Key Design Features

- Synthesizable, technology independent VHDL Core
- Converts industry standard BT.656 digital video to 24-bit RGB
- Integrated 4:2:2 YCbCr to RGB888 colour-space converter
- Both PAL and NTSC (576i and 480i) input formats supported
- All signals synchronous with the pixel clock
- Small implementation size ideal for all types of FPGA
- Compatible with a wide range of SD video decoder ICs

Applications

- BT.656 input video capture and processing
- PAL & NTSC SDTV interlaced format conversion
- Connectivity with a wide range of commercially available video decoder ICs
- Simple and cost-effective method for capturing digital video into your FPGA or ASIC

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
mode	Input video mode	integer	0: PAL (576i) 1: NTSC (480i)

Pin-out Description

Pin name	*I/O*	*Description*	*Active state*
clk	in	Pixel clock	rising edge
reset	in	Asynchronous reset	low
overflow	out	Pixel overflow error	high
video_in [7:0]	in	BT.656 input video (8-bit)	data
video_val	in	BT.656 input video valid	high
pixout [23:0]	out	24-bit RGB888 pixel	data
pixout_field	out	Field flag	0: odd field 1: even field
pixout_vsync	out	Vertical sync out	high
pixout_hsync	out	Horizontal sync out	high
pixout_val	out	Output pixel valid	high
pixout_rdy	in	Ready to accept output pixel (handshake signal)	high

Block Diagram

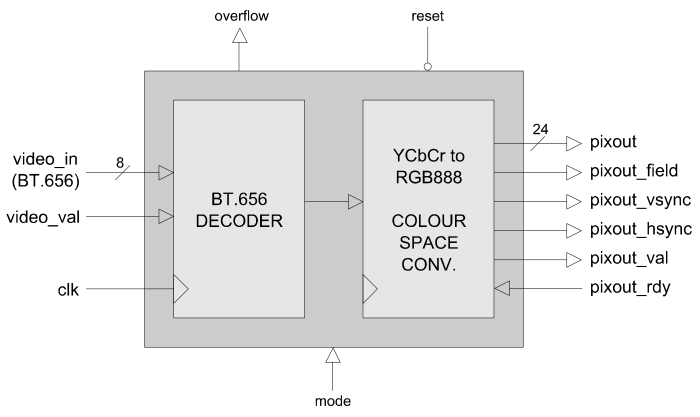


Figure 1: BT.656 Decoder architecture

General Description

BT_656_DECODER (Figure 1) is a digital video decoder with integrated colour-space converter. It's function is to extract the valid pixels from a BT.656 video stream and convert them to 24-bit RGB for subsequent processing.

Video decoding begins after reset is de-asserted and on the rising-edge of *clk* when the *video_val* signal is asserted high. (The signal *video_val* is a clock-enable signal that enables the sampling and processing of each input byte).

Pixels are extracted from the BT.656 input stream and converted to RGB888 format. These pixels are then presented at the output of the decoder together with field and sync flags. All signals are synchronous with the input clock.

The video output from the decoder follows a simple valid-ready streaming protocol that is common to all other Zipcores video IP. Output pixels and flags are sampled on a rising clock-edge when *pixout_val* and *pixout_rdy* are both high.

The decoder features an internal FIFO that is sufficient to buffer two lines of input video. If the *pixout_rdy* signal is de-asserted for more than two lines worth of active pixels then the FIFO will fill to capacity and the overflow flag will be asserted. In the event of an overflow then the decoder must be reset in order to cleanly re-align to the input video stream.

Note that the *pixout_rdy* signal may be tied high if it is known that the downstream interface can always accept output pixels.

BT.656 Decoder

The decoder samples the incoming BT.656 input and looks for the first active line in field '0' (odd field). Once this is detected, output pixels are generated on a line-by-line basis. After all the lines in the odd field have been decoded, operation continues with the decoding of all active lines in field '1' (even field). The decoder then reverts back to field '0' once again.

After a system reset, the decoder will revert to it's initial state and stop generating output pixels. Decoding will then begin again with the first active line of field '0'.

When the generic parameter *mode* is set to '0', the decoder expects an 576i interlaced video input with a resolution of 720 x 288 pixels per field. Conversely, when *mode* is set to '1' then the expected input is (480i) or 720 x 240 pixels per field[1].

During blanking periods, no valid output pixels are generated. Decoding is only concerned with extraction of active pixels from the BT.656 video stream. Valid 4:2:2 YCbCr pixels are passed to the Colour-Space Converter module where they are converted to 24-bit RGB.

Colour-space converter

Chroma values from the input pixels (Cb, Cr) are duplicated every second pixel and then converted to the RGB888 colour-space using the following formulas:

$$\begin{aligned} R &= 1.164(Y-16)+1.596(Cr-128) \\ G &= 1.164(Y-16)+0.813(Cr-128)-0.391(Cb-128) \\ B &= 1.164(Y-16)+2.018(Cb-128) \end{aligned}$$

In addition, the colour-space converter also generates correctly aligned vsync, hsync, field and valid flags. All output flags are qualified by the *pixout_val* signal.

The signal *pixin_field* is active for the duration of the field, with '0' indicating an odd field and '1' indicating an even field. The signal *pixin_vsync* is coincident with the first pixel of a field - irrespective of whether odd or even. The signal *pixin_hsync* is coincident with the first pixel of a line. Example timing waveforms are given in the functional timing section below.

Functional Timing

Figure 2 shows the input BT.656 video stream into the decoder. Bytes are sampled on a rising clock-edge when *video_val* is active high. When *video_val* is low then the input bytes are ignored by the decoder.

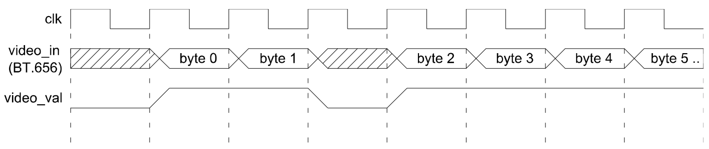
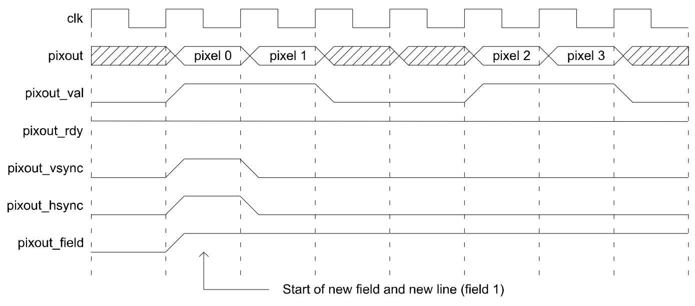


Figure 2: BT.656 input video timing

Example output decoder waveforms are shown in Figure 3. Output pixels and syncs are transferred on a rising clock-edge when *pixout_val* and *pixout_rdy* are both high. When *pixout_val* is low then the outputs should be ignored.

Note that during an active line, the output valid flag will have a 50% duty cycle. This is due to the fact that two bytes must be read from the stream for every generated output pixel.



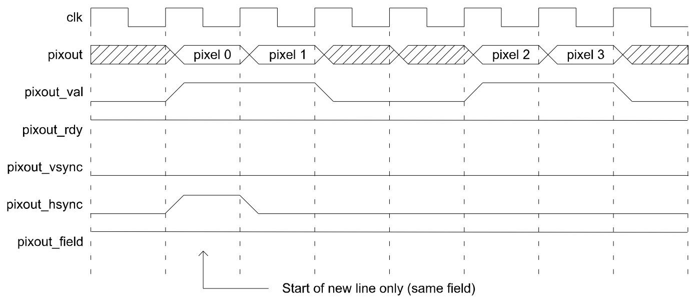


Figure 3: Output waveforms showing the start of a new field and the start of a new line (pixout_rdy shown tied high)

Source File Description

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

Source file	***Description***
bt_656_in.txt	BT.656 input video text file (8-bit)
pipeline_reg.vhd	Pipeline register element
fifo_sync.vhd	Synchronous FIFO
bt_656_file_reader.vhd	BT.656 text file reader
bt_656_dec.vhd	Main decoder component
bt_656_dec_csc.vhd	Colour-space converter
bt_656_decoder.vhd	Top-level component
bt_656_decoder_bench.vhd	Top-level testbench

Functional Testing

An example VHDL 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. bt_656_dec_csc.vhd
4. bt_656_dec.vhd
5. bt_656_decoder.vhd
6. bt_656_decoder_bench.vhd
7. bt_656_file_reader.vhd

1 Auto detect of the input video mode is an option. Please contact Zipcores for more information.