LogiCORE IP Image Statistics v6.0

Product Guide for Vivado Design Suite

PG008 December 18, 2013

XILINX®

XILINX®

Table of Contents

Introduction

The Xilinx LogiCORE™ IP Image Statistics core implements the computationally intensive metering functionality common in digital cameras, camcorders and imaging devices. This core generates a set of statistics for color histograms, mean and variance values, edge and frequency content for 16 user-defined zones on a per frame basis. The statistical information collected may be used in the control algorithms for Auto-Focus, Auto-White Balance, and Auto-Exposure for image processing applications.

Features

- Supports spatial resolutions from 32x32 up to 7680x7680
  - Supports 1080P60 in all supported device families (1)
  - Supports 4kx2k @ 24 Hz in supported high performance devices
- AXI4-Stream data interfaces
- AXI4-Lite control interface
- 16 programmable zones
- 8, 10, or 12-bit input precision
- Outputs for all zones and color channels:
  - Minimum and maximum color values
  - Sum and sum of squares for each color value
  - Low and high frequency content
  - Horizontal, vertical and diagonal edge content
- Outputs for pre-selected zone(s):
  - Y channel histogram
  - R,G,B channel histograms
  - Two-dimensional Cr-Cb histogram

1. Performance on low power devices may be lower.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale™ Architecture, Zynq®-7000, 7 Series
Supported User Interfaces	AXI4-Lite, AXI4-Stream(2)
Resources	See Table 2-1 through Table 2-3.
Provided with Core	
Design Files	Encrypted RTL
Example Design	Not Provided
Test Bench	Verilog (3)
Constraints File	XDC
Simulation Models	Encrypted RTL, VHDL or Verilog Structural, C Model (3)
Supported Software Drivers (4)	Standalone
Tested Design Flows (5)	
Design Entry Tools	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis Tools	Vivado Synthesis
Support	
Provided by Xilinx, Inc.	

1. For a complete listing of supported devices, see the Vivado IP Catalog.
2. Video protocol as defined in the *Video IP: AXI Feature Adoption* section o (UG761) *AXI Reference Guide* [Ref 4].
3. HDL test bench and C-Model available on the product page on Xilinx.com.
4. Standalone driver details can be found in the SDK directory (*<install_directory>*/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from //wiki.xilinx.com.
5. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Most digital cameras implement a form of automatic white balance (AWB) and automatic exposure (AE) control. Cameras with an adjustable focus (AF) lens also implement automatic focus. All of these algorithms have the need for sophisticated image statistics gathered from the image or video to process and implement controls and algorithms for these functions. Generating image statistics is a data intensive process, and the Image Statistics core provides an efficient, programmable solution. The resulting image statistics provide the basic data for software based AE, AWB and AF algorithms. The Image Statistics core supports multi-zone metering on rectangular regions, and 16 software defined zones, as shown in Figure 1-1.

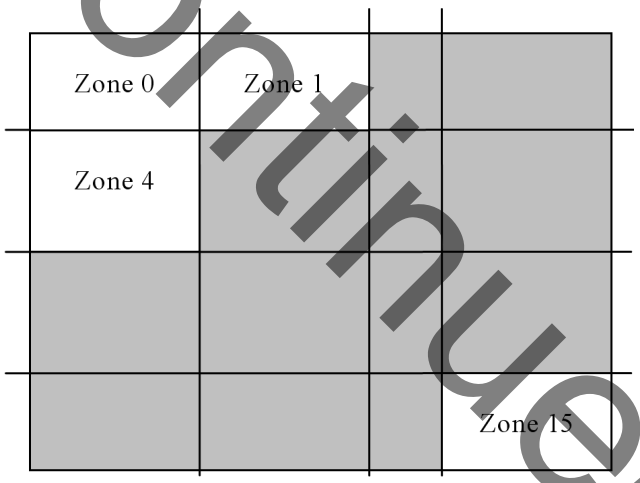


Figure 1-1: **16 Zone Metering**

Minimum, maximum, sum, and sum of squares values are calculated for all color channels for all 16 zones. Frequency and edge content is also calculated for all zones in luminance values provided by an RGB-to-YCrCb ITU 601 converter internal to the Image Statistics core. The RGB, Y, and two-dimensional Cr-Cb histograms are calculated over predefined sets of zones.

Feature Summary

The Image Statistics core provides data from video streams of a maximum resolution of 7680 columns by 7680 rows with 8, 10, or 12 bits per pixel and supports the bandwidth necessary for high-definition (1080p60) resolutions in all Xilinx FPGA device families. The

core generates data from the video stream in the form of minimum and maximum color values, sum and sum of squares for each color value, low and high frequency content and horizontal, vertical and edge content from 16 programmable zones. The core can also generate histograms for RGB channels, Y channel and two dimensional Cr-Cb channels.

The core can be configured and instantiated from Vivado design tools, CORE Generator or EDK tools. Core functionality is controlled dynamically through the AXI4-Lite interface.

Applications

- Automatic Exposure (AE) control
- Automatic Sensor Gain (AG) control
- Auto Focus (AF) control of the lens assembly
- Digital contrast/brightness adjustment
- Global histogram equalization
- White Balance correction

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado Design Suite/ISE Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

For more information, visit the Image Statistics product web page.

Information about other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The Image Statistics core is compliant with the AXI4-Lite interconnect standard as defined in *Video IP: AXI Feature Adoption* section of the (UG761) *AXI Reference Guide* [Ref 7].

Performance

The following sections detail the performance characteristics of the Image Statistics core.

Maximum Frequencies

The maximum achievable clock frequency can vary. The maximum achievable clock frequency and all resource counts can be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools and other factors. See Table 2-1 through Table 2-3 for device-specific information.

Latency

The processing latency of the core is one frame before statistical information can be delivered. Because the video stream goes through the core unmodified, there is only one clock cycle latency between in the input video and the output video.

Throughput

When the core is processing data from a video source which can always provide valid data, e.g. a frame buffer, the throughput of the core is essentially the rate of the video.

In numeric terms, 1080P/60 represents an average data rate of 124.4 MPixels/second (1080 rows x 1920 columns x 60 frames / second), and a burst data rate of 148.5 MPixels/sec. In order to ensure that the core can process 124.4 MPixels/second, the core needs to operate at the pixel clock rate.

The core is limited to how quickly it can calculate the data from the video stream. Resource Utilization provides an estimate of how quickly the core can operate in certain conditions.

Resource Utilization

For an accurate measure of the usage of primitives, slices, and CLBs for a particular instance, check the **Display Core Viewer after Generation**.

Table 2-1 through Table 2-3 were generated using Vivado Design Suite with default tool options for characterization data. UltraScale™ results are expected to be similar to 7 series results.

Table 2-1: **Resource Utilization and Target Speed for Kintex-7 Devices**

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	1920	Yes	Yes	Yes	2704	3968	1	5	19	212
10	1920	Yes	Yes	Yes	2978	4333	6	1	19	212
12	1920	Yes	Yes	Yes	3186	4695	16	1	19	212

Device and speed file: XC7K70T-1 ADVANCED 1.07b 2012-08-28

Table 2-2: **Resource Utilization and Target Speed for Artix-7 Devices**

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	1920	Yes	Yes	Yes	2710	3968	1	5	19	148
10	1920	Yes	Yes	Yes	2977	4333	6	1	19	140
12	1920	Yes	Yes	Yes	3190	4695	16	1	19	148

Device and speed file: XC7A100T-1 ADVANCED 1.05a 2012-08-31

Table 2-3: **Resource Utilization and Target Speed for Virtex-7 Devices**

Data Width	Max Rows Max Cols	Histogram			LUTs	FFs	RAM36	RAM18	DSP48s	Clock F_{MAX} (MHz)
		Y	RGB	CC						
8	1920	Yes	Yes	Yes	2706	3968	1	5	19	219
10	1920	Yes	Yes	Yes	2974	4333	6	1	19	226
12	1920	Yes	Yes	Yes	3189	4695	16	1	19	219

Device and speed file: XC7V585T-1 ADVANCED 1.07b 2012-08-28

Optional Histogram Calculation using Vivado Design Suite

Table 2-4 through Table 2-6 present resource utilization numbers for all possible combinations of optional histogram settings, for all supported device families, using 8-bit input data and the maximum numbers of rows and columns set to 1920.

Table 2-4: **Optional Histogram Calculation: Resource Utilization Artix-7 Devices (XC7A100T-1)**

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	CC	RGB						
8	1920	Yes	Yes	Yes	2710	3968	1	5	19	148
8	1920	Yes	Yes	No	2562	3563	1	2	19	140
8	1920	Yes	No	Yes	2552	3751	1	4	15	140
8	1920	Yes	No	No	2382	3346	1	1	15	148
8	1920	No	Yes	Yes	2664	3848	1	4	19	148
8	1920	No	Yes	No	2476	3443	1	1	19	140
8	1920	No	No	Yes	2483	3610	1	3	15	148
8	1920	No	No	No	2312	3202	1	0	15	148

Table 2-5: **Optional Histogram Calculation: Resource Utilization Kintex-7 Devices (XC7K70T-1)**

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	CC	RGB						
8	1920	Yes	Yes	Yes	2704	3968	1	5	19	212
8	1920	Yes	Yes	No	2565	3563	1	2	19	212
8	1920	Yes	No	Yes	2552	3751	1	4	15	226
8	1920	Yes	No	No	2382	3346	1	1	15	226
8	1920	No	Yes	Yes	2652	3848	1	4	19	219
8	1920	No	Yes	No	2478	3443	1	1	19	212
8	1920	No	No	Yes	2484	3610	1	3	15	226
8	1920	No	No	No	2311	3202	1	0	15	226

Table 2-6: **Optional Histogram Calculation: Resource Utilization Virtex-7 Devices (XC7V585T-1)**

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	CC	RGB						
8	1920	Yes	Yes	Yes	2706	3968	1	5	19	219
8	1920	Yes	Yes	No	2560	3563	1	2	19	219
8	1920	Yes	No	Yes	2546	3751	1	4	15	219

Table 2-6: **Optional Histogram Calculation: Resource Utilization Virtex-7 Devices (XC7V585T-1) *(Cont'd)***

Data Width	MAX Rows MAX Cols	Histogram			LUTs	FF	RAM36	RAM18	DSP 48s	Clock F_{MAX} (MHz)
		Y	CC	RGB						
8	1920	Yes	No	No	2379	3346	1	1	15	219
8	1920	No	Yes	Yes	2653	3848	1	4	19	226
8	1920	No	Yes	No	2474	3443	1	1	19	226
8	1920	No	No	Yes	2479	3610	1	3	15	219
8	1920	No	No	No	2313	3202	1	0	15	234

Port Descriptions

The Image Statistics core uses industry standard control and data interfaces to connect to other system components. The following sections describe the various interfaces available with the core. Figure 2-1 illustrates an I/O diagram of the Image Statistics core.

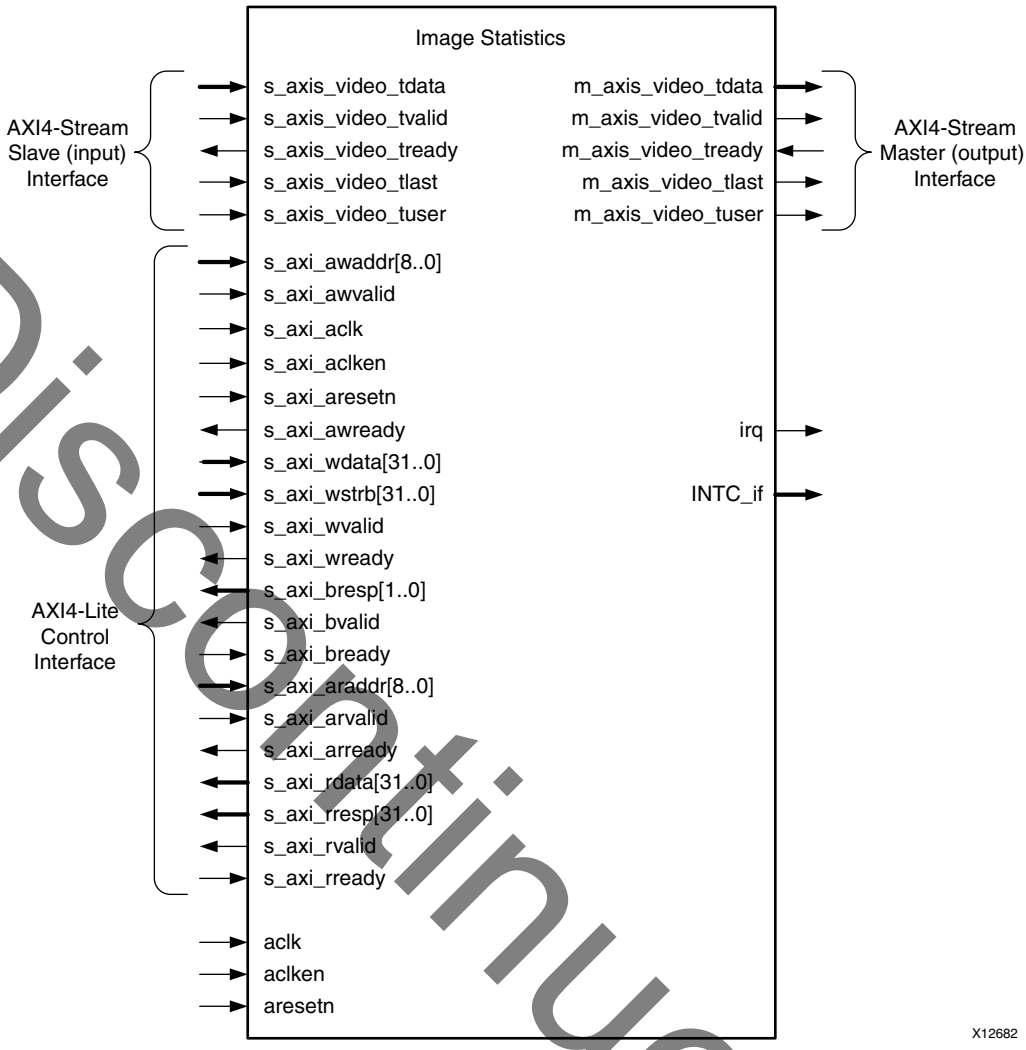


Figure 2-1: **Image Statistics Core Top-Level Signaling Interface**

Common Interface Signals

Table 2-7 summarizes the signals which are either shared by, or not part of the dedicated AXI4-Stream data or AXI4-Lite control interfaces.

Table 2-7: **Common Interface Signals**

Signal Name	Direction	Width	Description
ACLK	In	1	Video Core Clock
ACLKEN	In	1	Video Core Active High Clock Enable

Table 2-7: **Common Interface Signals *(Cont'd)***

Signal Name	Direction	Width	Description
ARESETn	In	1	Video Core Active Low Synchronous Reset
IRQ	Out	1	Optional Interrupt Request Pin. Available only when AXI4-Liter interface is selected on GUI.

The ACLK, ACLKEN and ARESETn signals are shared between the core and the AXI4-Stream data interfaces. The AXI4-Lite control interface has its own set of clock, clock enable and reset pins: S_AXI_ACLK, S_AXI_ACLKEN and S_AXI_ARESETn. Refer to Interrupt Subsystem for a detailed description of the INTC_IF and IRQ pins.

ACLK

The AXI4-Stream interface must be synchronous to the core clock signal ACLK. All AXI4-Stream interface input signals are sampled on the rising edge of ACLK. All AXI4-Stream output signal changes occur after the rising edge of ACLK. The AXI4-Lite interface is unaffected by the ACLK signal.

ACLKEN

The ACLKEN pin is an active-high, synchronous clock-enable input pertaining to AXI4-Stream interfaces. Setting ACLKEN low (de-asserted) halts the operation of the core despite rising edges on the ACLK pin. Internal states are maintained, and output signal levels are held until ACLKEN is asserted again. When ACLKEN is de-asserted, core inputs are not sampled, except ARESETn, which supersedes ACLKEN. The AXI4-Lite interface is unaffected by the ACLKEN signal.

ARESETn

The ARESETn pin is an active-low, synchronous reset input pertaining to only AXI4-Stream interfaces. ARESETn supersedes ACLKEN, and when set to 0, the core resets at the next rising edge of ACLK even if ACLKEN is de-asserted. The ARESETn signal must be synchronous to the ACLK and must be held low for a minimum of 32 clock cycles of the slowest clock. The AXI4-Lite interface is unaffected by the ARESETn signal.

Data Interface

The Image Statistics core receives and transmits data using AXI4-Stream interfaces that implement a video protocol as defined in the *Video IP: AXI Feature Adoption* section of the (UG761) *AXI Reference Guide* [Ref 7]

AXI4-Stream Signal Names and Descriptions

Table 2-8 lists the AXI4-Stream signal names and descriptions.

Table 2-8: **AXI4-Stream Data Interface Signal Descriptions**

Signal Name	Direction	Width	Description
s_axis_video_tdata	In	24, 32, 40	Input Video Data
s_axis_video_tvalid	In	1	Input Video Valid Signal
s_axis_video_tready	Out	1	Input Ready
s_axis_video_tuser	In	1	Input Video Start Of Frame
s_axis_video_tlast	In	1	Input Video End Of Line
m_axis_video_tdata	Out	24, 32, 40	Output Video Data
m_axis_video_tvalid	Out	1	Output Valid
m_axis_video_tready	In	1	Output Ready
m_axis_video_tuser	Out	1	Output Video Start Of Frame
m_axis_video_tlast	Out	1	Output Video End Of Line

Video Data

The AXI4-Stream interface specification restricts TDATA widths to integer multiples of 8 bits. Therefore, 10- and 12-bit sensor data must be padded with zeros on the MSB to form a 16-bit wide vector before connecting to s_axis_video_tdata. Padding does not affect the size of the core.

Similarly, RGB data on the Image Statistics' core output m_axis_video_tdata is packed and padded to multiples of 8 bits as necessary, as seen in Figure 2-2. Zero padding the most significant bits is only necessary for 10- and 12-bit wide data.

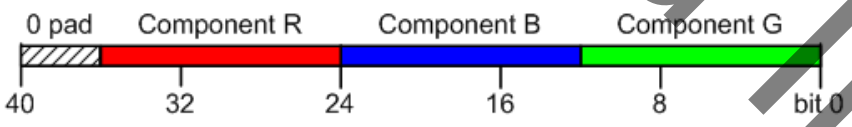


Figure 2-2: **RGB Data Encoding on s_axis_video_tdata and m_axis_video_tdata**

READY/VALID Handshake

A valid transfer occurs whenever READY, VALID, ACLKEN, and ARESETn are High at the rising edge of ACLK, as shown in Figure 2-3. During valid transfers, DATA only carries active video data. Blank periods and ancillary data packets are not transferred through the AXI4-Stream video protocol.

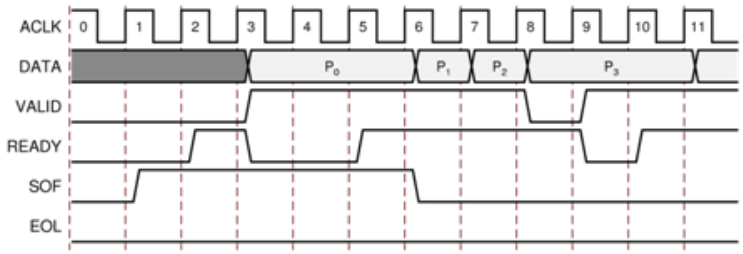


Figure 2-3: **Example of READY/VALID Handshake, Start of a New Frame**

Guidelines on Driving s_axis_video_tvalid

Once s_axis_video_tvalid is asserted, no interface signals (except the Image Statistics core driving s_axis_video_tready) can change value until the transaction completes (s_axis_video_tready, s_axis_video_tvalid and ACLKEN are High on the rising edge of ACLK). Once asserted, s_axis_video_tvalid can only be de-asserted after a transaction has completed. Transactions cannot be retracted or aborted. Following a transaction, s_axis_video_tvalid can either be de-asserted or remain asserted to initiate a new transfer.

Guidelines on Driving m_axis_video_tready

The m_axis_video_tready signal may be asserted before, during or after the cycle in which the Image Statistics core asserted m_axis_video_tvalid. The assertion of m_axis_video_tready may be dependent on the value of m_axis_video_tvalid. A slave that can immediately accept data qualified by m_axis_video_tvalid, should pre-assert its m_axis_video_tready signal until data is received. Alternatively, m_axis_video_tready can be registered and driven the cycle following VALID assertion.

RECOMMENDED: *It is recommended that the AXI4-Stream slave should drive READY independently, or pre-assert READY to minimize latency.*

Start of Frame Signals - m_axis_video_tuser, s_axis_video_tuser

The Start-Of-Frame (SOF) signal, physically transmitted over the AXI4-Stream TUSER signal, marks the first pixel of a video frame. The SOF pulse is one valid transaction wide, and must coincide with the first pixel of the frame, as seen in Figure 2-3. SOF serves as a frame synchronization signal, which allows downstream cores to re-initialize and detect the first pixel of a frame. The SOF signal may be asserted an arbitrary number of ACLK cycles before the first pixel value is presented on DATA, as long as a VALID is not asserted.

End of Line Signals - m_axis_video_tlast, s_axis_video_tlast

The End-Of-Line signal, physically transmitted over the AXI4-Stream TLAST signal, marks the last pixel of a line. The EOL pulse is one valid transaction wide, and must coincide with the last pixel of a scan-line, as shown in Figure 2-4.

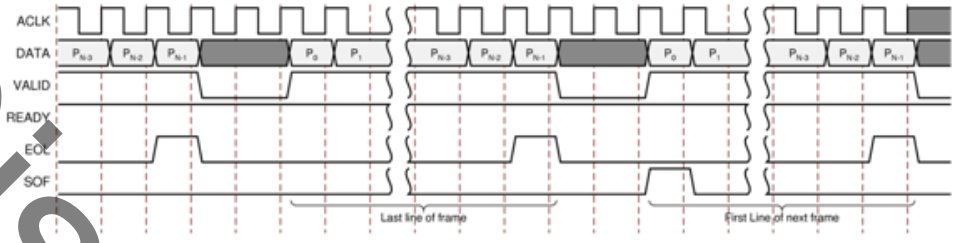


Figure 2-4: **Use of EOL and SOF Signals**

Control Interface

The AXI4-Lite slave interface facilitates integrating the core into a processor system, or along with other video or AXI4-Lite compliant IP, connected via AXI4-Lite interface to an AXI4-Lite master. The AXI4-Lite interface allows a user to dynamically control parameters within the core. Core configuration can be accomplished using an AXI4-Lite master state machine, or an embedded ARM or soft system processor such as a MicroBlaze processor.

The Image Statistics core can be controlled via the AXI4-Lite interface using read and write transactions to the Image Statistics register space.

Table 2-9: **AXI4-Lite Interface Signals**

Signal Name	Direction	Width	Description
s_axi_aclk	In	1	AXI4-Lite clock
s_axi_aclken	In	1	AXI4-Lite clock enable
s_axi_aresetn	In	1	AXI4-Lite synchronous Active Low reset
s_axi_awvalid	In	1	AXI4-Lite Write Address Channel Write Address Valid.
s_axi_awread	Out	1	AXI4-Lite Write Address Channel Write Address Ready. Indicates DMA ready to accept the write address.
s_axi_awaddr	In	32	AXI4-Lite Write Address Bus
s_axi_wvalid	In	1	AXI4-Lite Write Data Channel Write Data Valid.
s_axi_wready	Out	1	AXI4-Lite Write Data Channel Write Data Ready. Indicates DMA is ready to accept the write data.
s_axi_wdata	In	32	AXI4-Lite Write Data Bus

Table 2-9: **AXI4-Lite Interface Signals *(Cont'd)***

Signal Name	Direction	Width	Description
s_axi_bresp	Out	2	AXI4-Lite Write Response Channel. Indicates results of the write transfer.
s_axi_bvalid	Out	1	AXI4-Lite Write Response Channel Response Valid. Indicates response is valid.
s_axi_bready	In	1	AXI4-Lite Write Response Channel Ready. Indicates target is ready to receive response.
s_axi_arvalid	In	1	AXI4-Lite Read Address Channel Read Address Valid
s_axi_arready	Out	1	Ready. Indicates DMA is ready to accept the read address.
s_axi_araddr	In	32	AXI4-Lite Read Address Bus
s_axi_rvalid	Out	1	AXI4-Lite Read Data Channel Read Data Valid
s_axi_rready	In	1	AXI4-Lite Read Data Channel Read Data Ready. Indicates target is ready to accept the read data.
s_axi_rdata	Out	32	AXI4-Lite Read Data Bus
s_axi_rresp	Out	2	AXI4-Lite Read Response Channel Response. Indicates results of the read transfer.

S_AXI_ACLK

The AXI4-Lite interface must be synchronous to the S_AXI_ACLK clock signal. The AXI4-Lite interface input signals are sampled on the rising edge of ACLK. The AXI4-Lite output signal changes occur after the rising edge of ACLK. The AXI4-Stream interfaces signals are not affected by the S_AXI_ACLK.

S_AXI_ACLKEN

The S_AXI_ACLKEN pin is an active-high, synchronous clock-enable input for the AXI4-Lite interface. Setting S_AXI_ACLKEN low (de-asserted) halts the operation of the AXI4-Lite interface despite rising edges on the S_AXI_ACLK pin. AXI4-Lite interface states are maintained, and AXI4-Lite interface output signal levels are held until S_AXI_ACLKEN is asserted again. When S_AXI_ACLKEN is de-asserted, AXI4-Lite interface inputs are not sampled, except S_AXI_ARESETn, which supersedes S_AXI_ACLKEN. The AXI4-Stream interfaces signals are not affected by the S_AXI_ACLKEN.

S_AXI_ARESETn

The S_AXI_ARESETn pin is an active-low, synchronous reset input for the AXI4-Lite interface. S_AXI_ARESETn supersedes S_AXI_ACLKEN, and when set to 0, the core resets at the next rising edge of S_AXI_ACLK even if S_AXI_ACLKEN is de-asserted. The S_AXI_ARESETn signal must be synchronous to the S_AXI_ACLK and must be held low for a minimum of 32 clock cycles of the slowest clock. The S_AXI_ARESETn input is

resynchronized to the ACLK clock domain. The AXI4-Stream interfaces and core signals are also reset by S_AXI_ARESETn.

Register Space

The standardized Xilinx Video IP register space is partitioned to control-, timing-, and core-specific registers. The Image Statistics core uses only one timing related register, ACTIVE_SIZE (0x0020), which allows specifying the input frame dimensions. By setting the core's control register, the method for gathering the statistical data is set. The generated data is read through the registers.

Table 2-10: **Register Names and Descriptions**

Address (hex) BASEADDR +	Register Name	Access Type	Double Buffered	Default Value	Register Description
0x0000	CONTROL	R/W	No	Power-on-Reset: 0x0	Bit 0: SW_ENABLE Bit 1: REG_UPDATE Bit 2: CLR_STATUS Bit 5: TEST_PATTERN* Bit 6: READOUT Bit 30: FRAME_SYNC_RESET (1: reset) Bit 31: SW_RESET (1: reset)
0x0004	STATUS	R/W	No	0	Bit 0: PROC_STARTED Bit 1: EOF Bit 4: DONE (Frame acquisition complete) Bit 5: Block RAMs cleared Bit 16: SLAVE_ERROR
0x0008	ERROR	R/W	No	0	Bit 0: SLAVE_EOL_EARLY Bit 1: SLAVE_EOL_LATE Bit 2: SLAVE_SOF_EARLY Bit 3: SLAVE_SOF_LATE
0x000C	IRQ_ENABLE	R/W	No	0	16-0: Interrupt enable bits corresponding to STATUS bits
0x0010	VERSION	R	N/A	0x06000000	7-0: REVISION_NUMBER 11-8: PATCH_ID 15-12: VERSION_REVISION 23-16: VERSION_MINOR 31-24: VERSION_MAJOR
0x0014	SYSDEBUG0	R	N/A	0	31-0: Frame Throughput monitor*
0x0018	SYSDEBUG1	R	N/A	0	31-0: Line Throughput monitor*
0x001C	SYSDEBUG2	R	N/A	0	31-0: Pixel Throughput monitor*

Table 2-10: **Register Names and Descriptions *(Cont'd)***

Address (hex) BASEADDR +	Register Name	Access Type	Double Buffered	Default Value	Register Description
0x0020	ACTIVE_SIZE	R/W	Yes	Specified via GUI	12-0: Number of Active Pixels per Scan line 28-16: Number of Active Lines per Frame
0x0100	HMAX0	R/W	Yes	¼ ACTIVE_COLS	Position of the first vertical zone delimiter
0x0104	HAMX1	R/W	Yes	½ ACTIVE_COLS	Position of the second vertical zone delimiter
0x0108	HAMX2	R/W	Yes	¾ ACTIVE_COLS	Position of the third vertical zone delimiter
0x010C	VAMX0	R/W	Yes	¼ ACTIVE_ROWS	Position of the first horizontal zone delimiter
0x0110	VAMX1	R/W	Yes	½ ACTIVE_ROWS	Position of the second horizontal zone delimiter
0x0114	VAMX2	R/W	Yes	¾ ACTIVE_ROWS	Position of the third horizontal zone delimiter
0x0118	HIST_ZOOM_FACTOR	R/W	Yes	0	Bit 0 and 1 control CC histogram zooming: 00: No zoom, full Cb and Cr range 01: Zoom by 2 10: Zoom by 4 11: Zoom by 8
0x011C	RGB_HIST_ZONE_EN	R/W	Yes	0xFFFF	Bits 0-15 correspond to zones 0-15, enabling RGB histogramming for the selected zones
0x0120	YCC_HIST_ZONE_EN	R/W	Yes	0xFFFF	Bits 0-15 correspond to zones 0-15, enabling Y and CC histogramming for the selected zones
0x0124	ZONE_ADDR	R/W	Yes	0	Bits 0-3 select a zone for readout
0x0128	COLOR_ADDR	R/W	Yes	0	Bits 0-1 select a color channel for readout 00: Red 01: Green 1X: Blue
0x012C	HIST_ADDR	R/W	Yes	0	Bits 0-[DATA_WIDTH-1] address histograms.
0x0130	ADDR_VALID	R/W	Yes	0	Bit 0 qualifies ZONE_ADDR, COLOR_ADDR and HIST_ADDR

Table 2-10: **Register Names and Descriptions *(Cont'd)***

Address (hex) BASEADDR +	Register Name	Access Type	Double Buffered	Default Value	Register Description
0x0134	MAX	R	No	0	Maximum value measured for the currently selected zone and color channel
0x0138	MIN	R	No	0	Minimum value measured for the currently selected zone and color channel
0x013C	SUM_LO	R	No	0	Lower 32 bits of the sum of values for the currently selected zone and color channel
0x0140	SUM_HI	R	No	0	Upper 32 bits of the sum of values for the currently selected zone and color channel
0x0144	POW_LO	R	No	0	Lower 32 bits of the summed of square values for the currently selected zone and color channel
0x0148	POW_HI	R	No	0	Upper 32 bits of the summed of square values for the currently selected zone and color channel
0x014C	HSOBEL_LO	R	No	0	Lower 32 bits of the sum of absolute values for the horizontal Sobel Filter output, applied to luminance values of the currently selected zone and color channel
0x0150	HSOBEL_HI	R	No	0	Upper 32 bits of the sum of absolute values for the horizontal Sobel Filter output, applied to luminance values of the currently selected zone and color channel
0x0154	VSOBEL_LO	R	No	0	Lower 32 bits of the sum of absolute values for the vertical Sobel Filter output, applied to luminance values of the currently selected zone and color channel
0x0158	VSOBEL_HI	R	No	0	Upper 32 bits of the sum of absolute values for the vertical Sobel Filter output, applied to luminance values of the currently selected zone and color channel
0x015C	LSOBEL_LO	R	No	0	Lower 32 bits of the sum of absolute values for the diagonal Sobel Filter output, applied to luminance values of the currently selected zone and color channel

Table 2-10: **Register Names and Descriptions *(Cont'd)***

Address (hex) BASEADDR +	Register Name	Access Type	Double Buffered	Default Value	Register Description
0x0160	LSOBEL_HI	R	No	0	Upper 32 bits of the sum of absolute values for the diagonal Sobel Filter output, applied to luminance values of the currently selected zone and color channel
0x0164	RSOBEL_LO	R	No	0	Lower 32 bits of the sum of absolute values for the anti-diagonal diagonal Sobel Filter output, applied to luminance values of the currently selected zone and color channel
0x0168	RSOBEL_HI	R	No	0	Upper 32 bits of the sum of absolute values for the anti-diagonal Sobel Filter output, applied to luminance values of the currently selected zone and color channel
0x016C	HIFREQ_LO	R	No	0	Lower 32 bits of the sum of absolute values of the high frequency filter output, applied to luminance values of the currently selected zone
0x0170	HIFREQ_HI	R	No	0	Upper 32 bits of the sum of absolute values of the high frequency filter output, applied to luminance values of the currently selected zone
0x0174	LOFREQ_LO	R	No	0	Lower 32 bits of the sum of absolute values of the low frequency filter output, applied to luminance values of the currently selected zone
0x0178	LOFREQ_HI	R	No	0	Upper 32 bits of the sum of absolute values of the high frequency filter output, applied to luminance values of the currently selected zone
0x017C	RHIST	R	No	0	Red histogram values calculated over the zones selected by RGB_HIST_ZONE_EN
0x0180	GHIST	R	No	0	Green histogram values calculated over the zones selected by RGB_HIST_ZONE_EN
0x0184	BHIST	R	No	0	Blue histogram values calculated over the zones selected by RGB_HIST_ZONE_EN
0x0188	YHIST	R	No	0	Luminance histogram values calculated over the zones selected by YCC_HIST_ZONE_EN

Table 2-10: **Register Names and Descriptions *(Cont'd)***

Address (hex) BASEADDR +	Register Name	Access Type	Double Buffered	Default Value	Register Description
0x018C	CCHIST	R	No	0	Two-dimensional Cr-Cb chrominance histogram values calculated over the zones selected by YCC_HIST_ZONE_EN
0x0190	DATA_VALID	R	No	0	Bit 0 qualifies valid data in core registers corresponding to the address inputs

CONTROL (0x0000) Register

Bit 0 of the CONTROL register, SW_ENABLE, facilitates enabling and disabling the core from software. Writing '0' to this bit effectively disables the core halting further operations, which blocks the propagation of all video signals. After Power up, or Global Reset, the SW_ENABLE defaults to 0 for the AXI4-Lite interface. Similar to the ACLKEN pin, the SW_ENABLE flag is not synchronized with the AXI4-Stream interfaces: Enabling or Disabling the core takes effect immediately, irrespective of the core processing status.

Bit 1 of the CONTROL register, REG_UPDATE is a write done semaphore for the host processor, which facilitates committing all user and timing register updates simultaneously. For the double buffered registers in the Image Statistics core, one set of registers (the processor registers) is directly accessed by the processor interface, while the other set (the active set) is actively used by the core. New values written to the processor registers will get copied over to the active set at the end of the AXI4-Stream frame, if and only if REG_UPDATE is set. Setting REG_UPDATE to 0 before updating multiple register values, then setting REG_UPDATE to 1 when updates are completed ensures all registers are updated simultaneously at the frame boundary enabling consistency in data gathering between frames.

Bit 2 of the CONTROL register, CLR_STATUS resets values of the STATUS register to 0, thereby clearing any interrupt requests (irq pin) as well.

Bit 5 of the CONTROL register, TEST_PATTERN, switches the core to test-pattern generator mode if debug features are enabled. See Appendix C, Debugging for more information. If debug features were not included at instantiation, this flag has no effect on the operation of the core. Switching test-pattern generator mode on or off is not synchronized to frame processing, therefore can lead to image tearing.

Bit 6 of the CONTROL register, READOUT, directs the Image Statistics core to bypass register readout or to exit register readout when set to 0. When set to 1, READOUT directs the core to enter register readout mode thereby producing the statistical data that had been calculated by the core.

Bits 30 and 31 of the CONTROL register, FRAME_SYNC_RESET and SW_RESET facilitate software reset. Setting SW_RESET reinitializes the core to GUI default values, all internal registers and outputs are cleared and held at initial values until SW_RESET is set to 0. The

SW_RESET flag is not synchronized with the AXI4-Stream interfaces. Resetting the core while frame processing is progress will cause image tearing. For applications where the software reset functionality is desirable, but image tearing has to be avoided a frame synchronized software reset (FRAME_SYNC_RESET) is available. Setting FRAME_SYNC_RESET to 1 resets the core at the end of the frame being processed, or immediately if the core is between frames when the FRAME_SYNC_RESET was asserted.

After reset, the FRAME_SYNC_RESET bit is automatically cleared, so the core can get ready to process the next frame of video as soon as possible. The default value of both RESET bits is 0. Core instances with no AXI4-Lite control interface can only be reset via the ARESETn pin.

STATUS (0x0004) Register

All bits of the STATUS register can be used to request an interrupt from the host processor. In order to facilitate identification of the interrupt source, bits of the STATUS register remain set after an event associated with the particular STATUS register bit, even if the event condition is not present at the time the interrupt is serviced.

Bits of the STATUS register can be cleared individually by writing '1' to the bit position.

Bit 0 of the STATUS register, PROC_STARTED, indicates that processing of a frame has commenced via the AXI4-Stream interface.

Bit 1 of the STATUS register, End-of-frame (EOF), indicates that the processing of a frame has completed.

Bit 4 of the STATUS register indicates that data acquisition is complete.

Bit 5 of the STATUS register indicates that the block RAMs are completely cleared of their data.

Bit 16 of the STATUS register, SLAVE_ERROR, indicates that one of the conditions monitored by the ERROR register has occurred.

ERROR (0x0008) Register

Bit 16 of the STATUS register, SLAVE_ERROR, indicates that one of the conditions monitored by the ERROR register has occurred. This bit can be used to request an interrupt from the host processor. In order to facilitate identification of the interrupt source, bits of the STATUS and ERROR registers remain set after an event associated with the particular ERROR register bit, even if the event condition is not present at the time the interrupt is serviced.

Bits of the ERROR register can be cleared individually by writing '1' to the bit position to be cleared.

- Bit 0 of the ERROR register, EOL_EARLY, indicates an error during processing a video frame via the AXI4-Stream slave port. The number of pixels received between the latest and the preceding End-Of-Line (EOL) signal was less than the value programmed into the ACTIVE_SIZE register.
- Bit 1 of the ERROR register, EOL_LATE, indicates an error during processing a video frame through the AXI4-Stream slave port. The number of pixels received between the last EOL signal surpassed the value programmed into the ACTIVE_SIZE register.
- Bit 2 of the ERROR register, SOF_EARLY, indicates an error during processing a video frame through the AXI4-Stream slave port. The number of pixels received between the latest and the preceding Start-Of-Frame (SOF) signal was less than the value programmed into the ACTIVE_SIZE register.
- Bit 3 of the ERROR register, SOF_LATE, indicates an error during processing a video frame through the AXI4-Stream slave port. The number of pixels received between the last SOF signal surpassed the value programmed into the ACTIVE_SIZE register.

IRQ_ENABLE (0x000C) Register

Any bits of the STATUS register can generate a host-processor interrupt request through the IRQ pin. The Interrupt Enable register facilitates selecting which bits of STATUS register will assert IRQ. Bits of the STATUS registers are masked by corresponding bits of the IRQ_ENABLE register (using AND), and the resulting terms are combined together to generate IRQ (using OR).

Version (0x0010) Register

Bit fields of the Version register facilitate software identification of the exact version of the hardware peripheral incorporated into a system. The core driver can use this read-only value to verify that the software is matched to the correct version of the hardware.

SYSDEBUG0 (0x0014) Register

The SYSDEBUG0, or Frame Throughput Monitor, register indicates the number of frames processed since power-up or the last time the core was reset. The SYSDEBUG registers can be useful to identify external memory / Frame buffer / or throughput bottlenecks in a video system. See Appendix C, Debugging for more information.

SYSDEBUG1 (0x0018) Register

The SYSDEBUG1, or Line Throughput Monitor, register indicates the number of lines processed since power-up or the last time the core was reset. The SYSDEBUG registers can be useful to identify external memory / Frame buffer / or throughput bottlenecks in a video system. See Appendix C, Debugging for more information.

SYSDEBUG2 (0x001C) Register

The SYSDEBUG2, or Pixel Throughput Monitor, register indicates the number of pixels processed since power-up or the last time the core was reset. The SYSDEBUG registers can be useful to identify external memory / Frame buffer / or throughput bottlenecks in a video system. See Appendix C, Debugging for more information.

ACTIVE_SIZE (0x0020) Register

The ACTIVE_SIZE register encodes the number of active pixels per scan line and the number of active scan lines per frame. The lower half-word (bits 12:0) encodes the number of active pixels per scan line. Supported values are between 32 and the value provided in the Maximum number of pixels per scan line field in the GUI. The upper half-word (bits 28:16) encodes the number of lines per frame. Supported values are 32 to 7680. To avoid processing errors, restrict values written to ACTIVE_SIZE to the range supported by the core instance.

Setting Up Zone Boundaries

The zone boundaries for the 16 zones can be set up by programming the positions of three vertical and three horizontal delimiters as shown in Figure 2-5. Complemented by the constraints that the top-left corner of Zone 0 is flush with the left-top corner of the active image, and the bottom-right corner of Zone 15 is flush with the bottom-right corner of the active image, these values uniquely define the corners of all zones.

Data is collected during the active (and non-blank) period of the frame, and all zones traversed by the current scan-line are updated with the input data in parallel. Zone boundaries should be set up before acquiring the first frame of data by programming the HMAX and VMAX registers.

NOTE: **The minimum horizontal and vertical size of each zone must be at least 2**, along with the following geometric constraints:

- 0 < hmax0 < hmax1 < hmax2 < *MAX_COLS*
- 0 < vmax0 < vmax1 < vmax2 < *MAX_ROWS*

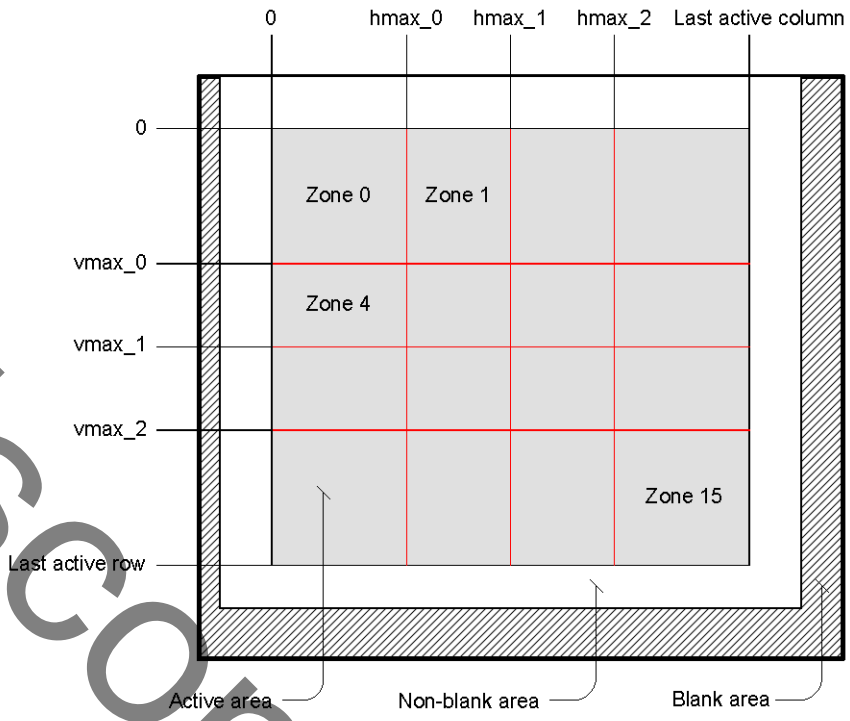


Figure 2-5: **Setting Up Zone Boundaries**

HMAX0 – HMAX2 (0x0100, 0x0104, 0x0108) Registers

Horizontal maximum registers involved in setting up the 16 zones used for gathering histogram data.

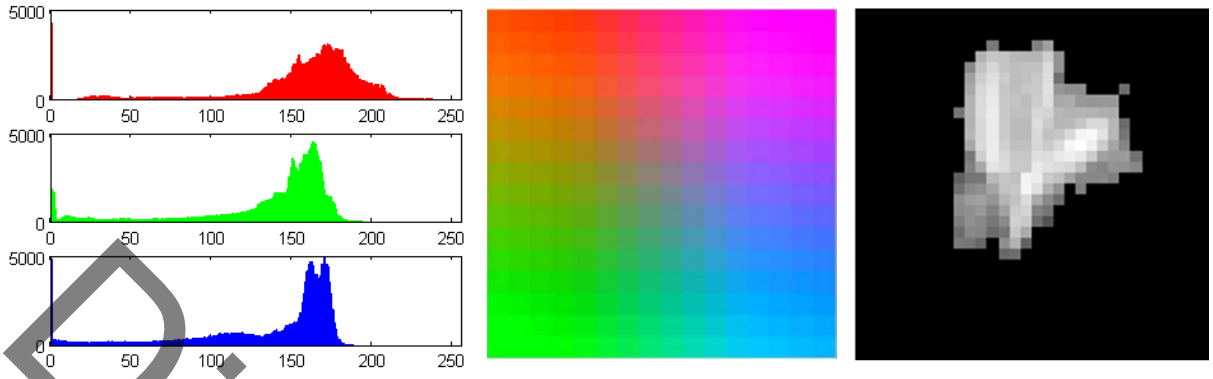
VMAX0 – VMAX2 (0x010C, 0x0110, 0x0114) Registers

Vertical maximum registers involved in setting up the 16 different zones used for gathering histogram data.

Histogram Data

For zones selected by a dynamically programmable register (`RGB_HIST_ZONE_EN`), the Image Statistics core bins R,G,B data and creates histograms shown in Figure 2-6a. Similarly, for zones selected by register `YCC_HIST_ZONE_EN`, Y and two-dimensional Cr-Cb histograms are calculated and shown in Figure 2-6c.

The two-dimensional Cr-Cb histogram (Figure 2-6c) contains information about the color content of a frame. Different hues have distinct locations in the Cr-Cb color-space (Figure 2-6b). The center location and variance of the color gamut can be derived from its two-dimensional Cr-Cb histogram. The bounding shape of the color gamut, along with the center location and variance of the two-dimensional Cr-Cb histogram, can be used to drive higher level algorithms [Ref 4] for white-balance correction.



a. Example RGB Histograms

b. Colors in Cr-Cb Space

c. Example Color Gamut in Cr-Cb Space

Figure 2-6: **Histogram Data**

For more information about histogram calculations, see Setting Up Histogram Calculations, page 26.

The resolution of the R,G,B and Y histograms is the same as the resolution of the input data. The two-dimensional Cr-Cb histogram contains the same number of bins, but due to the two-dimensional configuration, the resolution is $2^{DATA\_WIDTH/2}$ along the Cr-Cb axes.

The 2D histogram bin is setup with Cb on the X axis and Cr on the Y axis with the number of bins being 2^{DATA_WIDTH}. In Figure 2-6b, a `DATA_WIDTH` value of 8 is used which gives a total number of 256 bins numbered 0 - 255. The bins in Figure 2-6b are addressed left to right, bottom to top. The bin number represents the Cb and Cr value and is also the address of the block RAM. The contents of the bins are the number of pixels that contain that particular Cb, Cr value after the `HIST_ZOOM_FACTOR` is taken into consideration.

Setting Up Histogram Calculations

Histogram data is calculated and stored in block RAMs; hence calculating RGB and YCrCb histograms for all zones independently significantly increases the amount of FPGA resources required. Employing a mask to select which zones are involved in histogram calculation covers most typical applications.

- Calculating histogram values for one particular zone
- Calculating histogram values over an area of the image (such as the central zones, or zones in the corners)
- Calculating histogram values over the whole image

Figure 2-7 demonstrates how zones for RGB (red squares) and YCrCb (yellow circles) histogram calculations can be selected. For the example shown in Figure 2-7, zones 3, 4, 6, 7, 8 and 14 are selected for the Y and CrCb histograms. Correspondingly, bits 3,4,6,7,8 and 14 are set in `YCC_HIST_ZONE_EN`, resulting in a value of 0x000041D8.

Similarly, for the R,G, and B histograms, zones 1, 2, 3, 7, 8, 10, 11 and 14 are selected. Correspondingly bits 1, 2, 3, 7, 8, 10, 11 and 14 are set in `rgb_hist_zone_en`, resulting in a value of 0x00004D8E.

For the two-dimensional Cr-Cb histogram, there is another control, `STATS_HIST_ZOOM_FACTOR`, that helps tailor the Cr-Cb histogram calculation to the higher-level algorithm that consumes the 2D histogram results.

Consequently, Cr and Cb values have the same dynamic range as the input data. Cr and Cb are represented internally on `DATA_WIDTH` bits. A full precision Cr-Cb histogram would constitute a sparse 4k x 4k table that may be too large to implement within an FPGA. Therefore Cr and Cb are quantized to `DATA_WIDTH`/2 bits for histogramming. This quantization process inevitably involves loss of information. The use of the zoom factor (`STATS_HIST_ZOOM_FACTOR`) enables focusing on certain aspects of the histogram to minimize the effects of the information loss.

Figure 2-7: **Selecting Individual Zones for RGB and YCrCb Histograms**

Some higher level algorithms, such as gamut stretching [Ref 4], are concerned with the overall histogram, while other methods are concerned only with the central section (the area around the neutral point) to identify color casts. To support either type of algorithm, the histogram zoom factor allows the user to trade off resolution with range. The histogram zoom factor controls which bits of Cr and Cb values are selected for histogram binning. By setting the `HIST_ZOOM_FACTOR` to 0, the whole Cr-Cb histogram is represented at the output, as Cr and Cb values are simply quantized to `DATA_WIDTH`/2 bits. For example if `DATA_WIDTH` = 8, this quantization results in only the most significant four bits, bits 4, 5, 6 and 7, being used for histogram binning (see Table 2-11).

Table 2-11: **Histogram Zoom Bit Selections for DATA_WIDTH = 8 Bit Input Data**

Histogram Zoom Factor	Bits Used for Binning							
0	7	6	5	4	3	2	1	0
1	7	6	5	4	3	2	1	0
2	7	6	5	4	3	2	1	0
3	7	6	5	4	3	2	1	0

When HIST_ZOOM_FACTOR is set to a value other than 0, the resulting two-dimensional histogram represents only the central portion of the Cr-Cb histogram; pixels with extreme Cr-Cb values may fall outside the range represented by DATA_WIDTH/2 bits.

To enable further reduction of core footprint, RGB, Y, and Cr-Cb histograms can be individually enabled/disabled during generation time via the CORE Generator graphical user interface. If a particular type of histogram is not needed by the higher level algorithms, the core footprint can be reduced by 1,2, or 4 block RAMs depending on the input data resolution (DATA_WIDTH) and the target family.

HIST_ZOOM_FACTOR (0x0118) Register

Histogram zoom factor enables you to zoom into the center of the CrCb color space histogram (see Figures Figure 2-6b and Figure 2-6c) for greater resolution where the three color components converge.

The Image Statistics core treats the grey point of a CrCb color space as value 128. A HIST_ZOOM_FACTOR of 0 allows for the full CrCb color space range at the expense of resolution. As an example, if a DATA_WIDTH of 8 and a HIST_ZOOM_FACTOR of 0 are used, the possible color range is 0 - 255, for both Cb and Cr but only the four most significant bits are used giving 16 possible values for Cb and 16 possible values for Cr. On the 2D grid, the first bin will represent Cb, Cr values of 0 - 15, 0 - 15. Likewise the second bin will have Cb, Cr values of 16 - 31, 0 - 15. When a HIST_ZOOM_FACTOR of 1 is used, the color range for Cr and Cb is now 64 - 191 with Cb and Cr values of 64 - 71, 64 - 71 going into bin 0, 72 - 79, 64 - 71 going into bin 1, etc.

The formula representing the color space range for the Image Statistics core is as follows:

From: $2^{(DATA\_WIDTH - 1)} - 2^{(DATA\_WIDTH - (1 + HIST\_ZOOM\_FACTOR))}$

To: $2^{(DATA\_WIDTH - 1)} + 2^{(DATA\_WIDTH - (1 + HIST\_ZOOM\_FACTOR))} - 1$

The color space is divided into $2^{(DATA\_WIDTH/2)} \times 2^{(DATA\_WIDTH/2)}$ bins. Based on Figure 2-6b, the HIST_ADDR numbers the bins from left to right, bottom to top, $0 - 2^{(DATA\_WIDTH-1)}$. Each address location is associated with the Cb, Cr value and contains the number of pixels with that Cb, Cr color value.

RHIST (0x017C) Register

Histogram data for the specific zone and address selected for the red component.

GHIST (0x0180) Register

Histogram data for the specific zone and address selected for the green component.

BHIST (0x0184) Register

Histogram data for the specific zone and address selected for the blue component.

YHIST (0x0188) Register

Histogram data for the specific zone and address selected for the luma component.

CCHIST (0x018C) Register

Histogram data for the specific zone and address selected for the Cb and Cr components.

Addressing

Due to the large number of statistical data collected by the core, presenting all data simultaneously on core outputs is not feasible. Registers `ZONE_ADDR` and `COLOR_ADDR` facilitate reading out the max, min, sum and power result for specific zones and color channels.

The register `HIST_ADDR` facilitates addressing of histogram values.

The Image Statistics core provides a simple handshaking interface for reading out data. After setting the address registers as needed, setting bit 0 of the `ADDR_VALID` register signals to the core that valid addresses are present. In turn, the core fetches data corresponding to the addresses and marks valid data on the core outputs by writing a '1' to bit 0 of the DATA_VALID register (Figure 2-8).

All maximum, minimum, sum, sum of squares, Sobel and frequency contents can be read out by accessing zones 0-15 and color channels (coded 0,1,2) sequentially. If the host processor interface and the Image Statistics core are in the same CLK domain, addressing can be simplified, such that multiple addresses are supplied during the active portion of ADDR_VALID. When a sequence of valid addresses is presented to the core, the sequence of corresponding valid data becomes available with a latency of five CLK cycles (Figure 2-9).

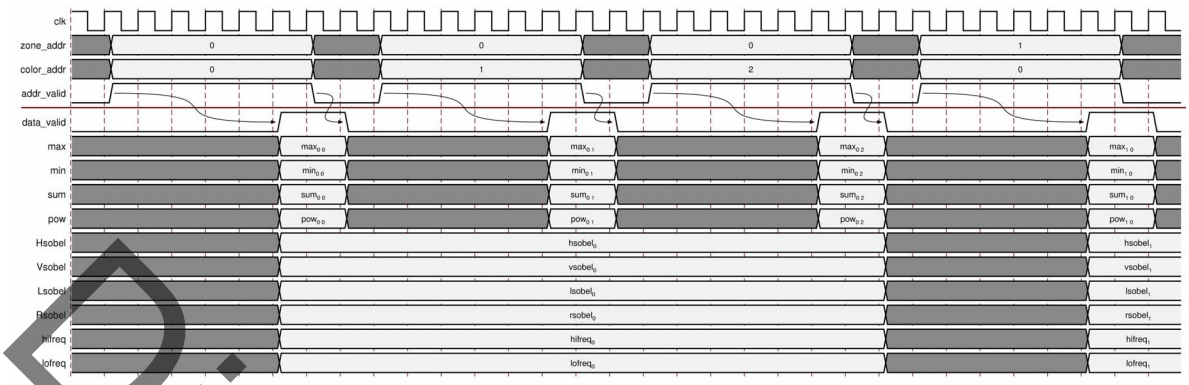


Figure 2-8: **Readout Addressing**

Figure 2-9 illustrates reading out histogram data. To shorten the readout period, histogram data can be read out in parallel with other statistical data.

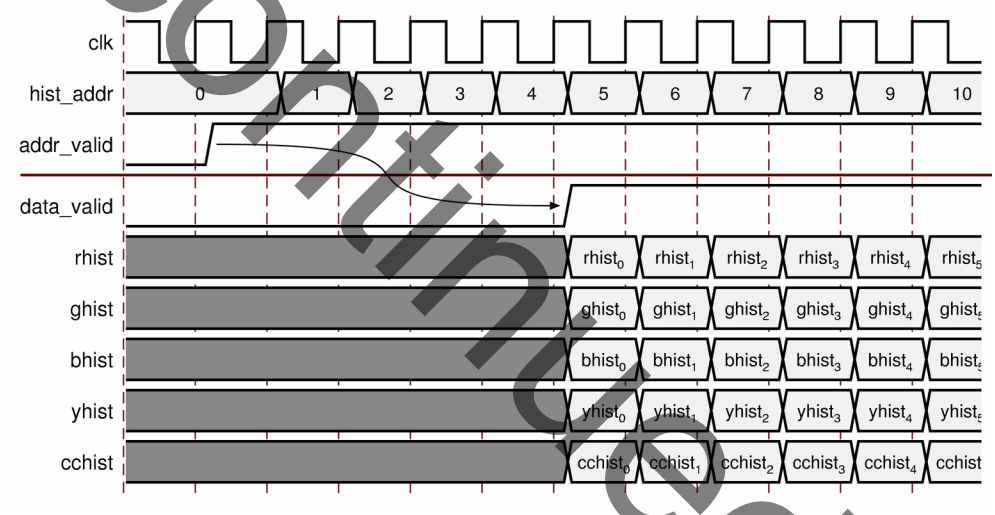


Figure 2-9: **Reading Out Histogram Data**

RGB_HIST_ZONE_EN (0x011C) Register

This register enables zones 0 -15 (setup using the HMAX and VMAX registers) for RGB histogram data gathering.

YCC_HIST_ZONE_EN (0x0120) Register

This register enables zones 0 -15 (setup using the HMAX and VMAX registers) for Y and CC histogram data gathering.

ZONE_ADDR (0x0124) Register

This register selects which one of the sixteen zones will be read out.

COLOR_ADDR (0x0128) Register

This register selects which color component of the RGB histogram will be read out:

- 00 – Red
- 01 – Green
- 1x – Blue

HIST_ADDR (0x012C) Register

Selects the specific address [0 – DATA_WIDTH-1] for histogram data.

ADDR_VALID (0x0130) Register

Bit 0 qualifies the `ZONE_ADDR`, `COLOR_ADDR`, and `HIST_ADDR`.

Minimum and Maximum Values

The minimum and maximum values can be useful for histogram stretching, Auto-Gain, Digital-Gain, Auto-Exposure, or simple White-Balance applications. These values are calculated for all zones and all R,G,B color channels simultaneously.

MAX (0x0134) Register

Maximum value measured for the currently selected zone and color channel.

MIN (0x0138) Register

Minimum value measured for the currently selected zone and color channel.

Sum of Color Values

The core provides the sum of color values for all zones (sum). The mean values for color channels can be calculated by dividing the sum value by the size of the zone (*N*):

$$\bar{x} = \frac{1}{N}\sum_{i=0}^{N-1} x_i = \frac{sum}{N}$$

Equation 2-1

SUM_LO/HI (0x013C, 0x0140) Register

Lower and upper values (64 bits total) containing the sum of the currently selected zone and color channel.

Sum of Squares of Color Values

The core provides the power output equal to the sum of squared values for all color channels and zones (*pow*), from which the signal power or the variance can be calculated:

$$\sigma^2 = \frac{1}{N}\left[\sum_{i=0}^{N-1} x_i^2\right] - \bar{x}^2 = \frac{pow}{N} - \bar{x}^2$$

Equation 2-2

POW_LO/HI (0x0144, 0x0148) Register

Lower and upper values (64 bits total) containing the sum of squares of the currently selected zone and color channel.

Edge Content

The Image Statistics core filters the luminance values calculated for all zones using the Sobel operators:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

Figure 2-10: **Horizontal, Vertical and Diagonal (Left and Right) Sobel Operators**

The Sobel operators are implemented without multipliers to reduce size and increase performance. The edge content outputs (*Hsobel*, *Vsobel*, *Lsobel*, *Rsobel*) provide the cumulative sums of absolute values of filtered luminance values:

$$Lsobel = \sum_{i=0}^{N-1} ABS\left(\frac{1}{32} FIR2D\left(x_i, \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}\right)\right)$$

Equation 2-3

Equation 2-3 describes the calculation of the upper-left to lower-right diagonal frequency content, *Lsobel*. Output values for *Vsobel*, *Lsobel*, and *Rsobel* are calculated similarly by using the corresponding coefficient matrixes from Figure 2-10.

HSOBEL_LO/HI (0x014C, 0x0150) Register

Contains the horizontal Sobel filter value. The LO register contains the lower 32 bits and the HI register contains the upper 32 bits.

VSOBEL_LO/HI (0x0154, 0x0158) Register

Contains the vertical Sobel filter value. The LO register contains the lower 32 bits and the HI register contains the upper 32 bits.

LSOBEL_LO/HI (0x015C, 0x0160) Register

Contains the top left to bottom right diagonal Sobel filter value. The LO register contains the lower 32 bits and the HI register contains the upper 32 bits.

RSOBEL_LO/HI (0x0164, 0x0168) Register

Contains the top right to bottom left diagonal Sobel filter value. The LO register contains the lower 32 bits and the HI register contains the upper 32 bits.

Frequency Content

The frequency content for each zone is calculated using the luminance channel. To calculate low-frequency content, luminance values are first low-pass filtered with a 7 tap FIR filter, with fixed coefficients [-1 0 9 16 9 0 -1]/32.

The low frequency power output (*LoFreq*) of the core provides the cumulative sum of the squared values of the FIR filter output for each zone:

$$LoFreq = \sum_{i=0}^{N-1} \left[\frac{FIR\left(x_i, \{-1,0,9,16,9,0,-1\}\right)}{32} \right]^2$$

Equation 2-4

In Equation , square brackets [] represent clipping at $max(x_i)=2^{DATA\_WIDTH-1}$ and clamping values at 0.

The high frequency power output (*HiFreq*) of the core provides the difference between the power of the original luminance values and the power of the low-pass filtered signal within each of these zones:

$$HiFreq = \lfloor pow - LoFreq \rfloor$$

Equation 2-5

In Equation 2-5, square brackets represent clamping values at 0.

HIFREQ_LO/HI (0x016C, 0x0170) Register

Contains the high frequency value. The LO register contains the lower 32 bits and the HI register contains the upper 32 bits.

LOFREQ_LO/HI (0x0174, 0x0178) Register

Contains the low frequency value. The LO register contains the lower 32 bits and the HI register contains the upper 32 bits.

DATA_VALID (0x0190) Register

Bit 0 qualifies valid data in core registers corresponding to the address inputs.

Interrupt Subsystem

STATUS register bits can trigger interrupts so embedded application developers can quickly identify faulty interfaces or incorrectly parameterized cores in a video system. Irrespective of whether the AXI4-Lite control interface is present or not, the core detects AXI4-Stream framing errors, as well as the beginning and the end of frame processing.

Events associated with bits of the STATUS register can generate a (level triggered) interrupt, if the corresponding bits of the interrupt enable register (IRQ_ENABLE) are set. Once set by the corresponding event, bits of the STATUS register stay set until the user application clears them by writing '1' to the desired bit positions. Using this mechanism the system processor can identify and clear the interrupt source.

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

General Design Guidelines

The Image Statistics core generates statistical information about the video data stream that passes through it. The statistical information is useful for applications such as Auto White Balance, Auto Exposure and Auto Gain. The core processes pixels provided through an AXI4-Stream slave interface, outputs pixels through an AXI4-Stream master interface, and can be controlled via an AXI4-Lite interface. The Image Statistics block cannot change the input/output image sizes, the input and output pixel clock rates, or the frame rate nor does it modify the video stream in any way.

RECOMMENDED: It is recommended that the Image Statistics core is used in conjunction with the Video In to AXI4-Stream and Video Timing Controller cores.

The Video Timing Controller core measures the timing parameters, such as number of active scan lines, number of active pixels per scan line of the image sensor. The Video In to AXI4-Stream core converts the incoming video data stream to AXI4-Stream.

Typically, the Image Statistics core is part of an Image Sensor Pipeline (ISP) System, as shown in Figure 3-1.

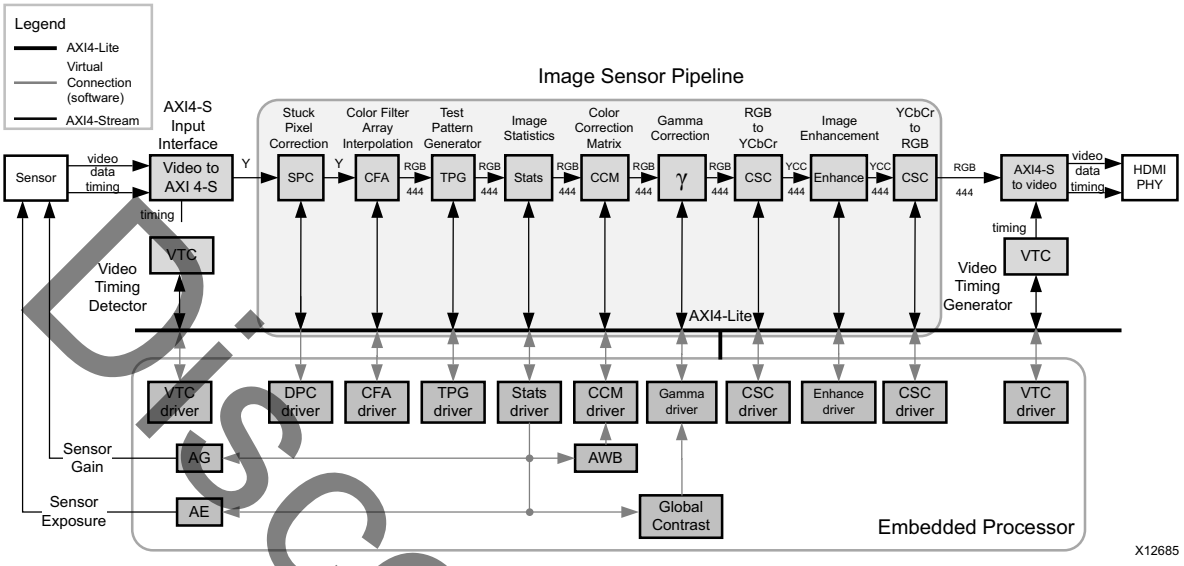


Figure 3-1: **Image Sensor Pipeline System with Image Statistics Core**

Clock, Enable, and Reset Considerations

ACLK

The master and slave AXI4-Stream video interfaces use the ACLK clock signal as their shared clock reference, as shown in Figure 3-2.

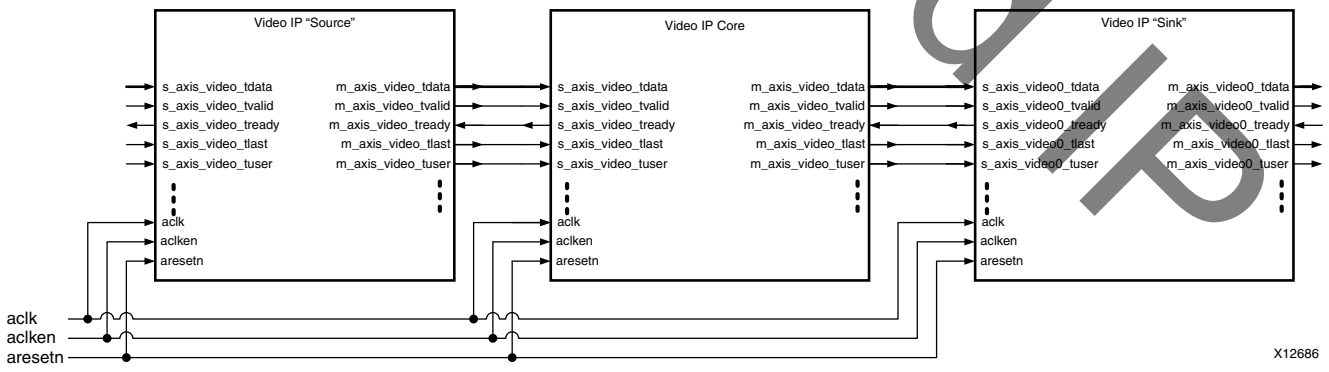


Figure 3-2: **Example of ACLK Routing in an ISP Processing Pipeline**

S_AXI_ACLK

The AXI4-Lite interface uses the A_AXI_ACLK pin as its clock source. The ACLK pin is not shared between the AXI4-Lite and AXI4-Stream interfaces. The Image Statistics core contains clock-domain crossing logic between the ACLK (AXI4-Stream and Video Processing) and S_AXI_ACLK (AXI4-Lite) clock domains. The core automatically ensures that the AXI4-Lite transactions completes even if the video processing is stalled with ARESETn, ACLKEN or with the video clock not running.

ACLKEN

The Image Statistics core has two enable options: the ACLKEN pin (hardware clock enable), and the software reset option provided through the AXI4-Lite control interface (when present).

ACLKEN may not be synchronized internally to AXI4-Stream frame processing therefore de-asserting ACLKEN for extended periods of time may lead to image tearing.

The ACLKEN pin facilitates:

- Multi-cycle path designs (high speed clock division without clock gating)
- Standby operation of subsystems to save on power
- Hardware controlled bring-up of system components

IMPORTANT: *When ACLKEN (clock enable) pins are used (toggled) in conjunction with a common clock source driving the master and slave sides of an AXI4-Stream interface, to prevent transaction errors the ACLKEN pins associated with the master and slave component interfaces must also be driven by the same signal (Figure 2-2).*

IMPORTANT: *When two cores connected through AXI4-Stream interfaces, where only the master or the slave interface has an ACLKEN port, which is not permanently tied high, the two interfaces should be connected through the AXI4-Stream Interconnect or AXI-FIFO cores to avoid data corruption (Figure 2-3).*

S_AXI_ACLKEN

The S_AXI_ACLKEN is the clock enable signal for the AXI4-Lite interface only. Driving this signal Low only affects the AXI4-Lite interface and does not halt the video processing in the ACLK clock domain.

ARESETn

The Image Statistics core has two reset source: the ARESETn pin (hardware reset), and the software reset option provided through the AXI4-Lite control interface (when present).

IMPORTANT: *ARESETn is not synchronized internally to AXI4-Stream frame processing. Deasserting ARESETn while a frame is being process leads to image tearing.*

The external reset pulse needs to be held for 32 ACLK cycles to reset the core. The ARESETn signal only resets the AXI4-Stream interfaces. The AXI4-Lite interface is unaffected by the ARESETn signal to allow the video processing core to be reset without halting the AXI4-Lite interface.

IMPORTANT: *When a system with multiple-clocks and corresponding reset signals are being reset, the reset generator has to ensure all signals are asserted/de-asserted long enough so that all interfaces and clock-domains are correctly reinitialized.*

S_AXI_ARESETn

The S_AXI_ARESETn signal is synchronous to the S_AXI_ACLK clock domain, but is internally synchronized to the ACLK clock domain. The S_AXI_ARESETn signal resets the entire core including the AXI4-Lite and AXI4-Stream interfaces.

System Considerations

The Image Statistics core needs to be configured for the actual image sensor frame size to operate properly. To gather the frame size information from the image sensor, connect the core to the Video In to AXI4-Stream input and the Video Timing Controller. The timing detector logic in the Video Timing Controller will gather the image sensor timing signals. The AXI4-Lite control interface on the Video Timing Controller allows the system processor to read out the measured frame dimensions, and program all downstream cores, such as the Image Statistics, with the appropriate image dimensions.

Clock Domain Interaction

The ARESETn and ACLKEN input signals will not reset or halt the AXI4-Lite interface. This allows the video processing to be reset or halted separately from the AXI4-Lite interface without disrupting AXI4-Lite transactions.

The AXI4-Lite interface will respond with an error if the core registers cannot be read or written within 128 S_AXI_ACLK clock cycles. The core registers cannot be read or written if the ARESETn signal is held low, if the ACLKEN signal is held low or if the ACLK signal is not connected or not running. If core register read does not complete, the AXI4-Lite read transaction will respond with **10** on the S_AXI_RRESP bus. Similarly, if a core register write

does not complete, the AXI4-Lite write transaction will respond with **10** on the S_AXI_BRESP bus. The S_AXI_ARESETn input signal resets the entire core.

Programming Sequence

If processing parameters (such as image size) need to be changed dynamically or the system needs to be reinitialized, there is a recommended sequence that should be followed. Pipelined Xilinx IP video cores should be disabled/reset from system output towards the system input, and these cores should be programmed/enabled from system input to system output. STATUS register bits allow system processors to identify the processing states of individual constituent cores, and successively disable a pipeline as one core after another is finished processing the last frame of data.

Error Propagation and Recovery

Parameterization and/or configuration registers define the dimensions of video frames video IP should process. Starting from a known state and based on configuration settings, the IP can predict when the beginning of the next frame is expected. Similarly, the IP can predict when the last pixel of each scan line is expected. SOF detected before it was expected (early), or SOF not present when it is expected (late), EOL detected before expected (early), or EOL not present when expected (late), signals error conditions indicative of either upstream communication errors or incorrect core configuration.

When SOF is detected early, the output SOF signal is generated early and this terminates the previous frame immediately. When SOF is detected late, the output SOF signal is generated according to the programmed values. Extra lines / pixels from the previous frame are dropped until the input SOF is captured.

Similarly, when EOL is detected early, the output EOL signal is generated early, and this terminates the previous line immediately. When EOL is detected late, the output EOL signal is generated according to the programmed values. Extra pixels from the previous line are dropped until the input EOL is captured.

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite environment.

Vivado Integrated Design Environment (IDE)

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click on the selected IP or select the Customize IP command from the toolbar or popup menu.

For details, see the sections, "Working with IP" and "Customizing IP for the Design" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3] and the "Working with the Vivado IDE" section in the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 7] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl console.

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

Interface

The Image Statistics core is easily configured to meet the developer's specific needs through the Vivado design tools interface. This section provides a quick reference to parameters that can be configured at generation time. Figure 4-1 shows the Image Statistics Vivado design tools interface.

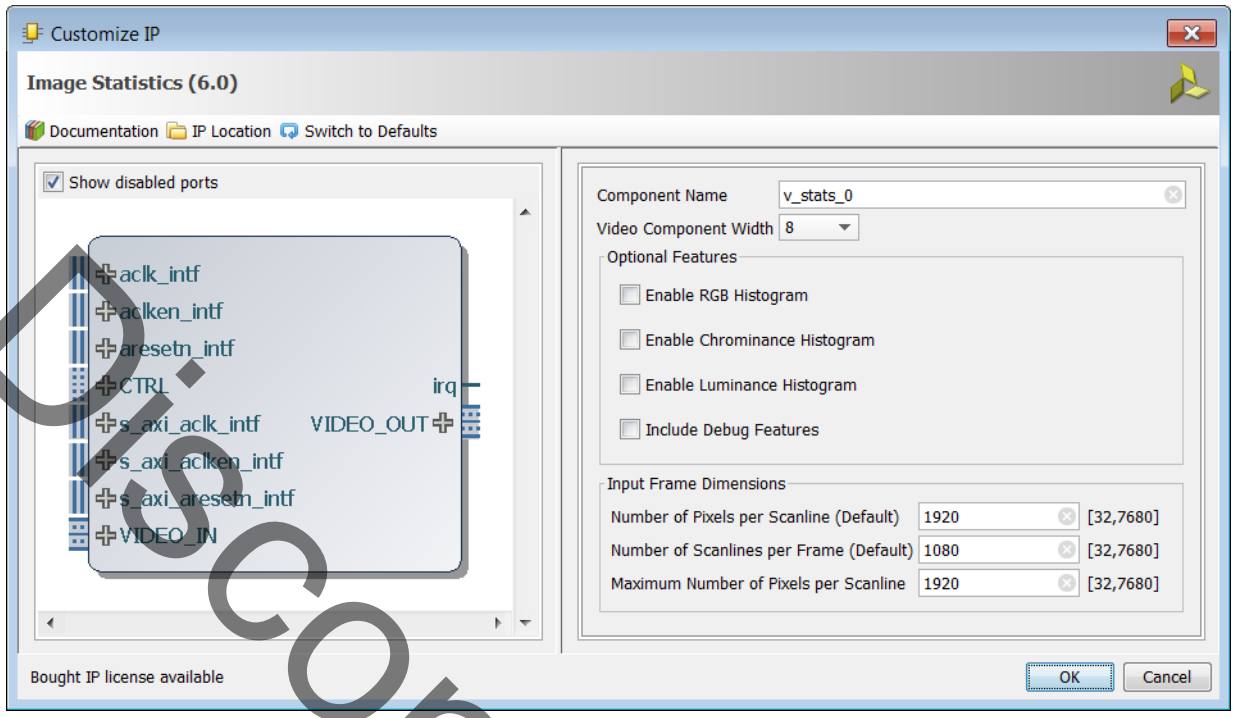


Figure 4-1: **Image Statistics Vivado IP Catalog GUI**

The GUI 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 "_". The name v_stats_v6_0 cannot be used as a component name.
- **Video Component Width:** Specifies the bit width of input data. Permitted values are 8, 10, or 12. When using IP Integrator, this parameter is automatically computed based on the Video Component Width of the video IP core connected to the slave AXI-Stream video interface.
- **Optional Features:** Storing and calculating histograms utilize block RAM resources in the FPGA. By specifying which histograms calculations are needed, this option can be used to reduce FPGA resources required for the generated core instance.
  - **Enable RGB Histograms:** The check box enables/disables instantiation of the R,G and B histogram calculating modules for zones pre-selected for RGB histogramming.
  - **Enable Chrominance Histogram:** The check box enables/disables instantiation of the two-dimensional chrominance (Cr-Cb) histogram calculating module for zones pre-selected for Y and CrCb histogramming

- **Enable Luminance Histogram:** The check box enables/disables instantiation of the luminance histogram calculating module for zones pre-selected for Y and CrCb histogramming.

IMPORTANT: *Storing and calculating histograms utilize block RAM resources in the FPGA. By specifying which histograms calculations are needed, this option can reduce FPGA resources required for the generated core instance.*

- **Include Debugging Features**: When selected, the core will be generated with debugging features, which simplify system design, testing and debugging. For more information, refer to Debugging Features in Appendix C.

IMPORTANT: *Debugging features are only available when the AXI4-Lite Register Interface is selected.*

- **Input Frame Dimensions**:
  - **Number of Active Pixels per Scan line**: The generated core will use the value specified in the CORE Generator GUI as the default value for the lower half-word of the ACTIVE_SIZE register.
  - **Number of Active Lines per Frame**: The generated core will use the value specified in the CORE Generator GUI as the default value for the upper half-word of the ACTIVE_SIZE register.
  - **Maximum Number of Active Pixels Per Scan line**: Specifies the maximum number of pixels per scan line that can be processed by the generated core instance. Permitted values are from 32 to 7680. Specifying this value is necessary to establish the depth of line buffers. The actual value selected for Number of Active Pixels per Scan line, or the corresponding lower half-word of the ACTIVE_SIZE register must always be less than the value provided by Maximum Number of Active Pixels Per Scan line. Using a tight upper-bound results in optimal block RAM usage.

Output Generation

For details, see "Generating IP Output Products" in the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Constraining the Core

This chapter contains information on applicable constraints for the Image Statistics core.

Required Constraints

The only constraints required are clock frequency constraints for the video clock, `clk`, and the AXI4-Lite clock, `s_axi_aclk`. Paths between the two clock domains should be constrained with a `max_delay` constraint and use the `datapathonly` flag, causing setup and hold checks to be ignored for signals that cross clock domains. These constraints are provided in the XDC constraints file included with the core.

Simulation

This chapter contains information about simulating IP in the Vivado® Design Suite environment. For comprehensive information about Vivado simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 12].

Synthesis and Implementation

For details about synthesis and implementation, see "Synthesizing IP" and "Implementing IP" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

C Model Reference

This chapter contains information for installing the Image Statistics Engine C-Model, and describes the file contents and directory structure.

Software Requirements

The Image Statistics v6.0 C models were compiled and tested with the following software:

Table 8-1: **Compilation Tools for the Bit Accurate C models**

Platform	C Compiler
Linux 32-bit and 64-bit	GCC 4.1.1
Windows 32-bit and 64-bit	Visual Studio 2008 (Visual C++ 8.0)

Installation

The installation of the C model requires updates to the PATH variable, as described below.

Linux

Ensure that the directory in which the `libIp_v_stats_v6_0_bitacc_cmodel.so` file is located is in your `$LD_LIBRARY_PATH` environment variable.

C Model File Contents

Unzipping the `v_stats_v6_0_bitacc_model.zip` file creates the directory structures and files shown in Table 8-2.

Table 8-2: **C Model Directory Structure and Contents**

Name	Description
/lin	Pre-compiled bit accurate ANSI C reference model for simulation on 32-bit Linux Platforms
libIp_v_stats_v6_0_bitacc_cmodel.lib	Image Statistics model shared object library (Linux platforms only)
run_bitacc_cmodel	Pre-compiled bit accurate executable for simulation on 32-bit Linux Platforms
/lin64	Pre-compiled bit accurate ANSI C reference model for simulation on 64-bit Linux Platforms
libIp_v_stats_v6_0_bitacc_cmodel.lib	Image Statistics model shared object library (Linux platforms only)
run_bitacc_cmodel	Pre-compiled bit accurate executable for simulation on 64-bit Linux Platforms
/nt	Pre-compiled bit accurate ANSI C reference model for simulation on 32-bit Windows Platforms
libIp_v_stats_v6_0_bitacc_cmodel.lib	Pre-compiled library file for win32 compilation (Windows platforms only)
run_bitacc_cmodel.exe	Pre-compiled bit accurate executable for simulation on 32-bit Windows Platforms
/nt64	Pre-compiled bit accurate ANSI C reference model for simulation on 64-bit Windows Platforms
libIp_v_stats_v6_0_bitacc_cmodel.lib	Pre-compiled library file for win64 compilation (Windows platforms only)
run_bitacc_cmodel.exe	Pre-compiled bit accurate executable for simulation on 64-bit Windows Platforms
README.txt	Release notes
pg008_v_stats.pdf	LogiCORE IP Image Statistics Product Guide
v_stats_v6_0_bitacc_cmodel.h	Model header file
run_bittacc_cmodel.c	Example code calling the C model
rgb_utils.h	Header file declaring the RGB image / video container type and support functions
bmp_utils.h	Header file declaring the bitmap (.bmp) image file I/O functions
video_utils.h	Header file declaring the generalized image / video container type, I/O, and support functions
Kodim11.bmp	768x512 sample test image from the True-color Kodak test images

Using the C Model

The bit-accurate C model is accessed through a set of functions and data structures, declared in the header file `v_stats_v6_0_bitacc_cmodel.h`.

Before using the model, the structures holding the inputs, generics and output of the Image Statistics instance have to be defined:

```
struct  xilinx_ip_v_stats_v6_0_generics stats_generics;
struct  xilinx_ip_v_stats_v6_0_inputs   stats_inputs;
struct  xilinx_ip_v_stats_v6_0_outputs  stats_outputs;
```

Declaration of the preceding structs can be found in `v_stats_v6_0_bitacc_cmodel.h`.

Table 8-3 lists the generic parameters taken by the Image Statistics v6.0 IP core bit accurate model, as well as the default values. For an actual instance of the core, these parameters can only be set in generation time through the GUI.

Table 8-3: **Model Generic Parameters and Default Values**

Generic Variable	Type	Default Value	Range	Description
DATA_WIDTH	int	8	8, 10, 12	Input / output data width
ACTIVE_COLS	int	1920	32 - 7680	Maximum number of columns in the active video
ACTIVE_ROWS	int	1080	32 - 7680	Maximum number of rows in the active video
MAX_COLS	int	1920	32 - 7680	Maximum number of columns that the input video will have. Must be greater than ACTIVE_COLS
HAS_RGB_HIST	int	0	0, 1	RGB histogramming feature
HAS_CC_HIST	int	0	0, 1	Chroma histogramming feature
HAS_Y_HIST	int	0	0, 1	Luma histogramming feature

Calling `xilinx_ip_v_stats_v6_0_get_default_generics(&stats_generics)` initializes the generics structure with the Image Statistics GUI default DATA_WIDTH value (8).

The structure `stats_inputs` defines the values of run-time parameters and the actual input image. The structure holds the following members:

Table 8-4: **Member Variables of the Input Structure**

Type	Name	Function
video_struct	video_in	Holds the input video stream (may contain multiple frames)
int	hmax0	Column index of the first vertical zone delineator (1)
int	hmax1	Column index of the second vertical zone delineator (1)
int	hmax2	Column index of the third vertical zone delineator (1)

Table 8-4: **Member Variables of the Input Structure *(Cont'd)***

Type	Name	Function
int	vmax0	Row index of the first horizontal zone delineator (1)
int	vmax1	Row index of the second horizontal zone delineator (1)
int	vmax2	Row index of the third horizontal zone delineator (1)
int	hist_zoom_factor	Controls CrCb histogram zoom around the gray (center point), which can be useful for white-balance algorithms. 0: No zoom, full Cb and Cr range represented (lowest resolution) 1: Zoom by 2 2: Zoom by 4 3: Zoom by 8 (highest resolution around gray point)
int	rgb_hist_zone_en	16 bit value, each bit controlling whether or not the corresponding zone is included in RGB histogram calculation.
int	ycc_hist_zone_en	16 bit value, each bit controlling whether or not the corresponding zone is included in YCC histogram calculation.
int	active_rows	Maximum number of rows in the active video
int	active_cols	Maximum number of columns in the active video

1. See Figure 3-2 for the definition of zone delineators.

`xilinx_ip_v_stats_v6_0_get_default_inputs(&stats_generics, &stats_inputs)` initializes members of the input structure with the Image Statistics GUI default values.

IMPORTANT: The `video_in` variable is not initialized, as the initialization depends on the actual test image to be simulated. The next chapter describes the initialization of the `video_in` structure.

IMPORTANT: Before calling `xilinx_ip_v_stats_v6_0_get_default_inputs()` it is advised to initialize the `video_in` structure by loading an image or set of video frames. The horizontal and vertical delineators are set by default such that the input images are split into zones with identical dimensions.

After the inputs are defined the model can be simulated by calling the function.

```
int xilinx_ip_v_stats_v6_0_bitacc_simulate(
      struct xilinx_ip_v_stats_v6_0_generics* generics,
      struct xilinx_ip_v_stats_v6_0_inputs*   inputs,
      struct xilinx_ip_v_stats_v6_0_outputs*  outputs).
```

Results are provided in the outputs structure, which contains only one member, type `video_struct`.

After the outputs were evaluated and/or saved, dynamically allocated memory for input and output video structures must be released by calling function

```
void xilinx_ip_v_stats_v6_0_destroy(
      struct xilinx_ip_v_stats_v6_0_inputs *input,
```

```
struct xilinx_ip_v_stats_v6_0_outputs *output).
```

Successful execution of all provided functions except for the destroy function return value 0; otherwise a non-zero error code indicates that problems were encountered during function calls.

Image Statistics Input and Output Video Structure

Input images or video streams can be provided to the Image Statistics reference model using the video_struct structure, defined in video_utils.h:

```
struct video_struct{
    int      frames, rows, cols, bits_per_component, mode;
    uint16*** data[5]; };
```

Table 8-5 lists video structure.

Table 8-5: **Member Variables of the Video Structure**

Member Variable	Designation
frames	Number of video/image frames in the data structure
rows	Number of rows per frame Pertaining to the image plane with the most rows and columns, such as the luminance channel for yuv data. Frame dimensions are assumed constant through the all frames of the video stream, however different planes, such as y,u and v may have different dimensions.
cols	Number of columns per frame Pertaining to the image plane with the most rows and columns, such as the luminance channel for yuv data. Frame dimensions are assumed constant through the all frames of the video stream, however different planes, such as y,u and v may have different dimensions.
bits_per_ component	Number of bits per color channel / component. All image planes are assumed to have the same color/component representation. Maximum number of bits per component is 16.
mode	Contains information about the designation of data planes. Named constants to be assigned to mode are listed in Table 8-6.
data	Set of 5 pointers to 3 dimensional arrays containing data for image planes. data is in 16 bit unsigned integer format accessed as data[plane][frame][row][col]

Table 8-6: **Named Constants for Video Modes with Corresponding Planes and Representations**

Mode	Planes	Video Representation
FORMAT_MONO	1	Monochrome – Luminance only.
FORMAT_RGB(1)	3	RGB image / video data
FORMAT_C444	3	444 YUV, or YCrCb image / video data

Table 8-6: **Named Constants for Video Modes with Corresponding Planes and Representations**

Mode	Planes	Video Representation
FORMAT_C422	3	422 format YUV video, (u,v chrominance channels horizontally sub-sampled)
FORMAT_C420	3	420 format YUV video, (u,v sub-sampled both horizontally and vertically)
FORMAT_MONO_M	3	Monochrome (Luminance) video with Motion.
FORMAT_RGBA	4	RGB image / video data with alpha (transparency) channel
FORMAT_C420_M	5	420 YUV video with Motion
FORMAT_C422_M	5	422 YUV video with Motion
FORMAT_C444_M	5	444 YUV video with Motion
FORMAT_RGBM	5	RGB video with Motion

1. The Image Statistics core supports the FORMAT_RGB mode

Initializing the Image Statistics Input Video Structure

The easiest way to assign stimuli values to the input video structure is to initialize it with an image or video stream. The bmp_util.h and video_util.h header files packaged with the bit accurate C models contain functions to facilitate file I/O.

Bitmap Image Files

The header bmp_utils.h declares functions which help access files in Windows Bitmap format (http://en.wikipedia.org/wiki/BMP_file_format). However, this format limits color depth to a maximum of 8 bits per pixel, and operates on images with 3 planes (R,G,B). Therefore, functions

```
int write_bmp(FILE *outfile, struct rgb8_video_struct *rgb8_video);
int read_bmp(FILE *infile, struct rgb8_video_struct *rgb8_video);
```

operate on arguments type `rgb8_video_struct`, which is defined in rgb_utils.h. Also, both functions support only true-color, non-indexed formats with 24 bits per pixel.

Exchanging data between rgb8_video_struct and general video_struct type frames/videos is facilitated by functions:

```
int copy_rgb8_to_video( struct rgb8_video_struct* rgb8_in,
                        struct video_struct* video_out );

int copy_video_to_rgb8( struct video_struct* video_in,
                        struct rgb8_video_struct* rgb8_out );
```

Note: All image / video manipulation utility functions expect both input and output structures initialized, for example, pointing to a structure that has been allocated in memory, either as static or dynamic variables. Moreover, the input structure must have the dynamically allocated container (data[ ] or r[ ],g[ ],b[ ]) structures already allocated and initialized with the input frame(s). If the output container structure is pre-allocated at the time of the function call, the utility functions verify and throw an error if the output container size does not match the size of the expected output. If the

output container structure is not pre-allocated, the utility functions will create the appropriate container to hold results.

Binary Image/Video Files

The header video_utils.h declares functions which help load and save generalized video files in raw, uncompressed format. Functions

```
int read_video( FILE* infile,  struct video_struct* in_video);
int write_video(FILE* outfile, struct video_struct* out_video);
```

effectively serialize the `video_struct` structure. The corresponding file contains a small, plain text header defining, "Mode", "Frames", "Rows", "Columns", and "Bits per Pixel". The plain text header is followed by binary data, 16 bits per component in scan line continuous format. Subsequent frames contain as many component planes as defined by the video mode value selected. Also, the size (rows, columns) of component planes may differ within each frame as defined by the actual video mode selected.

Working with video_struct Containers

Header file video_utils.h define functions to simplify access to video data in `video_struct`.

```
int video_planes_per_mode(int mode);
int video_rows_per_plane(struct video_struct* video, int plane);
int video_cols_per_plane(struct video_struct* video, int plane);
```

Function `video_planes_per_mode` returns the number of component planes defined by the mode variable, as described in Table 6. Functions `video_rows_per_plane` and `video_cols_per_plane` return the number of rows and columns in a given plane of the selected video structure. The following example demonstrates using these functions in conjunction to process all pixels within a video stream stored in variable `in_video`, with the following construct:

```
for (int frame = 0; frame < in_video->frames; frame++) {
  for (int plane = 0; plane < video_planes_per_mode(in_video->mode); plane++) {
    for (int row = 0; row < rows_per_plane(in_video,plane); row++) {
      for (int col = 0; col < cols_per_plane(in_video,plane); col++) {
        // User defined pixel operations on
        // in_video->data[plane][frame][row][col]
      }
    }
  }
}
```

Destroy the Video Structure

Finally, the video structure must be destroyed to free up memory used to store the video structure.

C Model Example Code

An example C file, run_bitacc_cmodel.c, is provided. This demonstrates the steps required to run the model. After following the compilation instructions, you will want to run the example executable.

The executable takes the path/name of the input file and the path/name of the output file as parameters. If invoked with insufficient parameters, the following help message is printed:

```
Usage: run_bitacc_cmodel in_file out_file
       in_file     : path/name of the input  BMP file
       out_file    : path/name of the output TXT file
```

During successful execution a text file, containing the statistical information by zones and color channels, is created. These output values are bit-true to the corresponding IP core output values, addressed by `ZONE_ADDR`, and `COLOR_ADDR`. Histogram values are also contained in the resulting text file in a tabulated format.

Compiling with the Image Statistics C Model

Linux

To compile the example code, first ensure that the directory in which the files `libIp_v_stats_v6_0_bitacc_cmodel.so` is located is present in your $LD_LIBRARY_PATH environment variable. This shared library is referenced during the compilation and linking process. Then cd into the directory where the header files, the library files and `run_bitacc_cmodel.c` were unpacked. The library and header files are referenced during the compilation and linking process.

Place the header file and C source file in a single directory. Then in that directory, compile using the GNU C Compiler:

```
gcc -m32 -x c++ ../run_bitacc_cmodel.c ../parsers.c -o run_bitacc_cmodel -L.
-lIp_v_stats_v6_0_bitacc_cmodel -Wl,-rpath,.

gcc -m64 -x c++ ../run_bitacc_cmodel.c ../parsers.c -o run_bitacc_cmodel -L.
-lIp_v_stats_v6_0_bitacc_cmodel -Wl,-rpath,.
```

Windows

Precompiled library `v_stats_v6_0_bitacc_cmodel.lib`, and top level demonstration code run_bitacc_cmodel.c should be compiled with an ANSI C compliant compiler under Windows. Here an example is presented using Microsoft Visual Studio.

In Visual Studio create a new, empty Win32 Console Application project. As existing items, add:

- `libIpv_stats_v6_0_bitacc_cmodel.lib` to the “Resource Files” folder of the project
- `run_bitacc_cmodel.c` to the “Source Files” folder of the project
- `v_stats_v6_0_bitacc_cmodel.h` header files to “Header Files” folder of the project (optional)

After the project has been created and populated, it needs to be compiled and linked (built) in order to create a win32 executable. To perform the build step, choose “Build Solution” from the Build menu. An executable matching the project name has been created either in the Debug or Release subdirectories under the project location based on whether “Debug” or “Release” has been selected in the “Configuration Manager” under the Build menu.

Detailed Example Design

No example design is available at this time. For a comprehensive listing of Video and Imaging application notes, white papers, related IP cores including the most recent reference designs available, see the Video and Imaging Resources page at:

www.xilinx.com/esp/video/refdes_listing.htm

Test Bench

This chapter contains information about the provided test bench in the Vivado® Design Suite environment.

Demonstration Test Bench

A demonstration test bench is provided with the core which enables you to observe core behavior in a typical scenario. This test bench is generated together with the core in Vivado Design Suite. You are encouraged to make simple modifications to the configurations and observe the changes in the waveform.

Directory and File Contents

The following files are expected to be generated in the in the demonstration test bench output directory:

- `axi4lite_mst.v`
- `axi4s_video_mst.v`
- `axi4s_video_slv.v`
- `ce_generator.v`
- `tb_<IP_instance_name>.v`

Test Bench Structure

The top-level entity is **`tb_<IP_instance_name>`**.

It instantiates the following modules:

- `DUT`

  The <IP> core instance under test.

- `axi4lite_mst`

The AXI4-Lite master module, which initiates AXI4-Lite transactions to program core registers.

- `axi4s_video_mst`

  The AXI4-Stream master module, which generates ramp data and initiates AXI4-Stream transactions to provide video stimuli for the core and can also be used to open stimuli files generated from the reference C models and convert them into corresponding AXI4-Stream transactions.

  To do this, edit `tb_<IP_instance_name>.v`:

  a. Add define macro for the stimuli file name and directory path
  `define STIMULI_FILE_NAME<path><filename>.`

  b. Comment-out/remove the following line:
  `MST.is_ramp_gen(`C_ACTIVE_ROWS, `C_ACTIVE_COLS, 2);`
  and replace with the following line:
  `MST.use_file(`STIMULI_FILE_NAME);`

  For information on how to generate stimuli files, see *Chapter 4, C Model Reference*.

- `axi4s_video_slv`

  The AXI4-Stream slave module, which acts as a passive slave to provide handshake signals for the AXI4-Stream transactions from the core output, can be used to open the data files generated from the reference C model and verify the output from the core.

  To do this, edit `tb_<IP_instance_name>.v`:

  a. Add define macro for the golden file name and directory path
  `define GOLDEN_FILE_NAME "<path><filename>".`

  b. Comment out the following line:
  `SLV.is_passive;`
  and replace with the following line:
  `SLV.use_file(`GOLDEN_FILE_NAME);`

  For information on how to generate golden files, see *Chapter 4, C Model Reference*.

- `ce_gen`

  Programmable Clock Enable (`ACLKEN`) generator.

Verification, Compliance, and Interoperability

This chapter contains details about verification and testing of the core.

Simulation

A highly parameterizable test bench was used to test the Image Statistics core. Testing included the following:

- Register accesses
- Processing of multiple frames of data
- AXI4-Stream bidirectional data-throttling tests
- Testing detection and recovery from various AXI4-Stream framing error scenarios
- Testing different `ACLKEN` and `ARESETn` assertion scenarios
- Testing of various frame sizes
- Varying parameter settings

Hardware Testing

The Image Statistics core has been tested in a variety of hardware platforms at Xilinx to represent a variety of parameterizations, including the following:

- A test design was developed for the core that incorporated a MicroBlaze™ processor, AXI4-Lite interconnect and various other peripherals. The software for the test system included pre-generated input and output data along with live video stream. The MicroBlaze processor was responsible for:
  - Initializing the appropriate input and output buffers
  - Initializing the Image Statistics core.
  - Launching the test.

- Comparing the output of the core against the expected results.
- Reporting the pass/fail status of the test and any errors that were found.

Migrating and Upgrading

This appendix contains information about migrating from an ISE design to the Vivado Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading their IP core, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migration to Vivado Design Suite, see *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 8].

Upgrading in Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Parameter Changes

There are no parameter changes.

Port Changes

There are no port changes.

Other Changes

Migrating from v3.0 to v4.00.a

From version v3.0 to v4.00.a of the Image Statistics core the following significant changes took place:

- XSVI interfaces were replaced by AXI4-Stream interfaces.

- The pCore and General Purpose Processor became obsolete and were removed.
- Native support for EDK was added. The Image Statistics core now appears in the EDK IP Catalog.
- Debugging features were added.
- The AXI4-Lite control interface register map is now standard for all Xilinx video IP cores.

Because of the complex nature of these changes, replacing a v3.0 version of the core in a customer design is not trivial. An existing EDK pCore instance can be converted from XSVI to AXI4-Stream, using the Video In to AXI4-Stream core or components from XAPP521, *Bridging Xilinx Streaming Video Interface with the AXI4-Stream Protocol*.

A v3.0 pCore instance in EDK can be replaced from v4.00.a directly from the EDK IP Catalog. However, the application software needs to be updated for the changed functionality and addresses of the IRQ_ENABLE, STATUS, ERROR, and the core's specific registers.

An ISE design using the General Purpose Processor interface, all of the following steps will be necessary:

- Timing detection and generation using the Video Timing Controller core.
- Replacing XSVI interfaces with conversion modules described in XAPP521.

Migrating from v4.00.a to v5.00.a

From v4.00.a to v5.00.a of the Image Statistics core, the following changes took place:

- The core originally had aclk, aclken and aresetn to control both the Video over AXI4-Stream and AXI4-Lite interfaces. Separate clock, clock enable and reset pins now control the Video over AXI4-Stream and the AXI4-Lite interfaces with clock domain crossing logic added to the core to handle the dissimilar clock domains between the AXI4-Lite and Video over AXI4-Stream domains.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the Image Statistics core, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support Web Case.

Documentation

This product guide is the main document associated with the Image Statistics. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core are listed below, and can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)

- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Answer Records for the Image Statistics Core

AR 54527

Contacting Technical Support

Xilinx provides technical support at www.xilinx.com/support for this 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.

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- A block diagram of the video system that explains the video source, destination and IP (custom and Xilinx) used.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Debug Tools

There are many tools available to address Image Statistics core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Lab Tools

Vivado® lab tools insert logic analyzer and virtual I/O cores directly into your design. Vivado lab tools allows you to set trigger conditions to capture application and integrated

block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx devices in hardware.

The Vivado lab tools logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

Reference Boards

Various Xilinx development boards support Image Statistics. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series evaluation boards
  - KC705
  - KC724

C Model Reference

See *C Model Reference* in this guide for tips and instructions for using the provided C model files to debug your design.

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado lab tools are a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Vivado lab tools for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.

- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the LOCKED port.
- If your outputs go to 0, check your licensing.

Core Bypass Option

The bypass option facilitates establishing a straight through connection between input (AXI4-Stream slave) and output (AXI4-Stream master) interfaces bypassing any processing functionality.

Flag BYPASS (bit 4 of the CONTROL register) can turn bypass on (1) or off, when the core instance Debugging Features were enabled at generation. Within the IP this switch controls multiplexers in the AXI4-Stream path.

In bypass mode the core processing function is bypassed, and the core repeats AXI4-Stream input samples on its output.

Starting a system with all processing cores set to bypass, then by turning bypass off from the system input towards the system output allows verification of subsequent cores with known good stimuli.

Built-in Test-Pattern Generator

The optional built-in test-pattern generator facilitates to temporarily feed the output AXI4-Stream master interface with a predefined pattern.

Flag TEST_PATTERN (bit 5 of the CONTROL register) can turn test-pattern generation on (1) or off, when the core instance **Debugging Features** were enabled at generation. Within the IP this switch controls multiplexers in the AXI4-Stream path, switching between the regular core processing output and the test-pattern generator. When enabled, a set of counters generate 256 scan-lines of color-bars, each color bar 64 pixels wide, repetitively cycling through Black, Green, Blue, Cyan, Red, Yellow, Magenta, and White colors till the end of each scan-line. After the Color-Bars segment, the rest of the frame is filled with a monochrome horizontal and vertical ramp.

Starting a system with all processing cores set to test-pattern mode, then by turning test-pattern generation off from the system output towards the system input allows successive bring-up and parameterization of subsequent cores.

Throughput Monitors

Throughput monitors enable monitoring processing performance within the core. This information can be used to help debug frame-buffer bandwidth limitation issues, and if possible, allow video application software to balance memory pathways.

Often times video systems, with multiport access to a shared external memory, have different processing islands. For example, a pre-processing sub-system working in the input video clock domain may clean up, transform, and write a video stream, or multiple video streams to memory. The processing sub-system may read the frames out, process, scale, encode, then write frames back to the frame buffer, in a separate processing clock domain.

Finally, the output sub-system may format the data and read out frames locked to an external clock.

Typically, access to external memory using a multiport memory controller involves arbitration between competing streams. However, to maximize the throughput of the system, different memory ports may need different specific priorities. To fine tune the arbitration and dynamically balance frame rates, it is beneficial to have access to throughput information measured in different video datapaths.

The `SYSDEBUG0` (0x0014) (or Frame Throughput Monitor) indicates the number of frames processed since power-up or the last time the core was reset. The `SYSDEBUG1` (0x0018), or Line Throughput Monitor, register indicates the number of lines processed since power-up or the last time the core was reset. The `SYSDEBUG2` (0x001C), or Pixel Throughput Monitor, register indicates the number of pixels processed since power-up or the last time the core was reset.

Priorities of memory access points can be modified by the application software dynamically to equalize frame, or partial frame rates.

Evaluation Core Timeout

The Image Statistics hardware evaluation core times out after approximately eight hours of operation. The output is driven to zero. This results in a black screen for RGB color systems and in a dark-green screen for YUV color systems.

Interface Debug

AXI4-Lite Interfaces

Table C-1 describes how to troubleshoot the AXI4-Lite interface.

Table C-1: **Troubleshooting the AXI4-Lite Interface**

Symptom	Solution
Readback from the Version Register through the AXI4-Lite interface times out, or a core instance without an AXI4-Lite interface seems non-responsive.	Are the S_AXI_ACLK and ACLK pins connected? The VERSION_REGISTER readout issue may be indicative of the core not receiving the AXI4-Lite interface.
Readback from the Version Register through the AXI4-Lite interface times out, or a core instance without an AXI4-Lite interface seems non-responsive.	Is the core enabled? Is s_axi_aclken connected to vcc? Verify that signal ACLKEN is connected to either net_vcc or to a designated clock enable signal.
Readback from the Version Register through the AXI4-Lite interface times out, or a core instance without an AXI4-Lite interface seems non-responsive.	Is the core in reset? S_AXI_ARESETn and ARESETn should be connected to vcc for the core not to be in reset. Verify that the S_AXI_ARESETn and ARESETn signals are connected to either net_vcc or to a designated reset signal.
Readback value for the VERSION_REGISTER is different from expected default values	The core and/or the driver in a legacy project has not been updated. Ensure that old core versions, implementation files, and implementation caches have been cleared.

Assuming the AXI4-Lite interface works, the second step is to bring up the AXI4-Stream interfaces.

AXI4-Stream Interfaces

Table C-2 describes how to troubleshoot the AXI4-Stream interface.

Table C-2: **Troubleshooting AXI4-Stream Interface**

Symptom	Solution
Bit 0 of the ERROR register reads back set.	Bit 0 of the ERROR register, EOL_EARLY, indicates the number of pixels received between the latest and the preceding End-Of-Line (EOL) signal was less than the value programmed into the ACTIVE_SIZE register. If the value was provided by the Video Timing Controller core, read out ACTIVE_SIZE register value from the VTC core again, and make sure that the TIMING_LOCKED flag is set in the VTC core. Otherwise, using Vivado Lab Tools, measure the number of active AXI4-Stream transactions between EOL pulses.
Bit 1 of the ERROR register reads back set.	Bit 1 of the ERROR register, EOL_LATE, indicates the number of pixels received between the last End-Of-Line (EOL) signal surpassed the value programmed into the ACTIVE_SIZE register. If the value was provided by the Video Timing Controller core, read out ACTIVE_SIZE register value from the VTC core again, and make sure that the TIMING_LOCKED flag is set in the VTC core. Otherwise, using Vivado Lab Tools, measure the number of active AXI4-Stream transactions between EOL pulses.

Table C-2: **Troubleshooting AXI4-Stream Interface *(Cont'd)***

Symptom	Solution
Bit 2 or Bit 3 of the ERROR register reads back set.	Bit 2 of the ERROR register, SOF_EARLY, and bit 3 of the ERROR register SOF_LATE indicate the number of pixels received between the latest and the preceding Start-Of-Frame (SOF) differ from the value programmed into the ACTIVE_SIZE register. If the value was provided by the Video Timing Controller core, read out ACTIVE_SIZE register value from the VTC core again, and make sure that the TIMING_LOCKED flag is set in the VTC core. Otherwise, using Vivado Lab Tools, measure the number EOL pulses between subsequent SOF pulses.
s_axis_video_tready stuck low, the upstream core cannot send data.	During initialization, line-, and frame-flushing, the core keeps its s_axis_video_tready input low. Afterwards, the core should assert s_axis_video_tready automatically. Is m_axis_video_tready low? If so, the core cannot send data downstream, and the internal FIFOs are full.
m_axis_video_tvalid stuck low, the downstream core is not receiving data	• No data is generated during the first two lines of processing. • If the programmed active number of pixels per line is radically smaller than the actual line length, the core drops most of the pixels waiting for the (s_axis_video_tlast) End-of-line signal. Check the ERROR register.
Generated SOF signal (m_axis_video_tuser0) signal misplaced.	Check the ERROR register.
Generated EOL signal (m_axis_video_tlast) signal misplaced.	Check the ERROR register.
Data samples lost between Upstream core and this core. Inconsistent EOL and/or SOF periods received.	• Are the Master and Slave AXI4-Stream interfaces in the same clock domain? • Is proper clock-domain crossing logic instantiated between the upstream core and this core (Asynchronous FIFO)? • Did the design meet timing? • Is the frequency of the clock source driving the ACLK pin lower than the reported Fmax reached?
Data samples lost between Downstream core and this core. Inconsistent EOL and/or SOF periods received.	• Are the Master and Slave AXI4-Stream interfaces in the same clock domain? • Is proper clock-domain crossing logic instantiated between the upstream core and this core (Asynchronous FIFO)? • Did the design meet timing? • Is the frequency of the clock source driving the ACLK pin lower than the reported Fmax reached?

If the AXI4-Stream communication is healthy, but the data seems corrupted, the next step is to find the correct configuration for this core.

Other Interfaces

Table C-3 describes how to troubleshoot third-party interfaces.

Table C-3: **Troubleshooting Third-Party Interfaces**

Symptom	Solution
Severe color distortion or color-swap when interfacing to third-party video IP.	Verify that the color component logical addressing on the AXI4-Stream TDATA signal is in according to *Data Interface* in Chapter 2. If misaligned: In HDL, break up the TDATA vector to constituent components and manually connect the slave and master interface sides.
Severe color distortion or color-swap when processing video written to external memory using the AXI-VDMA core.	Unless the particular software driver was developed with the AXI4-Stream TDATA signal color component assignments described in *Data Interface* in Chapter 2 in mind, there are no guarantees that the software correctly identifies bits corresponding to color components. Verify that the color component logical addressing TDATA is in alignment with the data format expected by the software drivers reading/writing external memory. If misaligned: In HDL, break up the TDATA vector to constituent components, and manually connect the slave and master interface sides.

Application Software Development

This appendix contains details about programming the core.

Processing States

The core distinguishes acquisition and readout periods to avoid modification of single-buffered measurement data while it is being read out. After readout, block RAMs and registers have to be re-initialized before the next acquisition cycle may commence.

After reset or power-up, the core cycles through the "Initialization," "Wait for Start of Frame," and "Data Acquisition" states.

Figure D-1 shows the top-level state diagram of the Image Statistics core.

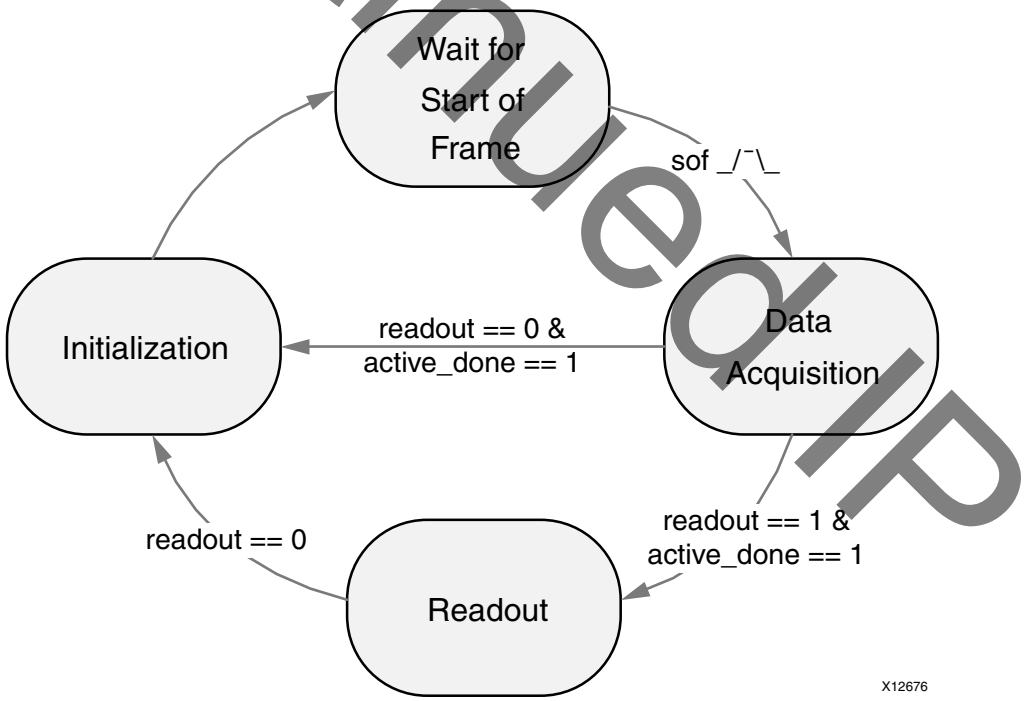


Figure D-1: **Image Statistics IP Core State Diagram**

Initialization

The one- and two-dimensional histograms are stored in block RAMs, which need to be cleared before the IP core can start data acquisition. Clearing of block RAMs may take 256, 1024 or 4096 CLK cycles corresponding to 8, 10, or 12-bit wide input data. Block RAM initialization may take several scan-lines, depending on the input resolution.

Once the core is finished with block RAM initialization, it progresses to the "Wait for Start of Frame" state where it remains until a rising edge on `sof` (Start of Frame) is detected, at which time it enters the "Data Acquisition" state.

Data Acquisition

In the "Data Acquisition" state, the core updates all internal measurement values with the pixel data presented on `video_data_in` when data is qualified with `active_video_in` = 1.

If the `READOUT` bit in the `CONTROL` register is set, the core proceeds to the "Readout" state (bit 6). Otherwise, the core proceeds to the Initialization state after the last active scan-line, which may occur several scan-lines before the rising edge on End of Frame (`eof`).

Readout

In the readout state, the core does not collect any more statistical information, and the multiplexing/addressing mechanism on the output is activated. Once the user provides addresses that are qualified valid by asserting the `addr_valid` pin, the core fetches and displays information on its output ports pertaining to the input addresses. Valid output data is identified by the `DATA_VALID` register (address offset 0x0190).

Reading Out Statistical Results

Figure D-3 shows the reg_update register being asserted to a '1' and then the HMAXn and VMAXn registers transitioning from their default values to the new values.

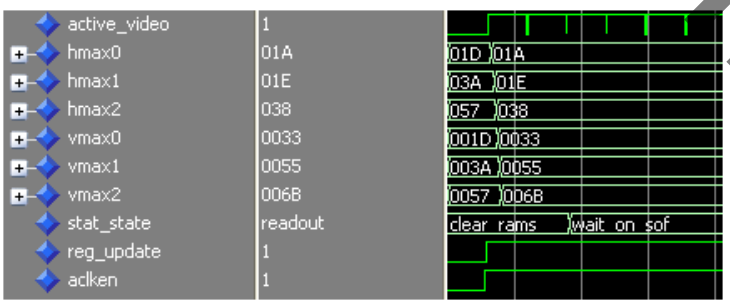


Figure D-2: **Register Update Example**

Figure D-3 shows an example of data readout timing and use of the control (specifically bits REG_UPADTE and READOUT) and STATUS (shown in hexadecimal format) registers. In this example, an empty frame followed by a test frame (the frames are bounded by the active_done signal) are processed by the core. The core cycles through the "Initialization" (clearing block RAMs) and "Wait on Start of Frame" stages, from which it transitions to the "Data Acquisition" state on the second falling edge of SOF.

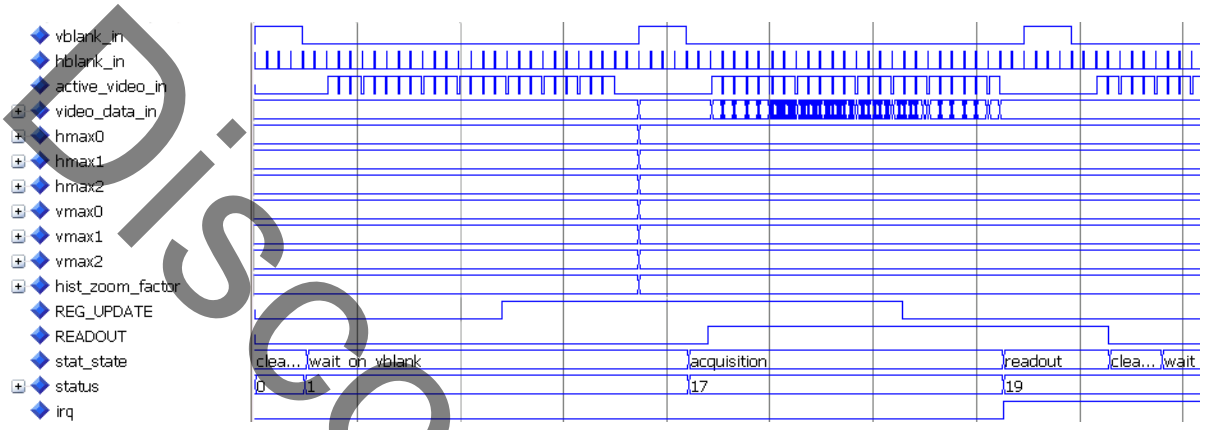


Figure D-3: **Frame and Readout Timing**

As discussed in the Synchronization, page 74 section, the Image Statistics core signals the end of data acquisition by asserting the DONE flag of the STATUS register, which can also trigger an interrupt if the IRQ_ENABLE register is set up to enable interrupts.

During the frame, bit 2 (READOUT) was asserted, which at the end of the active portion of the frame instructs the core to enter the "Readout" state.

Bit READOUT of the CONTROL register has to be set before the end of the frame; otherwise the core does not enter the readout mode but clears measurement data and arms itself for capturing the next frame.

After the host processor is finished reading out relevant statistical data, it programs bit 6 (READOUT) of the CONTROL register to 0, which instructs the core to re-initialize by entering the "clear-RAMs" state. The example in Figure D-3 also demonstrates the use of the REG_UPDATE flag. The user at any point could have modified values for input registers, such as HMAX0, HMAX1, HMAX2, VMAX0, VMAX1, VMAX2, RGB_HIST_ZONE_EN, YCC_HIST_ZONE_EN, or HIST_ZOOM_FACTOR. After setting all input registers to their desired value, REG_UPDATE was asserted, which resulted to all internal registers to latch in the user input at the rising edge of End of Frame. Registers HMAXn, VMAXn, RGB_HIST_ZONE_EN, YCC_HIST_ZONE_EN and HIST_ZOOM_FACTOR values displayed in Figure D-3 demonstrate how the values change simultaneously on the rising edge of End of Frame if REG_UPDATE is set to 1.

Synchronization

A semaphore-based mechanism is used to synchronize external frame timing with host processor register writes, data readouts and data acquisition.

The semaphores involved in synchronizing host processor activity with the incoming video stream are:

- DONE flag (bit 4 of the STATUS register)
- REG_UPDATE (bit 1 of the CONTROL register)
- READOUT (bit 6 of the CONTROL register)
- CLR_STATUS (bit 2of the CONTROL register)

The software flow diagram for normal system operation is shown in Figure D-4.

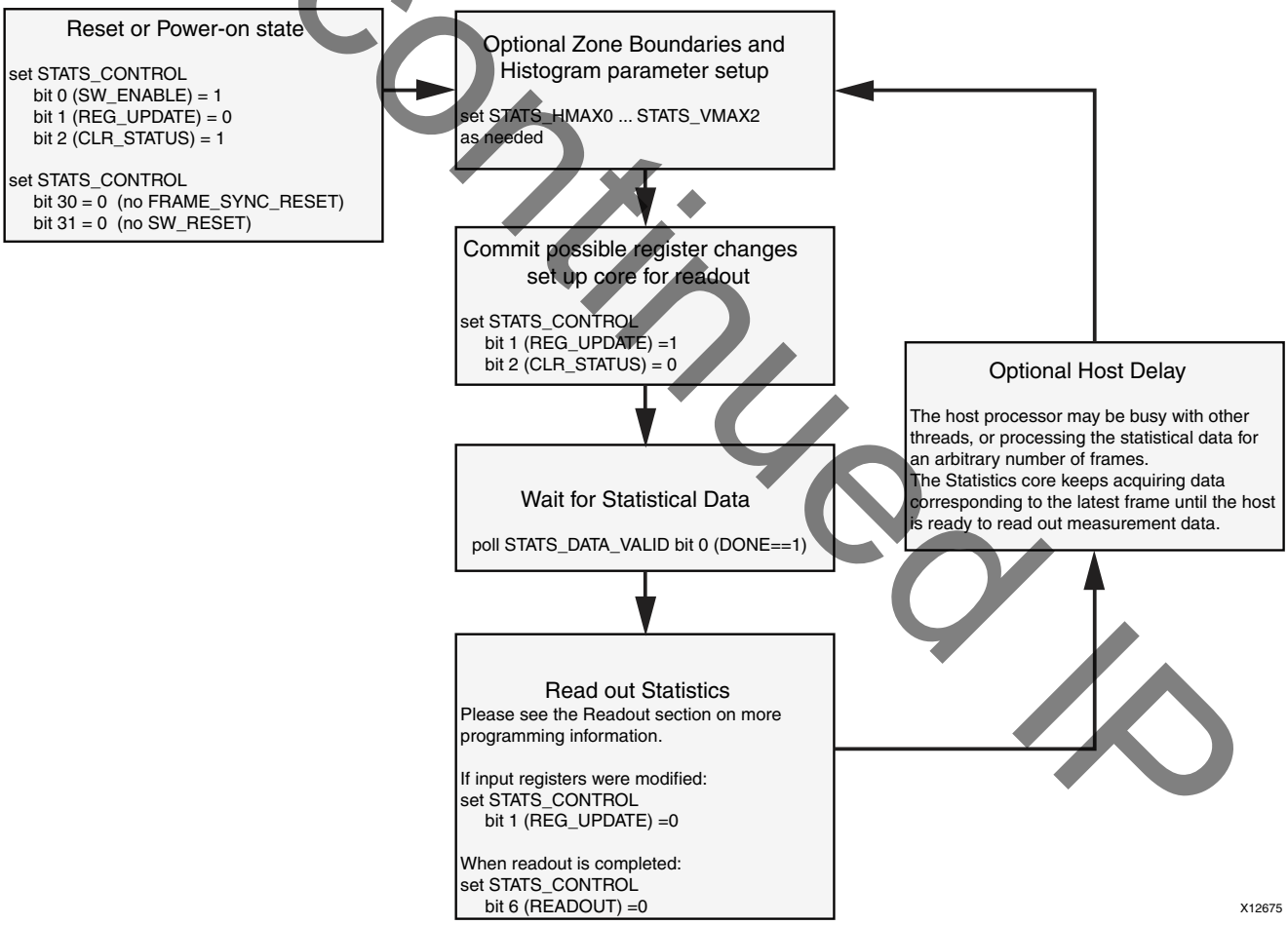


Figure D-4: **Software Flow Diagram for Normal System Operation**

When data acquisition is finished, the core asserts status flag DONE.

After the data acquisition state, depending on the state of the READOUT flag, the core either enters the readout state (READOUT = 1), or discards measurement data by re-initializing block RAMs and registers and preparing for acquiring data from the next frame (READOUT = 0).

This mechanism relieves the host processor from having to service the core when statistical data is not needed and allows the core to continuously process frames, so when the host processor is ready to poll statistical information, information from the last frame is available.

If the core enters the readout state, it remains there until the host processor signals being done with reading out measurement data, which may take a few lines, or several frames depending on the speed and the workload of the host processor. Therefore termination of the readout state is decoupled from the input video stream.

Once the host processor deasserts READOUT, the core immediately clears (re-initializes) block RAMs and registers and proceeds to acquire data from the next frame.

After deasserting READOUT, the host processor may assert it again immediately, enabling the core to enter the "Readout" state after acquisition of the current frame is complete.

IMPORTANT: *READOUT has to be asserted before the statistics core is done acquiring the next frame, or the core discards the data and self-triggers to acquire the next frame.*

IMPORTANT: *For the core to process every subsequent frame, the vertical blanking period has to be at least as long as the number of scan-lines it takes to initialize the block RAMs. For example, in the pessimistic case of using SD sensor (720 pixels per line) with 12 bit data (4k deep block RAMs), the minimum vertical blanking period has to be 4096/720 = 5.68, or at least six lines.*

Programmer Guide

The software API is provided to allow easy access to the Image Statistics AXI4-Lite registers defined in Table 2-10. To use the API functions, the following two header files must be included in the user C code:

```
#include "stats.h"
#include "xparameters.h"
```

The hardware settings of the system, including the base address of your Image Statistics core, are defined in the xparameters.h file. The stats.h file contains the macro function definitions for controlling the Image Statistics core.

For examples on API function calls and integration into a user application, the driver's subdirectory contains a file, example.c, in the stats_v6_0/example subfolder. This file is a sample C program that demonstrates how to use the Image Statistics API.

Table D-1: **Image Statistics Driver Function Definitions**

Function name and parameterization	Description
STATS_Enable(uint32 BaseAddress)	Enables the core instance.
STATS_Disable(uint32 BaseAddress)	Disables the core instance.
STATS_Reset(uint32 BaseAddress)	Immediately resets the core. The core stays in reset until the RESET flag is cleared.
STATS_ClearReset(uint32 BaseAddress)	Clears the reset flag of the core, which allows it to re-sync with the input video stream and return to normal operation.
STATS_FSyncReset(uint32 BaseAddress)	Resets the core instance at the end of the current frame being processed, or immediately if the core is not currently processing a frame.
STATS_ReadReg(uint32 BaseAddress, uint32 RegOffset)	Returns the 32-bit unsigned integer value of the register. Read the register selected by RegOffset (defined in Table 2-10).
STATS_WriteReg(uint32 BaseAddress, uint32 RegOffset, uint32 Data)	Write the register selected by RegOffset (defined in Table 2-10). Data is the 32-bit value to write to the register.
STATS_RegUpdateEnable(uint32 BaseAddress)	Enables copying double buffered registers at the beginning of the next frame. Please see section “Double Buffering” below.
STATS_RegUpdateDisable(uint32 BaseAddress)	Disables copying double buffered registers at the beginning of the next frame. Please see section “Double Buffering” below.
STATS_RegUpdateDisable(uint32 BaseAddress)	Clears the status register of the core by first asserting then deasserting the CLEAR_STATUS flag of STATS_CONTROL. This function only works when the core is enabled.

Software Reset

Software reset reinitializes registers of the AXI4-Lite control interface to their initial value, resets FIFOs, forces `m_axis_video_tvalid` and `s_axis_video_tready` to 0. `STATS_Reset`() and `STATS_FSync_Reset` () reset the core immediately if the core is not currently processing a frame. If the core is currently processing a frame calling `STATS_Reset`(), or setting bit 30 of the `CONTROL` register to 1 will cause statistical data to be lost for that frame.

After calling `STATS_Reset`(), the core remains in reset until `STATS_ClearReset`() is called. Calling `STATS_FSync_Reset`() automates this reset process by waiting until the core finishes processing the current frame, then asserting the reset signal internally, keeping the core in reset only for 32 `ACLK` cycles, then deasserting the signal automatically. After calling `STATS_FSync_Reset`(), it is not necessary to call `STATS_ClearReset`() for the core to return to normal operating mode.

IMPORTANT: *Calling STATS_FSync_Reset() does not guarantee prompt, or real-time resetting of the core.*

If the AXI4-Stream communication is halted mid frame, the core will not reset until the upstream core finishes sending the current frame or starts a new frame.

Double Buffering

Image Statistics core and ACTIVE_SIZE registers are double-buffered. Values from the AXI4-Lite interface are latched into processor registers immediately after writing, and processor register values are copied into the active register set at the Start Of Frame (SOF) signal. Double-buffering decouples AXI4-Lite register updates from the AXI4-Stream processing, allowing software a large window of opportunity to update processing parameter values.

If multiple register values are changed during frame processing, simple double buffering would not guarantee that all register updates would take effect at the beginning of the same frame. Using a semaphore mechanism, the RegUpdateEnable() and RegUpdateDisable() functions allows synchronous commitment of register changes.

The Image Statistics core will start using the updated ACTIVE_SIZE and core register values only if the REGUPDATE flag of the CONTROL register is set (1), after the next Start-Of-Frame signal (s_axis_video_tuser) is received. Therefore, it is recommended to disable the register update before writing multiple double-buffered registers, then enable register update when register writes are completed.

Reading and Writing Registers

Each software register that is defined in Table 2-10 has a constant that is defined in stats.h which is set to the offset for that register listed in Table D-2.

RECOMMENDED: *The application software uses the predefined register names instead of register values when accessing core registers.*

This ensures that future updates to the Image Statistics drivers that change register locations will not affect the application dependent on the driver.

Table D-2: **Predefined Constants Defined in stats.h**

Constant Name Definition	Value	Target Register
STATS_CONTROL	0x0000	CONTROL
STATS_STATUS	0x0004	STATUS
STATS_ERROR	0x0008	ERROR
STATS_IRQ_ENABLE	0x000C	IRQ_ENABLE
STATS_VERSION	0x0010	VERSION
STATS_SYSDEBUG0	0x0014	SYSDEBUG0
STATS_SYSDEBUG1	0x0018	SYSDEBUG1
STATS_SYSDEBUG2	0x001C	SYSDEBUG2
STATS_ACTIVE_SIZE	0x0020	ACTIVE_SIZE
STATS_HAMX0	0x0100	HMAX0
STATS_HAMX1	0x0104	HMAX1
STATS_HAMX2	0x0108	HMAX2
STATS_VAMX0	0x010C	VMAX0
STATS_VAMX1	0x0110	VMAX1
STATS_VAMX2	0x0114	VMAX2
STATS_ HIST_ZOOM_FACTOR	0x0118	HIST_ZOOM_FACTOR
STATS_ RGB_HIST_ZONE_EN	0x011C	RGB_HIST_ZONE_EN
STATS_ YCC_HIST_ZONE_EN	0x0120	YCC_HIST_ZONE_EN
STATS_ ZONE_ADDR	0x0124	ZONE_ADDR
STATS_COLOR_ADDR	0x0128	COLOR_ADDR
STATS_ HIST_ADDR	0x012C	HIST_ADDR
STATS_ ADDR_VALID	0x0130	ADDR_VALID
STATS_MAX	0x0134	MAX
STATS_MIN	0x0138	MIN
STATS_SUM_LO	0x013C	SUM_LO
STATS_ SUM_HI	0x0140	SUM_HI
STATS_POW_LO	0x0144	POW_LO
STATS_POW_HI	0x0148	POW_HI
STATS_ HSOBEL_LO	0x014C	HSOBEL_LO
STATS_ HSOBEL_HI	0x0150	HSOBEL_HI
STATS_VHSOBEL_LO	0x0154	VSOBEL_LO
STATS_ VSOBEL_HI	0x0158	VSOBEL_HI
STATS_ LSOBEL_LO	0x015C	LSOBEL_LO
STATS_ LSOBEL_HI	0x0160	LSOBEL_HI
STATS_ RSOBEL_LO	0x0164	RSOBEL_LO

Table D-2: **Predefined Constants Defined in stats.h *(Cont'd)***

Constant Name Definition	Value	Target Register
STATS_ RSOBEL_HI	0x0168	RSOBEL_HI
STATS_ HIFREQ_LO	0x016C	HIFREQ_LO
STATS_ HIFREQ_HI	0x0170	HIFREQ_HI
STATS_ LOFREQ_LO	0x0174	LOFREQ_LO
STATS_ LOFREQ_HI	0x0178	LOFREQ_HI
STATS_RHIST	0x017C	RHIST
STATS_GHIST	0x0180	GHIST
STATS_BHIST	0x0184	BHIST
STATS_YHIST	0x0188	YHIST
STATS_CCHIST	0x018C	CCHIST
STATS_DATA_VALID	0x0190	DATA_VALID

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

http://www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf.

For a comprehensive listing of Video and Imaging application notes, white papers, reference designs and related IP cores, see the Video and Imaging Resources page at:

http://www.xilinx.com/esp/video/refdes_listing.htm#ref_des.

References

1. AMBA AXI Protocol Version: 2.0 Specification
2. Vicent Caselles, Jose-Luis Lisani, Jean-Michel Morel, Guillermo Sapiro: *Shape Preserving Local Histogram Modification*
3. Joung-Youn Kim, Lee-Sup Kim, and Seung-Ho Hwang: *An Advanced Contrast Enhancement Using Partially Overlapped Sub-Block Histogram Equalization*
4. Simone Bianco, Francesca Gasparini and Raimondo Schettini: *Combining Strategies for White Balance*
5. G. Finlayson, M. Drew, and B. Funt, "Diagonal Transform Suffice for Color Constancy" in *Proc. IEEE International Conference on Computer Vision*, Berlin, pp. 164-171, 1993
6. Keith Jack: Video Demystified, 4th Edition, *ISBN 0-7506-7822-4*, pp 15-19
7. *AXI Reference Guide* (UG761)
8. *ISE to Vivado Design Suite Migration Guide* (UG911)
9. *Vivado Design Suite User Guide: Designing with IP* (UG896)

10. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
11. *Vivado Design Suite User Guide: Getting Started* (UG910)
12. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
13. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/19/2011	1.0	Initial Xilinx release.
04/24/2012	2.0	Updated core to v4.00a and ISE Design Suite to v14.1.
07/25/2012	3.0	Updated for core version. Added Vivado information.
10/16/2012	3.1	Updated for core version. Updated for ISE v14.3 and Vivado v2012.3 tools. Added Vivado test bench.
03/20/2012	3.2	Updated for core version. Removed ISE chapters.
10/02/2013	6.0	Synch document version with core version. Updated Constraints, Test Bench, and Migration chapters.
12/18/2013	6.0	Added UltraScale Architecture support.

Notice of Disclaimer