Introduction

The Xilinx Image Characterization LogiCORE™ IP calculates important statistical data for video input streams. The Image Characterization LogiCORE is an important processing block for many applications including face recognition and object detection. The statistics provided by this core include means and variances for luminance, chrominance, high and low frequencies, edges, and motion on both a global and block basis. The Image Characterization core supports 8-bit pixel data in YUV 4:2:2 or 4:2:0 as well as 8-bit motion data at up to 1080p 30 fps. The core is programmable through a comprehensive register interface for setting edge gains, high-pass gain, color selects (hue and saturation), and block size. The Image Characterization LogiCORE is available with two different interfaces: General Purpose Processor and EDK pCore (including device driver).

Features

- Programmable register control
- Selectable processor interface
  - EDK pCore
  - General Purpose Processor
- Global and Block Means and Variances for:
  - Luminance/Chrominance Content
  - Frequency Content
  - Edge Content
  - Motion Content
  - Color Content
- Global Histograms for:
  - Luminance
  - Chrominance
  - Hue

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family(1)	Spartan®-3A DSP, Spartan-6, Virtex®-5, Virtex-6				
Supported User Interfaces	General Processor Interface, EDK PLB 4.6				
	Resources(2)				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs(3)**	**Max. Freq.(4)**
1280x720 Max Frame Size, 16x16 block size, pCore Interface	8735	10488	35	16	225
Provided with Core					
Documentation	Product Specification				
Design Files	Netlist or EDK pCore				
Example Design	Not Provided				
Test Bench	Not Provided				
Constraints File	Not Provided				
Simulation Model	Not Provided				
Tested Design Tools					
Design Entry Tools	ISE® 12.3, XPS 12.3				
Simulation	ModelSim v6.5c, Xilinx® ISim 12.3				
Synthesis Tools	XST 12.3				
Support					
Provided by Xilinx, Inc.					

1. For a complete listing of supported devices, see the release notes for this core.
2. Resources listed here are for Virtex-6® devices. For more complete device performance numbers, see "Core Resource Utilization," page 34.
3. Based on 36K block RAMs.
4. Performance numbers listed are for Virtex-6 FPGAs. For more complete performance data, see "Performance," page 37.

- Support for 8-bit, YUV 4:2:2 or YUV 4:2:0 data
- Support for 8-bit motion data
- Support for block sizes of 4x4, 8x8, 16x16, 32x32 or 64x64 pixels
- Support for image sizes up to 1920x1080p @ 30 fps or 1280x720p @ 60 fps
- Support for streaming or frame buffer based processing
- PLB46 support for interrupts and register access
- For use with Xilinx CORE Generator™ 12.3 or later

Applications

- Video Surveillance
- Industrial Imaging
- Video Conferencing
- Machine Vision
- Automotive
- Other video applications requiring image analysis

Overview

The Image Characterization LogiCORE IP is comprised of a collection of blocks that work together to calculate statistical data that can be used to describe an image in the analytics domain. These statistics are calculated on a global basis for the entire image as well as on a block basis which is implemented as a 2-D grid of NxN subdivisions of the image. The resulting image statistics are written to memory and can be read by another IP core or by software to implement complex analytics applications.

The main global and block statistical measures performed are:

- $\bar{x} = \frac{1}{M}\sum_{i=0}^{M-1} x_i$, the mean values
- $\sigma^2 = \frac{1}{M}\sum_{i=0}^{M-1} x_i^2 - \bar{x}^2$, the variance values

These statistical tools are applied to image content, including:

- Luminance/Chrominance Content
- Low Frequency Content
- High Frequency Content
- Color Content
- Edge Content
- Motion Content

In addition to the preceding statistics, histograms are also calculated on the global image. Histograms are generated for the Luminance, Chrominance (Cr and Cb), and Hue components.

The Image Characterization LogiCORE IP shares a number of similarities with the Image Statistics LogiCORE IP. At first glance the two IP cores may seem redundant, but the practical uses for these IP cores are actually quite different. They are designed for use in different portions of the video processing pipeline.

The Image Statistics LogiCORE IP calculates a set of statistics for 16 user-defined zones. The particular statistics that the Image Statistics core gathers are optimized for use in control algorithms such as Auto-Focus or Auto-White balance.

In contrast, the Image Characterization core calculates a different set of statistics for a 2-D grid of NxN blocks. This results in a much finer-grained description of the image. The particular statistics that the Image Characterization core gathers have been optimized for use in analytics applications such as Video Surveillance or Road-Sign Recognition.

Architecture

The Image Characterization core is implemented as four subsystems: YC Processing, Block Stats, Global Stats, and Histograms. These subsystems are connected as shown in Figure 1.

The details of each sub-system are discussed in the following sections.

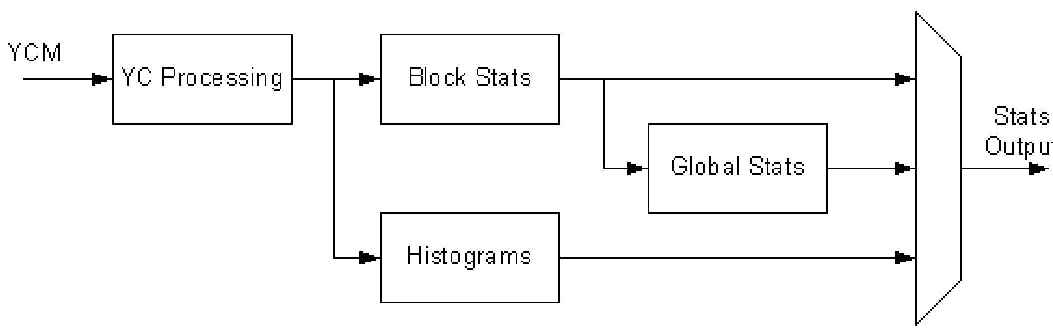

Figure 1: **Image Characterization Block Diagram**

YCM Data Bus

The Image Characterization core has a simple input interface that incorporates the YCM data bus. The YCM data bus uses a 24-bit word with Luma (Y), Chroma (C), and Motion (M) placed as noted in Table 1. Any unused portion of the YCM data bus can be simply tied to a constant value.

Table 1: **YCM Data Bus**

YCM[23:16]	YCM[15:8]	YCM[7:0]
Motion[7:0]	Chroma[7:0]	Luma[7:0]

Motion data is a magnitude measurement of how much a Luma pixel has changed between the current frame and the previous frame. The Xilinx Motion Adaptive Noise Reduction LogiCORE IP can be used to calculate the motion content of a video sequence and drive the YCM input to the Image Characterization core.

4:2:2 and 4:2:0 Formatting

The formatting of 4:2:2 and 4:2:0 data can often be a source of confusion. The formats used by the Image Characterization core are illustrated in Figure 2 and Figure 3. Figure 2 shows the alignment of Y, C and M in relation to each other and the `active_video_in` signal. This arrangement applies to both 4:2:2 and 4:2:0. The figure illustrates a line with 720 active pixels per line.

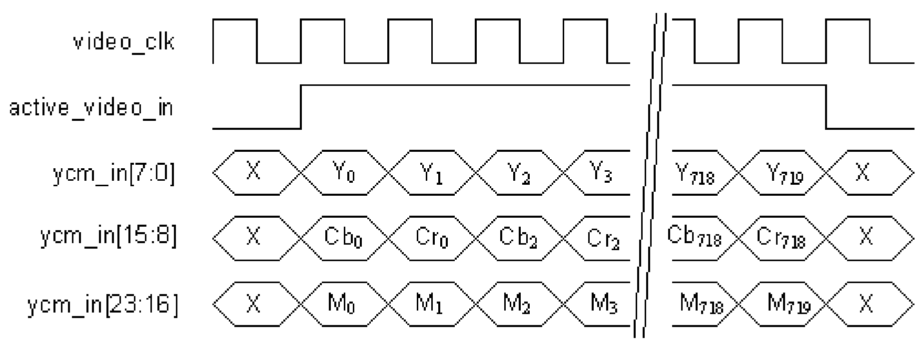

Figure 2: **4:2:2 or 4:2:0 Bus Format**

Figure 3 shows a high level view of a 4:2:0 YCM bus. Notice the arrangement of the bus in relation to the `chroma_in` signal. The `chroma_in` toggles every line. When `chroma_in` is '1' the values on the 'C' portion of the bus are valid. When `chroma_in` is '0' the values on the 'C' portion of the bus are invalid. The `chroma_in` signal only changes while the `active_video_in` signal is '0,' which denotes that the data bus is not valid. For a 4:2:2 YCM bus, the `chroma_in` signal would be '1' for every line instead of toggling.

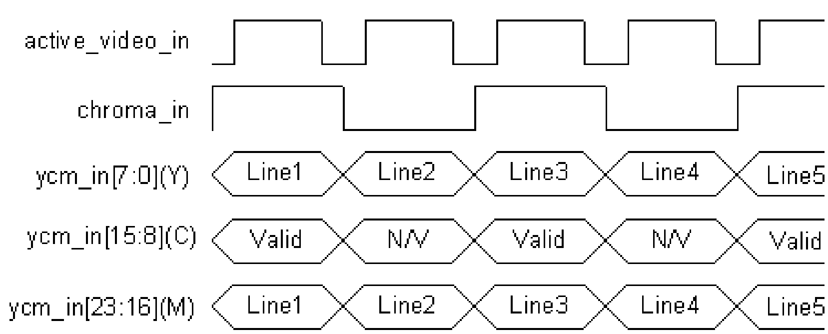

Figure 3: **4:2:0 High Level View**

YC Processing

The YC Processing subsystem receives the image data and calculates the following information for each pixel in the image:

- Frequency Content
- Edge Content
- Color Conversion

The Frequency, Edge, and Color calculations are implemented as separate processing pipelines as illustrated in Figure 4. The results are then passed to the Block Stats, Global Stats, and Histogram blocks which generate the various statistical data about the image.

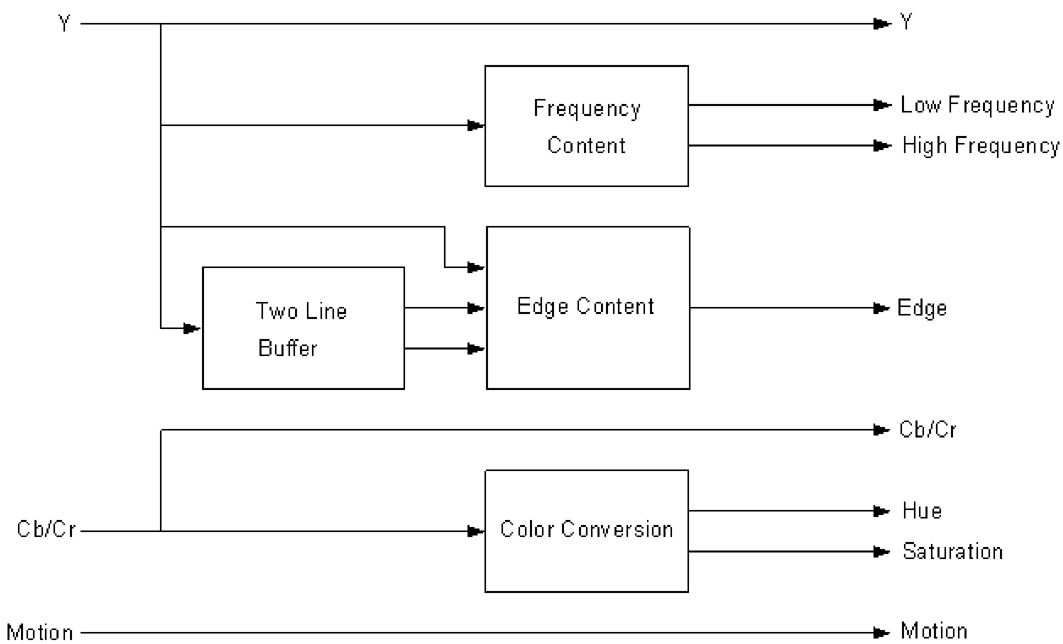

Figure 4: **YC Processing Block Diagram**

Frequency Content

The Frequency Content is calculated on only the Luminance or Y portion of the image. The frequency content that is calculated consists of the Low Frequency portion of the image and the High Frequency portion of the image.

The Low Frequency portion of the image is calculated by passing the data through a low pass filter to remove the high frequencies. As shown in Figure 5, the low pass filter is implemented as a 7-tap FIR filter with the following hard-coded filter coefficients: -1, 0, 9, 16, 9, 0, -1.

The High Frequency portion of the image is calculated by subtracting the low frequency portion of the image from the baseband image. The High Freq. Gain register allows the user to multiply the High Frequency data by 1, 2, 4, or 8 before the value is finally clamped to the range 0 : 255.

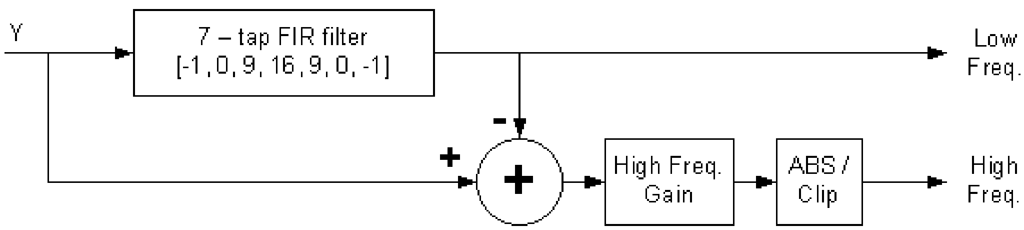

Figure 5: **Frequency Content Block Diagram**

Edge Content

The Edge Content is calculated on only the Y portion of the image. Four Sobel filters are used to look for horizontal, vertical, and diagonal edges in the image. As shown in Figure 6, each of the four components has a separate gain factor to allow for the emphasis of a particular type of edge. Valid gain values are 0, 1, 2, 4, and 8. The final result is a sum of the four edge components. This arrangement gives a good measure of the edges for any particular pixel in the image. Since 3x3 2D FIR filters are used to implement the Sobel filters, a line buffer capable of holding two lines of data is required, as shown in Figure 4. The size of the line buffer is based on the Maximum Frame Size parameter. See the "CORE Generator Graphical User Interface (GUI)" section for more details.

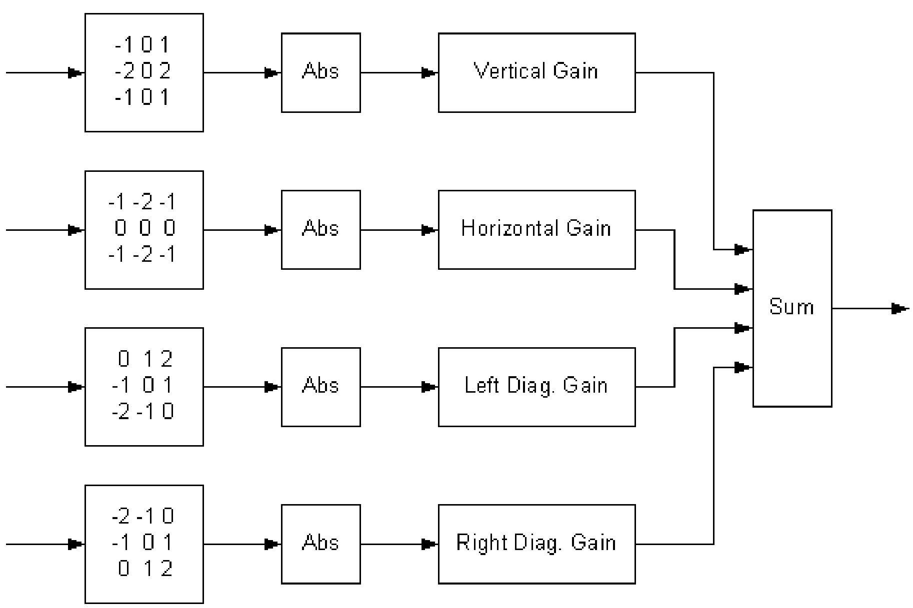

Figure 6: **Edge Content Block Diagram**

Color Conversion

The Color Content of the image is calculated from the Chrominance (C) portion of the image. For Chrominance, the color difference signals B-Y (Cb) and R-Y (Cr) are generated independently, but it is the color comprised by the combination of Cb and Cr that is of interest in characterization. To this effect, the magnitude (Saturation) and angle (Hue) of the Chrominance vector is calculated to provide the color content.

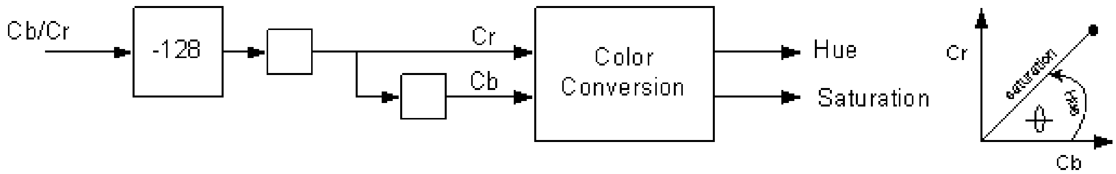

Figure 7: **Color Conversion Block Diagram**

Block Statistics

The Block Statistics subsystem receives data from the YC Processing subsystem. It then subdivides the image into an HxV number of horizontal and vertical blocks respectively. In the example shown in Figure 8, the image is subdivided into an 8x5 grid of blocks.

Blocks are measured in pixels. Valid block sizes are 4x4, 8x8, 16x16, 32x32, and 64x64. The user can specify the block size using the "Block_Size" register (see Table 6). Blocks are defined as starting at the upper left corner of the image, then moving left to right and top to bottom. If the image has a non-integer number of blocks, the partial blocks along the right and bottom edge of the image (the gray boxes in Figure 8) will be excluded from the video analytics analysis.

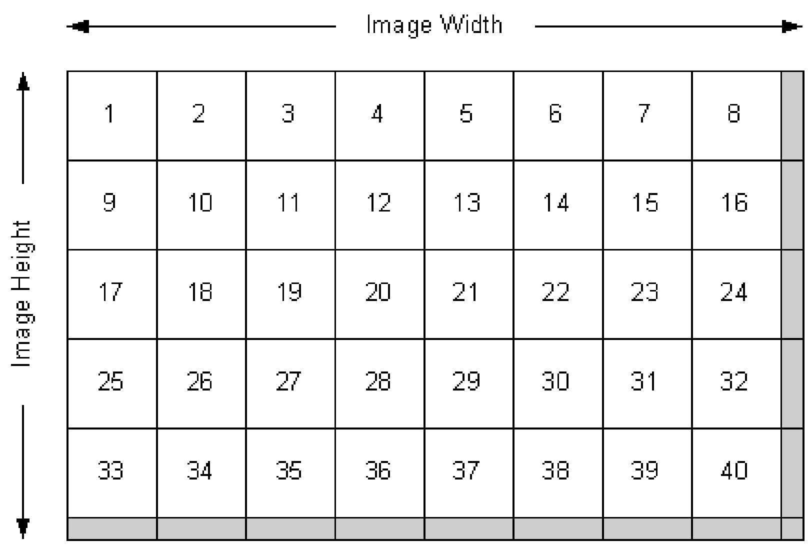

Figure 8: **Block Overlay of an Image Frame**

The following measurements are calculated for each block:

- Mean and Variance
  - Y
  - Cr
  - Cb
  - Low Frequency Content

    - High Frequency Content
    - Edge Content
    - Motion Content
    - Saturation
- Color Selection (x8)

A block diagram of the block statistics processing is illustrated in Figure 9. The mean and variance processing is implemented as eight independent processing pipelines.

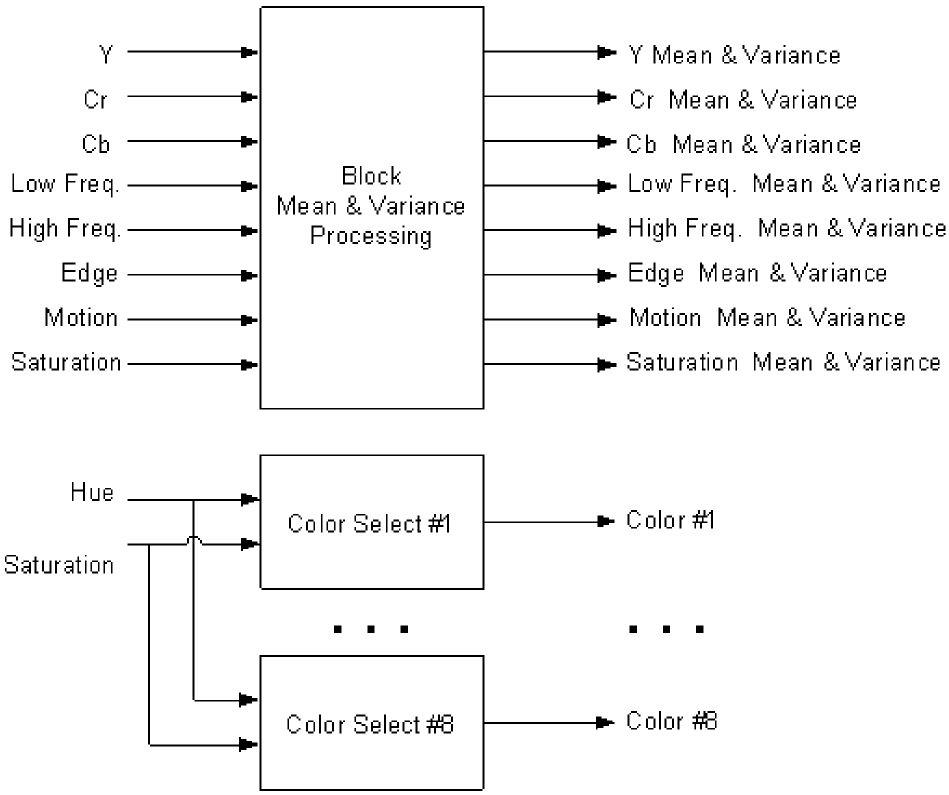

Figure 9: **Block Statistics Block Diagram**

The Mean, Variance, and Color Select calculations are each discussed in more detail in the following sections. Each measurement is implemented as an independent processing chain.

Block Mean

A block diagram of the block mean processing is illustrated in Figure 10. Block mean values are calculated by first scaling each pixel in the block by a "Block Scaling" factor. Once each pixel in the block has been scaled, it is summed in an accumulator.

The block scaling factor is set by the user through the "Block_Y_Scaling" register for the Y-based data streams and through the "Block_C_Scaling" register for the C-based data streams (see Table 6). The Block_Y_Scaling is typically calculated as (1/Num_Block_Pixels)*65536. Num_Block_Pixels is the number of pixels in a block. The Block_C_Scaling is typically calculated as (2/Num_Block_Pixels)*65536 for 4:2:2 data or as (4/Num_Block_Pixels)*65536 for 4:2:0 data.

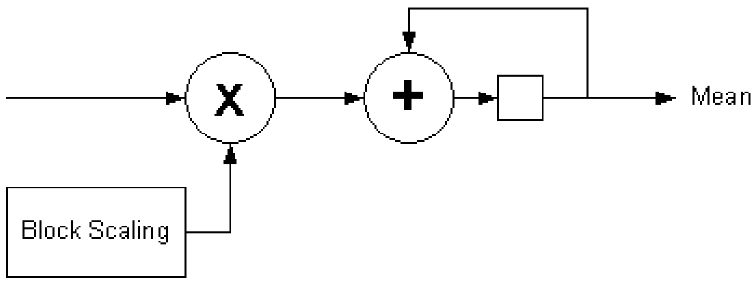

Figure 10: **Block Mean Block Diagram**

Block Variance

A block diagram of the block variance processing is illustrated in Figure 11. Block variance values are calculated by first squaring each pixel in the block and then scaling by the "Block Scaling" factor. Once each pixel has been scaled, it is summed in an accumulator. The last step is to subtract the square of the block mean value from the accumulated value.

The block scaling factor is set by the user through the "Block_Y_Scaling" register for the Y-based data streams and through the "Block_C_Scaling" register for the C-based data streams (see Table 6). Typical calculations for the block scale factors are discussed in the "Block Mean" section.

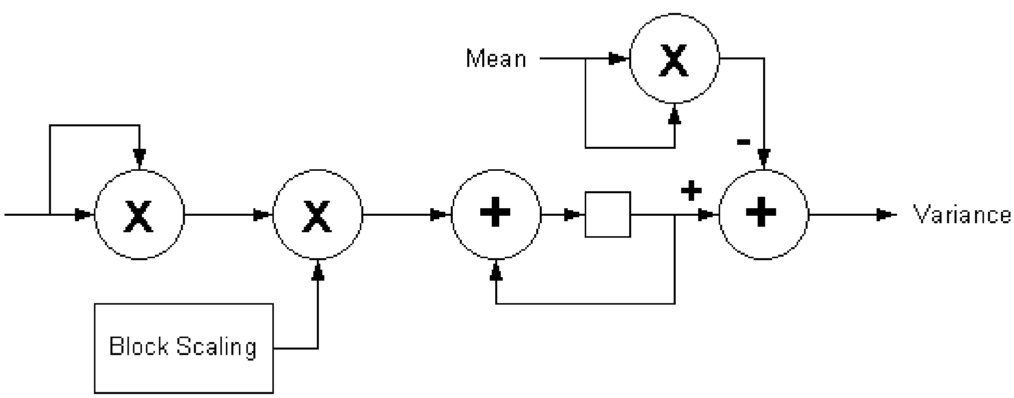

Figure 11: **Block Variance Block Diagram**

Color Select

Color Select uses the Hue and Saturation values for a pixel to detect if the pixel falls within a specified color range. If the pixel meets the specified color range, then the color is said to have been detected. There will be eight separate color selector circuits allowing for the detection of eight different colors.

Each color selector inputs a Hue Minimum and Maximum and a Saturation Minimum and Maximum. These values are set by the user using the Color Select #1-8 registers. See Table 6 for a full description of these registers.

To match the specified color, the Hue and Saturation of the pixel must fall within both the Hue Thresholds and the Saturation Thresholds. For each pixel in the block that matches the specified color, a counter is incremented. The final value from the Color Select is the number of pixels in the block that matched the specified color.

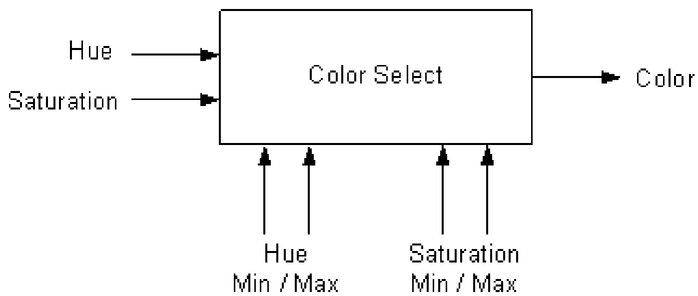

Figure 12: **Block Color Select Block Diagram**

Global Statistics

The Global Stats subsystem uses the Block mean and variance values to calculate Global means and variances of the full image. Global means and variances are calculated for the following values:

- Y
- Cr
- Cb
- Low Frequency Content
- High Frequency Content
- Edge Content
- Motion Content
- Saturation

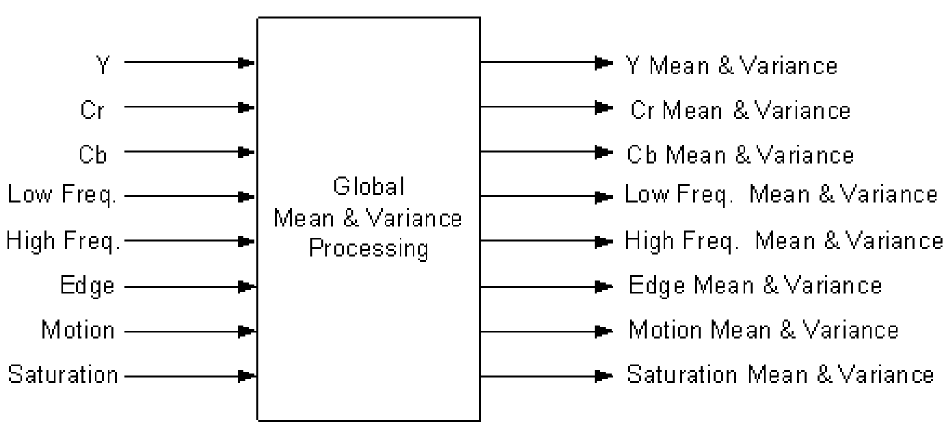

Figure 13: **Global Statistics Block Diagram**

To calculate a global mean or variance value, the value from each block is first scaled by the Global Width Scaling factor and the Global Height Scaling factor. After the scaling process, the value is summed by an accumulator for all of the blocks in the image. Figure 14 illustrates this process.

The Global Width Scaling factor is set by the user through the "Global_Y_Width_Scaling" register (see Table 6). Typically this value is calculated by (1/Num_Blocks_Wide)*65536. The Global Height Scaling factor is set by the user through the "Global_Y_Height_Scaling" register. Typically this value is calculated by (1/Num_Blocks_High)*65536.

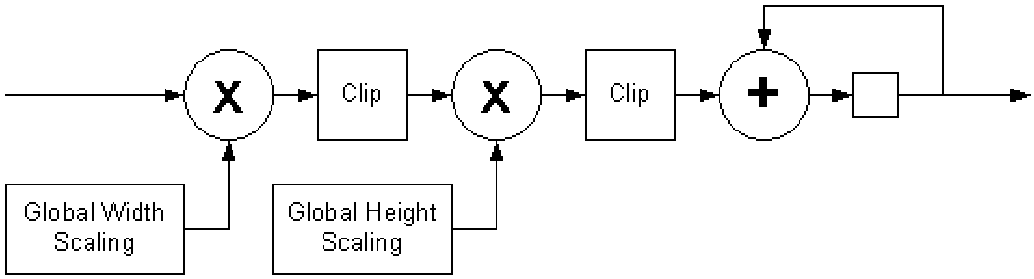

Figure 14: **Global Mean/Variance Block Diagram**

Histograms

The Histograms subsystem takes its input from the YC Processing subsystem. It calculates separate histograms over the entire frame for the following values:

- Y
- Cr
- Cb
- Hue

Since the Image Characterization core supports 8-bit data, each histogram contains 256 bins. The Histogram data is stored in memory starting with bin 0 and ending with bin 255.

Statistics Output

All of the statistics calculated by the Image Characterization core are written to external memory via the core dedicated Video Frame Buffer Controller (VFBC) interface. The data is written to two external memory buffers in a ping-pong fashion. The locations in memory of the two buffers are specified by the user through the "Stats_Start_Addr0" and "Stats_Start_Addr1" registers (see Table 6). The statistics are written to memory in a data structure that is specified in the "Statistics Data Structure" section.

Statistics Output Order

The Image Characterization core writes the calculated statistics to memory in the following order:

1. Frame Header Start
2. Block Statistics
3. Global Statistics
4. Histograms
5. Frame Header Final

The first step is writing the Frame Header Start block. The Header contains a Struct_Valid value. The Frame Header Start block writes a value of 0x0001 to the Struct_Valid value. This denotes that a new data structure has been started, but is not completed and should not be used for processing.

When the Image Characterization core completes all of the block statistics for a particular block of the image, those values are written to memory. Once all of the Block Statistics have been written to memory and the Global Statistics have been calculated, the Global Statistics are written to memory. Next the Histograms are written to memory. The final step is to write the Frame Header Final block. The Frame Header Final block is the same as the Frame Header Start block except that a value of 0xFFFF is written to the Struct_Valid value to denote that the data structure has been completed and is now ready to be used for processing.

Statistics Data Structure

The Statistics Data Structure defines how the image characterization statistics are organized when written to external memory. The data structure is made up of three pieces which are located contiguously in memory:

- Frame Header (see Table 3)
- Global Stats & Histograms (see Table 4)
- Block Stats (see Table 5)

The first two pieces are static in size. Both contain PAD values that are used to pad the size of structure to be a multiple of 128 bytes. This is done to accommodate that fact that the VFBC requires transfers be done in multiples of 128 bytes.

The size of the Block Stats structure is dependent on the number of blocks in the processed image. There will be one instance of the Block Stats data structure for each block in the image. The Block Stats data structures are arranged contiguously in memory. The order of the blocks corresponds to traversing through the blocks from left to right and from top to bottom.

The values in the Statistics Data structure use the following bit widths:

- Mean – 8-bits
- Variance – 16-bits
- Histogram – 32-bits (21-bits actual)
- Color_Select – 16-bits (12-bits actual)
- PAD – 32-bits (0x0000)

Table 2: **Statistics Data Structure**

Byte 3	Byte 2	Byte 1	Byte 0
Frame Header (32 words)			
Global Stats (32 words)			
Histograms (1024 words)			
Block Stats - Block #1 (14 words)			
…			
Block Stats - Block # HxV (14 words)			

Table 3: **Statistics Data Structure Frame Header**

Byte 3	Byte 2	Byte 1	Byte 0
Struct_Valid			
Frame_Index			
PAD (x30)			

Table 4: **Statistics Data Structure Global Stats**

Byte 3	Byte 2	Byte 1	Byte 0
Low_Freq_Mean	V_mean	U_Mean	Y_Mean
Saturation_Mean	Motion_mean	Edge_Mean	High_Freq_Mean
U_Var		Y_Var	
Low_Freq_Var		V_Var	
Edge_Var		High_Freq_Var	
Saturation_Var		Motion_Var	
PAD (x26)			
Y_Histogram (x256)			
U_Histogram (x256)			
V_Histogram (x256)			
Hue_Histogram (x256)			

Table 5: **Statistics Data Structure Block Stats**

Byte 3	Byte 2	Byte 1	Byte 0
Low_Freq_Mean	V_mean	U_Mean	Y_Mean
Saturation_Mean	Motion_mean	Edge_Mean	High_Freq_Mean
U_Var		Y_Var	
Low_Freq_Var		V_Var	
Edge_Var		High_Freq_Var	
Saturation_Var		Motion_Var	
Color_Sel_2		Color_Sel_1	
Color_Sel_4		Color_Sel_3	
Color_Sel_6		Color_Sel_5	
Color_Sel_8		Color_Sel_7	
Reserved			
Reserved			
Reserved			
Reserved			

Note: The Block Stats repeats once for each block in the image. For example a 1280x720 image with block size 16 would result in 3600 contiguous instances of Block Stats data.

CORE Generator Graphical User Interface (GUI)

The Xilinx Image Characterization LogiCORE IP is easily configured to meet the developer's specific needs through the CORE Generator graphical user interface (GUI). This section provides a quick reference to the parameters that can be configured at generation time. The GUI is shown in Figure 15.

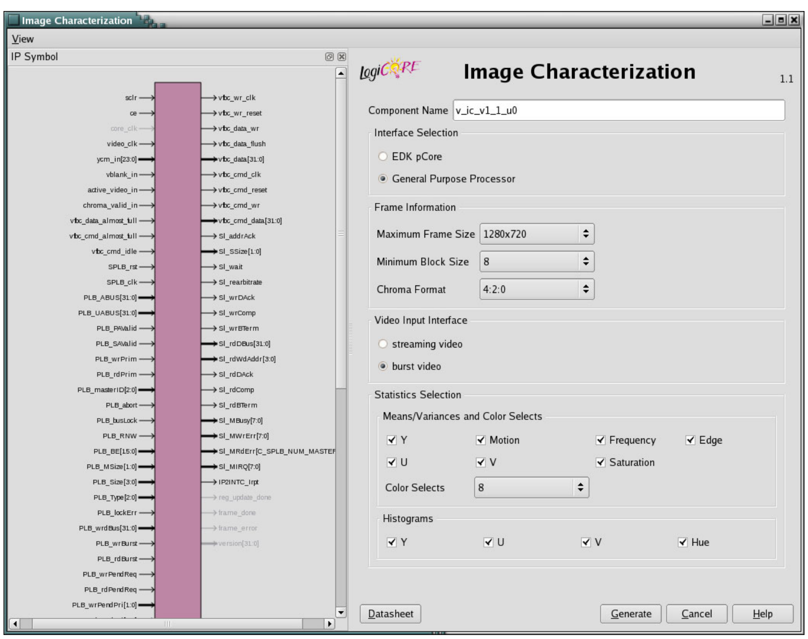

Figure 15: **Image Characterization CORE Generator GUI**

The screen displays a representation of the IP symbol on the left side, and the parameter assignments on the right side, described as follows:

- **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 "_". ***Note: The name "v_ic_v1_1" is not allowed.***
- **Interface Selection:** The Image Characterization core is generated with one of two processor interfaces.
  - **EDK pCore Interface:** CORE Generator software will generate the core as a pCore that can be easily imported into an EDK project as a hardware peripheral. The core registers can then be programmed in real-time via MicroBlaze™. See the "EDK pCore Interface" section for more information. When the EDK pCore is selected, the rest of the options are disabled and set to the default value. All modifications to the Image Characterization pCore are made with the EDK GUI.
  - **General Purpose Processor Interface:** CORE Generator software will generate a set of ports that can be used to program the core. See the "General Purpose Processor Interface" section for more information. When the General Purpose Processor interface is selected, the rest of the configuration options become active and can be used to generate a customized Image Characterization core.
- **Frame Information**
  - **Maximum Frame Size:** Sets the maximum frame size that the core will be instantiated to handle. Valid choices are 1920x1080, 1280x720, and 720x480. This value affects the number of resources used when the core is instantiated.
  - **Minimum Block Size:** Sets the minimum block size that the core can use. Valid choices are 4, 8, 16, 32, and 64. The smaller the block size, the more resources that are used.
  - **Chroma Format:** Sets the expected Chroma format. The valid choices are 4:2:0 and 4:2:2. The selection does not affect resource utilization.
- **Video Input Interface**
  - **Streaming video:** Select streaming video when interfacing a live video stream to the Image Characterization core. A Line buffer is added to the input interface to allow the core to be run at a higher rate than the clock rate of the input video source. The `core_clk` is a required input in this mode.
  - **Burst video:** Select burst video when the input to the Image Characterization core is driven by another processing core that can provide the video input at a higher rate than a live video stream. The `video_clk` that drives the video source is also used to drive the core processing.**stopped**
- **Statistics Selection:** Selected items will appear in the Image Characterization's statistics data structure. Unselected items will be replaced with a zero in the statistics data structure. Logic associated with an item is not instantiated when the item is unselected. Resources can be conserved by deselecting unneeded items.
  - **Means/Variances and Color Selects**
    - **Y**: When selected, global Y mean and variance values as well as block Y mean and variance values are included in the Image Characterization's statistics data structure.
    - **Motion**: When selected, global Motion mean and variance values as well as block Motion mean and variance values are included in the Image Characterization's statistics data structure.
    - **Frequency**: When selected, global Frequency mean and variance values as well as block Frequency mean and variance values are included in the Image Characterization's statistics data structure. The frequency values include Low Frequency and High Frequency.
    - **Edge**: When selected, global Edge Content mean and variance values as well as block Edge Content mean and variance values are included in the Image Characterization's statistics data structure.
    - **U**: When selected, global U mean and variance values as well as block U mean and variance values are included in the Image Characterization's statistics data structure.
    - **V**: When selected, global V mean and variance values as well as block V mean and variance values are included in the Image Characterization's statistics data structure.

- **Saturation**: When selected, global Saturation mean and variance values as well as block Saturation mean and variance values are included in the Image Characterization's statistics data structure.
- **Color Selects**: Sets the number of Color Select values that will be included in the Image Characterization's statistics data structure. Valid choices are 0, 4 and 8.

- **Histograms**
  - **Y**: When selected, the Y Histogram is included in the Image Characterization's statistics data structure.
  - **U**: When selected, the U Histogram is included in the Image Characterization's statistics data structure.
  - **V**: When selected, the V Histogram is included in the Image Characterization's statistics data structure.
  - **Hue**: When selected, the Hue Histogram is included in the Image Characterization's statistics data structure.

EDK pCore Graphical User Interface (GUI)

When the Xilinx Image Characterization LogiCORE IP is generated from CORE Generator as an EDK pCore, it is generated with each option set to the default value. All customizations of an Image Characterization pCore are done with the EDK pCore graphical user interface (GUI). Figure 16 illustrates the EDK pCore GUI for the Image Characterization pCore. All of the options in the EDK pCore GUI for the Image Characterization core correspond to the same options in the CORE Generator GUI. See the "CORE Generator Graphical User Interface (GUI)" section for details about each option.

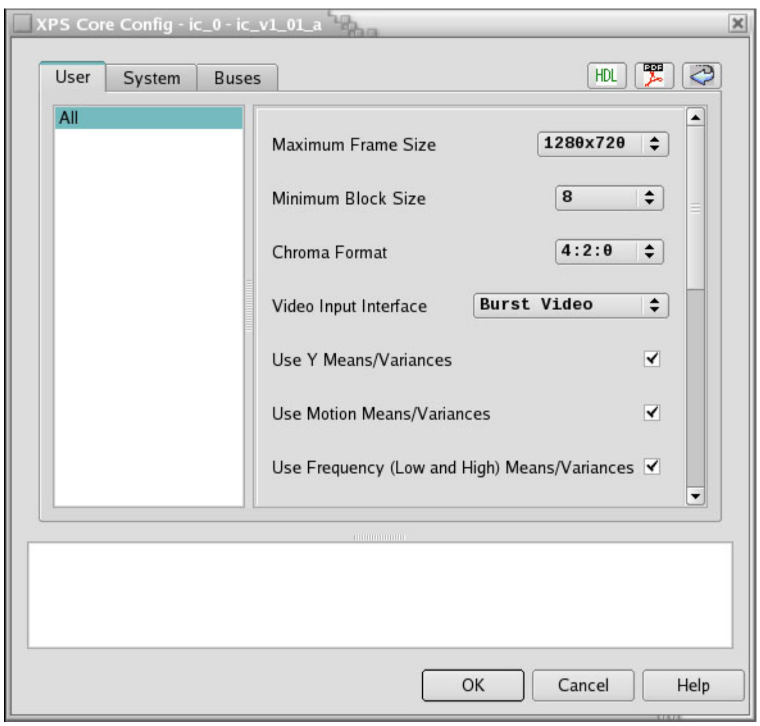

Figure 16: **Image Characterization pCore GUI**

Core Interfaces

There are many video systems developed that use an integrated processor system to dynamically control the parameters within the system. This is especially important when several independent image processing cores are integrated into a single FPGA. The Image Characterization core can be configured with one of two interfaces: an EDK pCore Interface or a General Purpose Processor Interface.

EDK pCore Interface

The pCore interface creates a core that can be easily added to an EDK Project as a hardware peripheral. This section describes the Register Set, the pCore Driver Files, and the I/O signals associated with the Image Characterization pCore.

Once generated by CORE Generator software, the new pCore is located in the CORE Generator project directory at <Component_Name>/pcores/ic_v1_01_a. The pCore should be copied to the user's <EDK_Project>/pcores directory or to a user pCores repository. The Image Characterization pCore driver software is located in the CORE Generator project directory at <Component_Name>/drivers/ic_v1_01_a. The driver software should be copied to the user's <EDK_Project>/drivers directory or to a user pCores repository.

pCore Register Set

The pCore interface provides a memory-mapped interface for the programmable registers within the core, which are defined in Table 6.

Table 6: **Image Characterization pCore Memory Mapped Register Set**

Address (hex)	Register Name	Access Type	Description	
BASEADDR + 0x0000	Control	R/W	General Control Register	
			31:2	Reserved
			1	Register Update Enable This bit communicates to the IP Core to take new values at the next frame vblank rising edge. Usage: This bit is cleared when the IP Core next vblank happens.
			0	Core Enable Enable the Characterization core on the next video frame.
BASEADDR + 0x0008	Status Error	R		
			31:1	Reserved
			0	Frame Error The Image Characterization core did not store all of the image statistics data before the beginning of the next frame. Usage: This bit is cleared when any value is written to the register.
BASEADDR + 0x000C	Status Done	R	Status Register	
			31:1	Reserved
			0	Done Done bit can be polled by software for end of image characterization operation. Usage: This bit is cleared when any value is written to the register.

Table 6: **Image Characterization pCore Memory Mapped Register Set *(Cont'd)***

<table>
<tr><th>Address (hex)</th><th>Register Name</th><th>Access Type</th><th colspan="2">Description</th></tr>
<tr><td rowspan="2">BASEADDR + 0x0010</td><td rowspan="2">Image Statistics Start Address 0</td><td rowspan="2">R/W</td><td colspan="2">Image Stats Address Register</td></tr>
<tr><td>31:0</td><td>Start address #1 for the Image Stats Data Structure
Default (0x00000000)</td></tr>
<tr><td rowspan="2">BASEADDR + 0x0014</td><td rowspan="2">Image Statistics Start Address 1</td><td rowspan="2">R/W</td><td colspan="2">Image Stats Address Register</td></tr>
<tr><td>31:0</td><td>Start address #2 for the Image Stats Data Structure
Default (0x00000000)</td></tr>
<tr><td rowspan="2">BASEADDR + 0x0018</td><td rowspan="2">Image Statistics Start Frame Index</td><td rowspan="2">R/W</td><td colspan="2">Image Stats Start Frame Index Register</td></tr>
<tr><td>31:0</td><td>Starting Frame Index for the Image Characterization Data Structure
Default (0x00000000)</td></tr>
<tr><td>BASEADDR + 0x001C</td><td>Reserved</td><td>R/W</td><td colspan="2">Reserved</td></tr>
<tr><td>BASEADDR + 0x0020</td><td>Reserved</td><td>R/W</td><td colspan="2">Reserved</td></tr>
<tr><td>BASEADDR + 0x0024</td><td>Reserved</td><td>R/W</td><td colspan="2">Reserved</td></tr>
<tr><td rowspan="5">BASEADDR + 0x0028</td><td rowspan="5">Frame Size</td><td rowspan="5">R/W</td><td colspan="2">Horizontal Size and Vertical Size of each Frame</td></tr>
<tr><td>31:27</td><td>Reserved</td></tr>
<tr><td>26:16</td><td>Frame Height</td></tr>
<tr><td>15:11</td><td>Reserved</td></tr>
<tr><td>10:0</td><td>Frame Width</td></tr>
<tr><td rowspan="3">BASEADDR + 0x002C</td><td rowspan="3">Block Size</td><td rowspan="3">R/W</td><td colspan="2">Size of each NxN block</td></tr>
<tr><td>31:7</td><td>Reserved</td></tr>
<tr><td>6:0</td><td>Block Size (4, 8, 16, 32, or 64)</td></tr>
<tr><td rowspan="5">BASEADDR + 0x0030</td><td rowspan="5">Number of Blocks</td><td rowspan="5">R/W</td><td colspan="2">Number of Horizontal and Vertical Blocks per Frame</td></tr>
<tr><td>31:26</td><td>Reserved</td></tr>
<tr><td>25:16</td><td>Number of Blocks High</td></tr>
<tr><td>15:10</td><td>Reserved</td></tr>
<tr><td>9:0</td><td>Number of Blocks Wide</td></tr>
<tr><td rowspan="3">BASEADDR + 0x0034</td><td rowspan="3">Global Y Width Scaling</td><td rowspan="3">R/W</td><td colspan="2">Global Y Width Scale Factor</td></tr>
<tr><td>31:17</td><td>Reserved</td></tr>
<tr><td>16:0</td><td>Scale factor for Global Luma and Chroma Means & Variances
Calculated as (1/Num_Blocks_Wide)*65536</td></tr>
<tr><td rowspan="3">BASEADDR + 0x0038</td><td rowspan="3">Global Y Height Scaling</td><td rowspan="3">R/W</td><td colspan="2">Global Y Height Scale Factor</td></tr>
<tr><td>31:17</td><td>Reserved</td></tr>
<tr><td>16:0</td><td>Scale factor for Global Luma and Chroma Means & Variances
Calculated as (1/Num_Blocks_High)*65536</td></tr>
</table>

Table 6: **Image Characterization pCore Memory Mapped Register Set *(Cont'd)***

Address (hex)	Register Name	Access Type	Description	
BASEADDR + 0x003c	Reserved	R/W	Reserved	
BASEADDR + 0x0040	Reserved	R/W	Reserved	
BASEADDR + 0x0044	Block Y Scaling	R/W	Block Y Scale Factor	
			31:17	Reserved
			16:0	Scale factor for Block Means & Variances Calculated as (1/Num_Block_Pixels)*65536
BASEADDR + 0x0048	Block C Scaling	R/W	Block Chroma (C) Scale Factor	
			31:17	Reserved
			16:0	Scale factor for Block Chroma based Means & Variances Calculated as (4/Num_Block_Pixels)*65536
BASEADDR + 0x004c	High Frequency Gain	R/W	High Frequency Gain Register	
			31:4	Reserved
			3:0	High Frequency Gain (Accepted values = 1, 2, 4, 8)
BASEADDR + 0x0050	Edge Gain	R/W	Edge Gain Register	
			31:28	Reserved
			27:24	Horizontal Edge Gain (Accepted values = 0, 1, 2, 4, 8)
			23:20	Reserved
			19:16	Vertical Edge Gain (Accepted values = 0, 1, 2, 4, 8)
			15:12	Reserved
			11:8	Left Diagonal Edge Gain Upper left to lower right (Accepted values = 0, 1, 2, 4, 8)
			7:4	Reserved
			3:0	Right Diagonal Edge Gain Upper right to lower left (Accepted values = 0, 1, 2, 4, 8)
BASEADDR + 0x0054	Color Select #1	R/W	Color Select #1 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation MInimum
			15:8	Hue Maximum
			7:0	Hue Minimum

Table 6: **Image Characterization pCore Memory Mapped Register Set *(Cont'd)***

Address (hex)	Register Name	Access Type	Description	
BASEADDR + 0x0058	Color Select #2	R/W	Color Select #2 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation Minimum
			15:8	Hue Maximum
			7:0	Hue Minimum
BASEADDR + 0x005c	Color Select #3	R/W	Color Select #3 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation Minimum
			15:8	Hue Maximum
			7:0	Hue Minimum
BASEADDR + 0x0060	Color Select #4	R/W	Color Select #4 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation Minimum
			15:8	Hue Maximum
			7:0	Hue Minimum
BASEADDR + 0x0064	Color Select #5	R/W	Color Select #5 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation Minimum
			15:8	Hue Maximum
			7:0	Hue Minimum
BASEADDR + 0x0068	Color Select #6	R/W	Color Select #6 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation Minimum
			15:8	Hue Maximum
			7:0	Hue Minimum
BASEADDR + 0x006c	Color Select #7	R/W	Color Select #7 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation Minimum
			15:8	Hue Maximum
			7:0	Hue Minimum
BASEADDR + 0x0070	Color Select #8	R/W	Color Select #8 Thresholds Register	
			31:24	Saturation Maximum
			23:16	Saturation Minimum
			15:8	Hue Maximum
			7:0	Hue Minimum

Table 6: **Image Characterization pCore Memory Mapped Register Set *(Cont'd)***

Address (hex)	Register Name	Access Type	Description	
BASEADDR + 0x0F0	Version Register	R	Reports the Version of the Image Characterization Core	
			28:31	Major Version Number. Set to 0x1.
			20:27	Minor Version Number. Set to 0x01.
			16:19	Revision Number. Set to 0xA.
			0:15	Reserved
BASEADDR + 0x0100	Software Reset	R/W	SW reset of core	
			31:1	Reserved
			0	1 resets core
BASEADDR +0x021C	GIER	R/W	Global Interrupt Enable Register	
			31	Mask to enable global interrupts
			30:0	Reserved
BASEADDR + 0x0220	ISR - Interrupt Status/Clear	R	Interrupt Status when read, Interrupt Clear when written	
			31:2	Reserved
			1	Edge sensitive interrupt for IP Core Done with video frame
			0	Edge sensitive interrupt for IP Core Error
BASEADDR + 0x0228	IER - Interrupt Enable	R/W	Interrupt Enable Mask For each bit: 0 = Mask Interrupt 1 = Enable Interrupt	
			31:2	Reserved
			1	Mask or Enable interrupt for IP Core Done with video frame
			0	Mask or Enable interrupt for IP Core Error

pCore Driver Files

The Image Characterization pCore includes a software driver written in the C programming language that the user can use to control the core. A high-level API is provided to hide the details of the Xilinx Image Characterization core, and application developers are encouraged to use it to access the device features. A low-level API is also provided in case developers prefer to access the devices directly through the system registers described in the previous section.

Table 7 lists the files included with the Image Characterization pCore driver.

Table 7: **Software Driver Files Provided with the Image Characterization pCore**

File name	Description
xic.c	Provides the API access to all features of the Image Characterization device driver.
xic.h	Provides the API access to all features of the Image Characterization device driver.
xic_g.c	Contains a template for a configuration table of Image Characterization devices.
xic_hw.h	Contains identifiers and register-level driver functions (or macros) that can be used to access the Image Characterization device.
xic_intr.c	Contains interrupt-related functions of the Image Characterization device driver.
xic_sint.c	Contains static initialization methods for the Image Characterization device driver.
example.c	Examples that demonstrate how to control the Image Characterization device.

pCore I/O Signals

The I/O signals for the Image Characterization pCore are shown in Table 8. The signals can be broken into two groups: Video signals and PLB v4.6 signals. The Video signals are specified in Table 9. The PLB v4.6 signals are specified in Table 10.

Table 8: **pCore I/O Signals**

Video Input	
video_clk	
core_clk [(1)]	
ycm_in (23:0)(Motion,Cb/Cr,Y)	
vblank_in	
chroma_in	
active_video_in	
VFBC Interface	
vfbc_wr_full	vfbc_wr_clk
vfbc_wr_almost_full	vfbc_wr_reset
vfbc_cmd_full	vfbc_wr_write
vfbc_cmd_almost_full	vfbc_wr_end_burst
vfbc_cmd_idle	vfbc_wr_flush
	vfbc_wr_data
	vfbc_cmd_clk
	vfbc_cmd_reset
	vfbc_cmd_data
	vfbc_cmd_write
	vfbc_cmd_end
PLB Interface	
SPLB_Clk	Sl_addrAck
SPLB_Rst	Sl_SSize[0:1]
PLB_ABus[0:C_SPLB_AWIDTH-1]	Sl_wait
PLB_PAValid	Sl_rearbitrate
PLB_masterID[0:C_SPLB_MID_WIDTH-1]	Sl_wrDAck
PLB_abort	Sl_wrComp
PLB_RNW	Sl_wrBTerm
PLB_BE[0:(C_SPLB_DWIDTH/8)-1]	Sl_rdDBus[0:C_SPLB_DWIDTH-1]
PLB_MSize[0:1]	Sl_rdWdAddr[0:3]
PLB_size[0:3]	Sl_rdDAck
PLB_type[0:2]	Sl_rdComp
PLB_wrDBus[0:C_SPLB_DWIDTH-1]	Sl_rdBTerm
PLB_wrBurst	Sl_MBusy[0:C_SPLB_NUM_MASTERS-1]
PLB_rdBurst	Sl_MrdErr[0:C_SPLB_NUM_MASTERS-1]
PLB_SAValid	Sl_MwrErr[0:C_SPLB_NUM_MASTERS-1]
PLB_UABus[0:31]	Sl_MIRQ[0:C_SPLB_NUM_MASTERS-1]
PLB_BusLock	IP2INTC_Irpt
PLB_LockErr	
PLB_TAttribute[0:15]	
PLB_RdPrim	
PLB_WrPrim	
PLB_RDPendPri[0:1]	
PLB_WrPendPri[0:1]	
PLB_RdPendReq	
PLB_WrPendReq	

1. The “core_clk” input is used only for “streaming video” mode.

Table 9: **Video Signals**

Name	Direction	Description
core_clk	In	Clock used to drive the core processing when Input_Video_Interface = Streaming Video
video_clk	In	Clock that drives the input video
ycm_in	In	Video Input Bus: ycm_in[7:0] = Luma (y) ycm_in[15:8] = Chroma (c) ycm_in[23:16] = Motion (m) If the Luma, Chroma, or Motion is not used, that portion of the bus can be tied to a constant value.
active_video_in	In	Video pixel valid
vblank_in	In	Vertical blank
chroma_in	In	Chroma valid
vfbc_wr_clk	Out	VFBC write port clock
vfbc_wr_reset	Out	VFBC write port reset
vfbc_wr_write	Out	VFBC write port write enable
vfbc_wr_end_burst	Out	VFBC write port end burst
vfbc_wr_flush	Out	VFBC write port flush port fifo
vfbc_wr_data	Out	VFBC write port data bus
vfbc_wr_data_be	Out	VFBC write port data byte enable
vfbc_wr_full	In	VFBC write port full flag
vfbc_wr_almost_full	In	VFBC write port almost full flag
vfbc_cmd_clk	Out	VFBC command port clock
vfbc_cmd_reset	Out	VFBC command port reset
vfbc_cmd_data	Out	VFBC command port data bus
vfbc_cmd_write	Out	VFBC command port write enable
vfbc_cmd_end	Out	VFBC command port end
vfbc_cmd_full	In	VFBC command port full flag
vfbc_cmd_almost_full	In	VFBC command port almost full flag
vfbc_cmd_idle	In	VFBC command port idle processing flag

Table 10: **Processor Local Bus (PLB) v4.6 Signals**

Name	Direction	Description
SPLB_Clk	In	Slave PLB clock
SPLB_Rst	In	Slave PLB reset
PLB_ABus[0:C_SPLB_AWIDTH-1]	In	PLB address bus
PLB_PAValid	In	PLB primary address valid indicator
PLB_masterID[0:C_SPLB_MID_WIDTH-1]	In	PLB current master identifier
PLB_abort	In	PLB abort bus request indicator
PLB_RNW	In	PLB read not write
PLB_BE[0:(C_SPLB_DWIDTH/8)-1]	In	PLB byte enables
PLB_MSize[0:1]	In	PLB master data bus size
PLB_size[0:3]	In	PLB transfer size
PLB_type[0:2]	In	PLB transfer type
PLB_wrDBus[0:C_SPLB_DWIDTH-1]	In	PLB write data bus
PLB_wrBurst	In	PLB burst write transfer indicator
PLB_rdBurst	In	PLB burst read transfer indicator
PLB_SAValid	In	PLB Secondary address valid
PLB_UABus[0:31]	In	PLB upper address bus
PLB_BusLock	In	PLB bus Lock
PLB_LockErr	In	PLB lock error
PLB_TAttribute[0:15]	In	PLB attribute
PLB_RdPrim	In	PLB read primary
PLB_WrPrim	In	PLB write primary
PLB_RDPendPri[0:1]	In	PLB read pending on primary
PLB_WrPendPri[0:1]	In	PLB write pending on primary
PLB_RdPendReq	In	PLB read pending request
PLB_WrPendReq	In	PLB write pending request
Sl_addrAck	Out	Slave address acknowledge
Sl_SSize[0:1]	Out	Slave data bus size
Sl_wait	Out	Slave wait indicator
Sl_rearbitrate	Out	Slave rearbitrate bus indicator
Sl_wrDAck	Out	Slave write data acknowledge
Sl_wrComp	Out	Slave write transfer complete indicator
Sl_wrBTerm	Out	Slave terminate write burst transfer
Sl_rdDBus[0:C_SPLB_DWIDTH-1]	Out	Slave read data bus
Sl_rdWdAddr[0:3]	Out	Slave read word address
Sl_rdDAck	Out	Slave read data acknowledge
Sl_rdComp	Out	Slave read transfer complete indicator
Sl_rdBTerm	Out	Slave terminate read burst transfer
Sl_MBusy[0:C_SPLB_NUM_MASTERS-1]	Out	Slave busy indicator

Table 10: **Processor Local Bus (PLB) v4.6 Signals *(Cont'd)***

Name	Direction	Description
Sl_MrdErr[0:C_SPLB_NUM_MASTERS-1]	Out	Slave read error indicator
Sl_MwrErr[0:C_SPLB_NUM_MASTERS-1]	Out	Slave write error indicator
Sl_MIRQ[0:C_SPLB_NUM_MASTERS-1]	Out	Slave Interrupt
IP2INTC_Irpt	Out	Interrupt signal

General Purpose Processor Interface

The other interface option is the General Purpose Processor (GPP) interface. The GPP Interface is shown in Table 11 and consists of the Video signals listed in Table 9 and the Control and Status signals detailed in Table 12. The signals in Table 12 correspond to the registers in Table 6.

The directly exposed control and status signals allow the user to wrap these signals with a user-defined bus interface targeting any arbitrary processor. The recommendation when using this functionality is to disable the Control[1] (Register Update enable) signal of the Control bus before updating the control signals. Once the control signals are ready to be updated in the core, the Control[1] signal should be enabled. Values are written into the core on the falling edge of the Vertical Blank (`vblank`) input.

Table 11: **Image Characterization General Purpose Processor I/O Diagram**

Video Input	
ce sclr video_clk core_clk ycm_in(23:0)(Motion,Cb/Cr,Y) vblank_in chroma_in active_video_in	
VFBC Command Interface	
vfbc_cmd_full vfbc_cmd_almost_full vfbc_cmd_idle	vfbc_cmd_clk vfbc_cmd_reset vfbc_cmd_data vfbc_cmd_write vfbc_cmd_end
VFBC Write Interface	
vfbc_wr_full vfbc_wr_almost_full	vfbc_wr_clk vfbc_wr_reset vfbc_wr_write vfbc_wr_end_burst vfbc_wr_flush vfbc_wr_data

Table 11: **Image Characterization General Purpose Processor I/O Diagram *(Cont'd)***

Control and Status	
control stats_start_addr0 stats_start_addr1 stats_start_index frame_size block_size num_blocks global_y_width_scaling global_y_height_scaling block_y_scaling block_c_scaling high_freq_gain edge_gain color_select_1 color_select_2 color_select_3 color_select_4 color_select_5 color_select_6 color_select_7 color_select_8	frame_done frame_error reg_update_done version

Table 12: **Control and Status Signals**

Name	Direction	Description	
ce	In	Clock Enable	
sclr	In	Synchronous Clear	
control	In	Control register	
		31:2	Reserved
		1	Register Update Enable This bit communicates to the IP core to take new values at the next frame vblank rising edge.
		0	Core Enable Enable the Characterization core on the next video frame.
stats_start_addr0	In	Statistics data structure start address #1	
stats_start_addr1	In	Statistics data structure start address #2	
stats_start_index	In	Statistics data structure start index	
frame_size	In	Horizontal Size and Vertical Size of each Frame	
		31:27	Reserved
		26:16	Frame Height
		15:11	Reserved
		10:0	Frame Width
block_size	In	Size of each NxN block	
		31:7	Reserved
		0:6	Block Size (4, 8, 16, 32, or 64)

Table 12: **Control and Status Signals *(Cont'd)***

Name	Direction	Description	
num_blocks	In	Number of Horizontal and Vertical Blocks per Frame	
		31:26	Reserved
		25:16	Number of Blocks High
		15:10	Reserved
		9:0	Number of Blocks Wide
global_y_width_scaling	In	Horizontal scale factor for global means and variances	
		31:17	Reserved
		16:0	Scale factor for Global Luma and Chroma Means & Variances Calculated as (1/Num_Blocks_Wide)*65536
global_y_height_scaling	In	Vertical scale factor for global means and variances	
		31:17	Reserved
		16:0	Scale factor for Global Luma and Chroma Means & Variances Calculated as (1/Num_Blocks_High)*65536
block_y_scaling	In	Scale factor for block Y means and variances	
		31:17	Reserved
		16:0	Scale factor for Block Means & Variances Calculated as (1/Num_Block_Pixels)*65536
block_c_scaling	In	Scale factor for block Chroma means and variances	
		31:17	Reserved
		16:0	Scale factor for Block Chroma based Means & Variances Calculated as (4/Num_Block_Pixels)*65536
high_freq_gain	In	High frequency gain	
		31:4	Reserved
		3:0	High Frequency Gain (1, 2, 4, 8)
edge_gain	In	Edge gains (Horizontal, Vertical, Left, and Right Diagonals)	
		31:28	Reserved
		27:24	Horizontal Edge Gain (0, 1, 2, 4, 8)
		23:20	Reserved
		19:16	Vertical Edge Gain (0, 1, 2, 4, 8)
		15:12	Reserved
		11:8	Left Diagonal Edge Gain (0, 1, 2, 4, 8)
		7:4	Reserved
		3:0	Right Diagonal Edge Gain (0, 1, 2, 4, 8)
color_select_1	In	Color select thresholds #1	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum

Table 12: **Control and Status Signals *(Cont'd)***

Name	Direction	Description	
color_select_2	In	Color select thresholds #2	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum
color_select_3	In	Color select thresholds #3	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum
color_select_4	In	Color select thresholds #4	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum
color_select_5	In	Color select thresholds #5	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum
color_select_6	In	Color select thresholds #6	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum
color_select_7	In	Color select thresholds #7	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum
color_select_8	In	Color select thresholds #8	
		31:24	Saturation Maximum
		23:16	Saturation Minimum
		15:8	Hue Maximum
		7:0	Hue Minimum
frame_done	Out	Frame done flag (Interrupt)	
frame_error	Out	Frame error flag (Interrupt)	
reg_update_done	Out	Denotes when the registers have been successfully updated	

Table 12: **Control and Status Signals *(Cont'd)***

Name	Direction	Description	
version	Out	Reports the Version of the Image Characterization core	
		31:24	Major Version Number. Set to 0x1.
		27:20	Minor Version Number. Set to 0x01.
		19:16	Revision Number. Set to 0xA.
		15:0	Reserved

Image Characterization Control and Timing

The basic operation of the Image Characterization core is the same regardless of how the core is instantiated. The process begins on the falling edge of the vertical blank (`vblank_in`). At this point, if the "Register Update Enable" (bit 1) of the Control register is set to '1' (see Table 6), then all of the system registers are updated. The `reg_update_done` signal in Figure 18 corresponds to the register update process. This mechanism allows the registers of the core to be double buffered.

At the beginning of each video frame, the Image Characterization core also writes the "Frame Header Start" block to the memory buffer to signify that a new buffer has been started. Next, the core begins to process the incoming frame. As the block statistics for each block finish processing, they are written out to external memory. The processing in Figure 18 uses a block size of 8. Notice the activity on the vfbc_data bus after eight lines of the input data have been processed. This continues until the entire frame has been processed and all of the block statistics data have been written to the memory buffer.

As the block statistics data is written to memory, it is also used to calculate the global statistics. Once all of the block data has been written to memory, the global data is written to memory followed by the histograms. When all of the statistics data has been written to memory, the core completes the process by writing the "Frame Header Final" block to denote that the entire statistics structure has been written to memory and is now ready to be used for further processing. At this point, the `frame_done` signal goes to '1' and remains there until the falling edge of `vblank` triggers the start of a new frame.

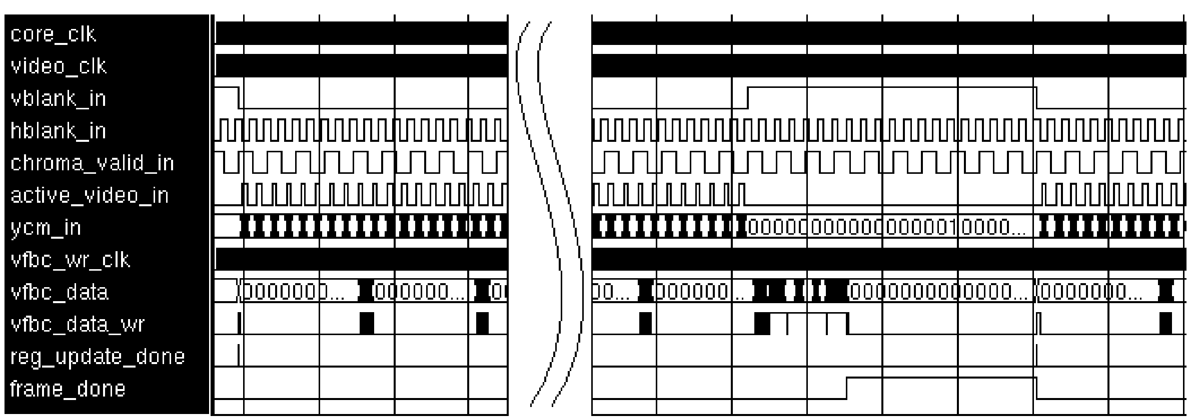

Figure 18: **Image Characterization Timing Diagram**

Streaming Video Mode

The streaming video mode is typically used when interfacing a real-time video source to the Image Characterization core. The streaming video mode provides two clock inputs: one clock (`video_clk`) for the input video and another clock (`core_clk`) for running the core processing. The `video_clk` is used to write the input into a line buffer. The `core_clk` is used to read the data from the buffer and to run the rest of the core processing. An entire line of a frame must be written to the line buffer before the Image Characterization core will begin processing it. The line buffer is sized to hold two lines of video so that the core can process one line while the next line is being written into the buffer. The Maximum Frame Size parameter in the CORE Generator GUI is used to determine the maximum line length that is to be buffered. For example, if the Maximum Frame Size is set to 720x480, then the maximum line length is 720.

The most common reason to use the streaming video mode is that it allows the Image Characterization core to run at a clock rate that is faster than the incoming video clock rate. The only hard requirement is that the `core_clk` must be equal to or faster than the `video_clk`. The faster the `core_clk` is relative to the `video_clk`, the easier it is for the core to run in real-time.

Another reason to use the streaming video mode to smooth out interruptions in the video stream. The Image Characterization core is most efficient when it can process an entire line of video without interruption. The streaming video mode buffers an entire line of video before it is processed by the core. This buffering stage removes the interruptions from the video stream.

Burst Video Mode

The burst video mode is typically used when interfacing the Image Characterization core to another processing core. Since the core can operate at the same clock rate as the incoming data, there is no need for the input line buffer that is used in the streaming video mode. In this burst video mode, the core only needs one clock (`video_clk`). The `core_clk` is not used.

The Image Characterization core is most efficient when it processes an entire line of video without interruption. If providing a continuous line of input is not possible, it may be beneficial to use the streaming video mode to buffer the data first so that the interruptions can be removed.

Video Blanking

The Image Characterization core processing is divided into to two independent pieces: the statistics processor and the output writer. The statistics processor calculates the image statistics of the video frame. The output writer transfers the calculated statistics to memory. Because these two processes run independently, synchronization is required to keep the core functioning correctly. As a result, the Image Characterization core requires the use of a vertical blanking (`vblank`) signal with the incoming frame of video. Calculating the amount of blanking that is necessary can be very challenging. When using a live-video stream, standardized blanking periods are sufficient for the Image Characterization core.

Horizontal Blanking and Core_clk Frequency

The statistics processor is pipelined such that it can run in real-time as the video is input to the core. The bulk of its time is spent calculating the block statistics. The block statistics are stored in temporary buffers until they can be transferred to external memory by the output writer. The system provides enough internal buffering to store the block statistics for two rows of blocks. For example if the Maximum_Frame_Size is 720x480, then the maximum image that can be process will be 720 pixels wide. If the Minimum_Block_Size is 16, then the image will have 45 blocks in each row of blocks. The core will provide enough internal buffering to store two rows of blocks, which would be 90 blocks in this instance. Providing storage for two rows of blocks allows the statistics processor to write to one row of the buffer while the output writer is transferring data from the other row. When the statistics processor finishes a row of boxes, it automatically begins processing the next row. As long as the output writer is able to keep up, there is no problem. If the output writer is unable to transfer the results to memory fast enough, then the statistics processor will eventually loop back around and catch up to the output writer. If this happens, the statistics processor will overwrite the buffer with new block statistics before the old block statistics have been transferred to memory. If this condition happens, the core will not produce the expected amount of data for that frame, which will result in a frame error (see "Frame Error").

For each block (regardless of block size), the Image Characterization core generates a set of block statistics (see "Block Statistics"). As soon as the statistics processor finishes calculating the statistics for a particular block, the output writer is allowed to begin transferring the data to the VFBC write buffer. If the VFBC write buffer is full, the `vfbc_wr_full` flag is used to hold off the output writer from transferring any statistics data to the VFBC. When the VFBC write buffer is not full, the output writer takes 17 cycles of the core clock to write out the statistics for one block. For large block sizes such as 64x64, 32x32, or 16x16, there should be plenty of time for the block statistics results to be written out as new blocks are being processed. For smaller block sizes such as 4x4 or 8x8, the output writer may not be able to keep up if the VFBC write buffer fills up even for a short period to time.

The Image Characterization core requires horizontal blanking (`hblank`) between each line of a frame to allow additional time for the output writer to transfer the block statistics to the VFBC and to relieve congestion that might result if the VFBC write buffer runs full at any point while the frame is being processed. This requirement holds for all block sizes. The minimum number of `hblank` cycles needed per line is difficult to calculate because it is related to the congestion that is caused by the VFBC write buffer running full. The VFBC write buffer running full is a product of the bandwidth of the MPMC and the priority setting of the particular VFBC to which the Image Characterization core is connected.

Worst case calculations for the Image Characterization core should use the largest frame resolution that will be used and the smallest block size that will be used. As an example, consider a system that will use a largest frame size of 720x480 and a smallest block size of 4x4. This combination will generate a 2-D grid of blocks that is 180 blocks wide by 120 blocks high. Since the block is 4 pixels wide by 4 pixels high, 16 clock cycles are needed to calculate the statistics for one block. The output writer requires a minimum of 17 clock cycles to transfer the results to memory. As a result, the absolute minimum amount of `hblank` is 45 cycles per line ((180*(17-16))/4). This absolute minimum is only a starting place. Additional margin should always be added. When using large block sizes that easily allow enough time to transfer the block statistics to memory while the block statistics are being calculated, an absolute minimum of 10 cycles of `hblank` per line is still required.

When using the Image Characterization core in Streaming mode, there is one additional factor to consider when calculating `hblank`. Streaming mode incorporates a line buffer. The incoming data is written into the buffer at the `video_clk` rate and read out of the buffer at the `core_clk` rate. An entire line must be written into the buffer before the core begins to process that line. If the `video_clk` and the `core_clk` are the same frequency, then the preceding `hblank` discussion still holds. If the `core_clk` is faster than the `video_clk`, then the core will process the line faster than a new line can be loaded into the buffer. As a result, the core will sit idle for a period to time after processing each line. This idle time is the equivalent of hblank since the core is not processing data, but is still free to transfer results to external memory. The faster the `core_clk` is relative to the `video_clk`, the greater the number of cycles that the core will sit idle between processing each line of video. An absolute minimum of 10 cycles of `hblank` per line is still required.

Vertical Blanking

The vblank period between the end of one frame and the beginning of the next frame is used to finish processing the frame and to transfer all remaining data to memory. One line of `vblank` is needed to finish processing the final row of blocks in the frame. Additional lines of `vblank` are needed to transfer the block statistics of the final row of blocks as well as to transfer the global statistics and the global historgrams.

As an example, consider a system processing a 720x480 frame with a block size of 4x4 and an `hblank` of 200 clock cycles per line. One line of vblank is needed to finish processing the final row of blocks. As the final row of blocks finishes processing, the results can be transferred to memory while the following blocks in line are being processed. For a block size of 4x4, each row of blocks will require 3060 cycles (180 blocks * 17 cycles/block) to transfer the block statistics to memory. The worst case would require transferring the statistics of both rows of temporary buffering which would be 6120 cycles. Dividing 6120 by 920 (720 pixels + 200 cycles of hblank) gives 6.7, so 7 lines of `vblank` are needed to transfer the remaining block statistics to memory. Since the block statistics can be transferred while the last row of blocks is being processed, the total number of lines of `vblank` required to finish the block statistics is 7 instead of 8.

Next the global data has to be transferred to memory. The global statistics, histograms and final frame header are static in size (1088 words). Dividing that value by the length of each line will yield the minimum number of additional lines of `vblank` that are needed. For this example, 1088/920 = 1.2 lines, so a minimum of 2 lines of `vblank` are needed to transfer the global data. The final total is 7 lines of `vblank` for block processing and 2 lines of `vblank` for transferring the global data. If possible, additional lines should be added as a safety margin. Regardless of the frame size or the block size, an absolute minimum of 3 lines of `vblank` is required.

Frame Done

When the Image Characterization core finishes writing all of the image statistics data to memory, it flags that it has completed processing the current frame. When using the General Purpose Processor interface, the `frame_done` signal is set to '1'. The `frame_done` signal is reset to '0' on the falling edge of `vblank`, which denotes the start of the next frame. When using the pCore interface, the `frame_done` signal is used to drive bit 1 of the interrupt controller. It also sets bit 0 of the Status Done Register. The value in the Status Done Register can be reset by writing any value to the register.

Frame Error

On the falling edge of `vblank` (which signifies the start of the next frame), the Image Characterization core checks to make sure that all of the image statistics from the previous frame were written to memory. If any of the data has not been written to memory, then the core flags a frame error. When using the General Purpose Processor interface, the frame error is shown by the `frame_error` signal going to '1'. The `frame_error` signal is reset to '0' on the next rising edge of `vblank`. When using the pCore interface, the `frame_error` signal is used to drive bit 0 of the interrupt controller. It also sets bit 0 of the Status Error Register. The value in the Status Error Register can be reset by writing any value to the register.

Use Model

Figure 19 shows an example of using the Image Characterization core in a larger system. In this setup, the Motion Adaptive Noise Reduction (MANR) LogiCORE IP calculates the motion in the video and drives the YCM data bus input of the Image Characterization core. The Image Characterization core writes the calculated statistics to memory via the VFBC port. The statistics are then read by either a processor or a "User-Developed Image Statistics Processing" block for higher level analysis and processing. In the figure, the "User-Developed" block is shown with dashed lines because it is an optional block for this system. Such a system can be easily built using the building blocks provided by Xilinx (VDMA, Timing Controller, OSD, etc.).

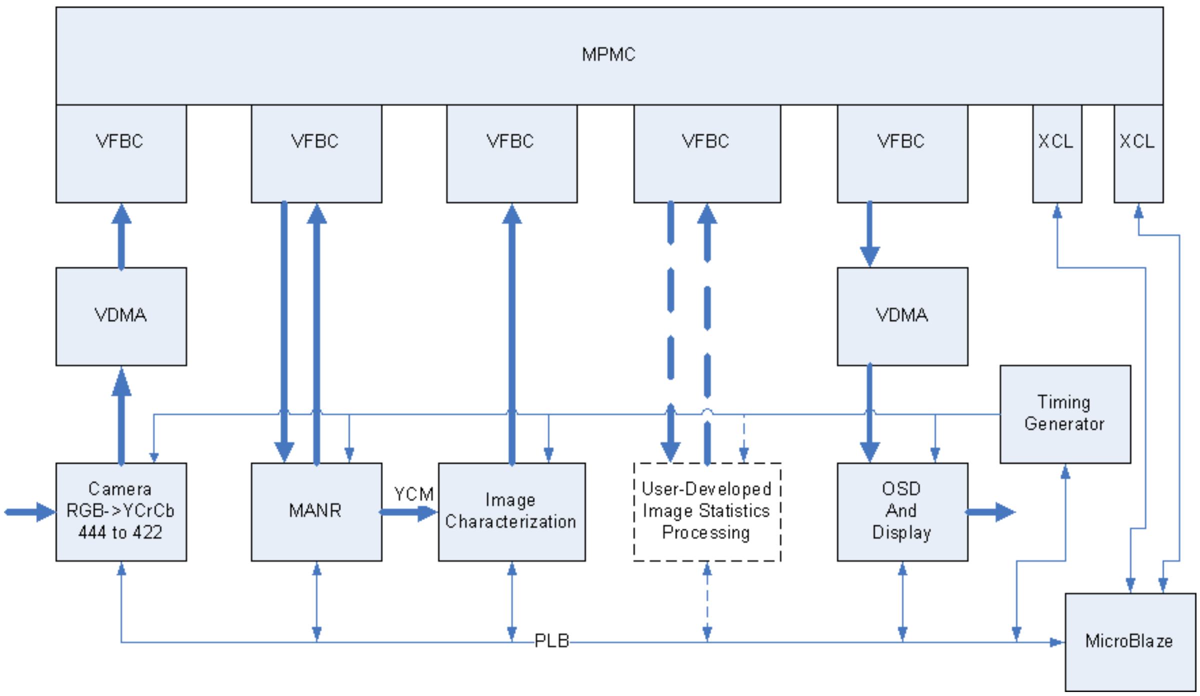

Figure 19: **Image Characterization Example Use Model**

Core Resource Utilization

Resources required for the Image Characterization core have been estimated for the Spartan-3A DSP (Table 13), Spartan-6 (Table 14), Virtex-5 (Table 15), and Virtex-6 (Table 16) These values were generated using the Xilinx CORE Generator tools v12.3. They are derived from *post-synthesis reports*, and may change during MAP and PAR.

Start by choosing the Family, Maximum Frame Size, and Minimum Block size of the core. If using the Streaming Video Interface, add the corresponding resources. If using the pCore Interface, add the corresponding resources. This gives an approximation of the resources that are used by the full core.

If any of the statistics values have been deselected, then the resources associated with that value can be subtracted from the resource calculation. Table 17 estimates the resources for each statistics value for Virtex-6. The same table can be used for Spartan-3A DSP, Spartan-6 or Virtex-5 as the resource estimates are very similar. To use Table 17, locate the statistics value that has been deselected in the table and subtract the associated resources from the resource calculation for the full core. Note the special case when Hue Histogram is deselected, Saturation Mean/Variance is deselected and Color Select = 0. In that case, do not use the individual line items, but use the group calculation at the bottom of the table.

Table 13: **Spartan-3A DSP Resource Estimates**

Feature		LUTs	FFs	Block RAMs	DSP48A
1920x1080 Maximum Frame Size	4x4 block size	7187	6707	29	35
	16x16 block size	7037	6613	17	35
	64x64 block size	6929	6593	17	35
	Streaming Video Interface	211	143	7	0
1280x720 Maximum Frame Size	4x4 block size	7158	6691	29	35
	16x16 block size	7117	6642	17	35
	64x64 block size	6781	6533	17	35
	Streaming Video Interface	95	97	7	0
720x480 Maximum Frame Size	4x4 block size	7161	6654	17	35
	16x16 block size	6876	6552	17	35
	64x64 block size	6824	6532	17	35
	Streaming Video Interface	53	57	3	0
pCore Interface		2421	3693	0	0

Table 14: **Spartan-6 Resource Estimates**

Feature		LUTs	FFs	Block RAMs	DSP48A1
1920x1080 Maximum Frame Size	4x4 block size	6666	6786	29	35
	16x16 block size	6445	6738	17	35
	64x64 block size	6282	6659	17	35
	Streaming Video Interface	176	110	7	0
1280x720 Maximum Frame Size	4x4 block size	6670	6785	29	35
	16x16 block size	6349	6701	17	35
	64x64 block size	6259	6643	17	35
	Streaming Video Interface	94	93	7	0
720x480 Maximum Frame Size	4x4 block size	6444	6735	16	35
	16x16 block size	6360	6677	16	35
	64x64 block size	6280	6613	16	35
	Streaming Video Interface	40	91	3	0
pCore Interface		2646	3660	0	0

Table 15: **Virtex-5 Resource Estimates**

Feature		LUTs	FFs	Block RAMs	DSP48E
1920x1080 Maximum Frame Size	4x4 block size	6314	6703	16	35
	16x16 block size	6247	6648	16	35
	64x64 block size	6047	6587	16	35
	Streaming Video Interface	60	51	3	0
1280x720 Maximum Frame Size	4x4 block size	6278	6645	16	35
	16x16 block size	6229	6632	16	35
	64x64 block size	5964	6517	16	35
	Streaming Video Interface	98	93	3	0
720x480 Maximum Frame Size	4x4 block size	6023	6629	16	35
	16x16 block size	6164	6587	16	35
	64x64 block size	5931	6520	16	35
	Streaming Video Interface	93	101	2	0
pCore Interface		2389	3771	0	0

Table 16: **Virtex-6 Resource Estimates**

Feature		LUTs	FFs	Block RAMs	DSP48E1
1920x1080 Maximum Frame Size	4x4 block size	6780	6775	16	35
	16x16 block size	6471	6717	16	35
	64x64 block size	6377	6653	16	35
	Streaming Video Interface	92	83	3	0
1280x720 Maximum Frame Size	4x4 block size	6704	6759	16	35
	16x16 block size	6235	6685	16	35
	64x64 block size	6142	6621	16	35
	Streaming Video Interface	89	81	3	0
720x480 Maximum Frame Size	4x4 block size	6481	6714	16	35
	16x16 block size	6175	66440	16	35
	64x64 block size	6314	6592	16	35
	Streaming Video Interface	96	85	2	0
pCore Interface		2500	3803	0	0

Table 17: **Virtex-6 Resources Removed When an Item is Not Selected**

Feature	LUTs	FFs	Block RAMs	DSP48E1
Y Histogram	174	211	1	0
U and V Histogram	240	235	1	0
Hue Histogram	162	212	1	0
Y Mean/Variance	175	188	1	3
U Mean/Variance	317	340	1	3
V Mean/Variance	315	339	1	3
Motion Mean/Variance	173	213	1	3
Frequency Mean/Variance	606	731	2	6
Edge Content Mean/Variance	801	1163	3	3
Saturation Mean/Variance	314	340	1	3
Color Select = 4	252	476	2	0
Color Select = 0	646	1104	4	0
Hue Histogram Saturation Mean/Variance Color Select = 0	1536	2703	6	4

Performance

The following are typical clock frequencies for the target families. The maximum achievable clock frequency could vary. The maximum achievable clock frequency and all resource counts may be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools, and other factors.

- Spartan-3A DSP: 150 MHz
- Spartan-6: 150 MHz
- Virtex-5: 225 MHz
- Virtex-6: 225 MHz

References

1. Processor Local Bus (PLB) v4.6 (www.xilinx.com/support/documentation/ip_documentation/plb_v46.pdf)

Support

Xilinx provides technical support for this LogiCORE 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 Xilinx Image Characterization LogiCORE system 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 the core functionality with your own design and demonstrates the various interfaces on 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 Characterization 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 provided when you purchase the core and provides full access to all core functionality both in simulation and in hardware, including:

- Functional simulation support
- Back-annotated gate-level 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:

1. Navigate to the product page for this core.
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.

Ordering Information

The Image Characterization v1.1 core is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator system v12.3 or higher. The CORE Generator system is shipped with the Xilinx ISE Design Suite development software. Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE modules and software. Information about additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
04/19/10	1.0	Initial Xilinx release.
09/21/10	2.0	Updated for 12.3 release.

Notice of Disclaimer