MACNICA

Intellectual Property

Chroma Histogram

For Vectorscopes

Version 1.1

January 2015

Corporate HQ & Design Center
380 Stevens Ave. Suite 206
Solana Beach, CA 92075
http://www.macnica-na.com

License and Terms of Use

Document Revision History

Revision	Date	Comments
0.01	July, 2014	Initial Draft
1.0	09/09/2014	Added Fmax performance
1.1	January, 2015	Changed name to Vectorscope, added content examples and diagrams. Corrected clearing of frame complete: writing a 0 to clear_frame_completed clears the bit, not a '1' as was stated. Update utilization to reflect Y table addition

1 Contents

2 Figures

3 References

3.1 Industry Standards and Specifications

Citations

3.2 Related Documents

Video and Image Processing User Guide, Altera Corp.

www.altera.com/literature/ug/ug_vip.pdf

Avalon Interface Specifications, Altera Corp.

www.altera.com/literature/manual/mnl_avalon_spec.pdf

4 Overview

The Vectorscope is a commonly used tool in video equipment that graphs the color content of an image or frame. The layout is similar to that of a color wheel. In the figure below you can see that various vectors extending from the center extend towards deeper levels of saturation of the various colors. The vectors show are for R, G and B and also Cyan, Magenta and Yellow.

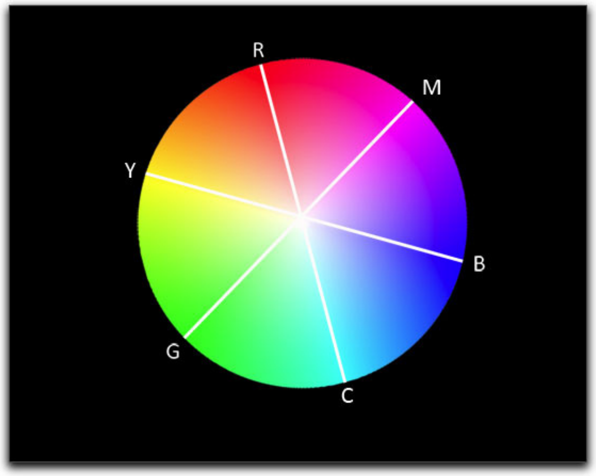


Figure 1 - Color Wheel

The Vectorscope is basically a graph of the CrCb values of the pixels in the image, with Cr being the vertical axis and Cb being the horizontal axis. Points are plotted in this X/Y coordinate system to represent the present of a particular Chroma value in the frame. In order to render such a graph, we need to collect statistics of the color content.

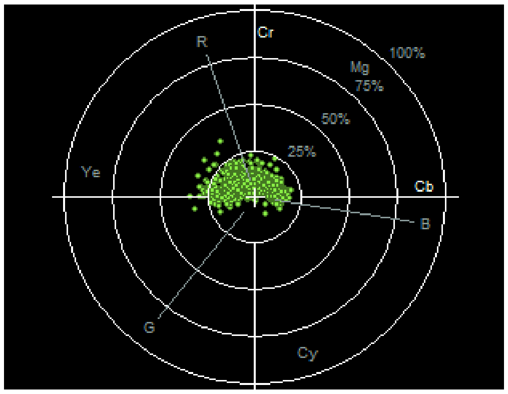


Figure 2- Vectorscope

This is the function of the Chroma Histogram. When enabled, it will observe the video stream and maintain a count of every Cr/Cb value combination to enable subsequent rendering or analysis by the host processor.

5 Functional Description

5.1 Implementation

The data is collected by counting the number of pixels in an image for each Cr-Cb value combination. If the Cr and Cb are represented by 8-bit values then we would have 2**16 or 64K "bins" that would each hold a count that can range up to the number of pixels in the entire frame. Essentially, we'd have 64K "counters". Such a task is clearly more well suited to hardware implementations, thus some considerations are made in the implementation.

5.1.1 Table-based

Instead of 64K counters – one for each Cr / Cb combination, a ram is used and read-modify-write operations are required to "increment" each counter or bin value. To additionally minimize the memory utilization, the table is reduced to 16K and only the most significant 7 bits of the Cr and Cb value are used to select the bin.

5.1.2 Subsampling and Averaging

Since it takes several clock cycles to perform such an operation, only every 4th pixel is evaluated. The incoming frame is subsampled by 1/4th horizontally. Vertical subsampling is also performed – which reduces the maximum number of counts that can occur in any bin.

During horizontal sampling, four pixel values are averaged and then the seven most significant bits of the result from Cr and Cb are used to index the table.

5.1.3 Avalon Sink

Also to minimize logic utilization, the Chroma Histogram is not a "flow through" device for the Avalon-ST data stream. It is an Avalon "Sink" only and sits downstream of a Splitter. As an Avalon-ST slave, it is always ready to accept data whether enabled or not.

5.1.4 Luminance Histogram

Additionally, since the incoming stream is most conveniently in the YCrCb color space, an additional histogram is generated for the luminance or Y value. It is a simple table that generates a count of all possible Y values – so 256 "bins" of the 4:1 subsampled input image.

5.2 Operation

The Chroma Histogram is always ready to accept an input stream, even when not enabled.

When the "Go" bit is set (register 0, bit 1) the capture is enabled and the following events occur:

1. The ram table must be cleared, by the hardware, by writing to every location. Each write operation takes two clocks, so a total of 64K clocks are required. Since this module is running at the video rate the clearing takes approximately 32 line times for a frame width of 2K pixels
2. An internal "capture enabled" goes true, and the logic awaits the beginning of a video frame
3. The beginning of a frame is received
4. The histogram table is enabled and captures data for one frame

5. The Capture Completed bit is set (register 1, bit 0). If the interrupt is enabled then an interrupt is generated to the processor and the Interrupt Pend bit is set (register 1, bit 1).

At this point the host will read the table and can enable another capture if desired. See the section Programming for an example

5.2.1 Chroma Table Format

The chroma histogram table is 32K entries by 20-bits. The table address is formulated by the concatenation of the averaged Cr value and the averaged Cb value.

Cr/Cb (avg)		**Table Address**	**Contents**
Cr	Cb		Counter Value
7-bits	7-bits	14 bits	20-bits
0	0	0x0000	# of pixel averages with Cr=0 and Cb=0
	1	0x0001	# of pixels for CrCb = {0,1}
	…		# of pixels for CrCb = {0,2}
	0x7F	0x007F	# of pixels for CrCb = {0,3}
1	0	0x0080	# of pixels for CrCb = {1,0}
	1	0x0081	
	2	0x0082	
	…		
	0x7F		
…	…	…	
0x7F	0x7F	0x3FFF	

Figure 3 – Table Organization

5.2.2 Luminance Table Format

The Luminance histogram is a simple 256 x 20-bits table. Each entry holds a count of the averaged and sub-sampled Y values for a Y value equal to the respective address.

5.3 Symbol

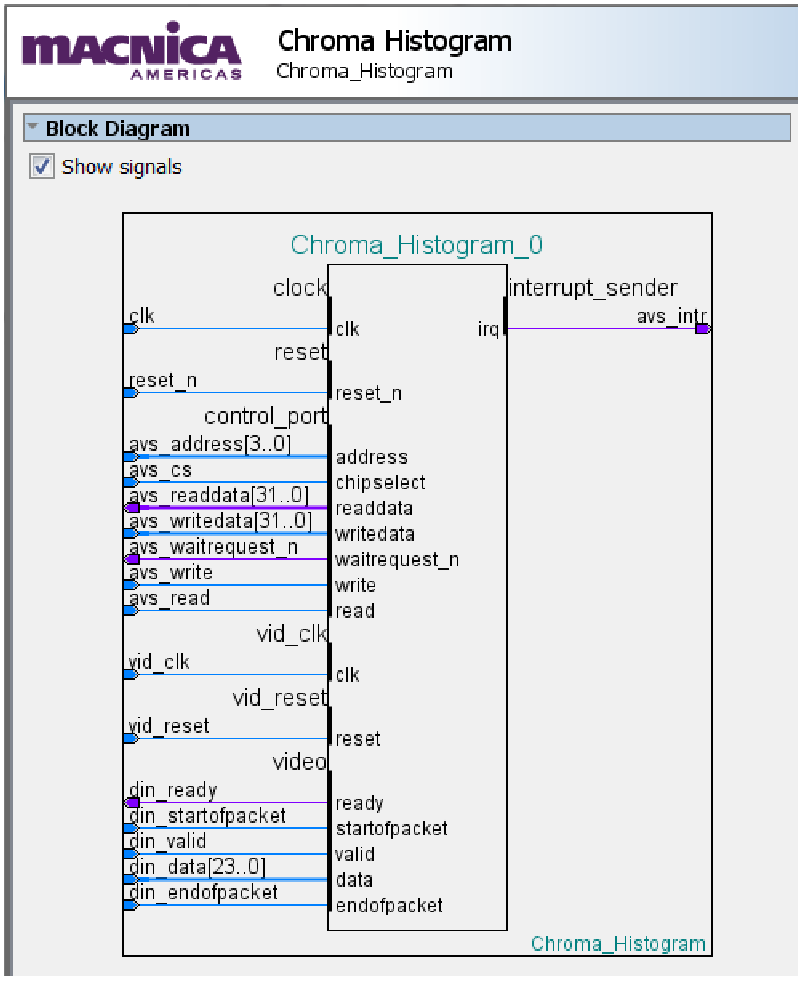


Figure 4 – Chroma Histogram top-level interfaces

5.4 Interface Description

5.4.1 Control

Interface	Description	Notes
Avalon Slave	A standard Avalon Slave interface for controlling the various settings and reading the histogram table. This interface is synchronous to the clock domain "clk" with reset signal "reset_n".	

5.4.2 Video

Interface	Description	Notes
Avalon Streaming (sink)	A standard Avalon-ST Video interface, 8 bits per color, three colors in parallel. This interface is synchronous to the "vid_clk" domain and uses reset signal "vid_reset".	8bpp, three symbols in parallel only

5.5 Registers

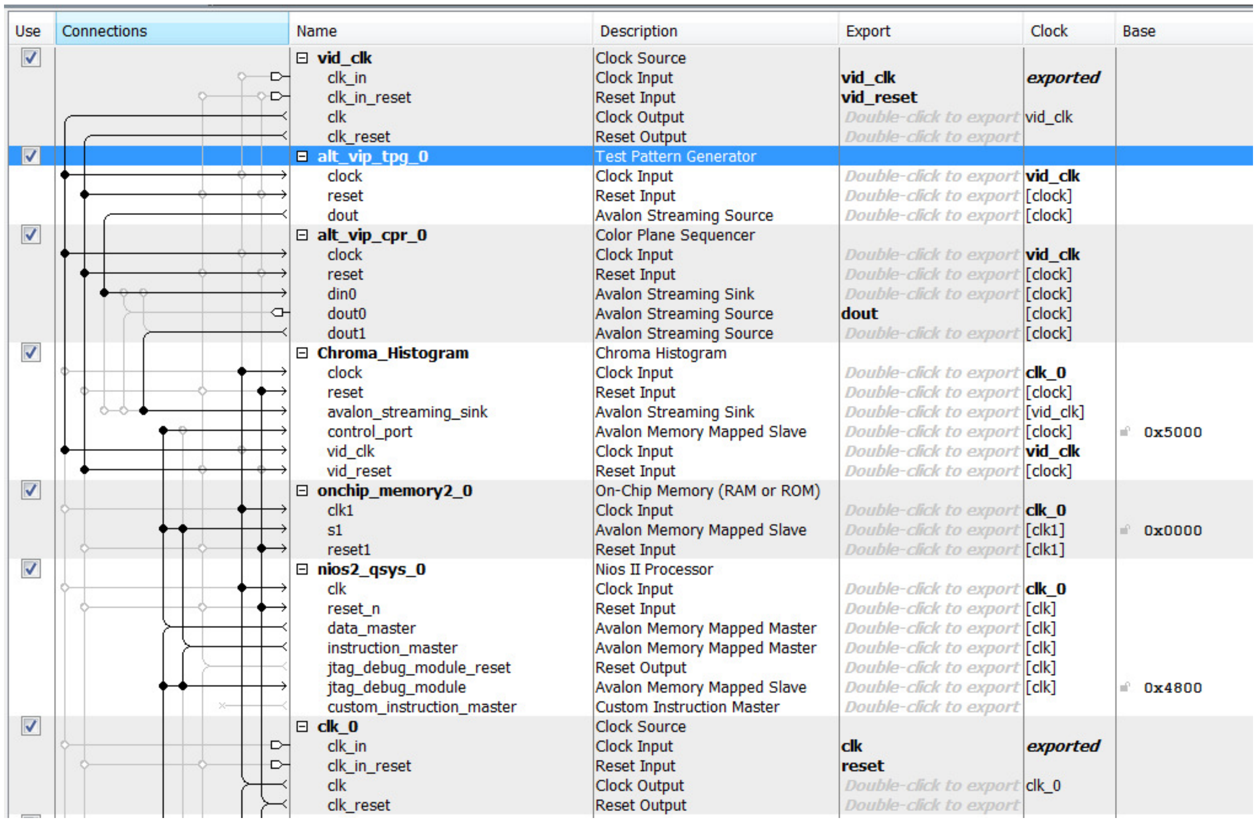
Register	Description	Notes
Control Register (address 0)	This is the standard VIP control register. As usual, bit '0' is the "Go" bit that enables the module. Additional bits are: [0] – The "go" enable bit [1] – interrupt enable	
Status (address 1)	[0] = '1' = frame completed. Writing a '0' to this bit will clear the bit and enable another capture if the "go" bit is still set. [1] = interrupt pending. Writing a '0' to this bit will clear the interrupt AND the frame completed status in bit 0; [2] = '1' => re-zero the host address to the tables. This is necessary after reading one table – e.g. the Y table – before starting to read out the CrCB table contents. *Usually then a 0x07 is written to zero the host address, since we don't want to write 0's just yet to the other bits which would clear then and enable another capture while we are about to read the table.*	
Chroma Histogram Data (address 2)	A write to this register will clear the histogram table contents for both the CrCb table and the Y histogram table. Reads from this address will return the data in the chroma histogram ram, starting from address zero. The address is auto-incrementing, and cleared internally when a frame capture is completed, or when a write to the Status Register is performed with bit [2] = 1; The table data is 20-bits and is stored in the LSB of the returned 32-bit value.	

Y Histogram Data (address 3)	Reads from this address will return the data in the Y histogram ram, starting from address zero. The address is auto-incrementing, and cleared internally when a frame capture is completed, or when a write to the Status Register is performed with bit [2] = 1; The table data is 20-bits and is stored in the LSB of the returned 32-bit value.	

5.6 Application Example

The Chroma Histogram is not a pass-through implementation and has no Avalon Streaming video Source output. Instead, it relies on a Color Plane Sequencer to split off the Chroma (Cr, Cb).

A sample verification testbench implementation is shown below.



Use	Connections	Name	Description	Export	Clock	Base
✓		⊟ **vid_clk**	Clock Source			
		clk_in	Clock Input	**vid_clk**	*exported*	
		clk_in_reset	Reset Input	**vid_reset**		
		clk	Clock Output	Double-click to export	vid_clk	
		clk_reset	Reset Output	Double-click to export		
✓		⊟ **alt_vip_tpg_0**	Test Pattern Generator			
		clock	Clock Input	Double-click to export	**vid_clk**	
		reset	Reset Input	Double-click to export	[clock]	
		dout	Avalon Streaming Source	Double-click to export	[clock]	
✓		⊟ **alt_vip_cpr_0**	Color Plane Sequencer			
		clock	Clock Input	Double-click to export	**vid_clk**	
		reset	Reset Input	Double-click to export	[clock]	
		din0	Avalon Streaming Sink	Double-click to export	[clock]	
		dout0	Avalon Streaming Source	**dout**	[clock]	
		dout1	Avalon Streaming Source	Double-click to export	[clock]	
✓		⊟ **Chroma_Histogram**	Chroma Histogram			
		clock	Clock Input	Double-click to export	**clk_0**	
		reset	Reset Input	Double-click to export	[clock]	
		avalon_streaming_sink	Avalon Streaming Sink	Double-click to export	[vid_clk]	
		control_port	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x5000
		vid_clk	Clock Input	Double-click to export	**vid_clk**	
		vid_reset	Reset Input	Double-click to export	[clock]	
✓		⊟ **onchip_memory2_0**	On-Chip Memory (RAM or ROM)			
		clk1	Clock Input	Double-click to export	**clk_0**	
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000
		reset1	Reset Input	Double-click to export	[clk1]	
✓		⊟ **nios2_qsys_0**	Nios II Processor			
		clk	Clock Input	Double-click to export	**clk_0**	
		reset_n	Reset Input	Double-click to export	[clk]	
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
		jtag_debug_module_reset	Reset Output	Double-click to export	[clk]	
		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x4800
		custom_instruction_master	Custom Instruction Master	Double-click to export		
✓		⊟ **clk_0**	Clock Source			
		clk_in	Clock Input	**clk**	*exported*	
		clk_in_reset	Reset Input	**reset**		
		clk	Clock Output	Double-click to export	clk_0	
		clk_reset	Reset Output	Double-click to export		

NOTE: Note that the Chroma Histogram has been renamed from the default Chroma_Histogram_0 convention. Due to an anomaly with QSYS, the source files are not properly copied to the simulation or synthesis sub-directories unless the _0 suffix is removed. Rename your instance as shown in the example above by right-clicking on the name and selected "rename" from the menu.

5.7 Programming

The programming flow is fairly simple:

1) Enable the Histogram by setting the “go” bit, and optionally the Interrupt Enable bit, in the Control register. The next single valid frame that is received will be analyzed.
2) The host then either waits for the completion interrupt, or checks the “frame complete” bit in the status register.
3) The host clears the interrupt bit (if used)
4) The host reads the contents of the table.
5) The host clears the table address by setting bit 2 of register 2
6) The host can now read the Y table.
7) The host initiates the clearing of the table by writing to register (2) with any value.
8) The host writes a ‘0’ to bit zero of the status register to enable another capture, or alternatively turns off the “go” bit to disable the function
9) The clearing of the histogram ram requires two clocks per location, so with this 16K-deep table approximately 33,000 clocks are required. This amounts to about 16 lines of incoming HD video, so it is very feasible to analyze every-other video frame. The frame complete and “go” bits are not evaluated until after the histogram table is cleared, so there is obviously ample time for the host CPU to set these bits as desired.

A very simple polled implementation is shown below:

```
#include "sys/alt_stdio.h"
#include <io.h>
#include "../chroma_test2_bsp/system.h"
int main()
{
  alt_putstr("Hello from Nios II!\n");
int frame_done,i,temp,table;
  /* Event loop never exits. */
  while (1){
        // enable histogram
        IOWR(CHROMA_HISTOGRAM_BASE,0,3);  // set GO bit and INTR_ENA
        frame_done = 0;
        while(!frame_done){
              frame_done=IORD(CHROMA_HISTOGRAM_BASE,1);
        }

        // See if the intr_pend bit is set
        if (IORD(CHROMA_HISTOGRAM_BASE,1) != 0x3)
              alt_putstr("Error reading Interrupt Status\n");
        else
              alt_putstr("Interrupt Status correct");

        //Clear the interrupt bit but not the frame completed bit
        // (which would enable another capture)
        IOWR(CHROMA_HISTOGRAM_BASE,1,0b001);  //
        // See if the intr_pend bit is cleared
        if (IORD(CHROMA_HISTOGRAM_BASE,1) != 0x1)
```

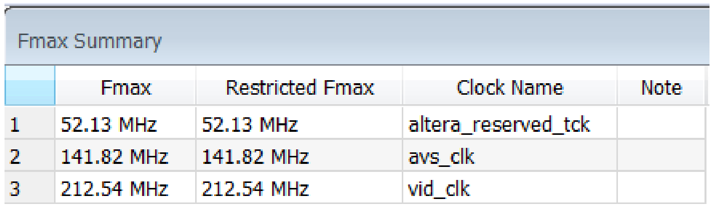
```
                alt_putstr("Error clearing Interrupt Status\n");
          else
                alt_putstr("Interrupt Status cleared correctly");




// Read some data from the CrCb table

        for(i=0;i<32;i++)
          temp = IORD(CHROMA_HISTOGRAM_BASE,2);


// read some data from the Y histogram.
// first have to reset the pointer!
// write bit 2 to address 1 to re-zero the host address.
// frame completed is cleared if bit 0 is a 0, so leave it as 1
        IOWR(CHROMA_HISTOGRAM_BASE,1,7);

        for(i=0;i<32;i++)
          temp = IORD(CHROMA_HISTOGRAM_BASE,3);


// clear the tables now
          IOWR(CHROMA_HISTOGRAM_BASE,2,1); // NOW we set bit 1, initiates table clear
        //NOW clear the frame completed bit.
        // The init_busy will prevent a new capture until clearing is done
          IOWR(CHROMA_HISTOGRAM_BASE,1,0);
  }

  return 0;
}
```

5.8 Specifications

5.8.1 Image Size

The implementation supports an image width of 2K pixels. There is no height restriction. Given that the table implements a 20-bit width for a maximum count of 2^20 counts, or a "mega"-count, a maximum of 20 Mpixels per frame could be supported without overflow.

5.8.2 Performance



Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	52.13 MHz	52.13 MHz	altera_reserved_tck	
2	141.82 MHz	141.82 MHz	avs_clk	
3	212.54 MHz	212.54 MHz	vid_clk	

5.8.3 Resource Utilization

Target Device Family	Variation	Memory	ALM
Cyclone V	standard YCrCb input	332,800 bits / 33 M10K	281