Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided “as is, where is” and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) 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. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.
Contents

1 Revision History ................................................................. 1
  1.1 Revision 2.0 ................................................................. 1
  1.2 Revision 1.0 ................................................................. 1

2 Introduction ........................................................................... 2

3 Image De-Noising Filter Hardware Implementation .................. 3
  3.1 Inputs and Outputs ......................................................... 4
  3.2 Configuration Parameters ............................................... 4
  3.3 FSM Implementation ..................................................... 4
  3.4 Test Bench .................................................................. 5
  3.5 Simulation Results ........................................................ 10
  3.6 Resource Utilization ...................................................... 11
Figures

Figure 1 Median-Based Denoising Filter Effect on a Noisy Image ........................................ 2
Figure 2 Image De-Noise Filter Hardware .......................................................... 3
Figure 3 Image De-Noise Filter FSM ........................................................................ 5
Figure 4 Create SmartDesign Testbench ................................................................. 6
Figure 5 Create New SmartDesign Testbench Dialog Box ............................................ 6
Figure 6 Image De-Noise Filter Core in Libero SoC Catalog ........................................ 6
Figure 7 Image De-Noise Filter Core on SmartDesign Test Bench Canvas .................. 7
Figure 8 Promote to Top Level Option ..................................................................... 7
Figure 9 Image De-Noise Filter Core Ports Promoted to Top Level .............................. 7
Figure 10 Generate Component Icon ....................................................................... 8
Figure 11 Import Files Option .................................................................................. 8
Figure 12 Input Image File Selection ....................................................................... 8
Figure 13 Input Image File in Simulation Directory .................................................... 9
Figure 14 Open Interactively Option ........................................................................ 9
Figure 15 ModelSim Tool with Image De-Noise Filter Test Bench File ..................... 10
Figure 16 Image De-Noise Filter Effect on a Noisy Image ........................................... 10
Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Image De-Noising Filter Ports</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Design Configuration Parameters</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Test Bench Configuration Parameters</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Resource Utilization</td>
<td>11</td>
</tr>
</tbody>
</table>
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 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 Cb and Cr 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 test bench were added. For more information, see Test Bench, 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 11.

1.2 Revision 1.0

Revision 1.0 was the first publication of this document.
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, a standard median operation is implemented by sliding a window of an odd size over the image. The $3 \times 3$ window size, considered to be effective for the most commonly used images, is chosen. 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. When this model is applied to grayscale images, pixels are ranked for their brightness or luminance. 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
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 five one-line buffers storing one horizontal video line each. The incoming data stream fills these five buffers, one by one. In the design illustrated in this document, the median filter is implemented on a 3 × 3 matrix, so three lines of video form the 3 × 3 window for 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 sends out 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.

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>
<thead>
<tr>
<th>Table 1 • Image De-Noising Filter Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port Name</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>RESET_n_I</td>
</tr>
<tr>
<td>SYS_CLK_I</td>
</tr>
<tr>
<td>Data_In_i</td>
</tr>
<tr>
<td>Data_In_Vld_i</td>
</tr>
<tr>
<td>Data_Out_o</td>
</tr>
<tr>
<td>Data_Out_Vld_o</td>
</tr>
</tbody>
</table>

3.2 Configuration Parameters

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

<table>
<thead>
<tr>
<th>Table 2 • Design Configuration Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>g_DATAWIDTH</td>
</tr>
<tr>
<td>g_X_RES_WIDTH</td>
</tr>
<tr>
<td>g_DISPLAY_RESOLUTION</td>
</tr>
<tr>
<td>g_VERT_DISPLAY_RESOLUTION</td>
</tr>
</tbody>
</table>

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

3.3 FSM Implementation

During the Image De-noising Filter finite state machine (FSM) implementation, the FSM goes through the following states.

- idle: After the module is reset, the FSM goes into idle state and waits for the third pixel of the third line to be read. It then proceeds to check_line_compl.
- check_line_compl: The FSM waits for the output pixel count to be equal to the display resolution. It then proceeds to last_line_written.
- last_line_written: The FSM waits for the last input line. If the last line is written, it proceeds to check_frame_compl, else it proceeds to check_input_data.
- check_input_data: The FSM waits for the fourth pixel of the last line to be written into the line buffer. It then proceeds to check_frame_compl.
- check_frame_compl: The FSM waits for the output line count to be equal to the vertical resolution width. If the output is the last line, it proceeds to last_line, else it goes to check_line_compl.
- last_line: The FSM waits for the last output pixel of the last line and then moves back to idle state.
The following illustration shows the FSM implementation for the Image De-noising filter.

**Figure 3 • Image De-Noising Filter FSM**

3.4 Test Bench

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

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

**Table 3 • Test Bench Configuration Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKPERIOD</td>
<td>Clock period</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Height of the image</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Width of the image</td>
</tr>
<tr>
<td>g_DATAWIDTH</td>
<td>Data bit width</td>
</tr>
<tr>
<td>WAIT</td>
<td>Number of clock cycles of delay between the transmission of one</td>
</tr>
<tr>
<td></td>
<td>line of the input image and the next</td>
</tr>
<tr>
<td>IMAGE_FILE_NAME</td>
<td>Input image name</td>
</tr>
</tbody>
</table>
The following steps describe how to simulate the core using the test bench.

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

   **Figure 4 • Create SmartDesign Testbench**
   
   ![Create SmartDesign Testbench](image)

2. Enter a name for the SmartDesign test bench, and click **OK**.

   **Figure 5 • Create New SmartDesign Testbench Dialog Box**
   
   ![Create New SmartDesign Testbench Dialog Box](image)

A SmartDesign test bench 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 test bench canvas.

   **Figure 6 • Image De-Noise Filter Core in Libero SoC Catalog**
   
   ![Image De-Noise Filter Core in Libero SoC Catalog](image)
The core appears on the canvas, as shown in the following figure.

**Figure 7**  Image De-Noise Filter Core on SmartDesign Test Bench Canvas

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

**Figure 8**  Promote to Top Level Option

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

**Figure 9**  Image De-Noise Filter Core Ports Promoted to Top Level
5. To generate the Image De-noising Filter SmartDesign component, click the Generate Component icon on the SmartDesign Toolbar, as shown in the following figure.

Figure 10 • Generate Component Icon

A sample test bench input image file is created at: 
\Project_name\component\Microsemi\SolutionCore\ImageDeNoiseFilter\2.0.0\Stimulus

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

Figure 11 • Import Files Option

7. Do one of the following:
   • To import the sample test bench input image, browse to the sample test bench 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 12 • Input Image File Selection
The input image file appears in the simulation directory, as shown in the following figure.

**Figure 13** • Input Image File in Simulation Directory

9. Click **Simulate Pre Synth Design**, and then click **Open Interactively**.

**Figure 14** • Open Interactively Option
The ModelSim tool appears with the test bench file loaded on to it, as shown in the following figure.

**Figure 15 • ModelSim Tool with Image De-Noise Filter Test Bench File**

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 test bench output image file (.txt) appears in the simulation folder.

### 3.5 Simulation Results

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

**Figure 16 • Image De-Noise Filter Effect on a Noisy Image**
3.6 Resource Utilization

In this design, the Image De-noising Filter is implemented on an M2S150T-1FC1152 SmartFusion®2 system-on-chip (SoC) FPGA. The following table provides resource utilization data for a 24-bit data width design after synthesis.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFFs</td>
<td>1785</td>
</tr>
<tr>
<td>4_input LUTs</td>
<td>2782</td>
</tr>
<tr>
<td>MACC</td>
<td>0</td>
</tr>
<tr>
<td>RAM1Kx18</td>
<td>15</td>
</tr>
<tr>
<td>RAM64x18</td>
<td>0</td>
</tr>
</tbody>
</table>