UG0643
User Guide
Image De-Noising Filter

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.

50200643. 3.0 9/20

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 3.0

The following is a summary of the changes in revision 3.0 of this document.

- Input Data_In_i is replaced with R_I, G_I and B_I to support RGB color format.
- Output Data_Out_o is replaced with R_O, G_O and B_O.
- Median Filter design logic is redesigned to support for (n x n) resolution with pipelined logics whereas previous design is implemented with Sequential FSM.

1.2 Revision 2.0

The following is a summary of the changes in revision 2.0 of this document.

- In Image De-Noising Filter Hardware Implementation, page 3:
  - *YCbCr* in signal names was replaced with *Data.*
  - The following text was deleted: The median filtering is only applied on the Y channel. The C_B and C_R signals are passed through the required pipe-lining registers to synchronize with Y channel. For the Y channel, three pixels from each of the three video lines are read into three shift-registers.
  - Details about the Image De-noising Filter testbench were added. For more information, see Testbench, page 5.
- The *Timing Diagrams* section and the appendix were deleted.
- The number of buffers in the hardware was updated from four to five. For more information, see Image De-Noising Filter Hardware Implementation, page 3.
- Information about port widths was added. For more information, see Inputs and Outputs, page 4.
- Resource utilization data was updated. For more information, see Resource Utilization, page 10.

1.3 Revision 1.0

The first publication of this document.

2 Introduction

Images captured from image sensors are affected by noise. Impulse noise is the most common type of noise, also called salt-and-pepper noise. It is caused by malfunctioning pixels in camera sensors, faulty memory locations in the hardware, or errors in data transmission.

Image denoising plays a vital role in digital image processing. Many schemes are available for removing noise from images. A good denoising scheme retrieves a clearer image even if the image is highly affected by noise.

Image denoising may either be linear or non-linear. A mean filter is an example of linear filtering, and a median filter is an example of non-linear filtering. While the linear model has traditionally been preferred for image denoising because of its speed, the limitation of this model is that it does not preserve the edges of the image. The non-linear model preserves the edges well compared to the linear model, but it is relatively slow.

Despite the slowness, non-linear filtering is a good alternative to linear filtering because it effectively suppresses impulse noise while preserving the edge information. The median filter ensures that each pixel in the image fits in with the pixels around it. It filters out samples that are not representative of their surroundings—the impulses. Therefore, it is very useful in filtering out missing or damaged pixels.

For 2D images, standard median operation is implemented by sliding a window over the image. The 3 × 3 window size, considered to be effective for the most commonly used images, is implemented in the IP. At each position of the window, the nine pixel values inside the window are copied and sorted. The value of the central pixel is replaced with the median value of the nine pixels in the window. The window slides right by one column after every clock cycle until the end of the line. The following illustration shows the effect of a median-based denoising filter on a noisy image.

***Figure 1 •* Median-Based Denoising Filter Effect on a Noisy Image**

3 Image De-Noising Filter Hardware Implementation

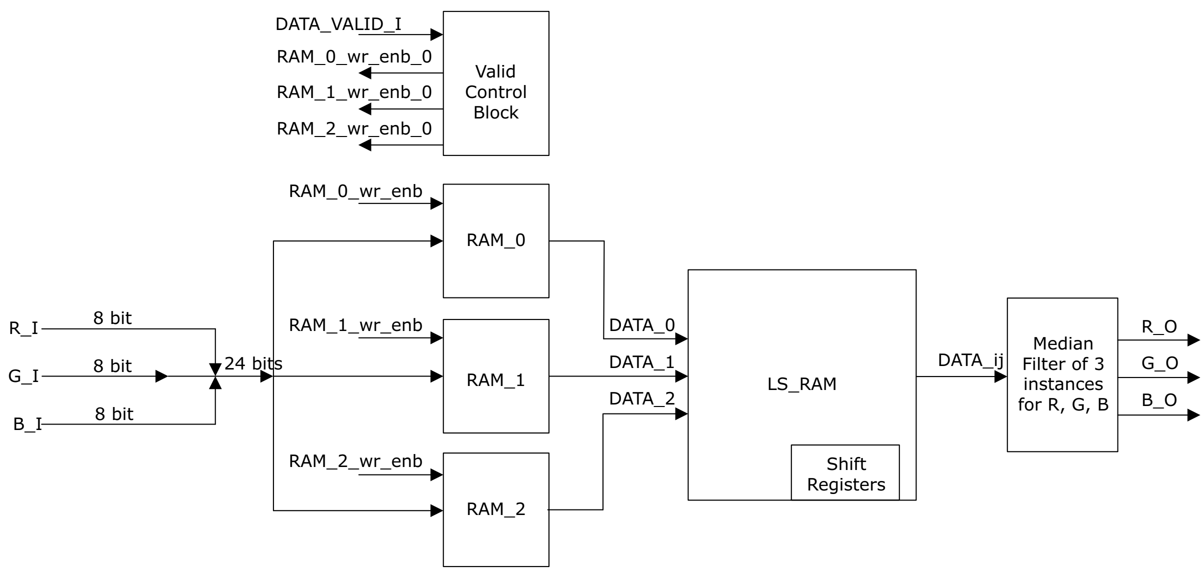
Microsemi Image De-noising Filter IP core—a part of Microsemi's imaging and video solutions IP suite—supports 3 × 3 2D median filtering and effectively removes impulse noise from images.

The Image De-noising Filter hardware contains three one-line buffers storing one horizontal video line each. The incoming data stream fills these three buffers, one by one. In the design illustrated in this document, the median filter is implemented on 3 × 3 matrix, so three lines of video form the 3 × 3 window for the median. When the third buffer contains three pixel values, the read process is initiated.

Three shift registers form the 3 × 3 2D array for median calculation. These shift registers are applied as input to the median finder, which contains 8-bit comparators that sort the nine input values in increasing order of magnitude and produce the median value, which is then updated into the output register. The new pixel column is shifted into the shift register, with the oldest data being shifted out. The 3 × 3 window moves from the left to right and from top to bottom for each frame.

The following illustration shows the block diagram of the Image De-noising Filter hardware with default RGB888 input.

Figure 2 • **Image De-Noising Filter Hardware**



3.1 Inputs and Outputs

The following table lists the input and output ports of the Image De-noising Filter.

Table 1 • **Image De-Noising Filter Ports**

Port Name	Direction	Width	Description
RESETN_I	Input		Active-low asynchronous reset signal to design
SYS_CLK_I	Input		System clock
R_I	Input	[(g_DATAWIDTH–1):0]	Data input – Red Pixel
G_I	Input	[(g_DATAWIDTH–1):0]	Data input – Green Pixel
B_I	Input	[(g_DATAWIDTH–1):0]	Data input – Blue Pixel
DATA_VALID_I	Input		Input data valid signal
R_O	Output	[(g_DATAWIDTH-1):0]	Data output - Red Pixel
G_O	Output	[(g_DATAWIDTH-1):0]	Data output - Green Pixel
B_O	Output	[(g_DATAWIDTH-1):0]	Data output - Blue Pixel
DATA_VALID_O	Output		Output data valid signal

3.2 Configuration Parameters

The following table lists the configuration parameters for the Image De-noising Filter design.

Table 2 • **Design Configuration Parameters**

Name	Description	Default
G_DATA_WIDTH	Data bit width	8
G_RAM_SIZE	Buffer size of RAM	2048 (for horizontal resolution of 1920)

Note: These are generic parameters that vary based on the application requirements.

3.3 Testbench

To demonstrate the functionality of the Image De-Noise Filter core, a sample testbench file (`image-denoise_test`) is available in the Stimulus Hierarchy (View > Windows > Stimulus Hierarchy), and a sample testbench input image file (`RGB_input.txt`) is available in the Libero® SoC Files window (View > Windows > Files).

The following table lists the testbench parameters that can be configured according to the application, if necessary.

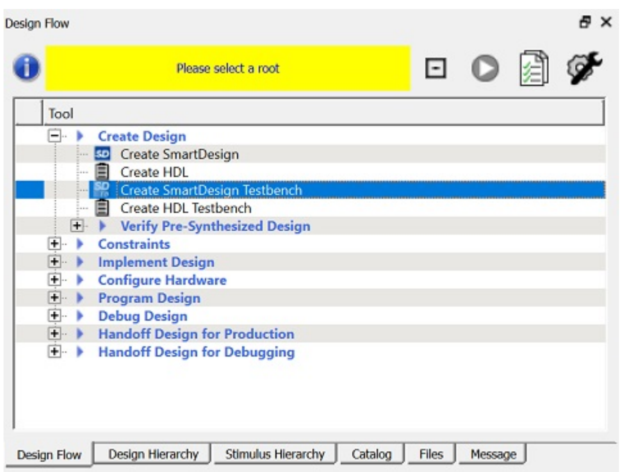
Table 3 • **Testbench Configuration Parameters**

Name	Description
CLKPERIOD	Clock period
HEIGHT	Height of the image
WIDTH	Width of the image
g_DATAWIDTH	Data bit width
WAIT	Number of clock cycles of delay between the transmission of one line of the input image and the next
IMAGE_FILE_NAME	Input image name

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

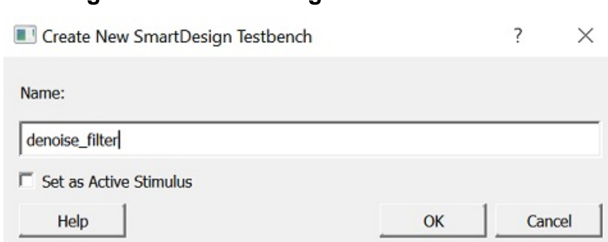
1. In the Libero SoC Design Flow window, expand **Create Design**, and double-click **Create SmartDesign Testbench,** as shown in the following figure.

Figure 3 • **Create SmartDesign Testbench**



2. Enter a name for the SmartDesign testbench and click **OK**.

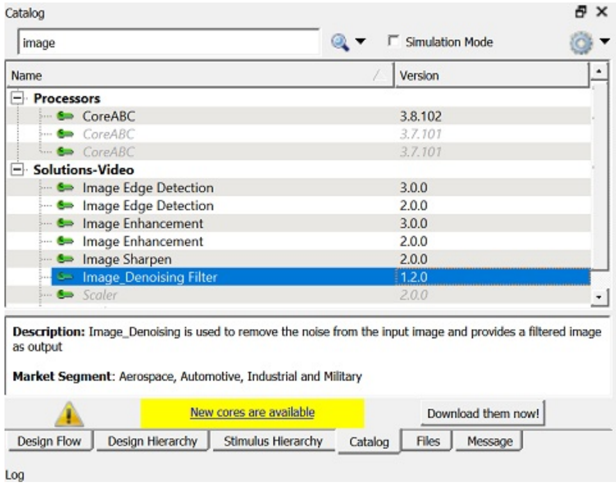
Figure 4 • **Create New SmartDesign Testbench Dialog Box**



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

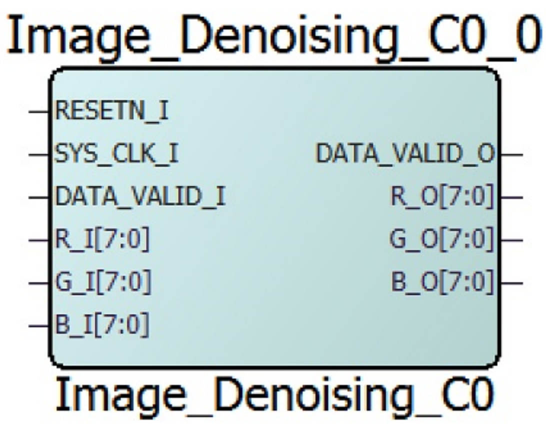
3. In the Libero SoC Catalog (View > Windows > Catalog), expand **Solutions-Video,** and drag the Image De-Noise Filter IP core onto the SmartDesign testbench canvas.

Figure 5 • **Image De-Noise Filter Core in Libero SoC Catalog**



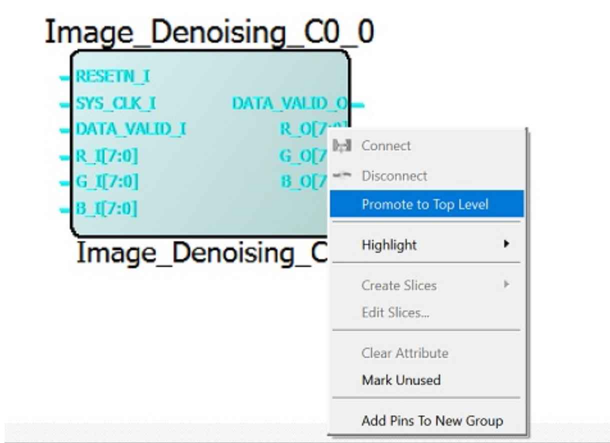
The core appears on the canvas, as shown in the following figure.

Figure 6 • **Image De-Noise Filter Core on SmartDesign Testbench Canvas**



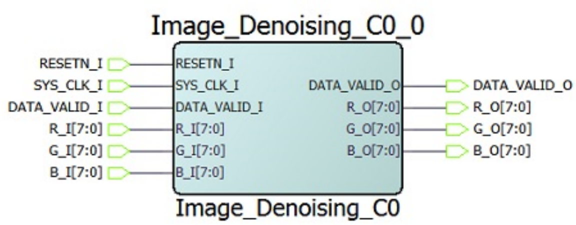
4. Select all the ports of the core, right-click, and click **Promote to Top Level**, as shown in the following figure.

Figure 7 • **Promote to Top Level Option**



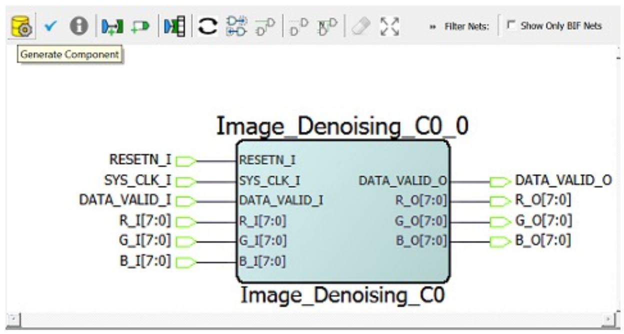
The ports are promoted to the top level, as shown in the following figure.

Figure 8 • **Image De-Noise Filter Core Ports Promoted to Top Level**



5. To generate the Image De-noising Filter SmartDesign component, click **Generate Component** icon on the SmartDesign Toolbar, as shown in the following figure.

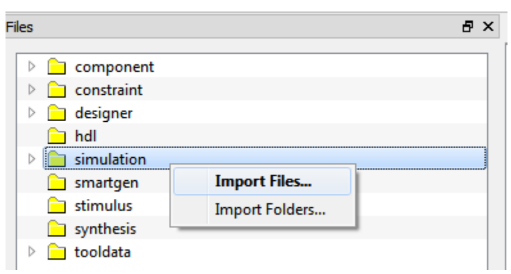
Figure 9 • **Generate Component Icon**



A sample testbench input image file is created at:
`…\Project_name\component\Microsemi\SolutionCore\Image_Denoising_Filter\1.2.0\Stimulus`

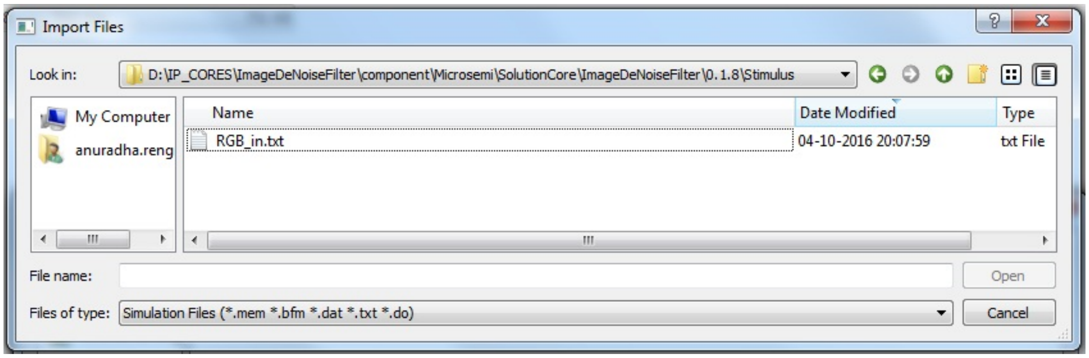
6. In the Libero SoC Files window, right-click the simulation directory, and click **Import Files...**, as shown in the following figure.

Figure 10 • **Import Files Option**



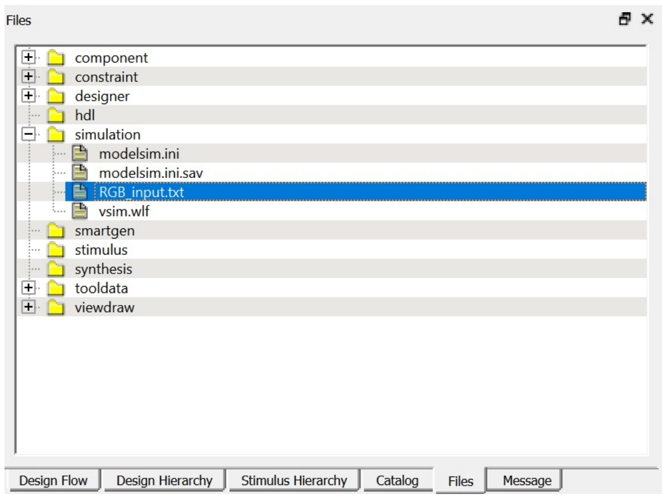
7. Do one of the following:
   - To import the sample testbench input image, browse to the sample testbench input image file, and click **Open**, as shown in the following figure.
   - To import a different image, browse to the desired image file, and click **Open**.

Figure 11 • Input Image File Selection



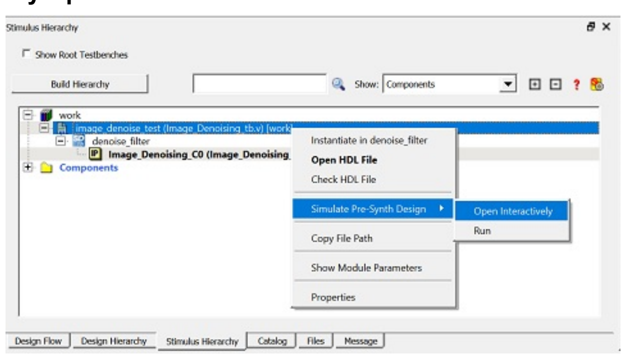
The input image file appears in the simulation directory, as shown in the following figure.

Figure 12 • Input Image File in Simulation Directory



8. In the Stimulus Hierarchy, expand **Work**, and right-click the Image De-noising Filter testbench file (`image_denoise_test.v`).
9. Click **Simulate Pre-Synth Design**, and then click **Open Interactively**.

Figure 13 • Open Interactively Option



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

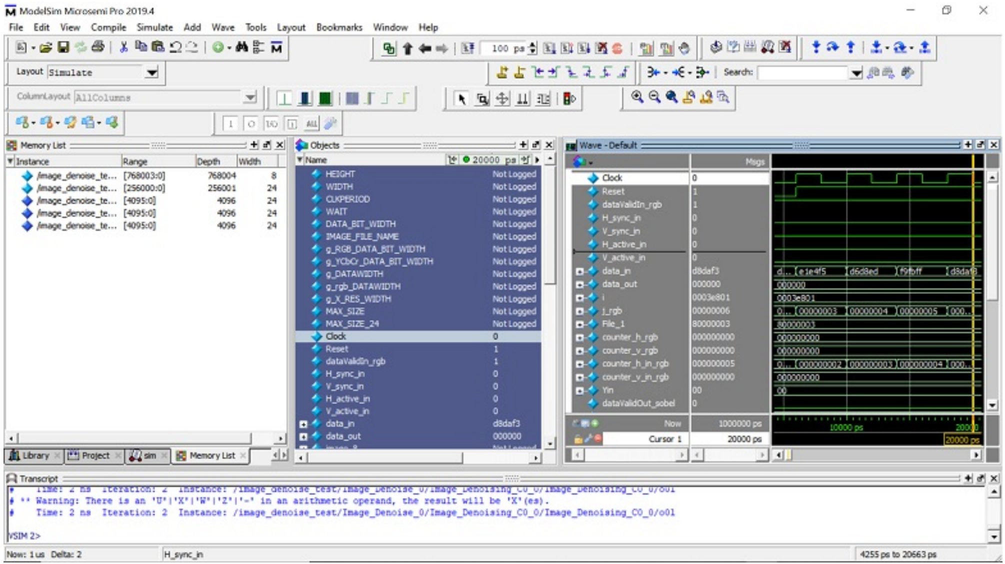


Figure 14 • **ModelSim Tool with Image De-Noising Filter Testbench File**

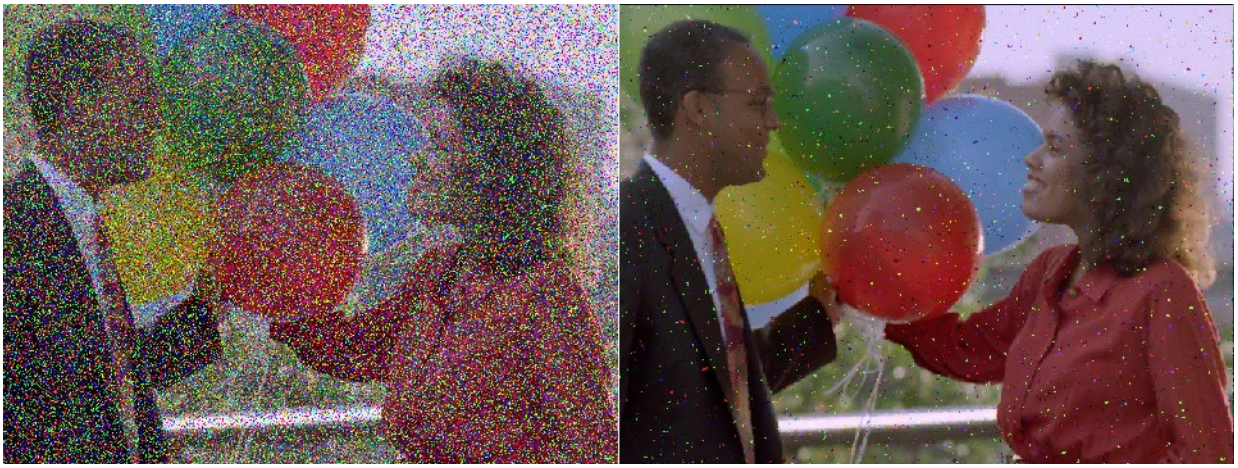
10. If the simulation is interrupted because of the runtime limit in the DO file, use the run -all command to complete the simulation.
    After the simulation is completed, the testbench output image file (.txt) appears in the simulation folder.

3.4 Simulation Results

The following illustration shows the effect of the Image De-noising Filter on a noisy image.

Figure 15 • **Image De-Noising Filter Effect on a Noisy Image 1**

Figure 16 • **Image De-Noising Filter Effect on a Noisy Image 2**



3.5 Resource Utilization

In this design, the Image De-noising Filter is implemented on an MPF300TS-1FCG1152I PolarFire System-on-Chip (SoC) FPGA. The following table provides resource utilization data for a 24-bit data width design after synthesis.

Note: Image De-noising Filter supports for SmartFusion2 and PolarFire FPGAs.

Table 4 • **Resource Utilization**

Resource	Utilization
DFFs	1961
4_input LUTs	2417
MACC	0
RAM1Kx18	15
RAM64x18	0