UG0640
User Guide
Bayer Interpolation

Microsemi
a Microchip company

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.

50200640. 5.0 2/19

Contents

1 Revision History 1
1.1 Revision 5.0 1
1.2 Revision 4.0 1
1.3 Revision 3.0 1
1.4 Revision 2.0 1
1.5 Revision 1.0 1

2 Introduction 2
2.1 Bilinear Interpolation 2

3 Hardware Implementation 4
3.1 Write LSRAM 4
3.2 Read LSRAM 4
3.3 Bilinear Interpolation 4

4 Interfaces 5
4.1 Ports 5
4.2 Configuration Parameters 5

5 Timing Diagrams 6

6 Test Bench 7
6.1 Simulation Steps 7

7 Simulation Results 12

8 Resource Utilization 13

Figures

Figure 1 Demosaicing of Bayer format Image . . . 2
Figure 2 Bayer Interpolation Block Diagram . . . 4
Figure 3 Bayer Interpolation Showing first and second frame . . . 6
Figure 4 Bayer Interpolation Showing first three lines of second frame . . . 6
Figure 5 Opening New SmartDesign Testbench . . . 7
Figure 6 Creating a SmartDesign Testbench . . . 8
Figure 7 Bayer Interpolation Core in Libero SoC Catalog . . . 8
Figure 8 Bayer Interpolation Core on SmartDesign Testbench Canvas . . . 8
Figure 9 Promote to Top-Level . . . 9
Figure 10 Generating Bayer Component with Ports Promoted to Top Level . . . 9
Figure 11 Import Files . . . 9
Figure 12 Imported File . . . 10
Figure 13 Simulating Pre-Synthesis Design . . . 10
Figure 14 ModelSim Simulation Window . . . 10
Figure 15 Input Bayer Image . . . 12
Figure 16 Output RGB Image . . . 12

Tables

Table 1 Input and Output Ports 5
Table 2 Configuration Parameters 5
Table 3 Testbench Configuration Parameters 7
Table 4 Resource Utilization on PolarFire 13
Table 5 Resource Utilization on SmartFusion2 13

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 5.0

The following is a summary of changes in this revision.

- Updated Introduction, page 2.
- Updated Figure 1, page 2, Figure 2, page 4, Figure 3, page 6, and Figure 4, page 6.
- Updated tables such Interfaces, page 5.
- Updated Resource Utilization, page 13.
- Updated Test Bench, page 7.
- Updated Simulation Results, page 12.

1.2 Revision 4.0

Updated the resource Utilization.

1.3 Revision 3.0

Updated the testbench information.

1.4 Revision 2.0

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

- Added the TestBench section.

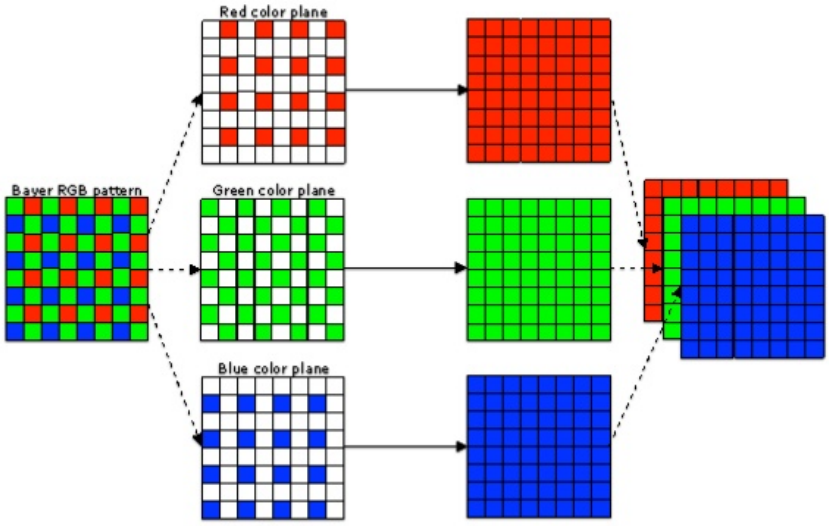
1.5 Revision 1.0

The first publication of this document.

2 Introduction

Bayer Interpolation coverts an image in Bayer color filter array format to RGB per pixel format. The following figure shows the demosaicing of a Bayer format image.

Figure 1 • **Demosaicing of Bayer format Image**



There are several standard interpolation methods. The simplest interpolation method is bilinear interpolation. The Bayer interpolation IP uses the bilinear interpolation methods to covert a Bayer format image to RGB format.

2.1 Bilinear Interpolation

The bilinear algorithm processes each pixel separately and finds out the missing components in it by applying linear interpolation to the available ones.

The formulas for calculating missing component at a particular pixel by considering 3x3 window are as follows.

Green component at red and blue pixel

$$G(i, j) = \frac{1}{4} \cdot \sum G(i + m, j + n)$$

where (m,n) = {(0,-1)(0,1)(-1,0)(1,0)}

Red component at blue pixel

$$R(i, j) = \frac{1}{4} \cdot \sum R(i + m, j + n)$$

where (i,j) = {(-1,-1)(-1,1)(1,-1)(1,1)}

Red component at green pixel

$$R(i, j) = \frac{1}{2} \cdot \sum R(i + m, j + n)$$

where (m,n) = {(0,-1)(0,1)} or (m,n) = {(-1,0)(1,0)}

Blue component at red pixel

$$B(i, j) = \frac{1}{4} \cdot \sum B(i + m, j + n)$$

where (m,n) = {(-1,-1)(-1,1)(1,-1)(1,1)}

Blue component at green pixel

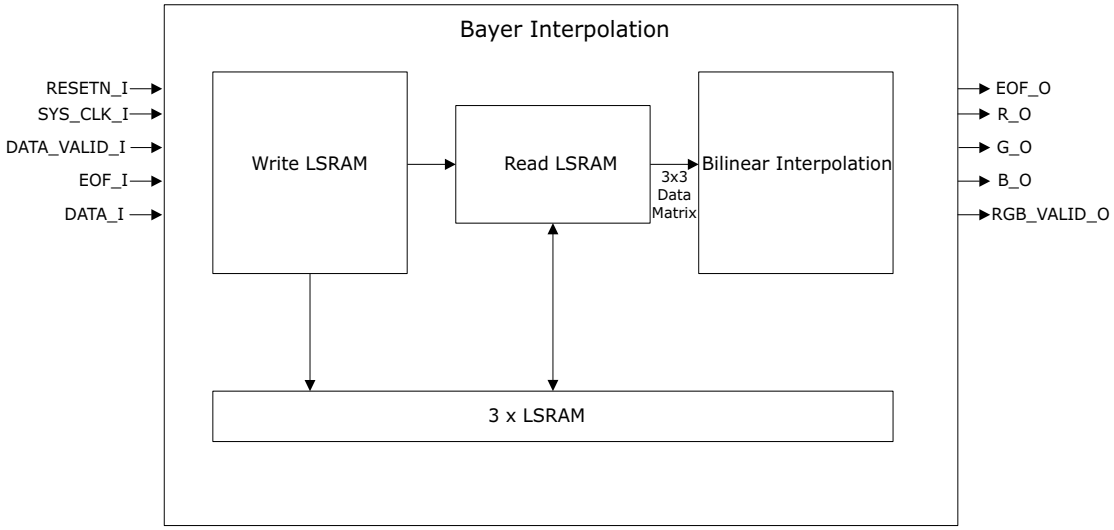
$$B(i, j) = \frac{1}{2} \cdot \sum B(i + m, j + n)$$

where (m,n) = {(0,-1)(0,1)} or (m,n) = {(-1,0)(1,0)}

3 Hardware Implementation

The following figure shows the block diagram of Bayer interpolation.

Figure 2 • **Bayer Interpolation Block Diagram**



The Bayer interpolation IP consists of the following three submodules.

- Write LSRAM, page 4
- Read LSRAM, page 4
- Bilinear Interpolation, page 4

3.1 Write LSRAM

The raw image data coming from camera sensor is written into 3 different LSRAM. The 1st, 4th, 7th …… line of the frame are written to LSRAM1, the 2nd, 5th, 8th …… line of the frame are written into LSRAM2 and the 3rd, 6th, 9th ….. line of the frame are written into LSRAM3. The LSRAM addresses and write enable signals are generated by write LSRAM submodule.

3.2 Read LSRAM

The read submodule generates the read enable signals and the addresses to read from LSRAM. It also has the 3x3 window logic which reads the 3x3 window from LSRAMs and feeds to the bilinear interpolation block. The pixel at which the color components are to be computed is placed at the center of the 3x3 window. Then the window slides right to compute the value of the next pixel in the line.

For the first line of the frame, the first row of the 3x3 window is all zeros, the second row is LSRAM1 data and third row is LSRAM2 data. For the second line, the first row is LSRAM1 data, second row is LSRAM2 data and third row is LSRAM3 data. For the third line, the first row is LSRAM2 data, second row is LSRAM3 data and third row is LSRAM1 data and so on.

3.3 Bilinear Interpolation

The bilinear interpolation module computes the R, G and B value for the center element of the 3x3 data matrix coming from read LSRAM module. It computes the R, G and B value based on the bilinear interpolation formulae described in Bilinear Interpolation, page 2.

The Bayer interpolation IP automatically detects the video resolution. The IP uses the data from first frame to compute the horizontal and vertical resolution. As a result, the IP does not generate output (data valid is zero) during the first frame.

4 Interfaces

This section describes the input/output ports and configuration parameters of the Bayer Interpolation IP.

4.1 Ports

The following figure shows the input and output ports of Bayer interpolation.

Table 1 • **Input and Output Ports**

Port Name	Type	Width	Description
RESETN_I	Input	1bit	Active low asynchronous reset signal to design
SYS_CLK_I	Input	1bit	System clock
DATA_VALID_I	Input	1bit	Asserted high when input data is valid
EOF_I	Input	1bit	End of frame input signal
DATA_I	Input	G_DATA_WIDTH bits	Bayer data input
RGB_VALID_O	Output	1bit	Asserted high when output data is valid
R_O	Output	G_DATA_WIDTH bits	Provides the red component output
G_O	Output	G_DATA_WIDTH bits	Provides the green component output
B_O	Output	G_DATA_WIDTH bits	Provides the blue component output
EOF_O	Output	1bit	End of frame output. The first EOF_I is skipped and subsequent EOF_I inputs are passed through.

4.2 Configuration Parameters

The following table shows the description of the configuration parameters used in the hardware implementation of Bayer Interpolation. These are generic parameters and can be varied as per the requirement of the application.

Table 2 • **Configuration Parameters**

Name	Description
G_DATA_WIDTH	Width of each pixel
G_RAM_SIZE	Size of the RAM to store one horizontal line Choose values which are powers of 2, such as 2048, and 4096.
G_BAYER_FORMAT[1]	Bayer format

1. If G_BAYER_FORMAT = 0, then Bayer format is RGGB
If G_BAYER_FORMAT = 1, then Bayer format is GRBG
If G_BAYER_FORMAT = 2, then Bayer format is GBRG
If G_BAYER_FORMAT = 3, then Bayer format is BGGR

5 Timing Diagrams

The following figure shows the timing diagram of Bayer Interpolation.

Figure 3 • **Bayer Interpolation Showing first and second frame**

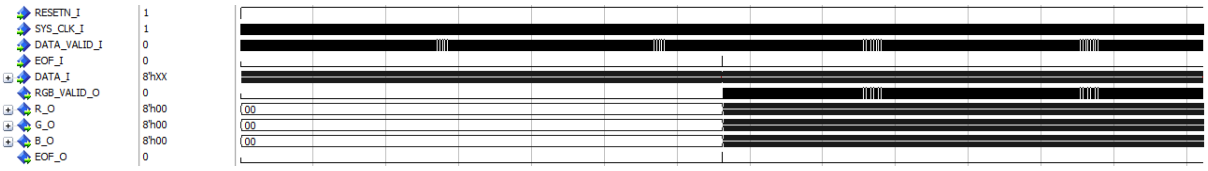
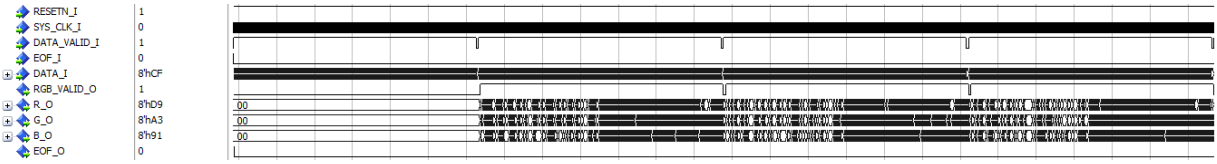


Figure 4 • **Bayer Interpolation Showing first three lines of second frame**



6 Test Bench

A testbench is provided to check the functionality of Bayer Interpolation IP. The following table shows the parameters that can be configured according to the application.

Table 3 • **Testbench Configuration Parameters**

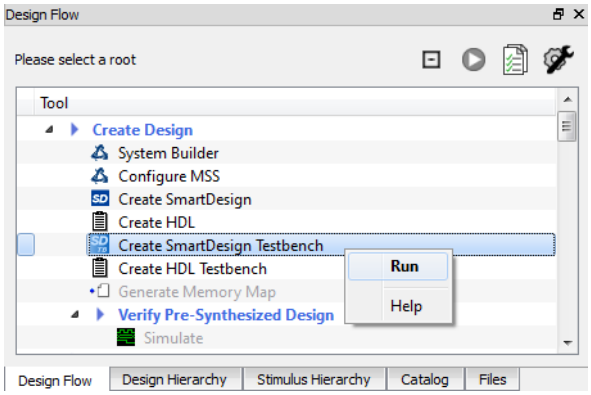
Name	Description
CLKPERIOD	Clock Period
g_DATAWIDTH	Width of each pixel
g_DISPLAY_RESOLUTION	Horizontal resolution
g_VERT_DISPLAY_RESOLUTION	Vertical resolution
WAIT	Number of clock cycles delay between transmission of two input lines
IMAGE_FILE_NAME	Input (image) file name

6.1 Simulation Steps

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

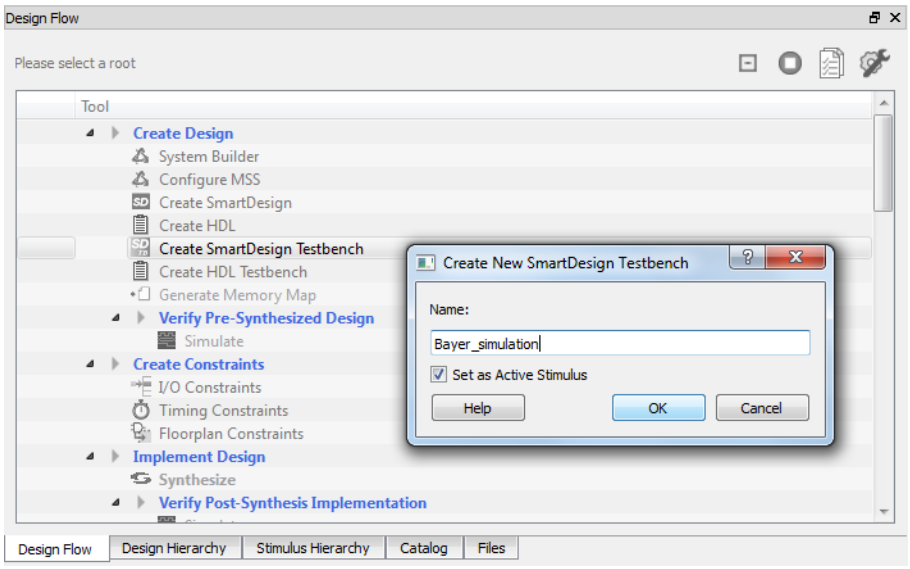
1. On Libero SoC Design Flow, expand **Create Design** and open **Create SmartDesign Testbench** as shown in the following figure.

Figure 5 • **Opening New SmartDesign Testbench**



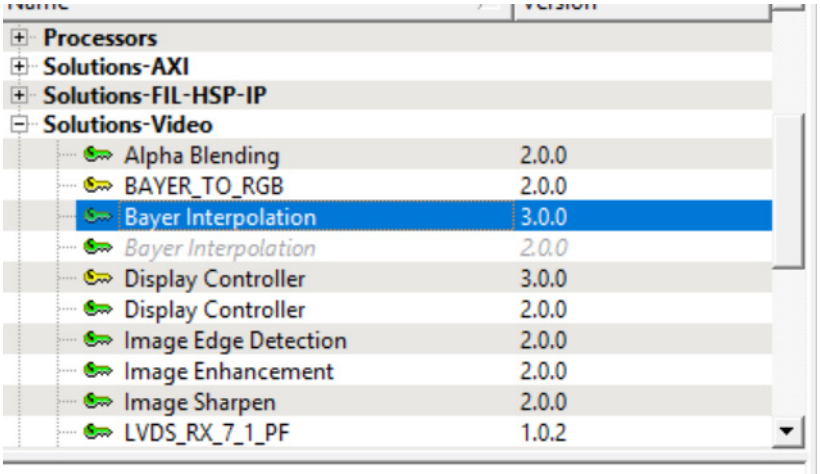
2. Enter a name for the SmartDesign testbench and click **OK** as shown in Figure 6, page 8.
The SmartDesign testbench is created, and a canvas appears to the right of the Design Flow pane.

Figure 6 • **Creating a SmartDesign Testbench**



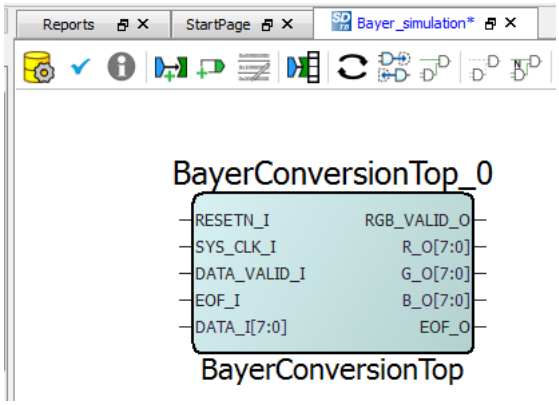
3. Go to Libero SoC Catalog > View > Windows > Catalog, and then expand **Solutions-Video**.

Figure 7 • **Bayer Interpolation Core in Libero SoC Catalog**



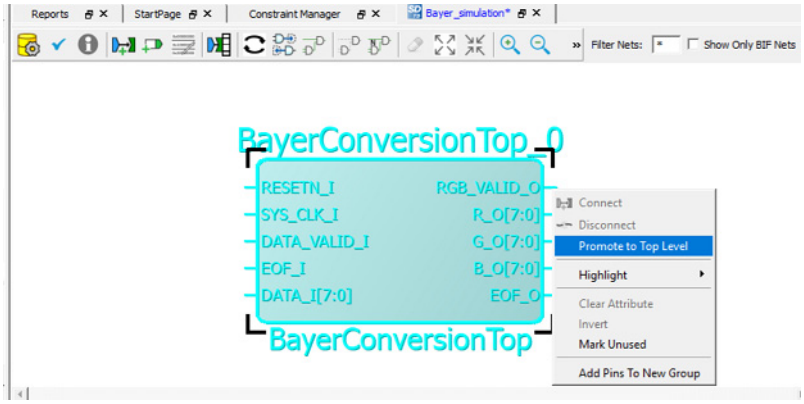
4. Drag and drop the Bayer IP core in to the new SmartDesign testbench canvas. The IP appears as shown in the following figure.

Figure 8 • **Bayer Interpolation Core on SmartDesign Testbench Canvas**



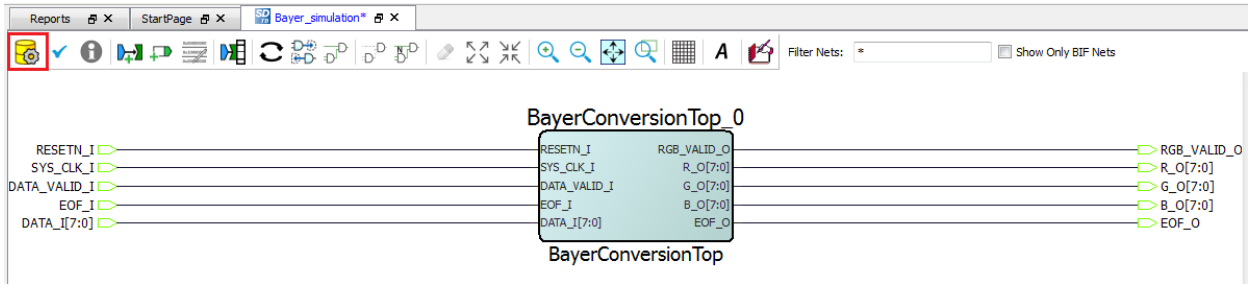
5. Select all of the ports and promote them to top level as shown in the following figure.

Figure 9 • **Promote to Top-Level**



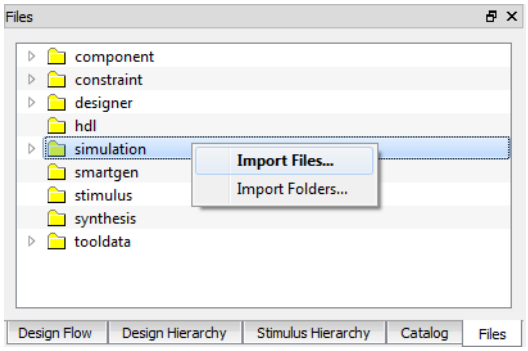
6. To generate the testbench component, select **Generate Component** from the SmartDesign toolbar, as highlighted in the following figure.

Figure 10 • **Generating Bayer Component with Ports Promoted to Top Level**



7. Go to the **Files** tab and select **simulation** > **Import Files** as shown in the following figure.

Figure 11 • **Import Files**

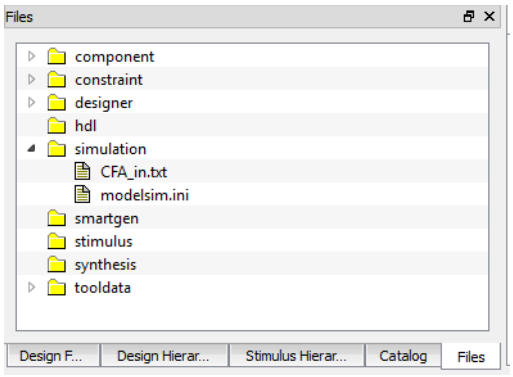


8. Import the CFA file from the following path:
   `..\<Project_name>\component\Microsemi\SolutionCore\BayerConversionTop\3.0.0\Stimulus`

   To import a different file, browse the folder that contains the required file, and click **Open**.

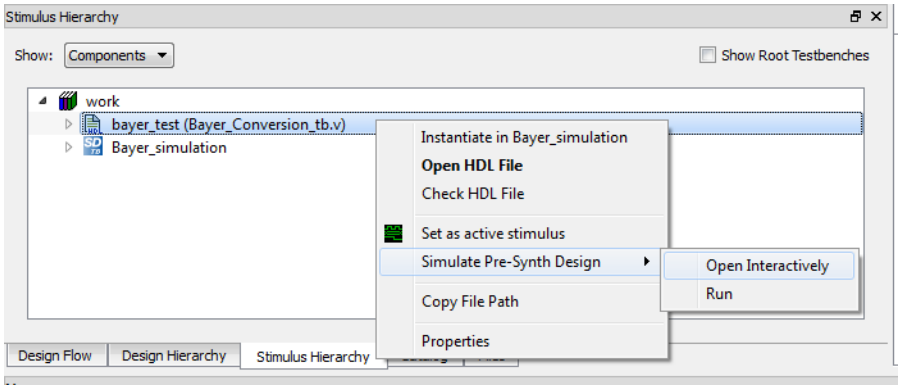
The imported file is listed under simulation as shown in the following figure.

Figure 12 • **Imported File**



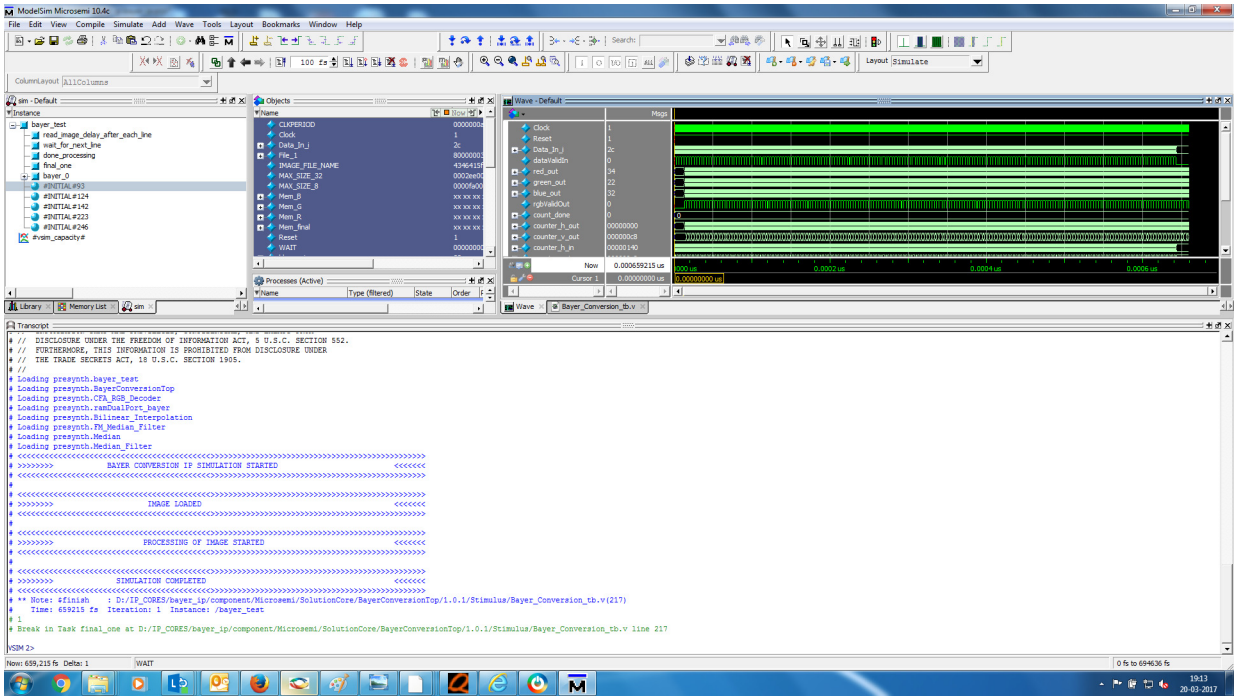
9. Go to the **Stimulus Hierarchy** tab and select **bayer_test (Bayer_interpolation_tb.v)** > **Simulate Pre-Synth Design** > **Open Interactively**. The IP is simulated for one frame.

Figure 13 • **Simulating Pre-Synthesis Design**



ModelSim opens with the testbench file as shown in Figure 14, page 10.

Figure 14 • **ModelSim Simulation Window**



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

The testbench output image file appears in the `Files/simulation` folder after the simulation completes.

7 Simulation Results

The following figure shows the input Bayer image.

Figure 15 • **Input Bayer Image**

Output RGB Image

The following figure shows the output RGB image.

Figure 16 • **Output RGB Image**

8 Resource Utilization

Bayer Interpolation is implemented on the SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) device (M2S150T-1152 FC package) and PolarFire® FPGA (MPF300TS - 1FCG1152E package). The following figure shows the resource utilization report after synthesis.

Table 4 • **Resource Utilization on PolarFire[1]**

Resource	Usage
DFFs	550
4LUTs	1020
LSRAM	3
MACC	0

1. For G_DATA_WIDTH = 8, G_RAM_SIZE = 2048 and G_BAYER_FORMAT = 0.

Table 5 • **Resource Utilization on SmartFusion2[1]**

Resource	Usage
DFFs	580
4LUTs	1060
RAM1K18	3
RAM64x18	0
MACC	0

1. for G_DATA_WIDTH = 8, G_RAM_SIZE = 2048 and G_BAYER_FORMAT = 0.