Key Design Features

- Technology independent soft IP Core for FPGA, ASIC and SoC devices
- Supplied as human-readable VHDL (or Verilog) source code
- Fully pipelined architecture with simple flow control and AXI4-compatible data streaming interfaces
- Supports all image resolutions up to 2^{16} x 2^{16} pixels
- Support for 5 different low-pass filter responses including, box blur, low-cost smoothing, Gaussian and custom kernels
- Features a 3 x 3 image filter
- Input / output rate of 1 x 24-bit pixel per clock
- No frame buffer required
- Small implementation size
- Support for 300 MHz+ operation on basic FPGA and SoC devices[1]

Applications

- Low-pass filtering of digital video to remove high frequency artefacts such as jagged edges, stepped lines and Moiré interference patterns
- Optimization of video from image sensors and cameras
- Forms an essential stage when downscaling and upscaling by large factors. For instance, 4K video to HD or SD and vice-versa

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
line_width	Width of linestores in pixels	integer	2^4 < pixels < 2^{16}
log2_line_width	Log2 of linestore width	integer	log2(line_width)
kernel_type	Anti-alias filter kernel selection	integer	0: Cheap box blur 1: Weighted center 2: Weighted cross hairs 3: Gaussian blur 4: True box blur

1 AMD / Xilinx® 7-series used as a benchmark

Block Diagram

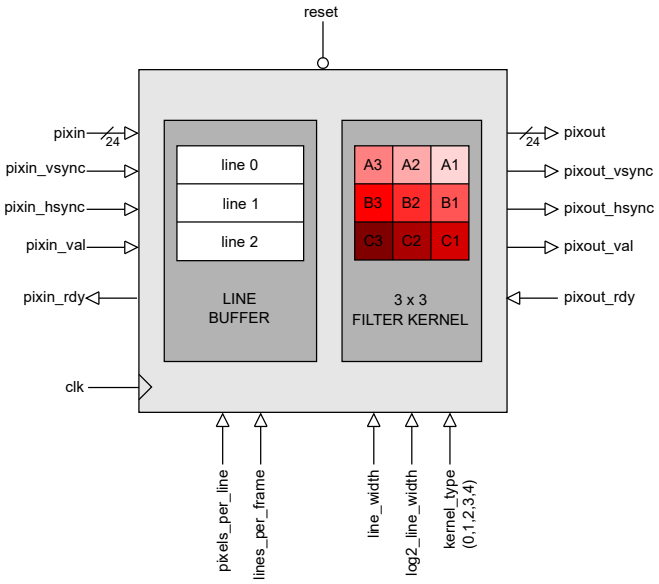

Figure 1: Simplified anti-alias digital video filter architecture

Pin-out Description

Pin name	*I/O*	*Description*	*Active state*
clk	in	Synchronous clock	rising edge
reset	in	Asynchronous reset	low
pixels_per_line	in	Number of pixels per line Range: 2^4 < lines < 2^{16}	data
lines_per_frame	in	Number of lines per frame Range: 2^4 < lines < 2^{16}	data
pixin [23:0]	in	24-bit RGB 8:8:8 pixel in	data
pixin_vsync	in	Vertical sync in (Coincident with first pixel of input frame)	high
pixin_hsync	in	Horizontal sync in (Coincident with first pixel of input line)	high
pixin_val	in	Input pixel valid	high
pixin_rdy	out	Ready to accept input pixel (Handshake signal)	high
pixout [23:0]	out	24-bit RGB 8:8:8 pixel out	data
pixout_vsync	out	Vertical sync out (Coincident with first pixel of output frame)	high
pixout_hsync	out	Horizontal sync out (Coincident with first pixel of output line)	high
pixout_val	out	Output pixel valid	high
pixout_rdy	in	Ready to accept output pixel (Handshake signal)	high

General Description

The ALIAS_FILTER IP Core (Figure 1) is a fully pipelined anti-aliasing filter for use in digital video applications. The design implements a low-pass filter response on the source video in order to alleviate problems such as jagged edges, stepped lines and Moiré interference patterns. This is especially important when downscaling and upscaling video by large factors.

Internally, the circuit uses a 3x3 pixel filter kernel with 5 different possible filter settings. The input and output pixels are 24-bit in RGB 8:8:8 format. The image size is fully programmable with support for up to 2^{16} x 2^{16} pixels.

Pixels flow into the IP Core in accordance with the valid-ready pipeline protocol[2]. Input pixels and syncs are sampled on the rising edge of *clk* when *pixin_val* and *pixin_rdy* are both high. Likewise, at the output interface, pixels and syncs are sampled on a the rising edge of *clk* when *pixout_val* and *pixout_rdy* are both high.

Filter kernel selection

The *kernel_type* parameter modifies the low-pass filter response of the anti-aliasing filter. Figure 2 below shows the 5 possible filter configurations available:

Type 0: Cheap box blur

1/8	1/8	1/8
1/8	0	1/8
1/8	1/8	1/8

Type 1: Weighted center

1/16	1/16	1/16
1/16	1/2	1/16
1/16	1/16	1/16

Type 2: Weighted cross hairs

0	1/8	0
1/8	1/2	1/8
0	1/8	0

Type 3: Gaussian blur

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Type 4: True box blur

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 2: Possible user filter settings

2 See Zipcores application note: app_note_zc001.pdf for more examples of how to use the valid-ready pipeline protocol

Example of image filtering

Figure 3 shows the general result of low-pass filtering the source image. Notice the absence of stepped lines and 'pixelated' edges in the output image.

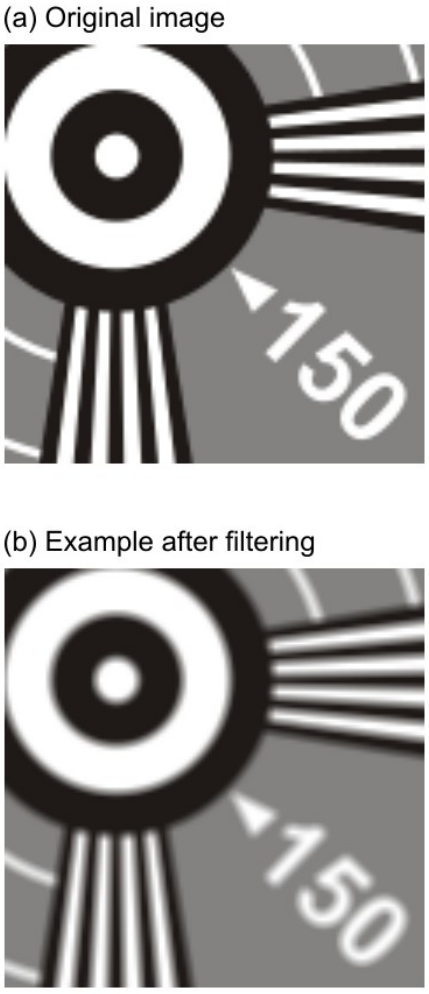

Figure 3: Anti-alias filtering example using the 'cheap box blur setting'. Image (a) before an (b) after filtering

Image resolution

The size of the input source image is fully programmable and is specified in the parameters: *pixels_per_line* and *lines_per_frame*. The maximum frame size supported is 2^{16} x 2^{16} pixels. The frame size parameters may be changed on a frame-by-frame basis if necessary.

It is recommended that a system reset is asserted once the parameters have been changed in order to avoid possible image corruption. This is often convenient to do during the vertical blanking period of a live video stream. After reset, the IP Core will start generating output pixels after the next clean input frame.

The generic parameters *line_width* and *log2_line_width* must also be set correctly to accommodate the maximum line length of the input image. The line width must be specified as the nearest power of 2. For instance, 1024, 2048, 4096 etc.

Functional Timing

The functional timing of the input and output interfaces follows a simple flow-control protocol with a 'valid' and 'ready' signal. These signals are compatible with the *t_valid* and *t_ready* signals used in the AMBA / AXI4-stream adopted by ARM and many other IP Core vendors.

Input interface timing

Figure 4 shows the signalling at the start of a new frame. The first line of a new frame begins with *pixin_vsync* and *pixin_hsync* asserted high together with the first pixel. Note that the signals *pixin*, *pixin_vsync* and *pixin_hsync* are only valid if *pixin_val* is also asserted high. For demonstration purposes, the diagram also shows what happens when *pixin_rdy* is de-asserted. In this case, the pipeline is stalled and the upstream interface must hold-off before *pixin_rdy* is re-asserted.

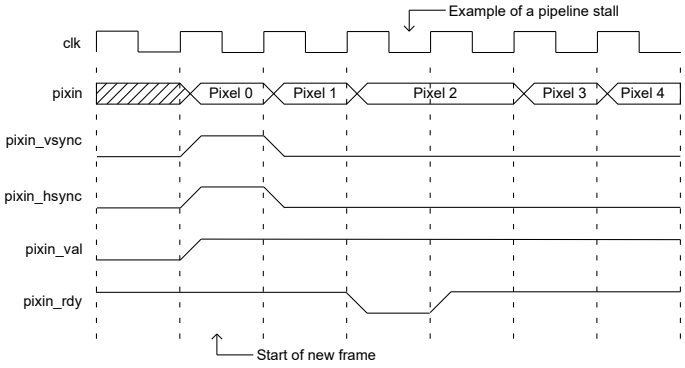

Figure 4: Start of a new input frame

Figure 5 shows the signalling at the start of a new line. Note that the timing diagram is the same as for the start of a new frame with the exception that *pixin_vsync* is held low while *pixin_hsync* is held high together with the first valid pixel.

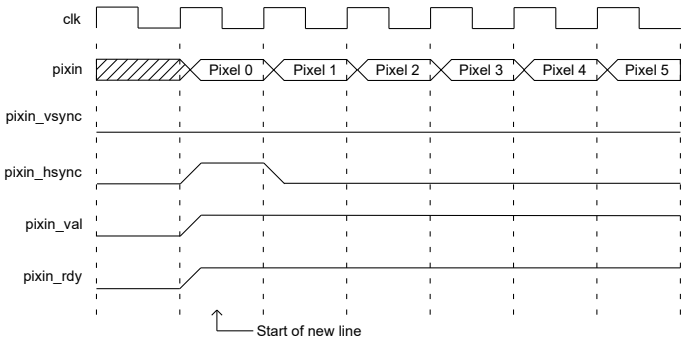

Figure 5: Start of a new input line

Output interface timing

The output interface timing is identical to the input timing. Figures 6 and 7 show the signals at the beginning of an output frame and output line respectively. In addition, Figure 7 gives an example of an invalid output pixel with *pixout_val* asserted low for one clock cycle. In this instance the downstream interface must ignore the pixel until *pixout_val* is re-asserted high.

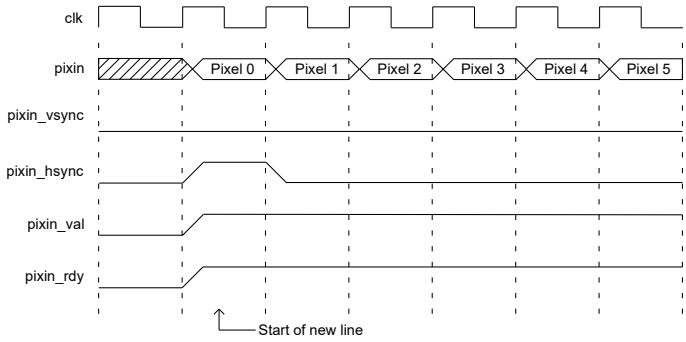

Figure 6: Start of a new output frame

As an additional note, if the downstream interface is always capable of accepting pixels, then the user may choose to tie the *pixout_rdy* signal to logic '1'. This will also result in a slightly optimized design in terms of resource use and maximum attainable clock frequency.

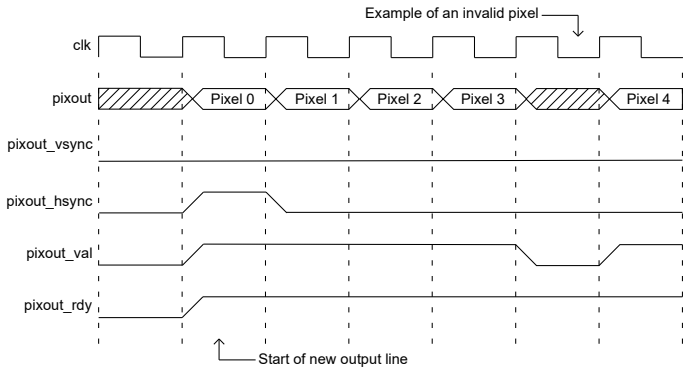

Figure 7: Start of a new output line

Source File Description

The source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

Source file	Description
video_in.txt	Text-based source image file
pipeline_reg.vhd	Pipeline register component
fifo_sync.vhd	Synchronous FIFO
video_file_reader.vhd	Reads image file into test bench
ram_dp_w_r.vhd	Dual-port RAM component
alias_buffer.vhd	Line buffer component
alias_kernel.vhd	3x3 anti-aliasing filter kernel
alias_filter.vhd	Top-level component
alias_filter_bench.vhd	Top-level test bench

Functional Testing

An example VHDL testbench is provided for use in a suitable hardware simulator. The compilation order of the source code is the same order as the source code file description (above).

The testbench instantiates the *alias_filter.vhd* component and the user may modify the generic parameters as required. In particular, the user may choose one of the five different filter kernel types available as described in figure 2. The frame size has been set to VGA (640x480) pixels.

The input source image for the simulation is generated by the *video_file_reader.vhd* component. This component reads a text file that contains the discrete RGB pixels. The text file is called *video_in.txt* and should be placed in the top-level simulation folder.

The text file follows a simple format which defines the state of the signals: *pixin_val, pixin_vsync*, *pixin_hsync* and *pixin (R, G, B)* on a clock-by-clock basis. An example file might be the following:

```
1 1 1 4F 52 55 # pixel 0 line 0 (start of frame)
1 0 0 12 73 71 # pixel 1
1 0 0 2C 52 18 # pixel 2
1 0 0 B9 72 88 # pixel 3

.
.
.

1 0 1 16 72 23 # pixel 0 line 1 (start of line)
1 0 0 18 50 AD # pixel 1
1 0 0 2E 71 F5 # pixel 2
1 0 0 29 4D 29 # pixel 3 etc ..
```

In this example the first line of the *video_in.txt* file asserts the input signals *pixin_val* = 1, *pixin_vsync* = 1, *pixin_hsync* = 1 and *pixin* = 0x4F5255

In the example simulation shipped with the source code, a sample 640x480 (VGA) image is processed by the anti-aliasing IP Core. The simulation must be run for at least 10 ms during which time an output text file called *video_out.txt* is generated[3]. This file contains a sequential list of 24-bit RGB output pixels in a similar format to *video_in.txt.*

Figure 8 shows the input and output images of the simulation with *kernel_type* set to '0'.

(Note: As it's difficult to see the key differences within the context of this low-resolution document, high quality images may be provided on request. Please contact Zipcores customer support for more information).

(a) Source image

(b) Output image

Figure 8: Simulation results before (a) and after (b) filtering

3 PERL scripts for generating and processing input and output text files are provided with the IP Core package

Synthesis and Implementation

The files required for synthesis and the design hierarchy is shown below:

- alias_filter.vhd
  - alias_buffer.vhd
    - ram_dp_w_r.vhd
  - alias_kernel.vhd
  - fifo_sync.vhd
    - pipeline_reg.vhd

The IP Core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the AMD / Xilinx® 7-series FPGA devices. Synthesis results for other FPGAs, SoCs and technologies can be provided on request.

There are no special constraints required for synthesis. The IP Core is completely technology independent.

Trial synthesis results are shown with the generic parameters set to: *line_width* = 2048, *log2_line_width* = 11, *kernel_type* = 0.

Resource usage is specified after place and route.

AMD / XILINX® 7-SERIES FPGAS

Resource type	***A-7***	***K-7***	***V-7***	***US+***
Registers	532	532	532	532
LUTs	715	715	788	847
Block RAM	7.5	7.5	7.5	7.5
DSPs	0	0	0	0
Occupied Slices	264	260	261	151 (CLB)
Clk freq. (approx)	200 MHz	300 MHz	350 MHz	500 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision.	12/08/2014
1.1	Addition of new filter kernel types.	29/04/2015
1.2	First official release.	16/08/2023
1.3	Included new test source images. Minor source-code changes including comments for the different filter kernel types.	19/12/2023