LogiCORE IP Defective Pixel Correction v3.0

DS721 March 1, 2011 **Product Specification**

The Xilinx LogiCORE™ IP Defective Pixel Correction performs real-time detection and correction of defective pixels in a camera image sensor array.

Features

- Programmable thresholds
- Selectable processor interface
    - EDK pCore
    - General Purpose Processor
- Up to 4k x 4k resolutions supported
- 8-, 10-, and, 12-bit input and output precision
- Temporal filtering without using an external frame buffer
- For use with Xilinx CORE Generator™ software v13.1 or later

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				
	Resources[2]				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs**	**Max. Freq.**
Data Width=8	1,698	1,503	1	2 (36) / 1 (18)	307
Data Width=10	1,868	1,691	1	2 (36) / 2 (18)	298
Data Width=12	2,057	1,879	1	2 (36) / 2 (18)	307
Provided with Core					
Documentation	Product Specification				
Design Files	Netlists, EDK pCore files, C drivers				
Example Design	Not Provided				
Test Bench	VHDL				
Constraints File	Not Provided				
Simulation Models	VHDL or Verilog Structural, C				
Tested Design Tools					
Design Entry Tools	CORE Generator™ tool, Platform Studio (XPS)				
Simulation	ModelSim v6.6c, Xilinx® ISim 13.1				
Synthesis Tools	XST 13.1				
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 (xc6vsx315t-1ff1156C), using the General Purpose Processor interface, and using speed file: PRODUCTION 1.13 2011-02-03.

Overview

An image sensor may have a certain number of defective pixels that may be the result of manufacturing faults, failures during normal operation, or variations in pixel voltage levels based on temperature or exposure. A wide class of pixel defects may be characterized as: dead (always low), hot (always high), or stuck (to a certain value). These anomalies can further be characterized as static (always present) or dynamic (as a function of exposure or temperature).

The Xilinx Defective Pixel Correction solution distinguishes between large stationary areas, which are likely to be non-changing parts of the image, and singular outliers, which are likely to be defective pixels. The Xilinx Defective Pixel Correction solution compares a pixel in the raw, Bayer sub-sampled domain to its neighboring, same color pixel values and keeps track of pixels that are sufficiently different from their neighbors. If the values of tracked outlier pixels stay in a predefined range for a predefined number of frames, then the tracked pixels are considered defective, and are replaced with values interpolated from neighboring pixels.

Spatial filtering first identifies potential defective pixels, and at the same time eliminates pixels that blend into their local neighborhoods, and therefore do not need to be substituted even if they are defective. Spatial filtering reduces the number of pixels, along with the amount of information, that needs to be stored for temporal filtering, therefore facilitating spatio-temporal filtering in embedded systems with limited or no access to external memory.

Selection of Threshold Values

The Xilinx Defective Pixel Correction solution provides three controls to influence the identification and monitoring of defective pixels.

Threshold value THRESH_SPATIAL_VAR defines how different a pixel needs to be from the surrounding pixels to be classified as an outlier. A practical value of $2^{\text{DATA\_WIDTH-5}}$ identifies pixels that visually stand out from their surroundings. A higher threshold value for THRESH_SPATIAL_VAR results in a lower number of outlier candidates and slower convergence time for identifying all outliers, but at the same time returns fewer false positives. If heuristics for the total number of outliers (M) are known, a feedback mechanism can be implemented that tunes THRESH_SPATIAL_VAR so that the number of outlier pixels identified, num_candidates, approximates M.

Threshold value THRESH_TEMPORAL_VAR, defines the range a pixel value needs to stay in to be classified as stuck. The lower the value, the lower the chance that slowly varying pixels get characterized as stuck. However, if the sensor image is loaded with noise, or blooming may modify the readout values of dead pixels, THRESH_TEMPORAL_VAR may need to be increased to identify all stuck pixels. As a practical value for THRESH_TEMPORAL_VAR, the square root of the maximum pixel value is suggested.

Threshold value, THRESH_PIXEL_AGE, defines the number of frames presumed outliers have to hold their values within THRESH_TEMPORAL_VAR range before an outlier pixel is considered defective, and replacement (interpolation) of the pixels begin. The higher the value of THRESH_PIXEL_AGE, the less flickering due to incorrect defective pixel correction the algorithm produces, but also the longer it takes for the algorithm to converge and start replacing defective pixels. Values in the range of several thousands allow virtually no flickering while identifying outliers within minutes.

Processor Interfaces

The Defective Pixel Correction core supports the following two processor interface options:

- EDK pCore Interface
- General Purpose Processor 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 have an embedded processor which can dynamically control the parameters within the core. The user can select an EDK pCore interface, which creates a pCore that 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 + 0x000	dpc_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 + 0x004	dpc_reg01_reset	R/W	0x00000000	Bit 0	Software reset • 0 – Not reset • 1 – Reset
BASEADDR + 0x008	dpc_reg02_status	R	0x00000001	Bit 0	FIFO Empty
				Bit 1	FIFO Full
				Bit 2-6	Reserved
				Bit 7	Timing lock output '1' indicates that the timing module of the core has locked on the input timing signals and is generating stable output timing signals
BASEADDR + 0x00C	dpc_reg03_thresh_temporal_var	R/W	2**(DATA_WIDTH-7)	Allowed inter-frame variance of defective pixels	
BASEADDR + 0x010	dpc_reg04_thresh_spatial_var	R/W	0x0000199A	Allowed spatial variance beyond which a pixel is characterized as an outlier	
BASEADDR + 0x014	dpc_reg05_thresh_pixel_age	R/W	0x000004B0	Number of frames an outlier pixel has to keep its value within the range specified	
BASEADDR + 0x018	dpc_reg06_num_candidates	R	0x00000000	Total number of potential defective pixel candidates stored in FIFO (in previous frame)	
BASEADDR + 0x01C	dpc_reg07_num_defective	R	0x00000000	Total number of pixels being actively interpolated (in previous frame)	

All of the registers are readable, enabling you to verify writes or read back current values.

The core has a feature that allows it to be enabled or disabled. This halts the operation of the core by blocking the propagation of all video signals. This function is controlled by setting the Software Enable, bit 0 of `dpc_reg00_control` register, to 0; the default value of Software Enable is 1 (enabled).

The core can be effectively reset in-system by asserting `dpc_reg01_reset` (bit 0), which resets the timing and returns the thresholds to their default values. The core control signals and output are forced to 0 until the software reset bit is deasserted.

All registers other than the `dpc_reg00_control`, `dpc_reg01_reset`, and `dpc_reg02_status` registers are double-buffered in hardware, to ensure no image tearing happens if the threshold values are modified in the active area of a frame. This double-buffering provides a more flexible and easier-to-use core because it decouples the register updates from the blanking period, allowing software a much larger window with which to update the parameter values. The updated values for the threshold registers are latched into the shadow registers immediately after writing them, while the actual thresholds used are stored in the working registers.

Any reads of registers during operation will return the values stored in the shadow 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 `dpc_reg00_control` is set to 1. This semaphore bit helps to prevent partially updated shadow registers from being copied over to the working registers.

Figure 1 shows a software flow diagram for updating registers during the operation of the core.

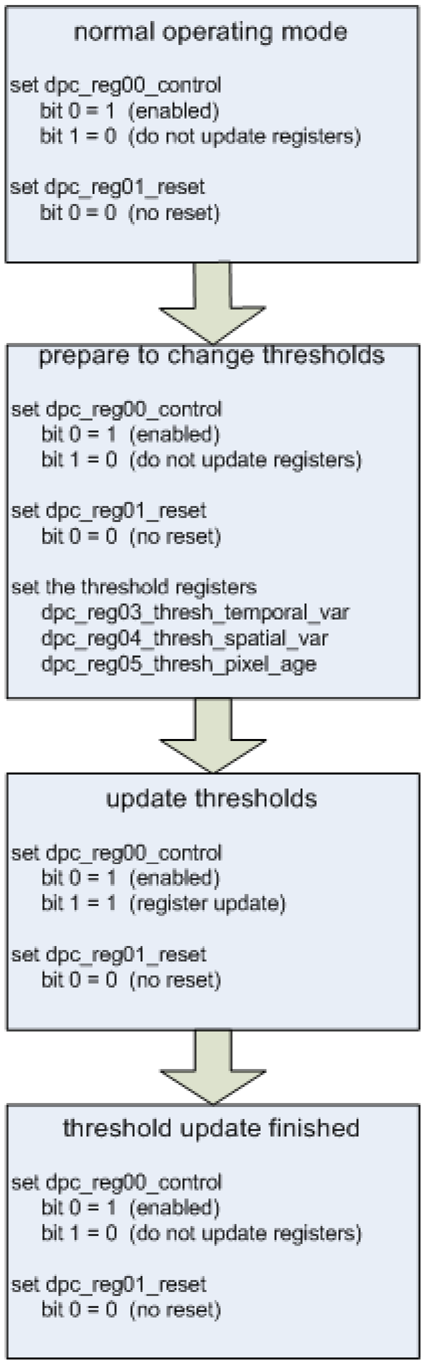


Figure 1: **Defective Pixel Correction Programming Flow Chart**

See the clk - clock: Master clock in the design, synchronous with, or identical to the video clock. section of Core Symbol and Port Descriptions

Programmer's Guide

A software API is provided to allow easy access to the Defective Pixel Correction 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 "dpc.h"
#include "xparameters.h"
```

The hardware settings of your system, including the base address of your Defective Pixel Correction core, are defined in the `xparameters.h` file. The `dpc.h` file contains the macro function definitions for controlling the Defective Pixel Correction 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 `dpc_v3_00_a/example` subfolder. This file is a sample C program that demonstrates how to use the Defective Pixel Correction pCore API.

EDK pCore API Functions

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

DPC_Enable(uint32 BaseAddress);

- This macro enables a Defective Pixel Correction instance.
- BaseAddress is the Xilinx EDK base address of the Defective Pixel Correction core (from `xparameters.h`).

DPC_Disable(uint32 BaseAddress);

- This macro disables a Defective Pixel Correction instance.
- BaseAddress is the Xilinx EDK base address of the Defective Pixel Correction core (from `xparameters.h`).

DPC_Reset(uint32 BaseAddress);

- This macro resets a Defective Pixel Correction instance. This reset effects the core immediately, and may cause image tearing.
- Reset affects the threshold registers, forces `video_data_out` to 0, and forces timing signal outputs to their reset state until DPC_ClearReset() is called.
- BaseAddress is the Xilinx EDK base address of the Defective Pixel Correction core (from `xparameters.h`)

DPC_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 Defective Pixel Correction core (from `xparameters.h`).

Reading and writing pCore Registers

Each software register defined in Table 1 has a constant defined in `dpc.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 = DPC_ReadReg(XPAR_DPC_0_BASEADDR, DPC_REG03_THRESH_TEMPORAL_VAR);
```

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

DPC_ReadReg(uint32 BaseAddress, uint32 RegOffset)

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

To write to a register, use the DPC_WriteReg() function using the base address of the Defective Pixel Correction pCore instance (from `xparameters.h`), the offset of the desired register, and the data to write. For example:

```
DPC_WriteReg(XPAR_DPC_0_BASEADDR, DPC_REG03_THRESH_TEMPORAL_VAR, 1);
```

The definition of this macro is:

DPC_WriteReg(uint32 BaseAddress, uint32 RegOffset, uint32 Data)

- Write the given register.
- BaseAddress is the Xilinx EDK base address of the Defective Pixel Correction 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.

DPC_RegUpdateEnable(uint32 BaseAddress);

- Calling RegUpdateEnable causes the Defective Pixel Correction to start using the updated threshold 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 Defective Pixel Correction core is enabled.
- BaseAddress is the Xilinx EDK base address of the Defective Pixel Correction core (from `xparameters.h`)

DPC_RegUpdateDisable(uint32 BaseAddress);

- Disabling the Register Update prevents the Defective Pixel Correction threshold registers from updating. It is recommended 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 Defective Pixel Correction core is enabled.
- BaseAddress is the Xilinx EDK base address of the Defective Pixel Correction core (from `xparameters.h`)

General Purpose Processor Interface

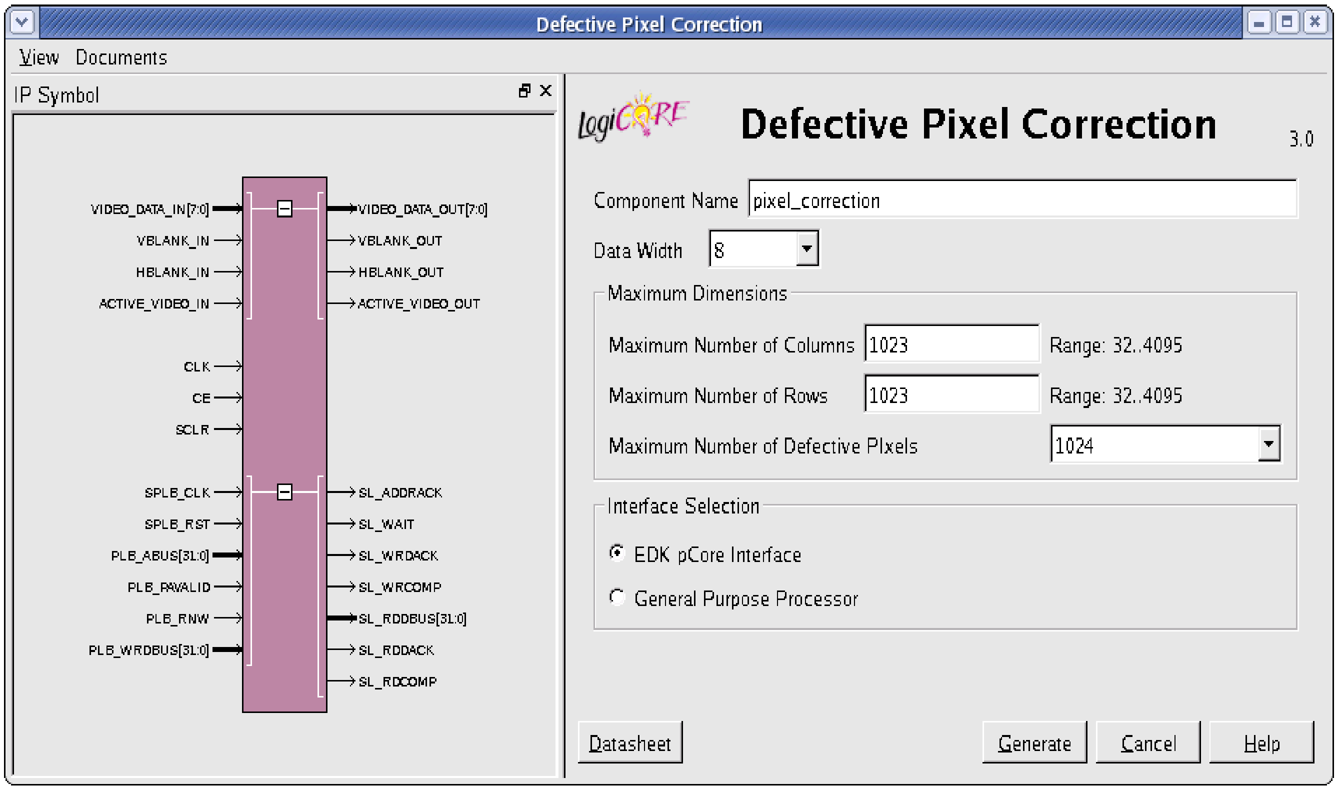
The General Purpose Processor Interface exposes the thresholds and control registers as ports. This option is very useful for users designing a system with a user-defined bus interface (decoding logic and register banks) to an arbitrary processor.

The threshold ports have the double-buffer control mechanism described in the previous section to prevent tearing or committing partially updated port values. However, the first set of registers (shadow register bank) has to be supplied by the user-defined bus interface. Values from this register bank (external to the Defective Pixel Correction core) are copied over to the internal registers at the rising edge of `vblank_in` when bit 1 of the `control` is set to 1.

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

CORE Generator – Graphical User Interface

The Defective Pixel Correction core is easily configured to meet the user's 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 2 shows the main Defective Pixel Correction screen.



Component Name: pixel_correction

Data Width: 8

Maximum Dimensions

Maximum Number of Columns: 1023 Range: 32..4095

Maximum Number of Rows: 1023 Range: 32..4095

Maximum Number of Defective PIxels: 1024

Interface Selection

[✓ EDK pCore Interface]

General Purpose Processor

Figure 2: **Defective Pixel Correction 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 (DATA_WIDTH)**: Specifies the bit width of the input channel. The allowed values are 8, 10, and 12.
- **Maximum Number of Columns**: Specifies the maximum number of columns that can be processed by the core. Permitted values are from 32 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. 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**: Specifies the maximum number of rows that can be processed by the core. Permitted values are from 32 to 4096. Specifying this value is necessary to establish the internal widths of counters and control-logic components. Feeding the configured Defective Pixel Correction instance timing signals that violate the MAX_ROWS constraint leads to data and output timing signal corruption.
- **Maximum Number of Defective Pixels**: Controls depth of RAM for storing defective pixels.

- **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 and customized in an EDK project as a hardware peripheral, and thresholds can be programmed via registers. 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 will generate a set of ports to be used to program the core. See the General Purpose Processor Interface section of Processor Interfaces

Core Symbol and Port Descriptions

The following set of signals is common to both interface options and to all video iPipe cores. Table 2 contains general port information, followed by a more detailed description of each port.

Table 2: **Port Descriptions**

Port Name	Port Width	Direction	Description
clk	1	IN	Rising-edge clock
ce	1	IN	Clock enable (active high)
sclr	1	IN	Synchronous clear – reset (active high)
video_data_in	DATA_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	DATA_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 - 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.
- **video_data_in**: This bus contains the video input in `DATA_WIDTH` bits wide unsigned integer representation. The input is assumed to be from a Bayer sub-sampled image.
- **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 Defective Pixel Correction core to detect 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.
- **video_data_out**: This bus contains video output represented as `DATA_WIDTH` bits wide unsigned integers. The output is in the format of Bayer sub-sampled data.
- **hblank_out**, **vblank_out**, and **active_video_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.

EDK pCore Interface

The EDK pCore Interface generates Processor Local Bus (PLB4.6) interface ports in addition to the common ports described in Table 2. 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 3. For more information on the PLB bus signals, see Processor Local Bus (PLB) v4.6.

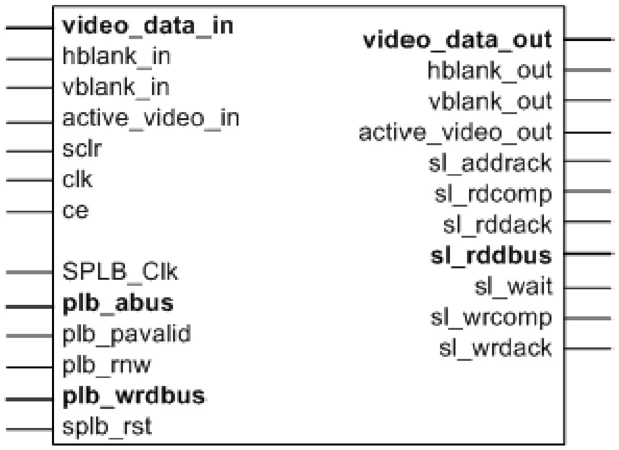


Figure 3: **Core Symbol for the EDK pCore Interface**

General Purpose Processor Interface

The General Purpose Processor Interface exposes the thresholds and control signals as ports. The Core Symbol for the General Purpose Processor Interface is shown in Figure 4. These ports are described in Table 3. The ports common to all interfaces are described in Table 2.

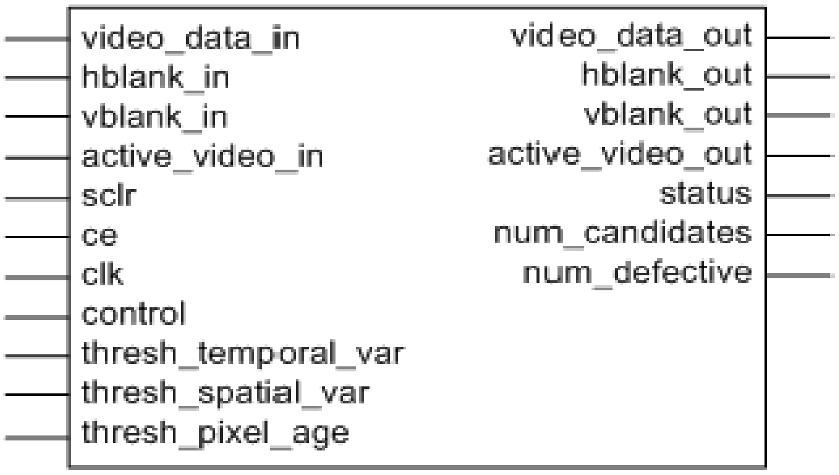


Figure 4: **Core Symbol for the General Purpose Processor Interface**

Table 3: **Additional Ports for the General Purpose Processor Interface**

Port Name	Port Width	Direction	Description
control	2	IN	• Bit 0: Software enable • Bit 1: Host processor write done semaphore • 0 indicates host processor actively updating registers • 1 indicates register update completed by host processor
thresh_temporal_var	DATA_WIDTH	IN	Allowed spatial variance beyond which a pixel is characterized as an outlier
thresh_spatial_var	16	IN	Allowed spatial variance beyond which a pixel is characterized as an outlier
thresh_pixel_age	16	IN	Number of frames an outlier pixel has to keep its value within the range specified
status	8	OUT	Status register • Bit 0: FIFO empty • Bit 1: FIFO full • Bit 2-6: Reserved • Bit 7: Timing lock output; '1' indicates that the timing module of the core has locked on the input timing signals and is generating stable output timing signals
num_candidates	STATUS_WIDTH	OUT	Total number of potential defective pixel candidates stored in FIFO (in previous frame)
num_defective	STATUS_WIDTH	OUT	Total number of pixels being actively interpolated (in previous frame)

Control Signals and Timing

The propagation delay of the Defective Pixel Correction core is two full scan lines and 18 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.

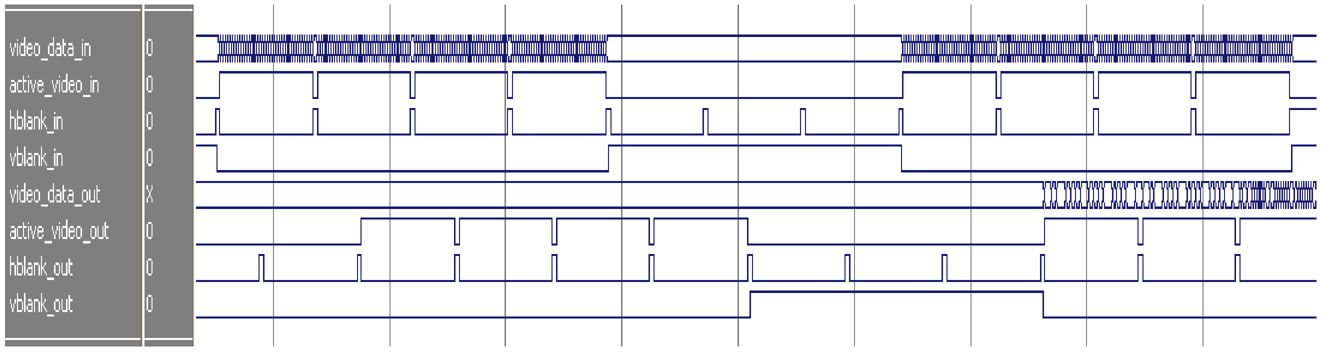


Figure 5: **Timing Example**

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
- Blanking signals are active high
- A minimum active frame size of eight rows and eight columns
- A minimum horizontal blanking period of two columns
- A minimum vertical blanking period of two rows

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

Core Resource Utilization and Performance

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 CORE Generator after generating the core.

Information provided in Table 4 - Table 7 is a guideline to the resource utilization of the Defective Pixel Correction core for Spartan-3A DSP, Spartan-6, Virtex-5, Virtex-6 FPGA families. This core does not use any XtremeDSP slices, block RAM, dedicated I/O, or clock resources. The design was tested using Xilinx ISE® v13.1 tools with default tool options, using timing constraints.

Table 4: **Resource Utilization and Target Speed for Spartan-3A DSP(1)(2)**

Data Width	Max Cols/Rows	Slices	LUTs	FFs	BRAM18s	DSP48As	Fmax(MHz)
8	1024	1,436	2,066	1,652	5	1	122
10	1024	1,603	2,326	1,856	6	1	122
12	1024	1,765	2,586	2,060	6	1	122
8	2200	1,436	2,066	1,652	5	1	122
10	2200	1,603	2,326	1,856	6	1	122
12	2200	1,765	2,586	2,060	6	1	122

1. Device: xc3sd3400a-4fg676
2. Speed file: PRODUCTION 1.33 2011-02-03

Table 5: **Resource Utilization and Target Speed for Spartan-6(1)(2)**

Data Width	Max Cols/Rows	LUT6-FF pairs	LUTs	FFs	RAM16/8	DSP48A1	Fmax(MHz)
8	1024	2,062	1,699	1,599	4/ 1	1	202
10	1024	2,210	1,888	1,797	5/ 1	1	202
12	1024	2,424	2,048	1,995	6/ 0	1	193
8	2200	2,062	1,699	1,599	4/ 1	1	202
10	2200	2,210	1,888	1,797	5/ 1	1	202
12	2200	2,424	2,048	1,995	6/ 0	1	193

1. Device: xc6slx150-2fgg676C
2. Speed file: PRODUCTION 1.17 2011-02-03

Table 6: **Resource Utilization and Target Speed for Virtex-5(1)(2)**

Data Width	Max Cols/Rows	LUT6-FF pairs	LUTs	FFs	RAM36/18	DSP48E	Fmax(MHz)
8	1024	2,276	1,800	1,602	2/ 1	1	246
10	1024	2,534	2,006	1,806	2/ 2	1	255
12	1024	2,828	2,230	2,010	2/ 2	1	255
8	2200	2,276	1,800	1,602	2/ 1	1	246
10	2200	2,534	2,006	1,806	2/ 2	1	255
12	2200	2,828	2,230	2,010	2/ 2	1	255

1. Device: xc5vsx50t-1ff665
2. Speed file: PRODUCTION 1.72 2011-02-03, STEPPING level 0

Table 7: **Resource Utilization and Target Speed for Virtex-6(1)(2)**

Data Width	Max Cols/Rows	LUT6-FF pairs	LUTs	FFs	RAM36/18	DSP48E1	Fmax(MHz)
8	1024	1,938	1,698	1,503	2/ 1	1	307
10	1024	2,206	1,868	1,691	2/ 2	1	298
12	1024	2,330	2,057	1,879	2/ 2	1	307
8	2200	1,938	1,698	1,503	2/ 1	1	307
10	2200	2,206	1,868	1,691	2/ 2	1	298
12	2200	2,330	2,057	1,879	2/ 2	1	307

1. Device: xc6vsx315t-1ff1156C
2. Speed file: PRODUCTION 1.13 2011-02-03

References

1. 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*.

For the latest information and important release notes about this core, refer to the IP Release Notes and known issues at http://www.xilinx.com/support/documentation/ip_documentation/xtp025.pdf.

License Options

The Defective Pixel Correction 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 Defective Pixel Correction 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 Defective Pixel Correction 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
04/24/09	1.0	Initial Xilinx release.
07/23/10	2.0	Updated with Core Version 2.0 information.
03/01/11	3.0	Updated with Core Version 3.0 information.

Notice of Disclaimer