

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
- Digital video outputs include FIELD, VSYNC, HSYNC and output valid flags
- 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
- Conversion of 'legacy' SDTV interlaced formats to fully progressive 576p and 480p video¹
- 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
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

¹ When used in combination with one of our video deinterlacer IP Cores. Please contact Zipcores for more information.

Block Diagram

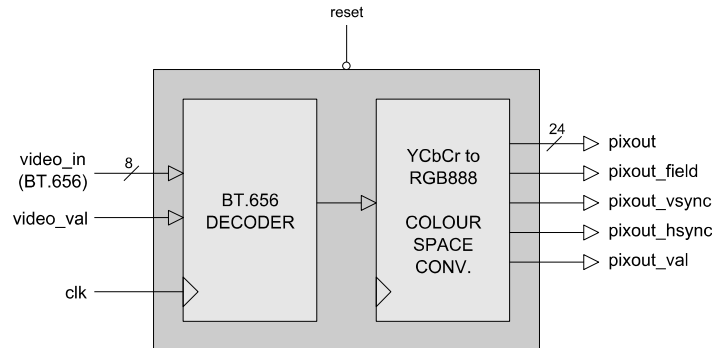


Figure 1: BT.656 Decoder Architecture

General Description

BT_656_DECODER (Figure 1) is a digital video decoder with integrated colour-space converter. Its 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, sync and valid flags. All signals are synchronous with the input clock.

The video output from the decoder is directly compatible with all other Zipcores video IP and provides a convenient way of capturing SDTV video into your FPGA or ASIC device.

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 state machine will revert to its initial state and stop generating output pixels. Decoding will then begin 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 720x288 pixels per field. Conversely, when *mode* is set to '1' then the expected input is (480i) or 720x240 pixels per field².

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.

² Auto detect of the input video modes is an option. Please contact Zipcores for more information.

YUV 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:

$$R = 1.164(Y - 16) + 1.596(Cr - 128)$$

$$G = 1.164(Y - 16) + 0.813(Cr - 128) - 0.391(Cb - 128)$$

$$B = 1.164(Y - 16) + 2.018(Cb - 128)$$

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 coincident with the first pixel of a 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 line³. Detailed timing waveforms are given in the functional timing section below.

Functional Timing

Example output decoder waveforms are shown in Figure 2 below. Output pixels and flags are valid on a rising clock-edge when *pixout_val* is 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 change of ratio from 4 bytes in to 2 pixels out.

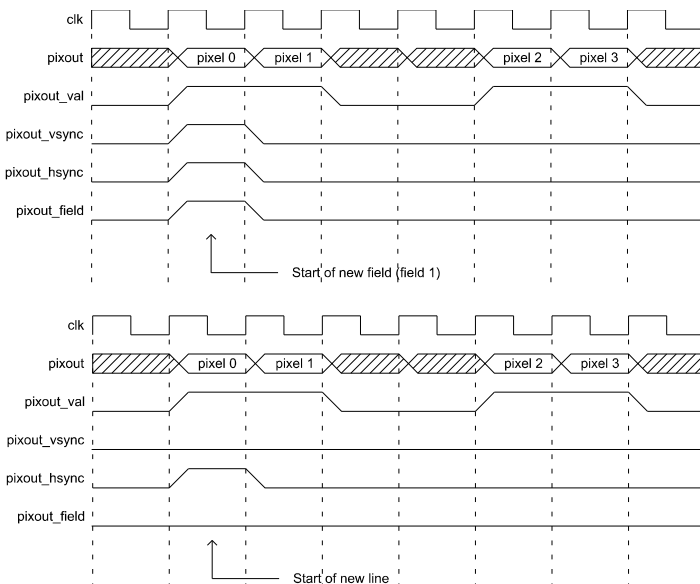


Figure 2: Output waveforms showing first pixel of a new field and first pixel of a new line

3 The vsync and hsync signals do not operate like 'traditional' analog syncs as such. Their purpose is to indicate the first pixel of a field and first pixel of a line only.

Figure 3 shows the input of the 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.

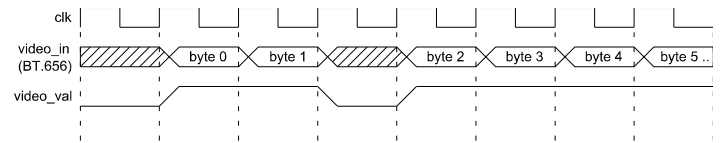


Figure 3: BT.656 input video timing

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)
bt_656_file_reader.vhd	BT.656 text file reader
bt_656_dec.vhd	Main decoder component
bt_656_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. bt_656_csc.vhd
2. bt_656_dec.vhd
3. bt_656_decoder.vhd
4. bt_656_decoder_bench.vhd
5. bt_656_file_reader.vhd

The VHDL testbench instantiates the BT_656_DECODER component with the video format set to 'PAL' or 576i.

The source video for the simulation is generated by the file-reader component. This component reads a text-based file which contains the BT.656 encoded data with each byte on a separate line. The text file is called *bt_656_in.txt* and should be placed in the top-level simulation directory.

The simulation must be run for at least 20 ms during which time all the outputs are captured to a text file called *video_out.txt*. This file contains a sequential list of output pixels and flags which may be processed and used to generate an output image in software.

Figure 4 below shows the results after decoding a PAL and NTSC video source. In both cases, the original interlaced frame is shown compared to the odd and even fields extracted by the BT.656 decoder.

Original full resolution bitmap images are available on request.

PAL (576i)



NTSC (480i)



Figure 4a & 4b: Results of decoding a full PAL interlaced frame into its constituent fields. Figure 4c & 4d: Results of decoding a full NTSC interlaced frame into its constituent fields.

Synthesis

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

- bt_656_dec.vhd
- bt_656_csc.vhd
- bt_656_decoder.vhd

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx Virtex 5 and the Altera Stratix III series of FPGA devices. The lowest and highest speed grade devices have been chosen in both cases for comparison.

Trial synthesis results are shown with the generic *mode* parameter set to '0' (576i).

Resource usage is specified after Place and Route.

VIRTEX 5

Resource type	Quantity used
Slice register	143
Slice LUT	133
Block RAM	0
DSP48	10
Clock frequency (worst case)	150 MHz
Clock frequency (best case)	200 MHz

STRATIX III

Resource type	Quantity used
Slice Register	171
Slice LUT	265
Block Memory bit	0
DSP block 18	12
Clock frequency (worse case)	166 MHz
Clock frequency (best case)	215 MHz

Revision History

Revision	Change description	Date
1.0	Initial revision	08/08/2010
1.1	Added CSC formulas	17/11/2010
1.2	Updated synthesis results in line with some internal code optimizations	25/01/2011
1.3	Made detection of SAV codes for blanking and active video more robust. Updated synthesis results	02/01/2012