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
- Decoding of a bayer-mapped image from an image sensor or Colour Filter Array (CFA)
- Supports all image resolutions from 16 x 16 pixels up to 2^{16} x 2^{16}
- Support for all sensor pixel widths from 2-bits and above
- Support for different CFA sensor alignments
- Features a 5 x 5 polyphase interpolation filter
- Output 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

- Bayer-mapped to 24-bit RGB decoding (de-mosaicing)
- Digital camera image processing
- Forms an essential first stage in most digital processing pipelines that contain an image sensor

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
dw	Sensor width in bits	integer	≥ 2
line_width	Width of linestores in pixels	integer	2^4 < pixels < 2^{16}
log2_line_width	Log2 of linestore width	integer	log2(line_width)
sensor_align	Bayer pattern sensor alignment	integer	0 : BGBG ... GRGR … 1 : GBGB ... RGRG ... 2 : GRGR ... BGBG … 3 : RGRG ... GBGB ...

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

Block Diagram

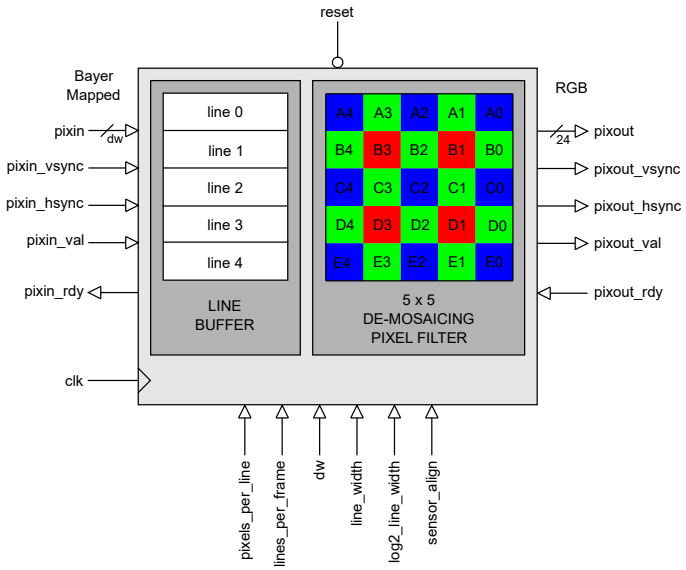

Figure 1: Bayer-to-RGB converter architecture

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 [dw - 1:0]	in	n-bit bayer-mapped 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 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

BAYER_TO_RGB (Figure 1) is a fully pipelined Bayer-mapped to RGB converter IP Core. The IP Core may be used to process the raw pixels from an image sensor or Colour Filter Array (CFA). These pixels are typically organized as a bayer pattern of discrete Red, Green and Blue values which must be interpolated to recover the original image - a process that is commonly known as de-mosaicing.

Internally, the circuit uses a 5 x 5 pixel filter with dynamic coefficients to interpolate the pixels from the CFA. The resulting output is a high-quality RGB image at 24-bits/pixel.

Bayer-mapped pixels flow into the design 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.

The image size is fully programmable with standard support for anything from 16x16 pixels and above. The width of the input pixels is specified using the generic parameter *dw*. Output pixels are standard 24-bit RGB.

Sensor alignment

The sensor alignment parameter modifies the central starting position of the 5 x 5 filter according to the alignment of the bayer pattern. Figure 2 below demonstrates the 4 possible sensor alignments in the CFA.

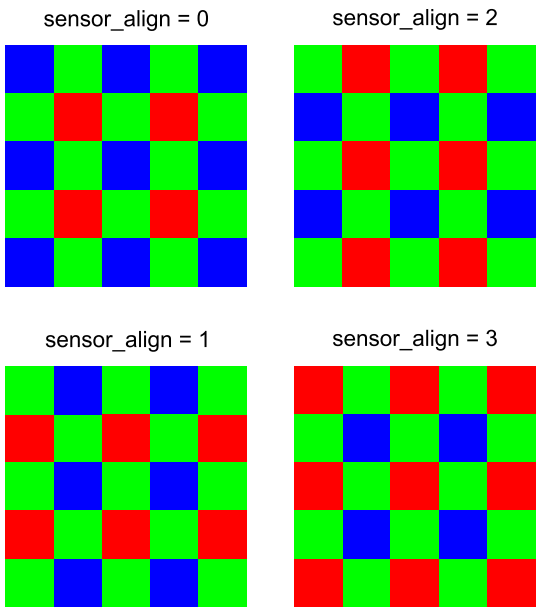

Figure 2: CFA sensor alignments

By setting the *sensor_align* parameter correctly, the design can adapt to the four possible patterns. If the alignment is wrong, then the colours in the output image will be corrupted.

Image resolution

The size of the image to be interpolated is fully programmable and is specified in the parameters: *pixels_per_line* and *lines_per_frame*. These parameters can 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 to avoid possible image corruption. 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 - e.g. 1024, 2048, 4096 etc.

De-mosaicing filter

The internal filter is a 5 x 5 pixel filter that is used to interpolate the bayer-mapped image. The filter architecture uses a polyphase filter design in which the filter kernel changes depending on the interpolation point in the bayer pattern. The filter kernels are based on the Malvar-He-Cutler algorithm which has been shown to give excellent results for a very reasonable resource cost. For optimum results it is recommended that the image sensor has a capacity of at least 5M pixels or better.

Functional Timing

Figure 3 shows the signalling at the input interface 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 further pixels are processed.

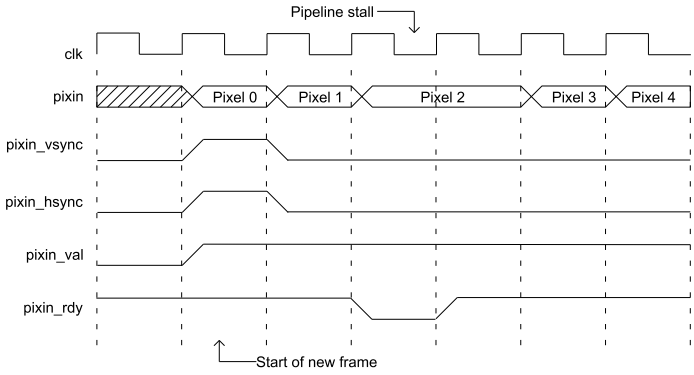

Figure 3: Start of new input frame

Figure 4 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. In this example, *pixin_rdy* is held high for the duration so the input interface does not stall.

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

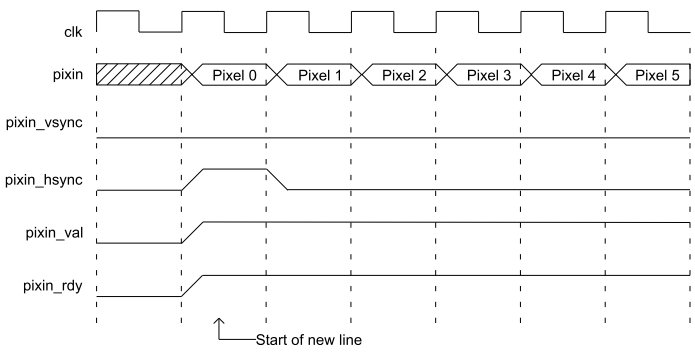

Figure 4: Start of new input line

The final timing waveform (Figure 5) shows the output signalling for the start of a new output frame. The output flow-control is identical to the input with the use of the same valid-ready protocol. In this example, the *pixout_rdy* signal is held high during the output of the frame. (Note that if the downstream interface can always accept pixels then *pixout_rdy* may be tied to logic '1').

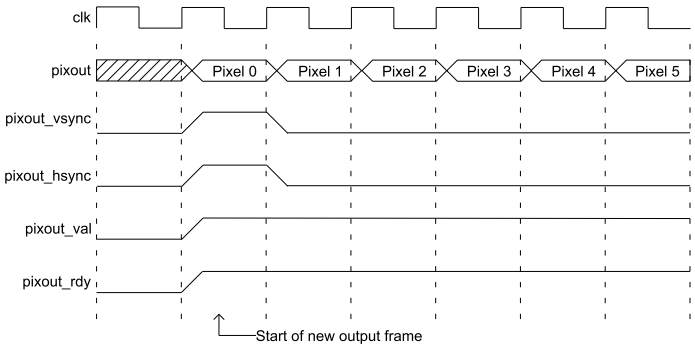

Figure 5: Start of new output frame

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***
bayer_in.txt	Text-based source image file
pipeline_reg.vhd	Pipeline register component
fifo_sync.vhd	Synchronous FIFO
bayer_file_reader.vhd	Reads image file into test bench
ram_dp_w_r.vhd	Dual-port RAM component
bayer_buffer.vhd	Line buffer component
bayer_filter.vhd	5 x 5 pixel filter component
bayer_to_rgb.vhd	Top-level component
bayer_to_rgb_bench.vhd	Top-level test bench

Functional Testing

An example testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. pipeline_reg.vhd
2. fifo_sync.vhd
3. ram_dp_w_r.vhd
4. bayer_buffer.vhd
5. bayer_filter.vhd
6. bayer_to_rgb.vhd
7. bayer_file_reader.vhd
8. bayer_to_rgb_bench.vhd

The testbench instantiates the BAYER_TO_RGB component and the user may modify the generic parameters as required. In particular, the user must ensure that the image dimensions and sensor alignment are correct for the input source image.

The input source image for the simulation is generated by the file reader component. The component reads a text file that contains the discrete R,G,B bayer-mapped pixels. The text file is called *bayer_in.txt* and should be placed in the top-level simulation folder.

The file *bayer_in.txt* follows a simple format which defines the state of the signals: *pixin_val, pixin_vsync*, *pixin_hsync* and *pixin* on a clock-by-clock basis. An example file might be the following:

```
1 1 1 0052      # pixel 0 line 0 (start of frame)
1 0 0 0073      # pixel 1
1 0 0 0052      # pixel 2
1 0 0 0072      # pixel 3
.
.
1 0 1 0072      # pixel 0 line 1 (start of line)
1 0 0 0050      # pixel 1
1 0 0 0071      # pixel 2
1 0 0 004d      # pixel 3 etc ..
```

In this example the first line of the *bayer_in.txt* file asserts the input signals *pixin_val* = 1, *pixin_vsync* = 1, *pixin_hsync* = 1 and *pixin* = 0x0052.

In the example simulation shipped with the source code, a sample VGA image is processed by the Bayer to RGB converter. The simulation must be run for at least 10 ms during which time an output text file called *bayer_out.txt* is generated[3]. This file contains a sequential list of 24-bit RGB output pixels in a similar format to *bayer_in.txt.*

Figure 6 and 7 show the images before and after de-mosaicing. High quality images can be provided on request.

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