# UG0693
# User Guide
# Image Edge Detection

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 5.0

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

- Updated Figure 1, page 2.
- Added the following sections.
  - Key Features, page 3
  - Supported Families, page 3
- Added Table 2, page 5.
- Updated Table 3, page 6, Table 5, page 13, and Table 6, page 13.
- Added the following sections.
  - IP Configurator, page 6
  - License, page 7
  - Encrypted, page 7
  - RTL, page 7
- Updated Simulation Steps, page 8.
- Replaced Figure 7, page 9 through Figure 10, page 9.

## 1.2 Revision 4.0

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

- Updated the block diagram and timing diagrams, see Figure 2, page 4, Figure 4, page 7.
- Updated the tables in Interfaces, page 5, Resource Utilization, page 13.
- Updated Testbench, page 8.
- Updated the input and output image of simulation, see Simulation Result, page 12.

## 1.3 Revision 3.0

Updated the resource utilization report.

## 1.4 Revision 2.0

Added the steps to simulate the core using the testbench.

## 1.5 Revision 1.0

The first publication of this document.

# 2 Image Edge Detection

Image Edge Detection is an image processing method for feature detection and extraction. This method is used to identify points in a digital image where the brightness changes sharply and discontinues. The edge detection method reduces the amount of data in an image and preserves the structural properties for further processing. In a gray level image, the edge is a local feature with in a neighborhood separate regions.The gray level is more or less uniform with in different values on the two sides of the edge. For a noisy image, it is difficult to detect edges as both edge and noise contains high frequency contents, which results in blurred and distorted images.

Microsemi offers the Image Edge Detection IP that enables designers to use the edge detection for image processing.

The Image Edge Detection IP implements Sobel filter, which is a classical algorithm in the field of image and video processing for the extraction of object edges. Sobel filter works on the premise of computing an estimate of the first derivative of an image to extract the edge information. By computing the x and y direction derivatives of a specific pixel against a neighborhood of surrounding pixels, it is possible to extract the boundary between two distinct elements in an image. Due to the computational load of calculating derivatives using the squaring and square root operators, fixed coefficient masks are adopted as a suitable approximation in computing the derivative at a specific point. In the case of Sobel, the masks used are shown in the following figure.

***Figure 1 •*** **Sobel Operator Horizontal and Vertical Kernels**

Vertical Mask

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Horizontal Mask

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | 1 |

These kernels can be combined together to find the absolute magnitude of the gradient at each point. The gradient magnitude is computed using:

$$G = \sqrt{Gx^2 + Gy^2}$$

Typically an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

This is much faster to compute. The Sobel operator has the advantage of simplicity in calculation.

## 2.1 Key Features

- Detects the edges in the image
- Uses 3x3 Sobel Kernel
- Supports data width of 8, 10, and 12
- Supports Native and AXI4 Stream Video Interface for video data transfer

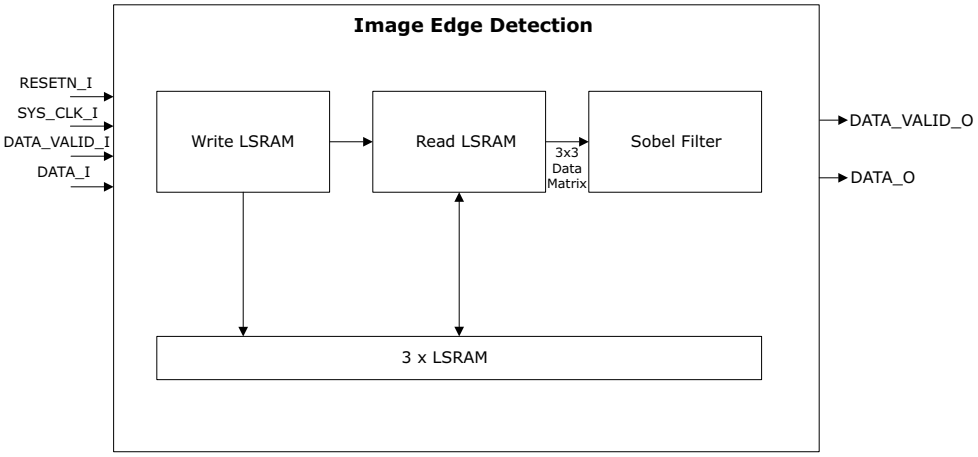
## 2.2 Supported Families

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

# 3 Hardware Implementation

The following figure shows the Image Edge Detection block diagram.

***Figure 2 •*** **Image Edge Detection Block Diagram**




As shown in Figure 2, page 4, the Image Edge Detection IP consists of the following subsystems:

## 3.1 Write LSRAM

The data input to the edge detection IP 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 Sobel filter interpolation block. The pixel at which the edge must 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 Sobel Filter

The Sobel filter performs the Sobel operation (as described in section 2) on the 3x3 window data coming from Read LSRAM block to produce the edge detected image.

# 4 Interfaces

This section describes the input/output ports and the configuration parameters of the Image Edge Detection IP.

## 4.1 Ports

The following table lists the input and output ports.

***Table 1 •*** **Input and Output Ports for Native Video Interface**

| 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 |
| DATA_I | Input | G_DATA_WIDTH bits | Input RGB data |
| DATA_VALID_O | Output | 1bit | Asserted high when output data is valid |
| DATA_O | Output | G_DATA_WIDTH bits | Provides the edge detected output |

***Table 2 •*** **Input and Output Ports for AXI4 Stream Video Interface**

| Port Name | Type | Width | Description |
|---|---|---|---|
| RESETN_I | Input | 1 bit | Active low asynchronous reset signal to design |
| SYS_CLK_I | Input | 1 bit | System clock |
| TDATA_I | Input | 3*G_DATA_WIDTH bit | Input Video Data |
| TVALID_I | Input | 1 bit | Input Video Valid |
| TREADY_O | Output | 1 bit | Output slave ready signal |
| TUSER_I | Input | 4 bits | bit 0 = End of frame<br>bit 1 = unused<br>bit 2 = unused<br>bit 3 = unused |
| TDATA_O | Output | 3*G_DATA_WIDTH bit | Output Video Data |
| TVALID_O | Output | 1 bit | Output Video Valid |
| TUSER_O | Output | 4 bits | bit 0 = End of frame<br>bit 1 = unused<br>bit 2 = unused<br>bit 3 = unused |
| TSTRB_O | Output | G_DATA_WIDTH/8 | Output Video Data strobe |
| TKEEP_O | Output | G_DATA_WIDTH/8 | Output Video Data Keep |
| TLAST_O | Output | 1 bit | Output End of frame |

## 4.2 Configuration Parameters

The following table lists the configuration parameters of the Image Edge Detection IP. These are generic parameters and can vary based on the application requirements.

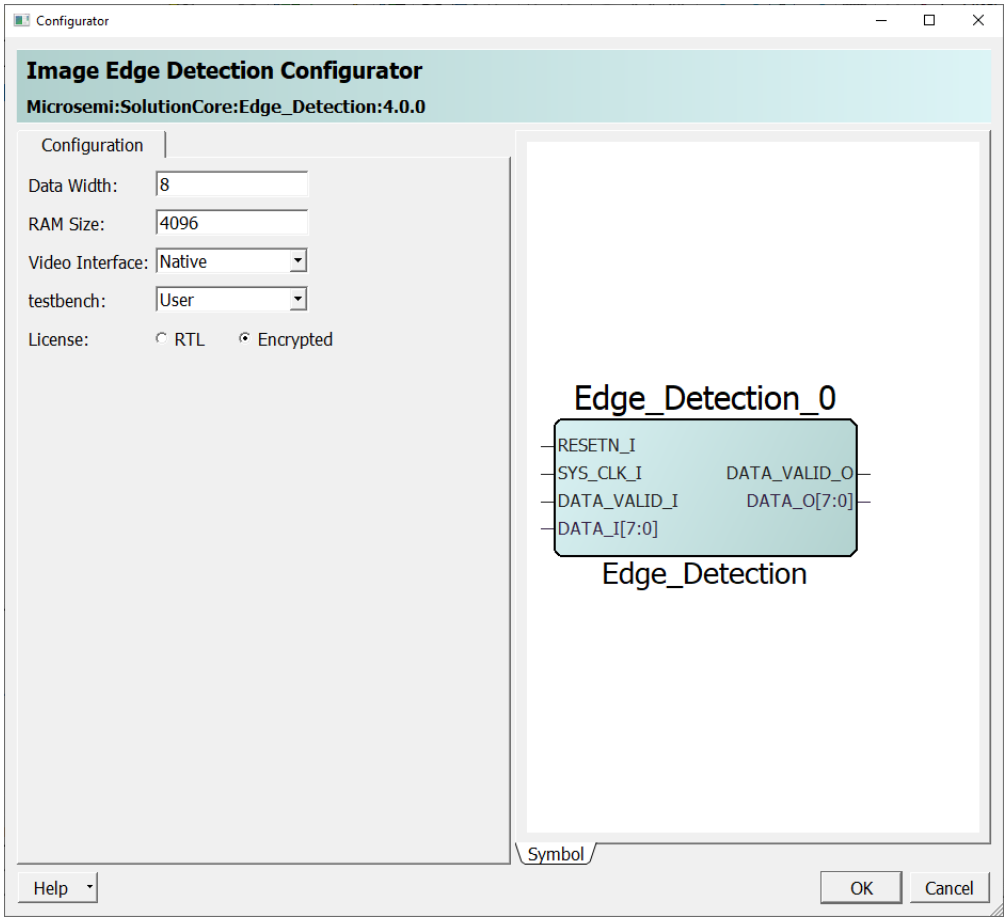
***Table 3 •*** **Configuration Parameters**

| Name | Description |
|---|---|
| Data Width | Width of each pixel |
| RAM Size | Size of the RAM to store one horizontal line<br>Choose values which are powers of 2, such as 2048, and 4096 |
| Video Interface | Native Video Interface and AXI4 Stream Video Interface |

## 4.3 IP Configurator

The IP configurator is shown in the following figure.

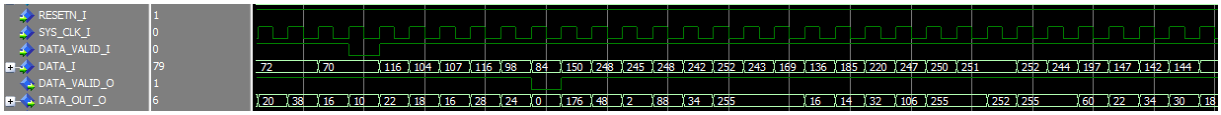
***Figure 3 •*** **IP Configurator**



# 5 Timing Diagram

The following figures show the timing diagram of the Image Edge Detection IP.

***Figure 4 •*** **Timing Diagram of Edge Detection Showing Valid Input and Output Data**




## 5.1 License

Image Edge Detection clear RTL is license locked, and the encrypted RTL available for free.

### 5.1.1 Encrypted

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

### 5.1.2 RTL

Complete RTL source code is provided for the core.

# 6 Testbench

A testbench is provided to check the functionality of the Image Edge Detection IP. The following table lists the parameters that can be configured according to the application.

***Table 4 •*** **Testbench Configuration Parameters**

| Name | Description |
|---|---|
| CLKPERIOD | Clock Period |
| HEIGHT | Height of the image |
| WIDTH | Width of the image |
| WAIT | Amount of delay after each line of input image |
| DATAWIDTH | Width of each pixel |
| IMAGE_FILE_NAME | Input image name |

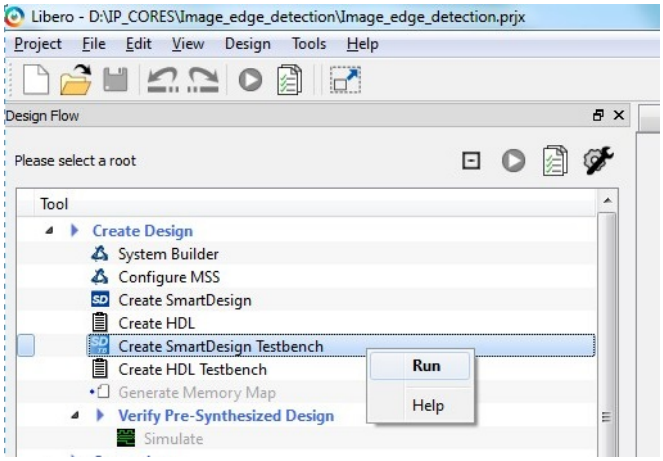
## 6.1 Simulation Steps

A testbench is provided to check the functionality of Image Edge Detection IP. This testbench is working only in Native Video Interface with a data width of 8.

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

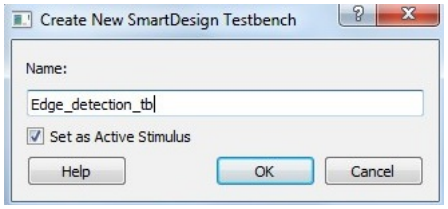
1. On the Design Flow window, expand Create Design and open Create SmartDesign testbench as shown in the following figure.

***Figure 5 •*** **Opening New SmartDesign TestBench**



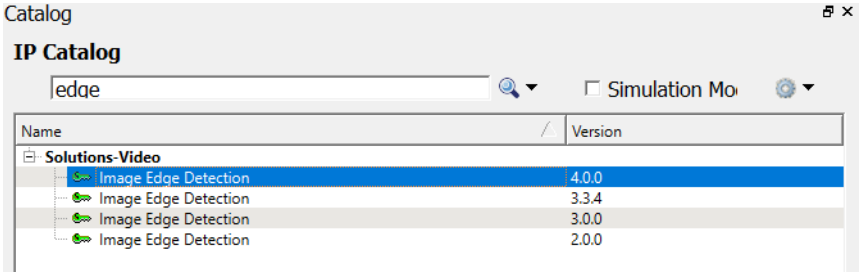

2. Enter a name for the SmartDesign testbench as shown in the following figure.

***Figure 6 •*** **Naming the Testbench**



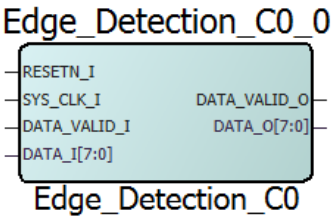
3. Go to the Catalog tab and in search bar type "edge". After that, drag the Image Edge Detection IP to the testbench canvas. The Image Edge Detection IP is highlighted in the following figure.

***Figure 7 •*** **IP Location**



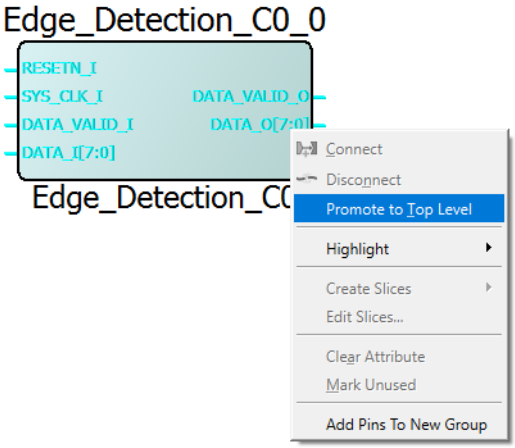

4. After dragging the IP to the testbench canvas, the IP appears as shown in the following figure.

***Figure 8 •*** **Instantiation of the IP**



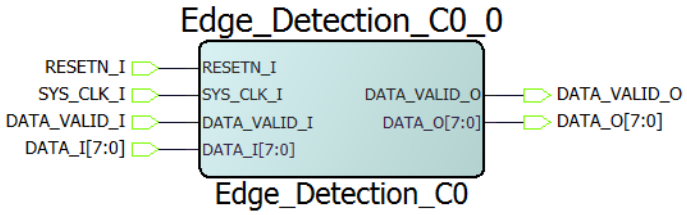

5. Select all of the ports and promote them to top level using the Promote to Top Level option, as shown in the following figure.

***Figure 9 •*** **Promote the Ports to Top Level**



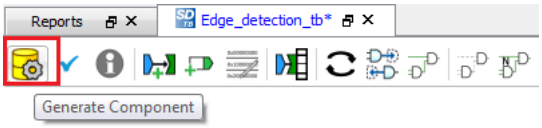

6. The ports are promoted to the top level, as shown in the following figure.

***Figure 10 •*** **Ports Promoted to Top-level**



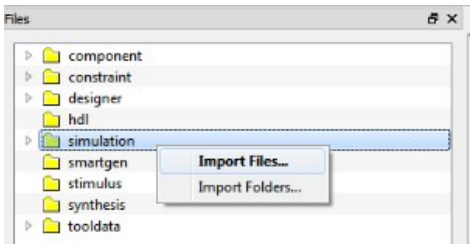
7. Click Generate Component to generate the testbench component. Generate Component is highlighted in the following figure.

***Figure 11 •*** **Generate Component Button**




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

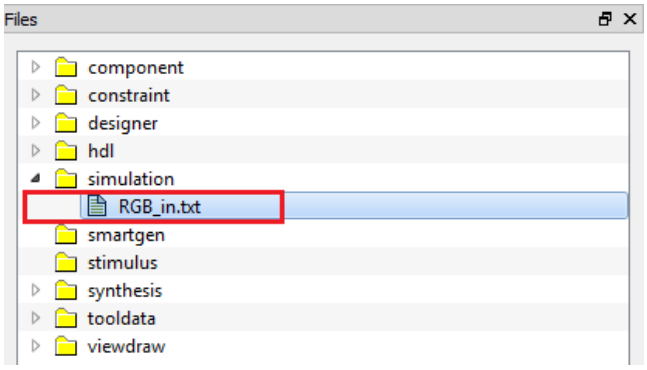
***Figure 12 •*** **The Import Files Option**




9. Import the RGB file from the following path:
   `..\<Project_name>\component\Microsemi\SolutionCore\Image_Edge_Detection\4.0.0\Stimulus`

   The imported file is listed under simulation as shown in Figure 13, page 10.

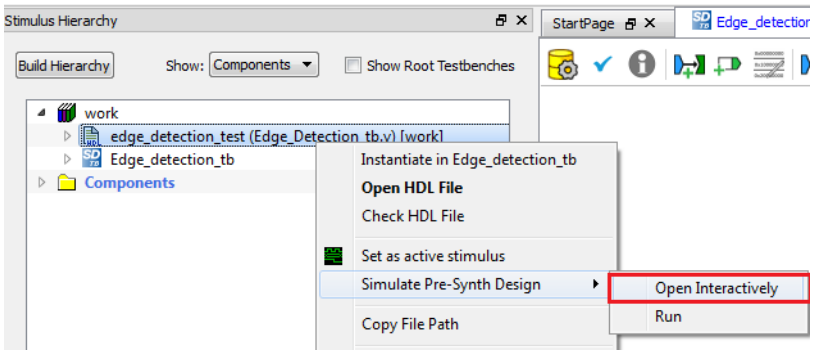
***Figure 13 •*** **Imported File**




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

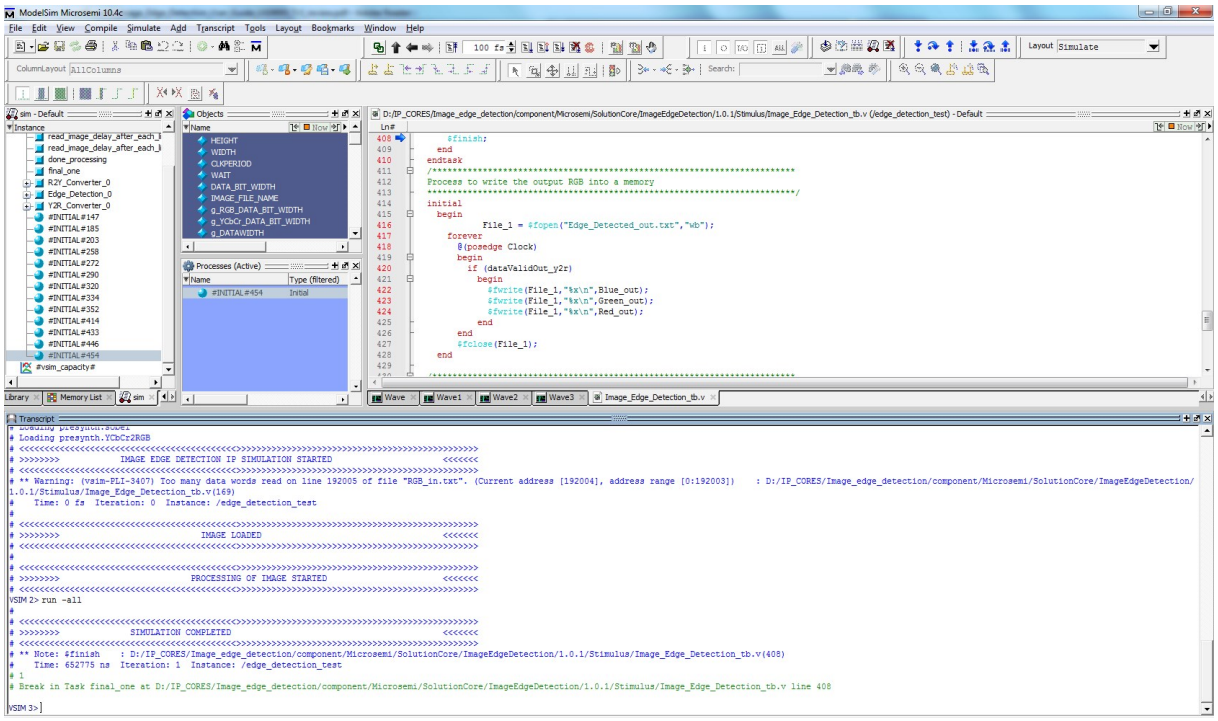
10. Go to the **Stimulus Hierarchy** tab and select **edge_detection_test (edge_detection_tb.v)** > **Simulate Pre-Synth Design** > **Open Interactively**. The core is simulated for one frame.

***Figure 14 •*** **Simulating Pre-Synthesis Design**




11. ModelSim opens with the testbench file as shown in the following figure.

***Figure 15 •*** **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.

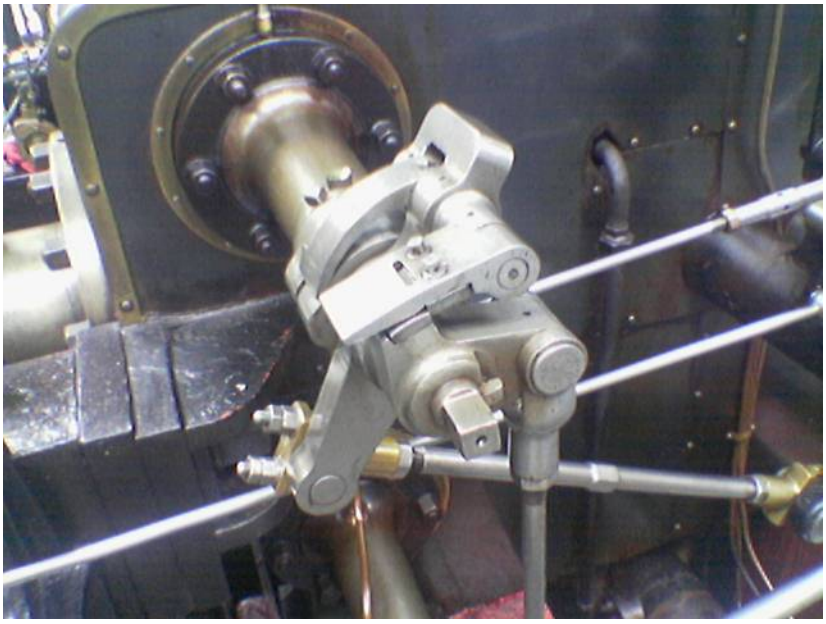
The testbench output image file appears in the `Files/simulation` folder after the simulation completes.

# 7 Simulation Result

This section shows an image before and after being processed using the Image Edge Detection IP.

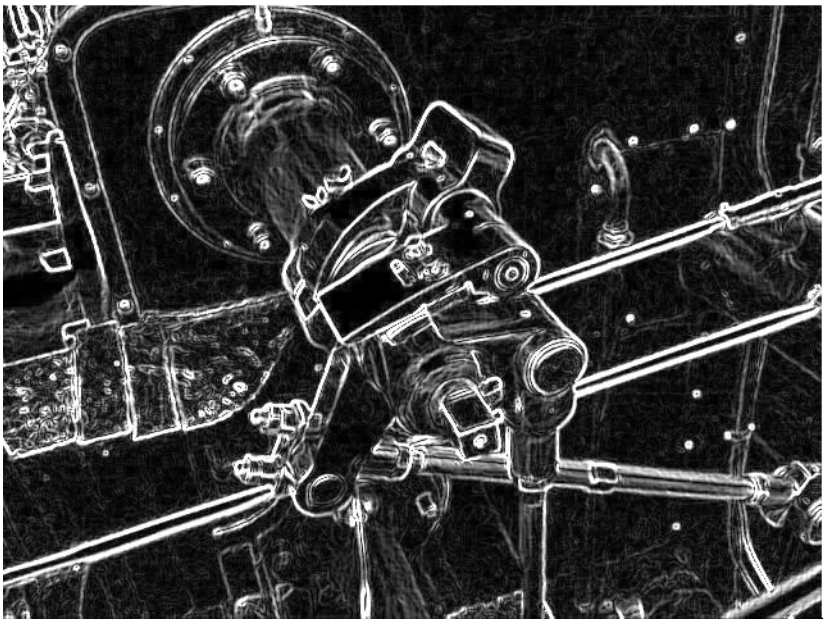
The following figure shows the input image.

***Figure 16 •*** **Input Image**



The following figure shows the output image after being processed using the Image Edge Detection IP.

***Figure 17 •*** **Output Image**



# 8 Resource Utilization

Image Edge Detection 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 5 •*** **Resource Utilization on PolarFire[1]**

| Resource | Usage |
|---|---|
| DFFs | 384 |
| 4LUTs | 671 |
| LSRAM | 3 |
| MACC | 0 |

1. For Data Width = 8 and RAM Size = 2048.

***Table 6 •*** **Resource Utilization on SmartFusion2[1]**

| Resource | Usage |
|---|---|
| DFFs | 400 |
| 4LUTs | 713 |
| RAM1K18 | 3 |
| RAM64x18 | 0 |
| MACC | 0 |

1. For Data Width = 8 and RAM Size = 2048.