Vision HDL Toolbox™ Release Notes

MATLAB®

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
Phone:	508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Vision HDL Toolbox™ Release Notes

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Contents

R2016a

R2015b

R2015a

R2016a

Version: 1.2

New Features

Bug Fixes

ROI Selector: Select a region of interest from a streaming video source

The new block, `ROI Selector`, selects a region of interest (ROI) from a video stream. You can specify one or more regions using input ports or mask parameters. The block returns each new region as streaming pixel data and corresponding `pixelcontrol` bus.

This release also includes an equivalent System object™, visionhdl.ROISelector.

Grayscale Morphology: Perform dilation, erosion, opening, and closing operations on grayscale inputs

Perform grayscale morphology using these new blocks and System objects:

- `Grayscale Closing`
- `Grayscale Dilation`
- `Grayscale Erosion`
- `Grayscale Opening`
- visionhdl.GrayscaleClosing
- visionhdl.GrayscaleDilation
- visionhdl.GrayscaleErosion
- visionhdl.GrayscaleOpening

Larger frame size for statistics computations

The `Image Statistics` block and visionhdl.ImageStatistics System object now support input regions up to 64^4 (16,777,216) pixels in size.

R2015b

Version: 1.1

New Features

Bug Fixes

Corner Detection Example: Detect intersecting edges with the Harris algorithm

This example uses the `Image Filter` block to implement the Harris & Stephens corner detection algorithm. See “Corner Detection” in Vision HDL Toolbox Examples.

MATLAB Compiler Integration: Generate standalone executables for System objects

All System objects in Vision HDL Toolbox™ support generating executables with MATLAB® Compiler™.

HDL code generation for structure arguments in MATLAB

HDL Coder™ now supports code generation for structure arguments of functions. For Vision HDL Toolbox, this simplifies the arguments of functions targeted for HDL code generation. Previously, you had to flatten the structure into the component control signals.

```
function [pixOut,hStartOut,hEndOut,vStartOut,vEndOut,validOut] = ...
      HDLTargetedDesign(pixIn,hStartIn,hEndIn,vStartIn,vEndIn,validIn)
```

With HDL code generation support for structures, the arguments can now include the control signal structure.

```
function [pixOut,ctrlOut] = HDLTargetedDesign(pixIn,ctrlIn)
```

The structure becomes individual control signals in the generated Verilog or VHDL code.

Improved line buffer performance

This release improves the HDL performance of blocks and objects that have internal line memory. The synthesized HDL code for the line buffer now supports HD video at 60fps on the Xilinx® Zynq®-7000 ZC702 board, and 4k video at 30fps on the Xilinx Zynq-7000 ZC706 board. The following blocks and System objects use the improved line buffer code:

- `Demosaic Interpolator`
- `Edge Detector`
- `Image Filter`
- `Median Filter`

- `Closing`
- `Dilation`
- `Erosion`
- `Opening`

For example, the table shows the R2015b performance of the `Demosaic Interpolator`, using **Gradient-corrected linear** interpolation, and synthesized with Xilinx Vivado® for these target boards.

Xilinx Zynq-7000 ZC702	Xilinx Zynq-7000 ZC706
HD input video	4k input video
200 MHz	375 MHz
Consumes: • no DSP48s • 2.5% of the LUTS • 1.5% of the slice registers • 8 BRAMS (4%)	Consumes: • no DSP48s • 0.6% of the LUTS • 0.4% of the slice registers • 8 BRAMS (1%)

In the previous release, the performance is shown below.

Xilinx Zynq-7000 ZC702	Xilinx Zynq-7000 ZC706
HD input video	4k input video
135 MHz (need 150 MHz for 60 fps)	230 MHz (need 300 MHz for 30 fps)
Consumes: • no DSP48s • 2.6% of the LUTS • 1.5% of the slice registers • 8 BRAMS (4%)	Consumes: • no DSP48s • 0.5% of the LUTS • 0.3% of the slice registers • 8 BRAMS (1%)

R2015a

Version: 1.0

New Features

Video synchronization signal controls for handling nonideal timing and resolution variations

Vision HDL Toolbox blocks and System objects accept and return video data as a serial stream of pixel data and control signals. The protocol mimics the timing of a video system, including inactive intervals between frames. Each block or object operates without full knowledge of the image format, and can tolerate imperfect timing of lines and frames. See Streaming Pixel Interface.

Configurable frame rates and sizes, including 60FPS for high-definition (1080p) video

To support HD video applications, Vision HDL Toolbox blocks and System objects generate HDL code capable of running at 150 MHz.

For supported video formats, see the Frame To Pixels block.

Frame-to-pixel and pixel-to-frame conversions to integrate with frame-based processing capabilities in MATLAB and Simulink

In MATLAB, use the visionhdl.FrameToPixels object to convert framed video data to a stream of pixels and control signals.

In Simulink®, use the Frame To Pixels block to convert framed video data to a stream of pixels and control signals.

Image processing, video, and computer vision algorithms with a pixel-streaming architecture, including image enhancement, filtering, morphology, and statistics

Vision HDL Toolbox blocks and System objects implement hardware-friendly architectures. For the list of blocks and System objects provided in this product, see HDL-Optimized Algorithm Design.

Implicit on-chip data handling using line memory

Some Vision HDL Toolbox blocks and System objects include internal memory for a small number of lines as required for the calculation at each image pixel.

The line memory stores *kernel size - 1*-by-*active pixels per line* pixels. Set **Line buffer size** to a power of two that accommodates *active pixels per line*.

Support for HDL code generation and real-time verification

Vision HDL Toolbox provides libraries of blocks and System objects that support HDL code generation. To generate HDL code from these designs, you must have a HDL Coder license. HDL Coder also enables you to generate scripts and test benches for use with 3rd party HDL simulators.

If you have a HDL Verifier™ license, you can use the FPGA-in-the-loop feature to prototype your HDL design on an FPGA board. HDL Verifier also enables you to cosimulate a Simulink model with an HDL design running in a 3rd party simulator.

See HDL Code Generation and Verification