LogiCORE IP Image Edge Enhancement v2.0

DS753 December 14, 2010 **Product Specification**

Introduction

The Xilinx Image Edge Enhancement LogiCORE™ IP provides users with an easy-to-use IP block to enhance the edges of objects within each frame of video. The core provides a set of standard Sobel and Laplacian filters with programmable gain that adjust the strength of the edge enhancement effect.

Features

- Support for:
  - High-definition (1080p60) resolutions
  - Up to 4096 total pixels and 4096 total rows
- Programmable gain for edge directions
- Selectable processor interface:
  - EDK pCore
  - General Purpose Processor
  - Constant Interface
- Support for 8-, 10-, or 12-bit input and output precision
- YCrCb or YUV 444 input and output
- For use with Xilinx CORE Generator™ software 12.4 or later
- Xilinx Streaming Video Interface (XSVI) bus simplifies connecting to other video IP

Applications

- Pre-processing Block for Image Sensors
- Video Surveillance
- Video Conferencing
- Video Capture Devices

LogiCORE IP Facts Table						
Core Specifics						
Supported Device Family(1)	Spartan®-3A DSP, Spartan-6, Virtex®-5, Virtex-6					
Supported User Interfaces	General Processor Interface, EDK PLB 4.6, Constant Interface					
	Resources(2)					**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs**	**DSP48E1s**	**Clock Frequency (MHz)(3)**
Data Width = 8	1008	1163	352	1(36K)+ 1(18K)	1	297
Data Width = 10	1151	1343	377	4(36K)	1	331
Data Width = 12	1268	1523	473	11(36K)	1	241
Provided with Core						
Documentation	Product Specification					
Design Files	Netlists, EDK pCore files, C drivers					
Example Design	Not Provided					
Test Bench	Not Provided					
Constraints File	Not Provided					
Simulation Model	Verilog and VHDL					
Tested Design Tools						
Design Entry Tools	CORE Generator™ tool, Platform Studio (XPS)					
Simulation	ModelSim v6.5c, Xilinx ISim 12.4					
Synthesis Tools	XST 12.4					
Support: Provided by Xilinx, Inc.						

1. For a complete listing of supported devices, see the release notes for this core.
2. Resources listed here are for Virtex-6 devices. For more complete performance data, see Core Resource Utilization and Performance.
3. Performance numbers listed are for Virtex-6 FPGAs. For more complete performance data, see Core Resource Utilization and Performance.

Overview

The edge enhancement core combines the outputs of Sobel and Laplacian operators with the original image to emphasize edge content as shown in Figure 1.

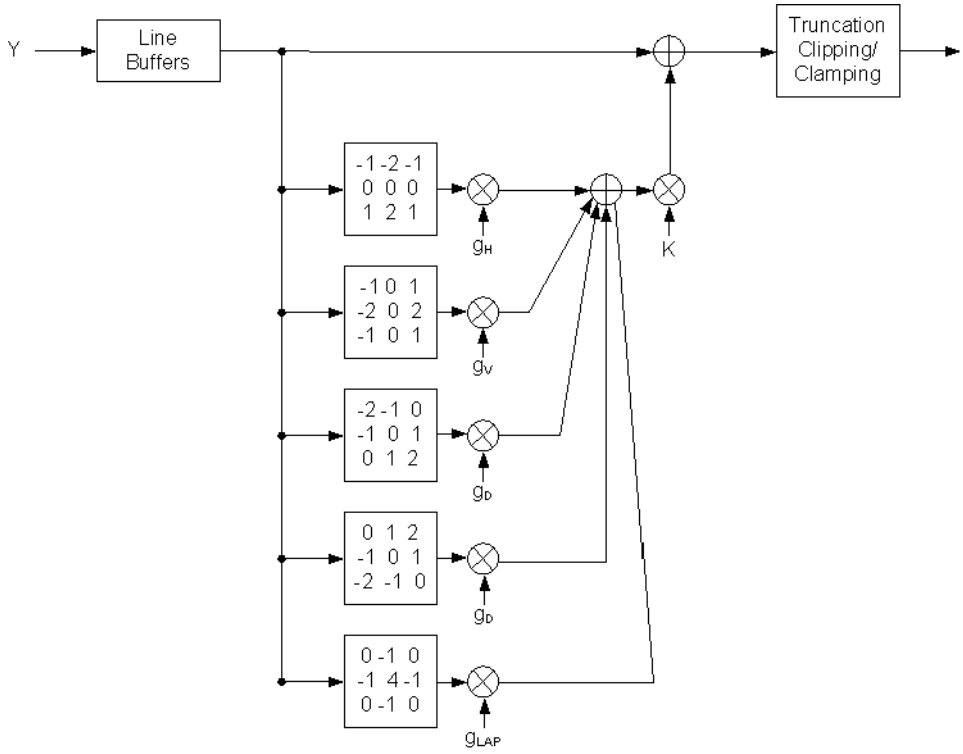

Figure 1: **Image Edge Enhancement**

The Sobel operators are defined in Equations 1, 2, and 3.

$$\text{Horizontal Sobel} = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \textit{Equation 1}$$

$$\text{Vertical Sobel} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \textit{Equation 2}$$

$$\text{Diagonal Sobels} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad \textit{Equation 3}$$

The Laplacian is defined in Equation 4.

$$\text{Laplacian} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{Equation 4}$$

Human visual systems detect the boundary of objects best when they are accompanied by sudden changes in brightness. The edge enhancement core exploits this and enhances only the luminance channel [Ref 1]. This has the added benefit of eliminating color shifts at the boundary of objects, which are common when enhancing the chrominance components by similar methods. The luminance component is processed through the core in two dimensions using two line buffers. The chrominance components are passed through the core with the proper delay to match luminance processing. This core can accept chrominance components represented as signed or unsigned integers with or without the 128 offset.

Defining Gains

The amount and direction of the edge enhancement can be controlled through programmable gains `gH` (horizontal), `gV` (vertical), `gD` (diagonal), and `gLap` (Laplacian). Here, a vertical edge is defined as a feature running from top to bottom of an image. Similarly, a horizontal edge runs from left to right across the image. The diagonal direction covers both upper left to lower right and upper right to lower left diagonals.

Gains can be set to values in the range of 0.0 to 2.0 (see the EDK pCore Interface and General Purpose Processor Interface sections for details). If a particular direction is not desired, that gain can be set to zero to eliminate emphasis in that direction. For example, if vertical edges do not need to be enhanced, the gain `gV` should be set to zero.

Additionally, there is an image content dependent gain, `K`, used to modify the Sobel and Laplacian output. In areas of the image that are smooth and of low contrast, the gain is low to avoid emphasizing noise. This gain is automatically and dynamically calculated by the core on a pixel basis, and it is designed to produce a good compromise between enhancement of features and undesired noise.

If the total gain used [(gH+gV+gD+gLap)*K] exceeds 1.0, clipping and clamping circuitry limits the enhancement of the edge content. Setting the gains with values greater than 1.0 allows over-enhancing the image to produce special effects like embossing.

The over-emphasis of edges may bring out noise at the edge transitions. Consequently, this core may be used in conjunction with noise reduction cores, such as the Image Noise Reduction LogiCORE IP, to improve the results.

Processor Interfaces

The Image Edge Enhancement core supports the following three processor interface options:

- EDK pCore Interface
- General Purpose Processor Interface
- Constant Interface

The processor interfaces provide the system designer with the ability to dynamically control the parameters within the core.

EDK pCore Interface

Many imaging applications include an embedded processor to dynamically control the parameters within an integrated system. CORE Generator software can generate the core with a pCore interface, which can be added to an EDK project as a hardware peripheral. This pCore provides a memory- mapped interface for the programmable registers within the core, which are described in Table 1.

Table 1: **EDK pCore Interface Register Descriptions**

Address Offset (hex)	Register Name	Access Type	Default Value (hex)	Description	
BASEADDR + 0x800	enhance_reg00_control	R/W	0x00000001	Bit 0	Software enable • 0 – Not enabled • 1 – Enabled
				Bit 1	Host processor write done semaphore • 0 – Host processor actively updating registers • 1 – Register update completed by host processor
BASEADDR + 0x804	enhance_reg01_reset	R/W	0x00000000	Bit 0	Software reset • 0 – Not reset • 1 – Reset
BASEADDR + 0x808	enhance_reg02_status	R	0x00000000	Bit 7	Timing locked • 1 – Indicates that the timing module of the core has locked on the input timing signals and is generating stable output timing signals
BASEADDR + 0x80C	enhance_reg03_gain_H	R/W	From GUI	Gain of Horizontal Sobel filter	Allowed values are 0 to 2 in increments of 0.25 represented by four unsigned bits with two integer bits and two fractional bits Bits: Gain value 0000: 0.00 0001: 0.25 0010: 0.50 0011: 0.75 0100: 1.00 0101: 1.25 0110: 1.50 0111: 1.75 1XXX: 2.00
BASEADDR + 0x810	enhance_reg04_gain_V	R/W	From GUI	Gain of Vertical Sobel filter	
BASEADDR + 0x814	enhance_reg05_gain_D	R/W	From GUI	Gain of Diagonal Sobel filters	
BASEADDR + 0x818	enhance_reg06_gain_Lap	R/W	From GUI	Gain of Laplacian filter	

All of the registers are readable, enabling the MicroBlaze™ processor to verify writes or read current values contained within the registers. The default values of some of the registers are defined in the CORE Generator – Graphical User Interface section.

This core supports an enable/disable function. When disabled, the normal operation of the hardware is halted and video signals are not propagated. This function is controlled by setting Software Enable, bit 0 of `enhance_reg00_control` register, to 0. The default value of Software Enable is 1 (enabled).

The in-system reset of the core is controlled by asserting `enhance_reg01_reset` (bit 0), which returns the gains to their default values, specified through the Graphical User Interface when the core is instantiated. The core control signals and output are forced to 0 until the software reset bit is deasserted.

The gain registers are double buffered in hardware to ensure no image tearing happens if the gain values are modified in the active area of a frame. This double buffering provides system control that is more flexible and easier to use by decoupling the register updates from the blanking period, allowing software a much larger window in which to update the parameter values. The updated values for the gain registers are latched into the shadow registers immediately after writing, while the actual gains used are stored in the working registers. The rising edge of `vblank_in` triggers the values from the shadow registers to be copied to the working registers when bit 1 of `enhance_reg00_control` is set to 1. This semaphore bit helps to prevent changing the gains mid-frame.

Any reads of registers return the values stored in the shadow registers.

Figure 2 shows a software flow diagram for updating registers during the operation of the core.

See the EDK pCore Interface section of Core Symbol and Port Descriptions.

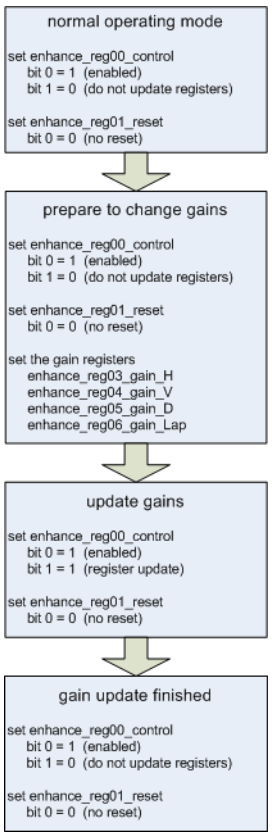

Figure 2: **Image Edge Enhancement Programming Flow Chart**

Programmer's Guide

The software API is provided to allow easy access to the Image Edge Enhancement pCore's registers defined in Table 1. To utilize the API functions provided, the following two header files must be included in the user C code:

```
#include "enhance.h"
#include "xparameters.h"
```

The hardware settings of your system, including the base address of your Image Edge Enhancement core, are defined in the `xparameters.h` file. The `enhance.h` file contains the macro function definitions for controlling the Image Edge Enhancement pCore.

For examples on API function calls and integration into a user application, the drivers subdirectory of the pCore contains a file, `example.c`, in the `enhance_v2_00_a/example` subfolder. This file is a sample C program that demonstrates how to use the Image Edge Enhancement pCore API.

EDK pCore API Functions

This section describes the functions included in the C driver (`enhance.c` and `enhance.h`) generated for the EDK pCore API.

ENHANCE_Enable(uint32 BaseAddress);

- This macro enables an Image Edge Enhancement instance.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`).

ENHANCE_Disable(uint32 BaseAddress);

- This macro disables an Image Edge Enhancement instance.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`).

ENHANCE_Reset(uint32 BaseAddress);

- This macro resets an Image Edge Enhancement instance. This reset effects the core immediately, and may cause image tearing. Reset affects the gain registers, forces `video_data_out` to 0, and forces timing signal outputs to their reset state until ENHANCE_ClearReset() is called.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`)

ENHANCE_ClearReset(uint32 BaseAddress);

- This macro clears the reset flag of the core, which allows it to re-sync with the input video stream and return to normal operation.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`).

Reading and Writing pCore Registers

Each software register defined in Table 1 has a constant defined in `enhance.h` that is set to the offset for that register. Reading a value from a register uses the base address and offset for the register:

```
Xuint32 value = ENHANCE_ReadReg(XPAR_ENHANCE_0_BASEADDR, ENHANCE_REG03_GAIN_H);
```

This macro returns the 32-bit unsigned integer value of the register. The definition of this macro is

ENHANCE_ReadReg(uint32 BaseAddress, uint32 RegOffset)

- Read the given register.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`).
- RegOffset is the register offset of the register (defined in Table 1).

To write to a register, use the ENHANCE_WriteReg() function using the base address of the Image Edge Enhancement pCore instance (from `xparameters.h`), the offset of the desired register, and the data to write. For example:

```
ENHANCE_WriteReg(XPAR_ENHANCE_0_BASEADDR, ENHANCE_REG03_GAIN_H, 1);
```

The definition of this macro is:

ENHANCE_WriteReg(uint32 BaseAddress, uint32 RegOffset, uint32 Data)

- Write the given register.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`).
- RegOffset is the register offset of the register (defined in Table 1).
- Data is the 32-bit value to write to the register.

ENHANCE_RegUpdateEnable(uint32 BaseAddress);

- Calling RegUpdateEnable causes the Image Edge Enhancement to start using the updated gain values on the next rising edge of `VBlank_in`. The user must manually disable the register update after a sufficient amount of time to prevent continuous updates.
- This function only works when the Image Edge Enhancement core is enabled.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`)

ENHANCE_RegUpdateDisable(uint32 BaseAddress);

- Disabling the Register Update prevents the Image Edge Enhancement gain registers from updating. Xilinx recommends that the Register Update be disabled while writing to the registers in the core, until the write operation is complete. While disabled, writes to the registers are stored, but do not affect the core's behavior.
- This function only works when the Image Edge Enhancement core is enabled.
- BaseAddress is the Xilinx EDK base address of the Image Edge Enhancement core (from `xparameters.h`)

General Purpose Processor Interface

The General Purpose Processor Interface exposes the gain registers as ports. The General Purpose Processor Interface is provided as an option to design a system with a user-defined bus interface (decoding logic and register banks) to an arbitrary processor.

The gain ports have the double-buffer control mechanism described in the previous section to prevent tearing. However, an external register bank (shadow register bank) has to be supplied by the user-defined bus interface. Values from this register bank (external to the Image Edge Enhancement core) are copied over to the internal registers at the rising edge of `vblank_in` when bit 1 of the `enhance_reg00_control` register is set to '1'.

See the General Purpose Processor Interface section of Core Symbol and Port Descriptions.

Constant Interface

The Constant Interface does not provide an option for the gains to be changed in system. Also, there is no processor interface and the core is not programmable, but can be reset and enabled/disabled using the `SCLR` and `CE` ports. The ports for the Constant Interface are described in detail in the Constant Interface section of Core Symbol and Port Descriptions.

CORE Generator – Graphical User Interface

The Image Edge Enhancement core is configured to meet user-specific needs through the CORE Generator graphical user interface (GUI). This section provides a quick reference to the parameters that can be configured at generation time. Figure 3 shows the main Image Edge Enhancement screen.

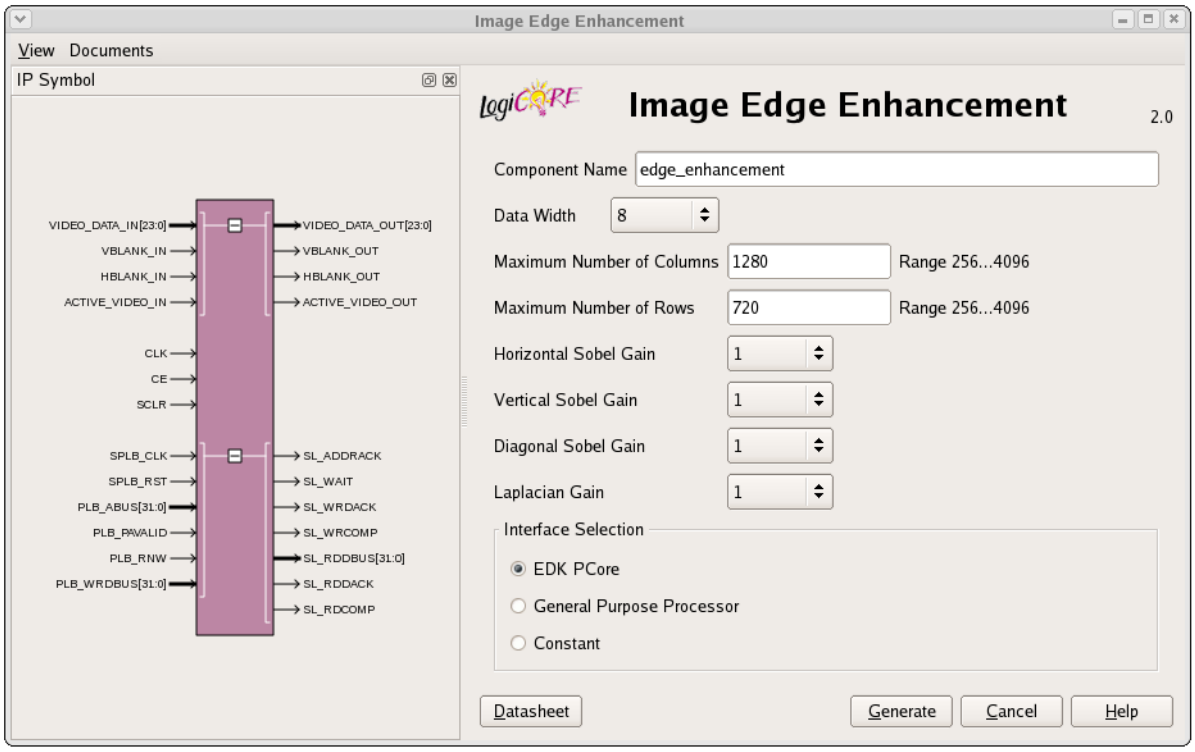

Figure 3: **Image Edge Enhancement Main Screen**

The GUI displays a representation of the IP symbol on the left side, and the parameter assignments on the right side, which are described as follows:

- **Component Name:** The component name is used as the base name of output files generated for the module. Names must begin with a letter and must be composed from characters: a to z, 0 to 9 and "_".
- **Data Width (WIDTH):** Specifies the bit width of the input channel for each component. The allowed values are 8, 10, and 12.
- **Maximum Number of Columns (MAX_COLS)**: Specifies the maximum number of columns that can be processed by the core. Permitted values are from 256 to 4096. Specifying this value is necessary to establish the internal widths of counters and control-logic components as well as the depth of line buffers. Using a tight upper-bound on possible values of `MAX_COLS` results in optimal block RAM usage. However, feeding the configured Image Edge Enhancement instance timing signals that violate the `MAX_COLS` constraint leads to data and output timing signal corruption.
- **Maximum Number of Rows (MAX_ROWS):** Specifies the maximum number of rows that can be processed by the core. Permitted values are from 256 to 4096. Specifying this value is necessary to establish the internal widths of counters and control-logic components. Feeding the configured Image Edge Enhancement instance timing signals that violate the `MAX_ROWS` constraint leads to data and output timing signal corruption.
- **Horizontal Sobel, Vertical Sobel, Diagonal Sobel, and Laplacian Gains:** Specifies the default gain to be applied for each filter. The possible values are 0.0 to 2.0 in increments of 0.25.

- **Interface Selection:** As described in the previous sections, this option allows for the configuration of two different interfaces for the core.
  - **EDK pCore Interface:** CORE Generator software generates a pCore that can be easily imported into an EDK project as a hardware peripheral, and gains can be programmed via a register. Double buffering is used to eliminate tearing of output images. See the EDK pCore Interface section of Processor Interfaces.
  - **General Purpose Processor Interface:** CORE Generator software generates a set of ports to be used to program the core. See the General Purpose Processor Interface section of Processor Interfaces.
  - **Constant Interface:** The gains are constant, and therefore no programming is necessary. The constant value is set in the GUI.

Core Symbol and Port Descriptions

The Image Edge Enhancement core can be configured with three different interface options, each resulting in a slightly different set of ports. The Image Edge Enhancement IP core uses a set of signals that is common to all of the Xilinx Video IP cores called the Xilinx Streaming Video Interface (XSVI). The XSVI signals are common to all interface options and are described in Table 2.

Xilinx Streaming Video Interface

The Xilinx Streaming Video Interface (XSVI) is a set of signals common to all of the Xilinx video cores used to stream video data between IP cores. XSVI is also defined as an Embedded Development Kit (EDK) bus type so that the tool can automatically create input and output connections to the core. This definition is embedded in the pCore interface provided with the IP, and it allows an easy way to cascade connections of Xilinx Video Cores. The Image Edge Enhancement IP core uses the following subset of the XSVI signals:.

The Image Edge Enhancement IP Core uses the following sub-set of the XSVI signals:

- video_data
- vblank
- hblank
- active_video

Other XSVI signals on the XSVI input bus, such as video_clk, vsync, hsync, field_id, and active_chr do not affect the function of this core.

Note: These signals are neither propagated, nor driven on the XSVI output of this core.

The following is an example EDK Microprocessor Peripheral Definition (.MPD) file definition:

Input Side:

```
BUS_INTERFACE BUS = XSVI_ENHANCE_IN, BUS_TYPE = TARGET, BUS_STD = XSVI

PORT hblank_i       =hblank,      DIR=I,                          BUS=XSVI_ENHANCE_IN

PORT vblank_i       =vblank,      DIR=I,                          BUS=XSVI_ENHANCE_IN

PORT active_video_i =active_video,DIR=I,                          BUS=XSVI_ENHANCE_IN

PORT video_data_i   =video_data,  DIR=I,VEC=[C_DATA_WIDTH-1:0], BUS=XSVI_ENHANCE_IN
```

Output Side:

```
BUS_INTERFACE BUS = XSVI_ENHANCE_OUT, BUS_TYPE = INITIATOR,  BUS_STD = XSVI
PORT hblank_o       =hblank,      DIR=I,                          BUS=XSVI_ENHANCE_OUT
PORT vblank_o       =vblank,      DIR=I,                          BUS=XSVI_ENHANCE_OUT
PORT active_video_o =active_video,DIR=I,                          BUS=XSVI_ENHANCE_OUT
PORT video_data_o   =video_data, DIR=I,VEC=[3*C_DATA_WIDTH1:0],BUS=XSVI_ENHANCE_OUT
```

The Image Edge Enhancement IP core is fully synchronous to the core clock, clk. Consequently, the input XSVI bus is expected to be synchronous to the input clock, clk. Similarly, to avoid clock resampling issues, the output XSVI bus for this IP is synchronous to the core clock, clk. The video_clk signals of the input and output XSVI buses are not used.

Constant Interface

The Constant Interface has no ports other than the Xilinx Streaming Video Interface, clk, ce, and sclr, as this interface does not provide additional programmability. The Constant Interface Core Symbol is shown in Figure 4 and described in Table 2. The Xilinx Streaming Video Interface is a set of signals that is common in all interface options.

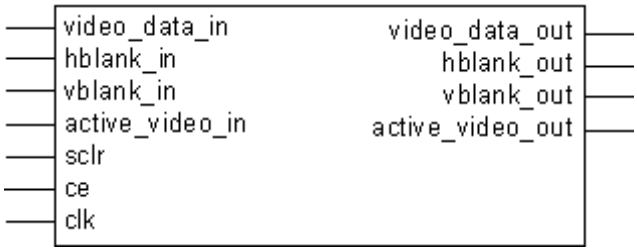

Figure 4: **Core Symbol for the Constant Interface**

Table 2 contains the Constant Interface port descriptions. Detailed descriptions of the ports are provided following the table.

Table 2: **Port Descriptions for the Constant Interface**

Port Name	Port Width	Direction	Description
video_data_in	*3*WIDTH*	*IN*	Data input bus
hblank_in	*1*	*IN*	Horizontal blanking input
vblank_in	*1*	*IN*	Vertical blanking input
active_video_in	*1*	*IN*	Active video signal input
video_data_out	*3*WIDTH*	*OUT*	Data output bus
hblank_out	*1*	*OUT*	Horizontal blanking output
vblank_out	*1*	*OUT*	Vertical blanking output
active_video_out	*1*	*OUT*	Active video signal output
clk	*1*	*IN*	Rising-edge clock
ce	*1*	*IN*	Clock enable (active high)
sclr	*1*	*IN*	Synchronous clear – reset (active high)

- **video_data_in:** This bus contains the luminance and chrominance inputs in the following order from MSB to LSB [Cb ; Cr : Y] or [U ; V; Y]. Each component is expected in `WIDTH` bits wide unsigned integer representation.

Table 3:

Bits	*3WIDTH-1:2WIDTH*	*2WIDTH-1:WIDTH*	*WIDTH-1:0*
Video Data Signals	Cb or U	Cr or V	Y

- **hblank_in:** The `hblank_in` signal conveys information about the blank/non-blank regions of video scan lines.
- **vblank_in:** The `vblank_in` signal conveys information about the blank/non-blank regions of video frames, and is used by the Image Edge Enhancement core to detect the end of a frame, when user registers can be copied to active registers to avoid visual tearing of the image.
- **active_video_in:** The `active_video_in` signal is high when valid data is presented at the input. Input data to the core, `video_data_in`, is ignored when `active_video_in` is low.
- **clk - clock:** Master clock in the design, synchronous with, or identical to, the video clock.
- **ce - clock enable:** Pulling `CE` low suspends all operations within the core. Outputs are held, and no input signals are sampled except for reset (`SCLR` takes precedence over `CE`).
- **sclr - synchronous clear:** Pulling `SCLR` high results in resetting all output pins to zero or their default values. Internal registers within the XtremeDSP™ slice and D-flip-flops are cleared.
- **video_data_out:** This bus contains the luminance and chrominance outputs in the following order from MSB to LSB [Cb ; Cr : Y] or [U ; V; Y]. Each component is expected in `WIDTH` bits wide unsigned integer representation.

Table 4:

Bits	*3WIDTH-1:2WIDTH*	*2WIDTH-1:WIDTH*	*WIDTH-1:0*
Video Data Signals	Cb or U	Cr or V	Y

- **hblank_out** and **vblank_out:** The corresponding input signals are delayed so blanking outputs are in phase with the video data output, maintaining the integrity of the video stream.
- **active_video_out:** The `active_video_out` signal is high when valid data is present at the output. When `active_video_out` is low, `video_data_out` is not valid even if it is non-zero.

EDK pCore Interface

The EDK pCore Interface generates Processor Local Bus (PLB4.6) interface ports in addition to the sclr and Xilinx Streaming Video Signals. The PLB bus signals are automatically connected when the generated pCore is inserted into an EDK project. The Core Symbol for the EDK pCore Interface is shown in Figure 5. The Xilinx Streaming Video Interface is described in the previous section (Table 2). For more information on the PLB bus signals, see Processor Local Bus (PLB) v4.6 [Ref 2].

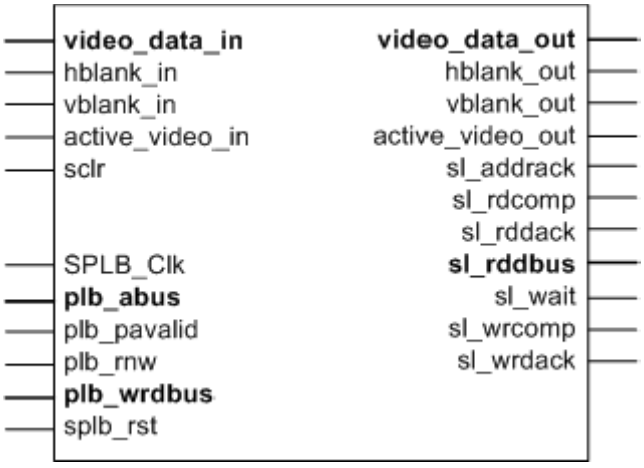

Figure 5: **Core Symbol for the EDK pCore Interface**

General Purpose Processor Interface

The General Purpose Processor Interface exposes the gains as a port. The Core Symbol for the General Purpose Processor Interface is shown in Figure 6. The Xilinx Streaming Video Interface is described in the previous section (Table 2). The ports are described in Table 5.

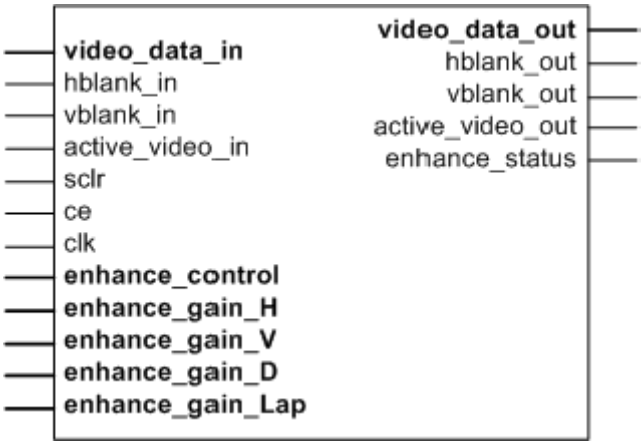

Figure 6: **Core Symbol for the General Purpose Processor Interface**

Table 5: **Optional Ports for the General Purpose Processor Interface**

Port Name	Port Width	Direction	Description
enhance_control	2	IN	Bit 0: Software enable Bit 1: Host processor write done semaphore • 0 – Host processor actively updating registers • 1 – Register update completed by host processor
enhance_gain_H	4	IN	Gain of Horizontal Sobel filter Possible bit values: 0000 to 1000 in increments of 0001 Gain is represented as four unsigned bits with two integer bits and two fractional bits (1.0 is represented as 0100)
enhance_gain_V	4	IN	Gain of Vertical Sobel filter Possible bit values: 0000 to 1000 in increments of 0001 Gain is represented as four unsigned bits with two integer bits and two fractional bits (1.0 is represented as 0100)
enhance_gain_D	4	IN	Gain of Diagonal Sobel filters Possible bit values: 0000 to 1000 in increments of 0001 Gain is represented as four unsigned bits with two integer bits and two fractional bits (1.0 is represented as 0100)
enhance_gain_Lap	4	IN	Gain of Laplacian filter Possible bit values: 0000 to 1000 in increments of 0001 Gain is represented as four unsigned bits with two integer bits and two fractional bits (1.0 is represented as 0100)
enhance_status	16	OUT	Status Bit 7: Timing Locked ‘1’ indicates that the timing module of the core has locked on the input timing signals and is generating stable output timing signals

Control Signals and Timing

Figure 7 shows a typical timing example with two frames of data.

The propagation delay of the Image Edge Enhancement core is one full scan line and 19 video clock cycles. The output timing signals (vblank_out, hblank_out, and active_video_out) are delayed appropriately so that the output video data is framed correctly by the timing signals.

Deasserting CE suspends processing, which may be useful for data-throttling, to temporarily cease processing of a video stream to match the delay of other processing components.

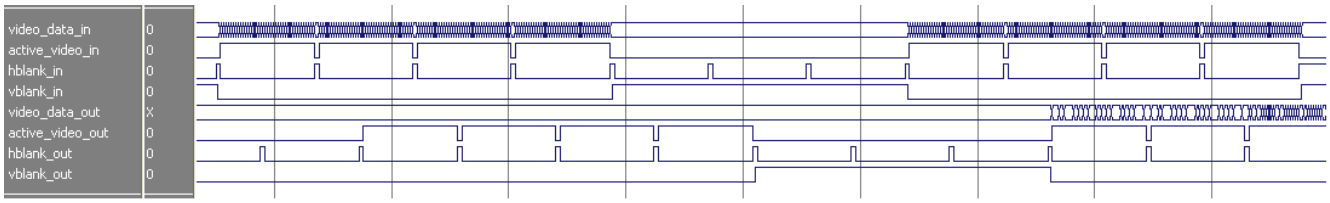

Figure 7: **Timing Example**

The control signals vblank_out, hblank_out, and active_video_out are created using a timing detector and generator within the core. The internal timing module assumes the following:

- One horizontal blanking period per row
- One vertical blanking period per frame
- A minimum active frame size of four rows and eight columns
- A minimum horizontal blanking period of two columns
- A minimum vertical blanking period of three rows

During the detection of the timing control signals, the core cannot guarantee the correct video data output. Consequently, the data output, video_data_out, of the first frame of data is set to zero even though active_video_out is high.

When SCLR is asserted, all data and control signal outputs are forced to zero. If the input control signal was high at the time SCLR was asserted, the corresponding output control signal goes low and stays low until the next expected rising edge.

Core Resource Utilization and Performance

The information in Table 6 to Table 9 is a guideline to the resource utilization of the Image Edge Enhancement core for Spartan-6, Virtex-6, Virtex-5, and Spartan-3ADSP FPGA families. The General Purpose Processor interface was selected. This core does not use any dedicated I/O or clock resources. The design was tested using Xilinx ISE® v12.4 tools with area constraints and default tool options.

For an accurate measure of the usage of device resources (for example, block RAMs, flip-flops, and LUTs) for a particular instance, click **View Resource Utilization** in the CORE Generator GUI after generating the core

Table 6: **Resource Utilization and Target Speed for Spartan-6 - xc6slx150-2fgg676**

Data Width	Maximum Number of Columns and Rows	FFs	LUTs	Slices	Block RAMs	DSP48A1s	Clock Frequency (MHz)
8	1024	1195	990	435	3(16K)	1	162
8	2200	1260	1085	441	9(16K)	1	198
10	1024	1402	1137	508	7(16K)+1(8K)	2	150
10	2200	1466	1227	482	14(16K) + 1(8K)	2	155
12	1024	1552	1247	523	21(16K)	2	154
12	2200	1617	1366	535	30(16K)	2	157

1. Speedfile: PRODUCTION 1.13b 2010-11-15

Table 7: **Resource Utilization and Target Speed for Virtex-6 - xc6vsx315t-1ff1156**

Data Width	Maximum Number of Columns and Rows	FFs	LUTs	Slices	Block RAMs	DSP48E1s	Clock Frequency (MHz)
8	1024	1163	1008	352	1(36K)+1(18K)	1	297
8	2200	1227	1099	393	4(36K)+1(18K)	1	250
10	1024	1343	1151	377	4(36K)	1	331
10	2200	1407	1211	458	7(36K)	1	230
12	1024	1523	1268	473	11(36K)	1	241
12	2200	1587	1356	509	15(36K)	1	254

1. Speedfile: PRODUCTION 1.11d 2010-11-15

Table 8: **Resource Utilization and Target Speed for Spartan-3A DSP - xc3sd3400a-5fg676**

Data Width	Maximum Number of Columns and Rows	FFs	LUTs	Slices	Block RAMs	DSP48As	Clock Frequency (MHz)
8	1024	1082	874	832	3	1	154
8	2200	1126	915	878	9	1	153
10	1024	1248	1037	887	8	2	150
10	2200	1292	1078	988	15	2	158
12	1024	1378	1171	921	21	2	153
12	2200	1422	1215	1082	30	2	153

1. Speedfile: PRODUCTION 1.33 2010-11-15

Table 9: **Resource Utilization and Target Speed for Virtex-5 - xc5vsx50t-1ff665**

Data Width	Maximum Number of Columns and Rows	FFs	LUTs	Slices	Block RAMs	DSP48Es	Clock Frequency (MHz)
8	1024	1062	957	470	1(36K)+1(18K)	1	256
8	2200	1126	1030	470	4(36K)+1(18K)	1	270
10	1024	1229	1093	511	4(36K)	1	232
10	2200	1292	1168	529	7(36K)	1	230
12	1024	1399	1250	565	11(36K)	1	261
12	2200	1458	1324	580	15(36K)	1	227

1. Speedfile: PRODUCTION 1.33 2010-11-15

Known Issues

For for the latest Known Issues see XTP025.

References

1. Sharpening Spatial Filters, “Digital Image Processing,” by Rafael Gonzales and Richard Woods, Third Edition. Pearson Education, Prentice Hall 2008
2. Processor Local Bus (PLB) v4.6

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

License Options

The Image Edge Enhancement core provides the following three licensing options:

- Simulation Only
- Full System Hardware Evaluation
- Full

After installing the required Xilinx ISE software and IP Service Packs, choose a license option.

Simulation Only

The Simulation Only Evaluation license key is provided with the Xilinx CORE Generator tool. This key lets you assess core functionality with either the example design provided with the Image Edge Enhancement core, or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically generated HDL structural model.)

No action is required to obtain the Simulation Only Evaluation license key; it is provided by default with the Xilinx CORE Generator software.

Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place-and-route the design, evaluate timing, and perform functional simulation of the Image Edge Enhancement core using the example design and demonstration test bench provided with the core.

In addition, the license key lets you generate a bitstream from the placed and routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before timing out (ceasing to function), at which time it can be reactivated by reconfiguring the device.

To obtain a Full System Hardware Evaluation license, do the following:

1. Navigate to the product page for this core.
2. Click Evaluate.
3. Follow the instructions to install the required Xilinx ISE software and IP Service Packs.

Full

The Full license key is available when you purchase the core and provides full access to all core functionality both in simulation and in hardware, including:

- Functional simulation support
- Full implementation support including place and route and bitstream generation
- Full functionality in the programmed device with no time outs

To obtain a Full license key, you must purchase a license for the core. Click on the "Order" link on the Xilinx.com IP core product page for information on purchasing a license for this core. After doing so, click the "How do I generate a license key to activate this core?" link on the Xilinx.com IP core product page for further instructions.

Installing Your License File

The Simulation Only Evaluation license key is provided with the ISE CORE Generator system and does not require installation of an additional license file. For the Full System Hardware Evaluation license and the Full license, an email will be sent to you containing instructions for installing your license file. Additional details about IP license key installation can be found in the ISE Design Suite Installation, Licensing and Release Notes document.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
12/02/09	1.0	Initial Xilinx release.
07/23/10	1.1	Fixed CR 54061 by adding Xilinx Streaming Video Interface (XSVI) information.
12/14/2010	2.0	Updated for core version 2.0.

Notice of Disclaimer