UG0944
User Guide
Histogram

Microsemi
a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

The first publication of this document.

2 Introduction

Histogram is traditionally used to show a set of data graphically. In the video and image processing domain, the histogram is used to visualize the distribution of pixel intensities in an image or a video frame. For an image with an 8-bit wide pixel, 256 intensity values are possible ranging from 0 to 255. The histogram will show how many times a pixel of particular intensity occurs in the entire image. Histograms are useful to understand the exposure and light distribution of the image and possibly correct it in image processing. One way of correcting the histogram of an image is by using intensity curves. The curve is adjusted through an interactive graphical interface such that the histogram has the pixel intensities spread over the entire range. The Histogram IP provides data that can be used to plot the histogram and can also correct the pixel intensity distribution from user inputs. The histogram IP takes video frames as input and generates a plot of intensities of all the pixels in one frame. The histogram IP computes the histogram data and also corrects the intensity of DATA_Y_I input. The Cb_I and Cr_I are passed through the IP with one cycle delay. The IP can be used to plot luma intensity or R, G, B intensity by using three instances of the IP. In case of IP is being used for R, G, B, the inputs Cb and Cr can be left unused. The histogram IP starts taking pixels with start of video frame and continues till end of the frame. During each pixel the logic inside the IP keeps track of previous pixels with same intensity and increments by one. The histogram IP can be used to see affects of changes in brightness, contrast, and color balance in image.

Histogram memory: FPGA LSRAM is used to store the pixel intensities. With every incoming pixel, the corresponding RAM address content is incremented by one. This process is followed till the end of the frame. This way the RAM addresses become pixel intensities and content becomes a number of times it occurs in one frame. Histogram memory is 24 bits wide and 256 addresses deep. The histogram is computed for 1 out of every 4 frames. For the remaining 3 frames, the histogram data is held constant, and it can be read by the user. The HIST_RDY_O output is held high for 3 frames, during which the user can read the histogram data on the CURVE_RDATA_O output signal using HIST_ADDR_I as the address for pixel intensity.

Curve memory: FPGA LSRAM is used to store correction factors for each intensity value. By default, it has linear numbers 0 to 255 initially. The curve memory has a RAM write interface from where these values can be reprogrammed at run time. Curve memory is 8 bits wide and 256 addresses deep.

2.1 Key Features

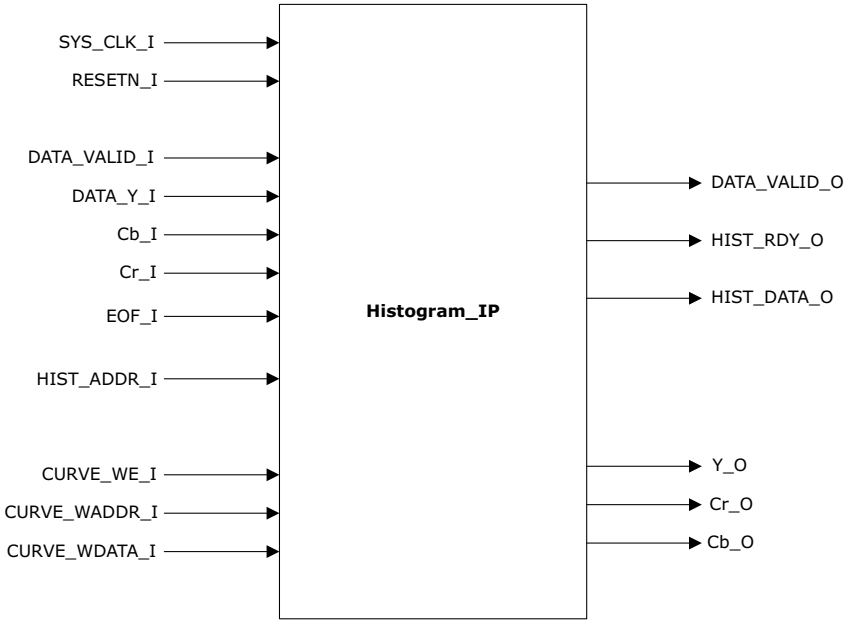
- Histogram plotting of real-time video
- Curve based correction of live video

2.2 Supported Families

- PolarFire® SoC
- PolarFire®
- RTG4™
- IGLOO®2
- SmartFusion®2

2.3 Inputs and Outputs

Figure 1 • **Histogram IP Block Diagram**



The following table lists the input and output ports of the Histogram IP.

Table 1 • **Input and Output Ports**

Port Name	Direction	Width	Description
SYS_CLK	Input	1 bit	System Clock. This must be the same as the pixel clock.
RESETN_I	Input	1 bit	Active low asynchronous reset signal to design
DATA_VALID_I	Input	1 bit	Input Data Valid Signal. This signal should be asserted when the data is valid.
DATA_Y_I	Input	8 bits	Input video data intensity
EOF_I	Input	1 bit	Input video frame end
HIST_ADDR_I	Input	8 bits	Histogram memory read address for reading histogram plot data
HIST_RDY_O	Output	1 bit	Ready indication when histogram memory can be read
HIST_DATA_O	Output	24 bits	Histogram memory data which is the number of pixels of intensity represented by the HIST_ADDR_I
CURVE_WE_I	Input	8 bits	Curve memory write enable
CURVE_WADDR_I	Input	8 bits	Curve memory write address
CURVE_WDATA_I	Input	8 bits	Curve memory write data
Cb_I	Input	8 bits	Chroma value for blue color
Cr_I	Input	8 bits	Chroma value for red color
Y_O	Output	8 bits	Output intensity after curve correction
Cb_O	Output	8 bits	Output for Cb_I with one clock cycle delay
Cr_O	Output	8 bits	Output for Cr_I with one clock cycle delay

2.4 Configuration Parameters

There are no configuration parameters for this IP.

2.5 License

Histogram IP clear RTL is license locked and the obfuscated RTL available for free.

2.5.1 Obfuscated

Complete RTL code is provided for the core, allowing the core to be instantiated with the SmartDesign tool. Simulation, synthesis, and layout can be performed within Libero® System-on-Chip (SoC). The RTL code for the core is obfuscated.

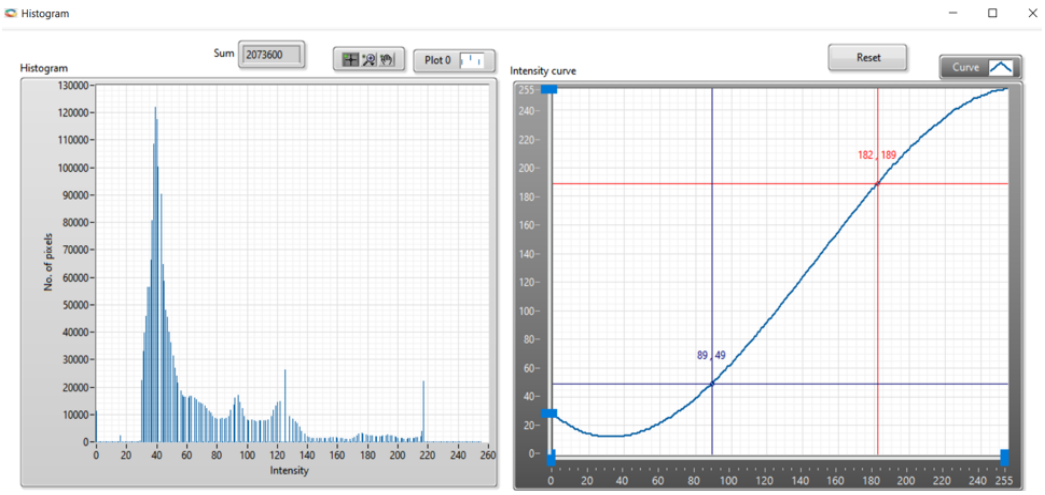
2.5.2 RTL

Complete RTL source code is provided for the core.

3 Histogram Plot

Using video control GUI, the following plot was observed for a 1920X1080 input image.

Figure 2 • **Histogram Plot**



4 Resource Utilization

Histogram IP is implemented on PolarFire FPGA (MPF300TS -1FCG1152I package). The following table shows the resource utilization report after synthesis.

Table 2 • **Resource Utilization**

Resource	Usage
DFFs	29
4LUTs	88
LSRAM18K	2
MACC	0

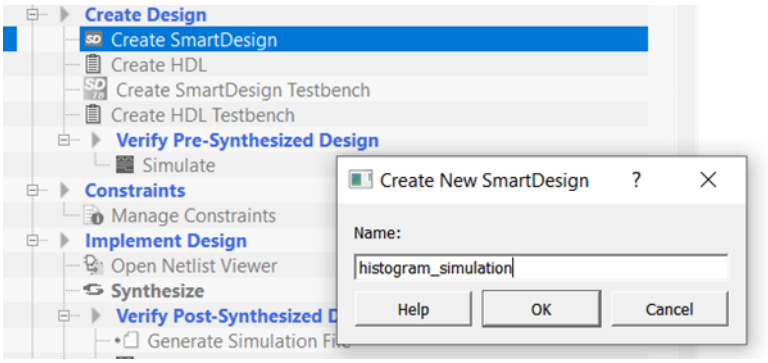
5 Testbench

A testbench signal generator hist_stimulus is provided to check the functionality of the histogram IP. This stimulus must be imported into the testbench to ensure histogram functionality.

The following steps describe how to simulate the core using the testbench.

1. In the **Design Flow** window, expand **Create Design**. Right-click **Create SmartDesign** and click **Run**, as shown in the following figure.

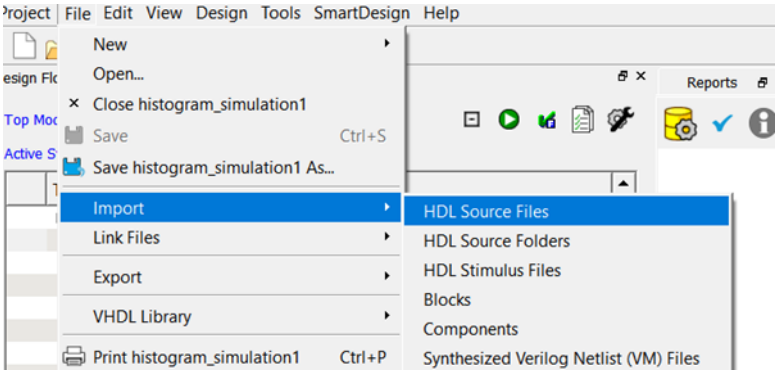
Figure 3 • **Create New SmartDesign**



SmartDesign is created, and a canvas appears to the right of the Design Flow pane.

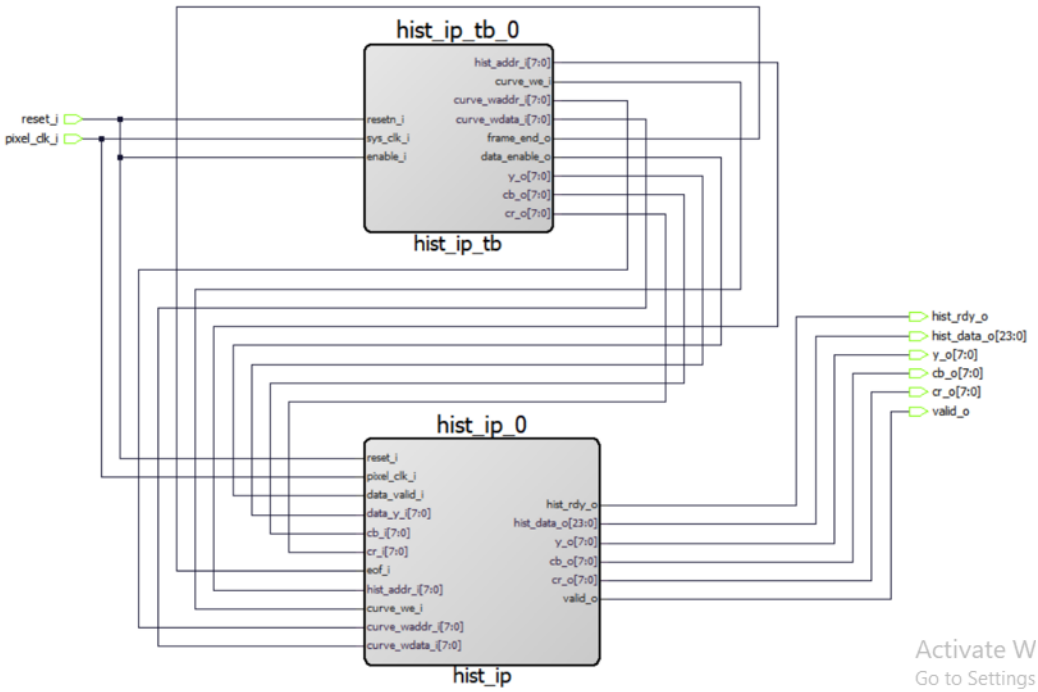
2. In the Libero tool menu go to **File -> Import -> HDL Source Files**. The `hist_ip.vhd`, `hist_stimulus.vhd`, and `ram2port.vhd` files have to be imported into the project.

Figure 4 • **Import HDL Source Files**



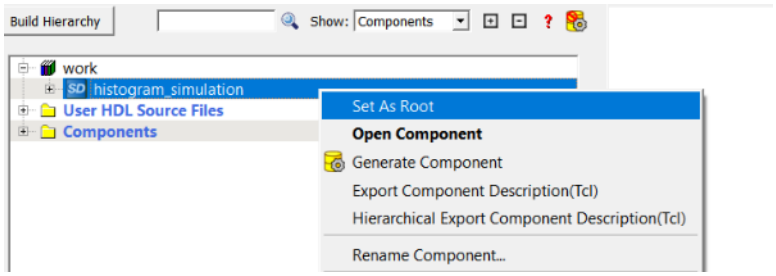
3. Click **Build Hierarchy** in the leftmost pane in the Libero.
4. Go to the Design Hierarchy tab and drag hit_ip and hist_stimulus in the SmartDesign area of the histogram_simulation.
5. Connect hist_ip and hist_stimulus blocks, as shown in the following figure.

Figure 5 • **Connect hist_ip and hist_stimulus blocks**



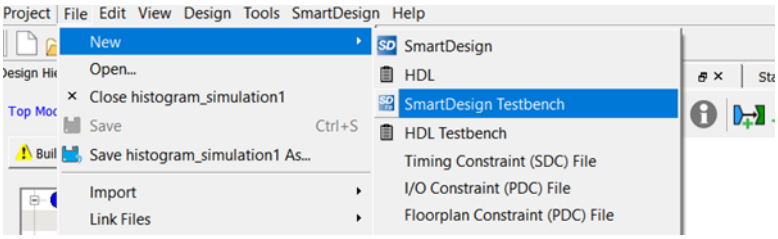
6. Click **Generate Component** and Build Hierarchy.
7. Go to design hierarchy, right click on the histogram_simulation, and select **Set As Root**.

Figure 6 • **Design Hierarchy**



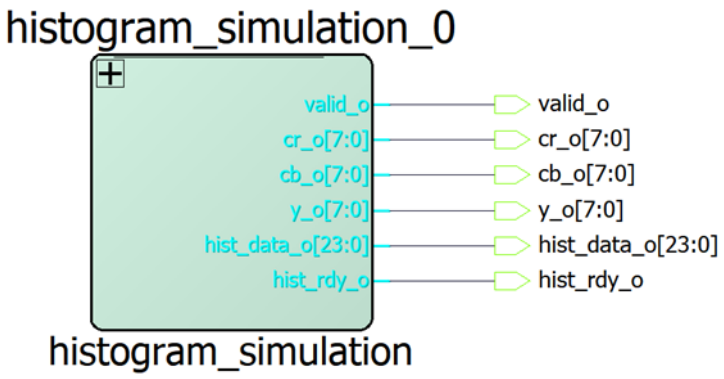
8. Create a testbench with the name hist_tb.

Figure 7 • **Create testbench**



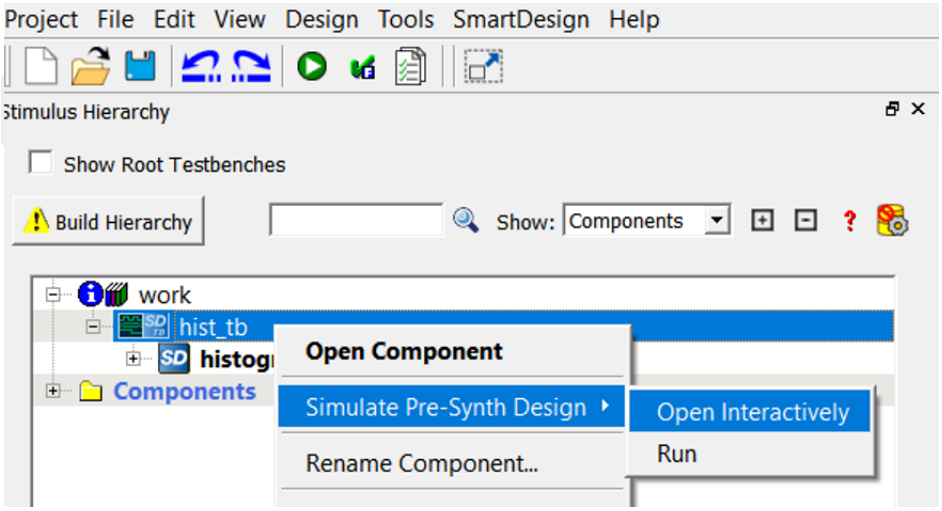
9. Save the project and click **Build Hierarchy**.
10. Promote the histogram_simulation ports to the top level.

Figure 8 • **Histogram Simulation**



11. Click **Generate Component**.
12. On the Stimulus Hierarchy tab, right-click the hist_tb testbench file and click **Open Interactively** from Simulate Pre-Synth Design.

Figure 9 • **Stimulus Hierarchy**



The ModelSim tool appears with the testbench file loaded onto it, as shown in the following figure.

If the simulation is interrupted because of the runtime limit in the DO file, use the run `-all` command to complete the simulation.

Figure 10 • **Simulation Result**