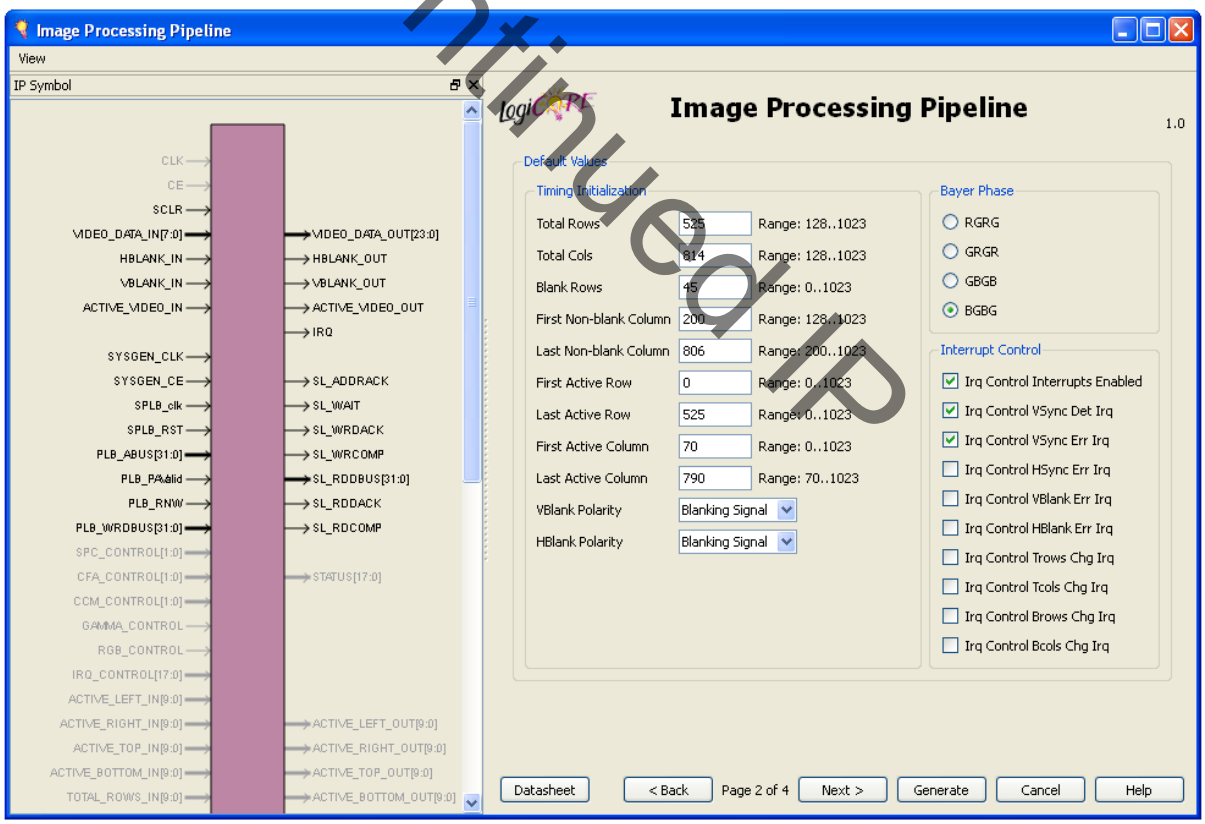

Figure 9: **Color Filter Array Interpolation, Default Values Screen**

- **Timing Initialization**: For the definition of Timing Initialization generic parameters, see the "Timing Parameters" section.
- **Bayer Phase**: Based on the data sheet of the particular image sensor used and the particular register settings of the sensor, identify where the top-left corner of total area falls on the CFA matrix. For the first two samples, four combinations are possible. For RGB sensors, these are RG, GR, BG, GB. For CMY sensors the combinations are MY, YM, CY, YC. See the "Bayer Phase Register" section.
- **Interrupt Control**: For the definition of Interrupt Control generic parameters, see "Using the Interrupt Subsystem."

Color Correction Matrix Default Values

See Figure 10.

- **Coefficient Matrix**: Enter the floating-point coefficients ranging from [-4, 4) (K in Equation 1) which are represented by 18 bit coefficients with 15 fractional bits. The entered values will be the default used to initialize the core and the values used when the core is reset.
- **Offsets**: Enter the offset coefficients (O in Equation 1). These signed coefficients are OWIDTH+1 bits wide.

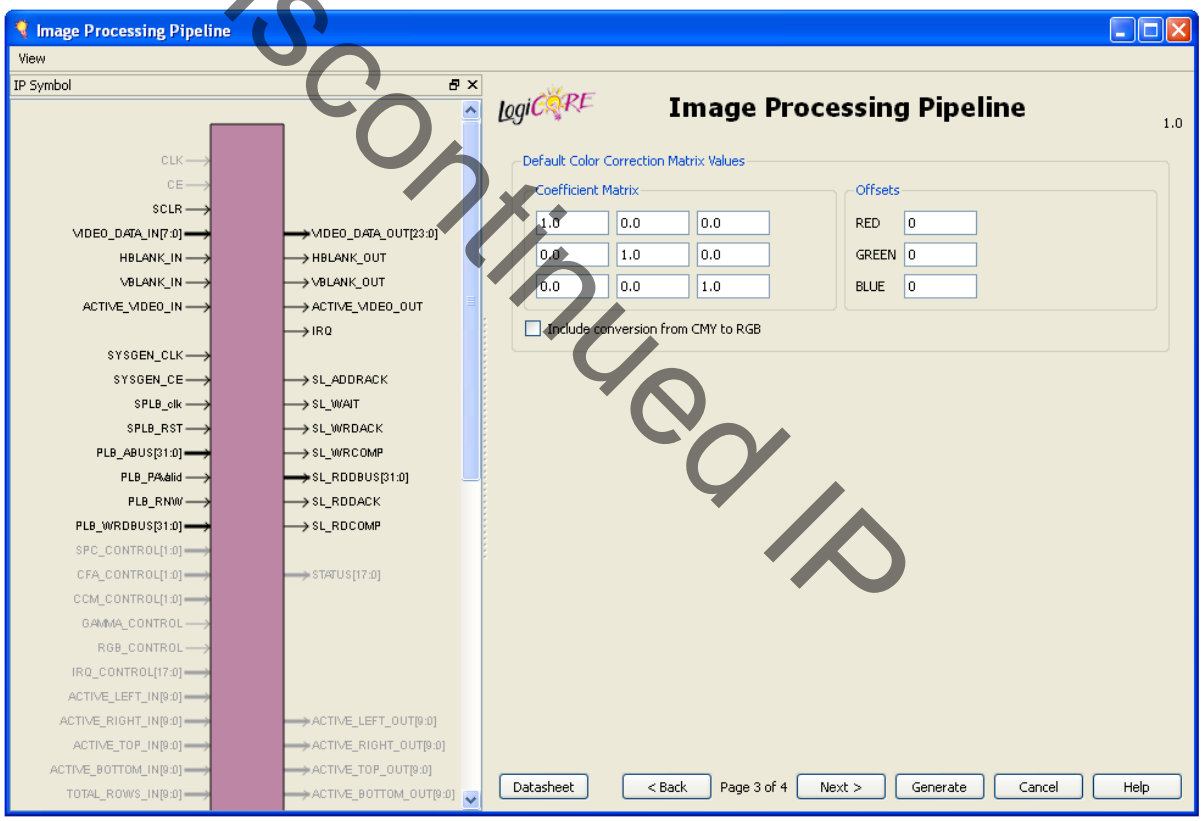

Figure 10: **Color Correction Matrix, Default Values Screen**

- **Include conversion from CMY to RGB**: The Color Correction Matrix block can accept either RGB or CMY inputs. When selecting this conversion, the core will modify the coefficient matrix to also convert the CMY input to RGB output. The equation to calculate the new coefficient matrix is shown in Equation 2:

Equation 2

$$K' = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

Defective Pixel Correction Default Values

See Figure 11.

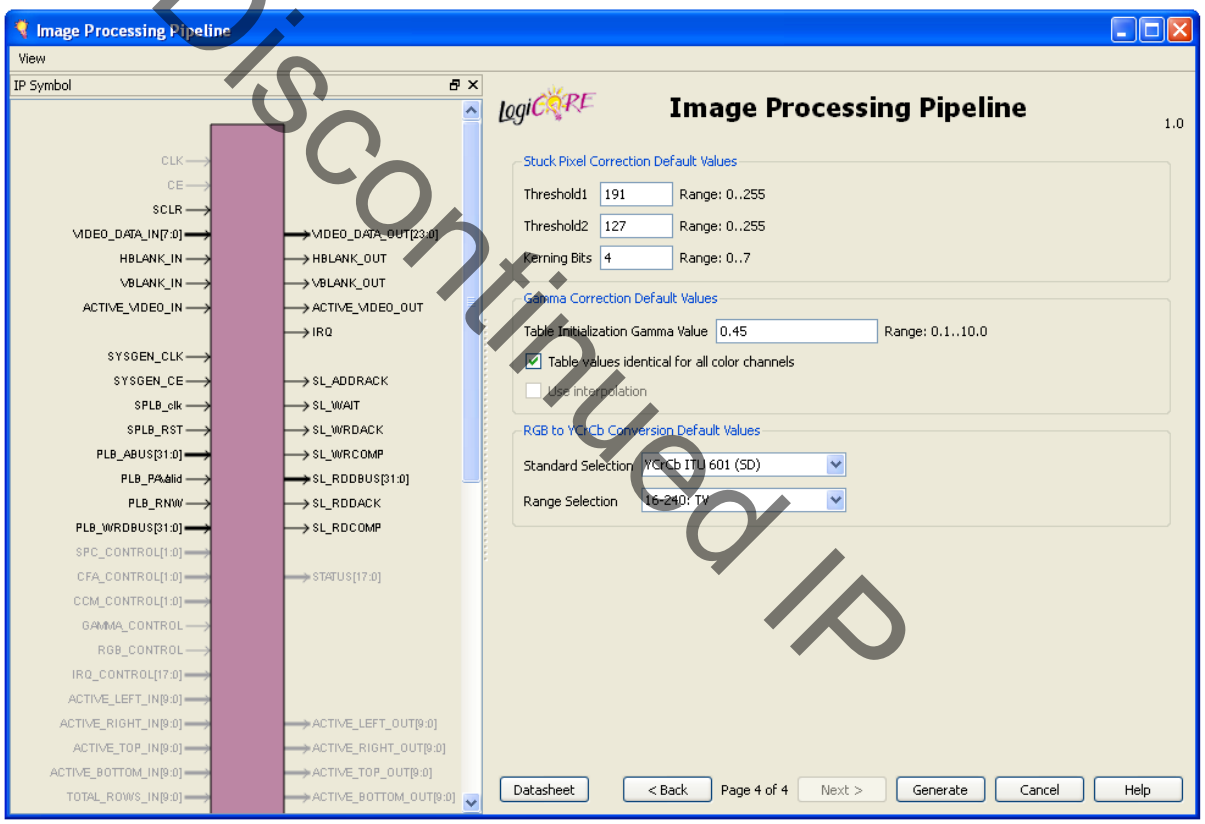

Figure 11: **Defective Pixel Correction and Gamma Correction, Default Values Screen**

- **Threshold1:** Specifies the threshold for comparing against the average of nearest neighbor sample values minus the current sample value scaled to match the Kerning Bits.
- **Threshold 2**: Specifies the threshold for comparing against the average of same color sample values minus the current sample value scaled to match the Kerning Bits.
- **Kerning Bits**: Specifies the number of least significant bits to drop during calculations. Threshold1 and Threshold2 must be scaled accordingly.

Gamma Correction Default Values

- **Table Initialization Gamma Value**: Specifies the gamma value for initializing the look up tables. Permitted values are from 0.1 to 10.

  Optimization: Specifies options to reduce memory usage (see the "Optimization Options" section of the Xilinx Gamma Correction core data sheet (www.xilinx.com/products/ipcenter/EF-DI-GAMMA.htm).

  - **Tables identical for all color channels**: The red, green, and blue channels all have the same gamma table.
  - **Use interpolation**: Interpolation is used to reduce block RAM counts.

RGB to YCrCb Color Space Conversion

Standard Selection: Select the standard to be implemented. The RGB to YCrCb module has coefficients and offset pre-programmed to support the following standards:

- YCrCb ITU 601 (SD)
- YCrCb ITU 709 (HD) 1125/60 (PAL)
- YCrCb ITU 709 (HD) 1250/50 (NTSC)
- YUV

When **No RGB to YCrCb Conversion** is selected, the conversion module is bypassed, resulting in savings of logic resources. In certain applications the Color Correction Matrix block may also perform an additional color-space conversion operation, if the conversion coefficients can be factored into the color correction matrix being used and the gamma correction functions can be pre-distorted to reflect the effect of linear operations before gamma operations.

Output Range Selection: This selection governs the range of outputs Y, Cr, and Cb by affecting the conversion coefficients as well as the clipping and clamping values. The core supports the following typical output ranges:

- 16 to 235, typical for studio equipment
- 16 to 240, typical for broadcast or television
- 0 to 255, typical for computer graphics

Output clipping and clamping values are the same for Luminance and Chrominance channels. These ranges are characteristic of 8-bit outputs. If 10- or 12-bit outputs are used, the ranges are extended proportionally. For example, 16 to 240 mode for 10-bit outputs will result in output values ranging from 64 to 960.

Core Symbol and Port Descriptions

The Image Processing Pipeline core can be configured with three different interface options, each resulting in a slightly different set of ports. The Streaming Video Interface is a set of signals which is common to all three interface options and to all video iPipe cores. It is described in Table 9.

Constant Interface

The Constant Interface has no ports other than the Streaming Video Interface, as this interface does not provide any additional programmability. The Constant Core Symbol is shown in Figure 12 and is described by Table 9. The Streaming Video Interface is a set of signals which is common in all interface options.

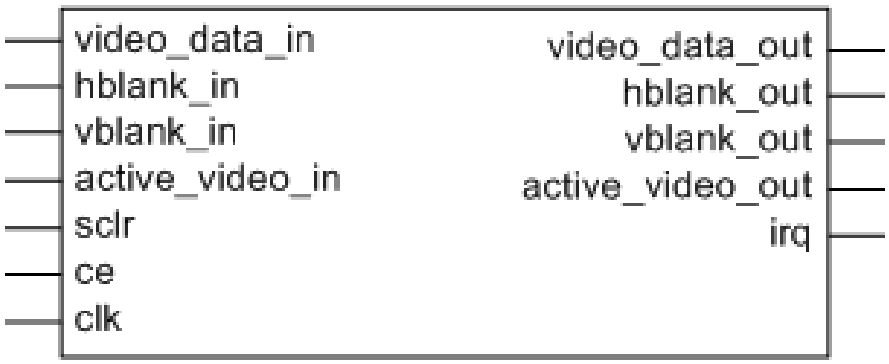

Figure 12: **Core Symbol for the Constant Interface**

Table 9: **Port Descriptions for the Constant Interface**

Port Name	Port Width	Direction	Description
video_data_in	IWIDTH	IN	Data input bus
hblank_in	1	IN	Horizontal blanking input
vblank_in	1	IN	Vertical blanking input
active_video_in	1	IN	Active video signal input
video_data_out	*3*OWIDTH*	OUT	Data output bus
hblank_out	1	OUT	Horizontal blanking output
vblank_out	1	OUT	Vertical blanking output
active_video_out	1	OUT	Active video signal output
irq	1	OUT	Interrupt request
clk	1	IN	Rising-edge clock
ce	1	IN	Clock enable (active high)
sclr	1	IN	Synchronous clear – reset (active high)

- **video_data_in**: This is the sample input bus for Bayer patterned data. IWIDTH bits wide color values are expected in unsigned integer representation.
- **hblank_in**: The `hblank_in` signal conveys information about the blank/non-blank regions of video scan lines.
- **vblank_in**: The `vblank_in` signal conveys information about the blank/non-blank regions of video frames, and is used by the core to detect end of a frame, when user registers can be copied to active registers to avoid visual tearing of the image.
- **active_video_in**: This signal is high when valid data is presented at the input.
- **clk - clock**: Master clock in the design, synchronous with, or identical to the video clock.
- **ce - clock enable**: Pulling `CE` low suspends all operations within the core. Outputs are held, no input signals are sampled, except for reset (`SCLR` takes precedence over `CE`).
- **sclr - synchronous clear**: Pulling `SCLR` high results in resetting all output ports to zero or their default values. Internal registers within the XtremeDSP slice and D-flip-flops are cleared. However, the core uses SRL16/SRL32 based delay lines for `hblank_out`, `vblank_out`, and `active_video_out` generation, which are not cleared by `SCLR`. This may result in non-zero outputs after `SCLR` is deasserted, until the contents of SRL16/SRL32s are flushed. Unwanted results can be avoided if `SCLR` is held active for the processing latency of the core which is seven clock cycles.

- **video_data_out**: This bus contains output in the order shown below. Color values are represented as OWIDTH bits wide unsigned integers.

Bits	3*OWIDTH*-1:2*OWIDTH*	2*OWIDTH*-1:*OWIDTH*	*OWIDTH*-1:0
video_data_out signals	Red / Cyan / Cb	Blue / Magenta / Cr	Green / Yellow / Y

- **hblank_out** and **vblank_out**: The corresponding input signals are delayed so blanking outputs are in phase with the video data output, maintaining the integrity of the video stream.
- **active_video_out**: The `active_video_out` signal is high when valid data is present at the output. The `active_video` signal does not affect the processing behavior of the core. Asserting or deasserting it will not stall processing or the video stream, nor will it force video outputs to zero.
- **irq**: The interrupt request port can drive the interrupt input of the host processor to signal exceptional events. For more information see "Using the Interrupt Subsystem."

EDK pCore Interface

The EDK pCore Interface generates Processor Local Bus (PLB4.6) interface ports in addition to the Streaming Video Signals. The PLB bus signals are automatically connected when the generated pCore is inserted into an EDK project. The core symbol for the EDK pCore Interface is shown in Figure 13. The Streaming Video Interface is described in the previous section (Table 9). For information about programming the EDK pCore, see the "EDK pCore Interface" section of "Processor Interfaces." For more information on the PLB bus signals, see Processor Local Bus (PLB) v4.6.

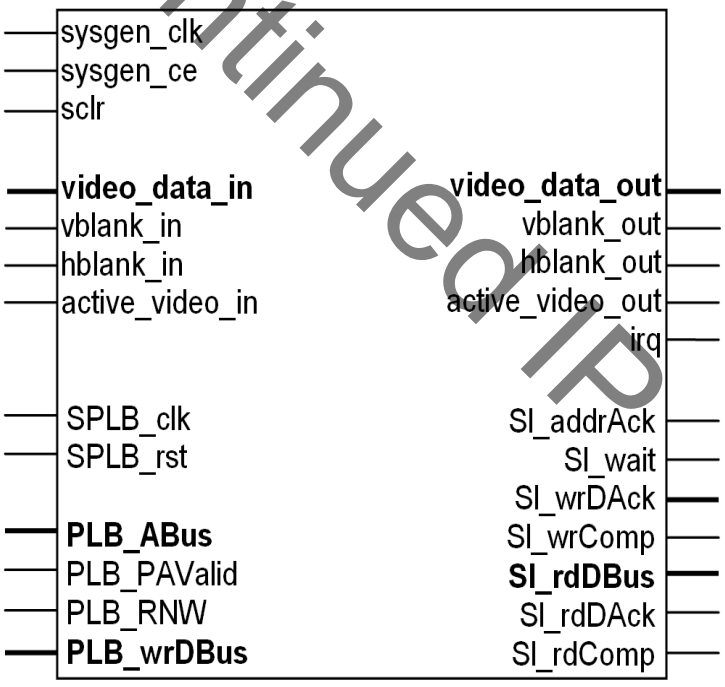

Figure 13: **Core Symbol for the EDK pCore Interface**

General Purpose Processor Interface

The General Purpose Processor Interface exposes all programmable registers as ports. The Core Symbol for the General Purpose Processor Interface is shown in Figure 14. The Streaming Video Interface is described Table 9. The General Purpose Processor ports are described in Table 10.

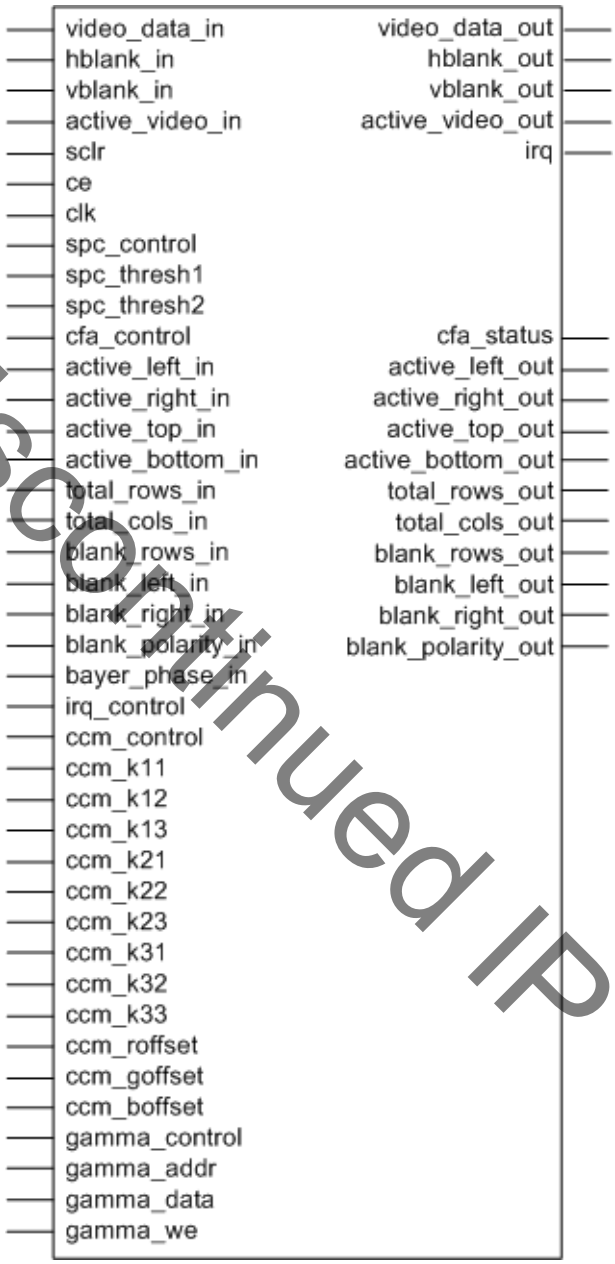

Figure 14: **Core Symbol for the General Purpose Processor Interface**

Table 10: **Optional Ports for the General Purpose Processor Interface**

Port Name	Port Width	Direction	Description
spc_control	2	IN	Bit 0: Software enable Bit 1: Host processor write done semaphore • '0' indicates host processor actively updating registers • '1' indicates register update completed by host processor
spc_thresh1	WIDTH	IN	Threshold for comparing against nearest neighbors
spc_thresh2	WIDTH	IN	Threshold for comparing against same color pixels
cfa_control	4	IN	Bit 0: Software enable Bit 1: Host processor write done semaphore. Bit 2: Clear status registers (clears interrupt source) Bit 3: Reserved
active_left_in	COLS_WIDTH	IN	User defined value for ACTIVE_LEFT(1)
active_right_in	COLS_WIDTH	IN	User defined value for ACTIVE_RIGHT(1)
active_top_in	ROWS_WIDTH	IN	User defined value for ACTIVE_TOP(1)
active_bottom_in	ROWS_WIDTH	IN	User defined value for ACTIVE_BOTTOM(1)
total_rows_in	COLS_WIDTH	IN	User defined value for TOTAL_ROWS(1)
total_cols_in	COLS_WIDTH	IN	User defined value for TOTAL_COLS(1)
blank_rows_in	ROWS_WIDTH	IN	User defined value for BLANK_ROWS(1)
blank_left_in	COLS_WIDTH	IN	User defined value for BLANK_LEFT(1)
blank_right_in	COLS_WIDTH	IN	User defined value for BLANK_RIGHT(1)
blank_polarity_in	1	IN	Blanking polarity input Bit 0: Horizontal blanking Bit 1: Vertical blanking • '0' forces active low (blanking) signaling(2) • '1' forces active high (valid) signaling(2)
bayer_phase_in	2	IN	See section "Bayer Phase Register"
irq_control	18	IN	See section "Using the Interrupt Subsystem"
cfa_status	18	OUT	Status register
active_left_out	COLS_WIDTH	OUT	Value actively used for ACTIVE_LEFT(1)
active_right_out	COLS_WIDTH	OUT	Value actively used for ACTIVE_RIGHT(1)
active_top_out	ROWS_WIDTH	OUT	Value actively used for ACTIVE_TOP(1)
active_bottom_out	ROWS_WIDTH	OUT	Value actively used for ACTIVE_BOTTOM(1)
total_rows_out	COLS_WIDTH	OUT	Value actively used for TOTAL_ROWS(1)
total_cols_out	COLS_WIDTH	OUT	Value actively used for TOTAL_COLS(1)
blank_rows_out	ROWS_WIDTH	OUT	Value actively used for BLANK_ROWS(1)
blank_left_out	COLS_WIDTH	OUT	Value actively used for BLANK_LEFT(1)
blank_right_out	COLS_WIDTH	OUT	Value actively used for BLANK_RIGHT(1)

Table 10: **Optional Ports for the General Purpose Processor Interface *(Cont'd)***

blank_polarity_out	1	OUT	Blanking polarity output Bit 0: Horizontal blanking • Bit 1: Vertical blanking • '0' forces active low (blanking) signaling(2) • '1' forces active high (valid) signaling(2)
ccm_control	2	IN	Bit 0: Software enable Bit 1: Host processor write done semaphore • '0' indicates host processor actively updating registers • '1' indicates register update completed by host processor
ccm_k11	18	IN	Matrix coefficient K1,1(3)
ccm_k12	18	IN	Matrix coefficient K1,2(3)
ccm_k13	18	IN	Matrix coefficient K1,3(3)
ccm_k21	18	IN	Matrix coefficient K2,1(3)
ccm_k22	18	IN	Matrix coefficient K2,2(3)
ccm_k23	18	IN	Matrix coefficient K2,3(3)
ccm_k31	18	IN	Matrix coefficient K3,1(3)
ccm_k32	18	IN	Matrix coefficient K3,2(3)
ccm_k33	18	IN	Matrix coefficient K3,3(3)
ccm_roffset	OWIDTH+1	IN	Red offset
ccm_goffset	OWIDTH+1	IN	Green offset
ccm_boffset	OWIDTH+1	IN	Blue offset
gamma_control	1	IN	Software enable
gamma_addr	IWIDTH+2	IN	LUT address bus (two additional MSBs are necessary only when no optimizations are used)
gamma_data	OWIDTH	IN	LUT initialization data corresponding to address provided by gamma_addr
gamma_we	1	IN	LUT write enable

1. See the section "Timing Parameters," page 4.
2. See the section "Blanking Signal Polarities," page 5.
3. 18 bit fixed point numbers with 15 fractional bits. For example, the floating point value of 1.5 is represented as 18'b011000000000000000 or 0x00018000.

Control Signals and Timing

Figure 15 presents an example for a very short video frame, having only six non-blank lines. The active area covers the full non-blank area.

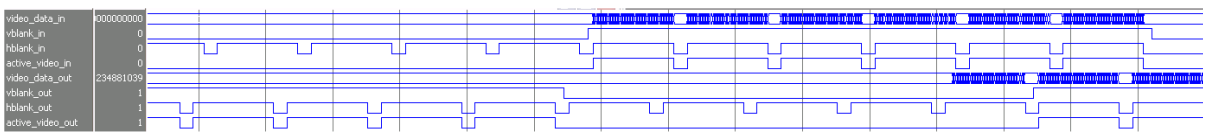

Figure 15: **Vertical Timing Example**

The propagation delay of the Image Processing Pipeline core depends on actual parameterization, but is at least four full line-times. Deasserting CE suspends processing, which may be useful for data-throttling, to temporarily cease processing of a video stream to match the delay of other processing components.

Timing Tolerances

Due to state-machine setup and reset constraints internal to the Color Filter Array Interpolation block, the following limitations have to be observed when configuring the image sensor to be used in conjunction with the core:

1. BLANK_ROWS > 4
2. ACTIVE_LEFT > 3
3. BLANK_LEFT <= ACTIVE_LEFT
4. ACTIVE_RIGHT < TOTAL_COLS - 5
5. BLANK_RIGHT >= ACTIVE_RIGHT

Figure 16 presents an example where the preceding conditions are met.

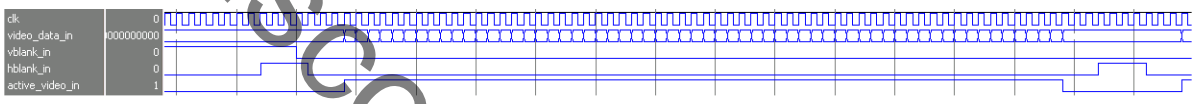

Figure 16: **Minimum Horizontal Timing Tolerances**

Timing parameters for Figure 16 are as follows:

BLANK_LEFT =1

(clock cycles between a falling edge of vblank_in and the next falling edge of hblank_in)

ACTIVE_LEFT = 4

(clock cycles between a falling edge of vblank_in and the next rising edge of active_video_in)

ACTIVE_RIGHT = 63

(clock cycles between a falling edge of vblank_in and the next falling edge of active_video_in)

BLANK_RIGHT = 66

(clock cycles between a falling edge of vblank_in and the next rising edge of hblank_in)

TOTAL_COLS = 70

(clock cycles between falling edges of hblank_in)

The non-blanked horizontal area can be flush with the active area (ACTIVE_LEFT = BLANK_LEFT; ACTIVE_RIGHT = BLANK_RIGHT), which enables the user to create active_video_in using a simple logical assignment in case the particular image sensor does not provide the active video signal.

active_video_in = (hblank_in XNOR hblank_polarity) AND (vblank_in XNOR vblank_polarity)

Core Resource Utilization and Performance

For an accurate measure of the usage of device resources (for example, block RAMs, flip-flops, and LUTs) for a particular instance, click **View Resource Utilization** in CORE Generator after generating the core.

Information presented in Tables 11 and 12 is a guideline to the resource utilization of the Image Processing Pipeline core (with a Constant Interface and no Color-Space Conversion) for Virtex-5 and Spartan-3ADSP FPGA families. This core does not use any dedicated IO or clock resources. The design was tested using Xilinx ISE® v11.1i tools with area constraints (see table footnotes) and default tool options.

Table 11: **Resource Utilization and Target Speed for Virtex-5 - xc5vsx50t-1ff665(1)(2)**

Input Width	Output Width	Maximum Number of Columns and Rows	FFs	LUTs	Slices	Xtreme DSPs	Block RAMs	Clock Frequency (MHz)
8	8	1024	1199	4878	4851	17	13	235
8	10	1024	1209	4865	4834	17	13	232
8	12	1024	1192	4870	4835	17	13	230
10	8	1024	1372	4401	4373	17	14	234
10	10	1024	1325	3916	3884	17	14	231
10	12	1024	1361	4404	4368	17	14	233
12	8	1024	1546	4866	4837	17	18	242
12	10	1024	1550	4968	4935	17	21	234
12	12	1024	1516	4298	4261	17	21	237
8	8	2048	1227	4874	4847	17	17	232
8	10	2048	1196	4871	4840	17	17	233
8	12	2048	1202	4870	4835	17	17	234
10	8	2048	1358	4399	4371	17	20	223
10	10	2048	1429	3913	3881	17	20	227
10	12	2048	1402	3912	3876	17	20	227
12	8	2048	1612	4298	4269	17	24	233
12	10	2048	1572	4971	4938	17	27	231
12	12	2048	1569	3811	3774	17	27	226

1. Speedfile: PRODUCTION 1.64 2009-02-18, STEPPING level 0
2. Area constraint: 59 rows x 22 columns in the CLB matrix

Table 12: **Resource Utilization and Target Speed for Spartan3A DSP - xc3sd3400a-5fg676(1)(2)**

Input Width	Output Width	Maximum Number of Columns and Rows	FFs	LUTs	Slices	Xtreme DSPs	Block RAMs	Clock Frequency (MHz)
8	8	1024	3679	4878	4851	17	13	160
8	10	1024	3685	4865	4834	17	13	163
8	12	1024	3691	4870	4835	17	13	160
10	8	1024	4255	4403	4375	17	14	160
10	10	1024	4259	3916	3884	17	14	158
10	12	1024	4265	4400	4364	17	14	158
12	8	1024	4875	4870	4841	17	18	158
12	10	1024	4881	4966	4933	17	21	154
12	12	1024	4887	3810	3773	17	21	154
8	8	2048	3679	4876	4849	17	13	157
8	10	2048	3741	4871	4840	17	17	159
8	12	2048	3747	4870	4835	17	17	158
10	8	2048	4309	4399	4371	17	20	158
10	10	2048	4315	3913	3881	17	20	156
10	12	2048	4323	3912	3876	17	20	155
12	8	2048	4931	4298	4269	17	24	158
12	10	2048	4937	4971	4938	17	27	155
12	12	2048	4943	3811	3774	17	27	156

1. Speedfile: PRODUCTION 1.33 2009-02-18
2. Area constraint: 43 rows x 28 columns in the CLB matrix

Simulator and Operating System Support

Simulator Support

- ISE Simulator (ISIM) v11.1
- Mentor Graphics ModelSim 6.4b and above

OS Support

- Generation only on Windows-XP 32-bit

Known Issues

For for the latest Known Issues see XTP025.

www.xilinx.com/support/documentation/user_guides/xtp025.pdf

References

1. Xilinx Defective Pixel Correction LogiCORE IP Data Sheet (DS721)
   www.xilinx.com/products/ipcenter/EF-DI-DEF-PIX-CORR.htm
2. Xilinx Color Filter Array Interpolation LogiCORE IP Data Sheet (DS722)
   www.xilinx.com/products/ipcenter/EF-DI-CFA.htm
3. Xilinx Color Correction Matrix LogiCORE IP Data Sheet (DS720)
   www.xilinx.com/products/ipcenter/EF-DI-CCM.htm
4. Xilinx Gamma Correction LogiCORE IP Data Sheet (DS719)
   www.xilinx.com/products/ipcenter/EF-DI-GAMMA.htm
5. Xilinx RGB to YCrCb Color-Space Conversion Data Sheet (DS657))
   www.xilinx.com/products/ipcenter/RGB_to_YCrCb.htm
6. *ITU Recommendation BT.601-5*, International Telecommunication Union, 1995.
7. *ITU Recommendation BT.709-5*, International Telecommunication Union, 2002.
8. Eastman Kodak Company: KAC - 1310, 1280 x 1024 SXGA CMOS Image Sensor Technical Data
9. Aptina MT9P031: 1/2.5-Inch 5Mp Digital Image Sensor Features
10. Processor Local Bus (PLB) v4.6

Support

Xilinx provides technical support for this Xilinx LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

License Options

The Image Processing Pipeline core provides three licensing options. After installing the required Xilinx ISE software and IP Service Packs, choose a license option.

Simulation Only

The Simulation Only Evaluation license key is provided with the Xilinx CORE Generator tool. This key lets you assess core functionality with either the example design provided with the Image Processing Pipeline core, or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically generated HDL structural model.)

Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place-and-route the design, evaluate timing, and perform functional simulation of the Image Processing Pipeline core using the example design and demonstration test bench provided with the core.

In addition, the license key lets you generate a bitstream from the placed and routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before timing out (ceasing to function), at which time it can be reactivated by reconfiguring the device.

Full

The Full license key is available when you purchase the core and provides full access to all core functionality both in simulation and in hardware, including:

- Functional simulation support
- Full implementation support including place and route and bitstream generation
- Full functionality in the programmed device with no time outs

Obtaining Your License Key

This section contains information about obtaining a simulation, full system hardware, and full license keys.

Simulation License

No action is required to obtain the Simulation Only Evaluation license key; it is provided by default with the Xilinx CORE Generator software.

Full System Hardware Evaluation License

To obtain a Full System Hardware Evaluation license, do the following:

1. Navigate to the product page for this core from:
   www.xilinx.com/products/ipcenter/EF-DI-IMG-PIPE.htm
2. Click Evaluate.
3. Follow the instructions to install the required Xilinx ISE software and IP Service Packs.

Obtaining a Full License

To obtain a Full license key, you must purchase a license for the core. After doing so, click the "Access Core" link on the Xilinx.com IP core product page for further instructions.

Installing Your License File

The Simulation Only Evaluation license key is provided with the ISE CORE Generator system and does not require installation of an additional license file. For the Full System Hardware Evaluation license and the Full license, an email will be sent to you containing instructions for installing your license file. Additional details about IP license key installation can be found in the ISE Design Suite Installation, Licensing and Release Notes document.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
04/09/09	0.5	Early access release
04/24/09	1.0	Initial Xilinx release

Notice of Disclaimer