Key Design Features

- Synthesizable, technology independent VHDL IP Core
- Converts 24-bit RGB digital video to an industry standard 8-bit ITU-R BT.656 stream
- Integrated RGB888 to 4:2:2 YCbCr colour-space converter
- Both PAL and NTSC (576i and 480i) 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 encoder ICs

Applications

- BT.656 output video generation
- PAL & NTSC SDTV video format conversion
- Connectivity with a wide range of commercially available video encoder ICs
- Simple and cost-effective method for generating digital video outputs from your FPGA or ASIC

Generic Parameters

Generic name	***Description***	***Type***	***Valid range***
mode	Output 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
underflow	out	Pixel underflow error	high
video_out [7:0]	out	BT.656 output video (8-bit)	data
video_val	out	BT.656 output video valid	high
pixin [23:0]	in	24-bit RGB888 pixel	data
pixin_field	in	Field flag	0: odd field 1: even field
pixin_vsync	in	Vertical sync in	high
pixin_hsync	in	Horizontal sync in	high
pixin_val	in	Input pixel valid	high
pixin_rdy	out	Ready to accept input pixel (handshake signal)	high

Block Diagram

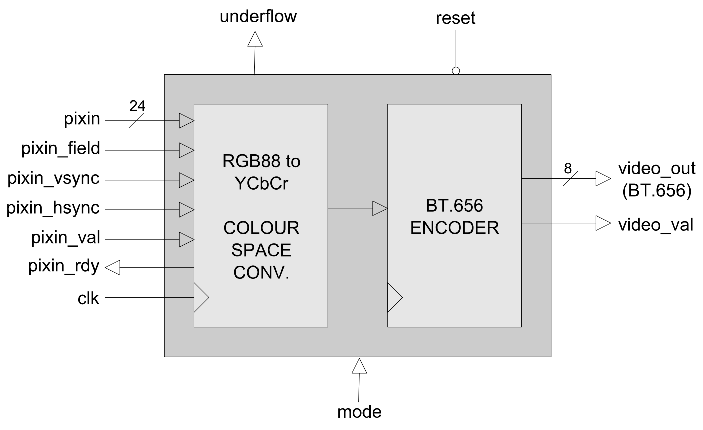

Figure 1: BT.656 Encoder architecture

General Description

BT_656_ENCODER (Figure 1) is a digital video encoder with integrated colour-space converter. The encoder accepts 24-bit RGB pixels from sequential odd and even fields. These pixels are then mapped to the YCbCr colour-space and formatted correctly into a BT.656 output stream.

The encoder begins operation after *reset* is de-asserted and on detection of input field '0'. Input pixels are sampled on the rising-edge of *clk* when the *pixin_val* and *pixin_rdy* signals are both asserted high. The *pixin_vsync* and the *pixin_hsync* flags are coincident with the first pixel of a field and line. The *pixin_field* flag indicates whether the field is odd or even.

The valid-ready flow-control interface shares a common format with all other Zipcores video IP and allows easy connectivity between IP cores. The valid-ready interface also allows simple connectivity with an input FIFO or external frame buffer.

The output of the encoder generates an industry standard ITU-R BT.656 format video stream together with a *video_val* signal that is asserted with the first valid byte of the output stream.

If the encoder is starved of pixels during the generation of an active line then the *underflow* flag will be asserted. In the event of an underflow condition, then design must be reset by asserting the reset signal low for at least one clock cycle. Operation will then resume as normal when the next input field is detected.

BT.656 Encoder

The encoder samples the incoming RGB pixels and looks for the first active line in field '0' (odd field). Once this is detected the generation of the BT.656 stream begins. Only active input pixels are processed by the encoder.

After all the lines in the odd field have been encoded, operation continues with the encoding of all active lines in field '1' (even field). The encoder then reverts back to field '0' once again.

After a system reset, the encoder will revert to it's initial state and stop generating the output stream. Encoding will then resume again with the first active line of field '0'.

When the generic parameter *mode* is set to '0', the encoder 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.

Note that only active pixels should be sent to the encoder. The encoder input is not concerned with input pixels during periods of vertical or horizontal blanking. All video timing information is automatically embedded in the BT.656 output stream.

Colour-space converter

The encoder features an integrated colour-space converter that converts the RGB inputs to YCbCr 4:2:2 format. Chroma values are decimated every second pixel to generate the 4:2:2 video. The colour-space conversion is done according to the following formula:

$$Y = 16 + 0.257\mathrm{R} + 0.504\mathrm{G} + 0.098\mathrm{B}$$
$$Cb = 128 - 0.148\mathrm{R} - 0.291\mathrm{G} + 0.439\mathrm{B}$$
$$Cr = 128 + 0.439\mathrm{R} - 0.368\mathrm{G} - 0.071\mathrm{B}$$

Functional Timing

Example input waveforms are shown in Figure 2. Input pixels and syncs are sampled on a rising clock-edge when *pixin_val* and *pixin_rdy* are both high. When *pixin_val* is low then the inputs are ignored by the encoder.

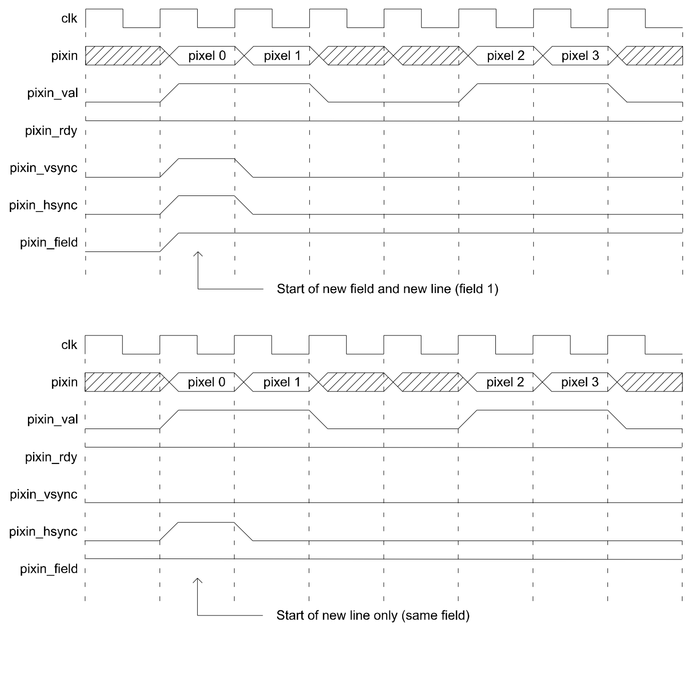

Figure 2: Input waveforms showing the start of a new field and start of a new line (pixin_val shown with 50% duty-cycle)

Figure 3 shows the output BT.656 video stream from the encoder. Bytes are transferred on a rising clock-edge when *video_val* is active high.

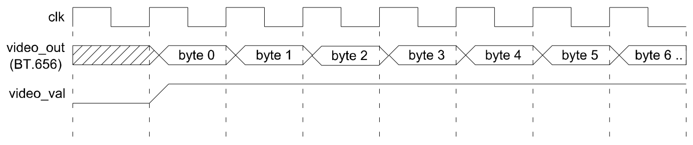

Figure 3: BT.656 output video stream

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***
video_in.txt	Source input video text file
pipeline_reg.vhd	Pipeline register element
fifo_sync.vhd	Synchronous FIFO
video_file_reader.vhd	Input video text file reader
bt_656_enc_sof.vhd	Field sync component
bt_656_enc_csc.vhd	Colour-space converter
bt_656_enc_422.vhd	Chroma re-sampler
bt_656_enc_pack.vhd	Pixel packer component
bt_656_enc_unpack.vhd	Pixel unpacker component
bt_656_enc_enc.vhd	Main BT.656 formatter
bt_656_encoder.vhd	Top-level component
bt_656_encoder_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_enc_sof.vhd
4. bt_656_enc_csc.vhd
5. bt_656_enc_422.vhd
6. bt_656_enc_pack.vhd
7. bt_656_enc_unpack.vhd
8. bt_656_enc.vhd
9. bt_656_encoder.vhd
10. bt_656_encoder_bench.vhd
11. video_file_reader.vhd

The VHDL testbench instantiates the BT_656_ENCODER 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 RGB pixels and flags on consecutive lines. The text file is called *video_in.txt* and should be placed in the top-level simulation directory.

The simulation must be run for at least 50 ms during which time the output BT.656 stream is captured to a text file called *bt_656_out.txt*.

Figure 4 below shows the results after encoding the a PAL video source into the two separate interlaced fields. The image also shows the video blanking regions and the vertical blue lines where the EAV and SAV codes are positioned in the video stream.

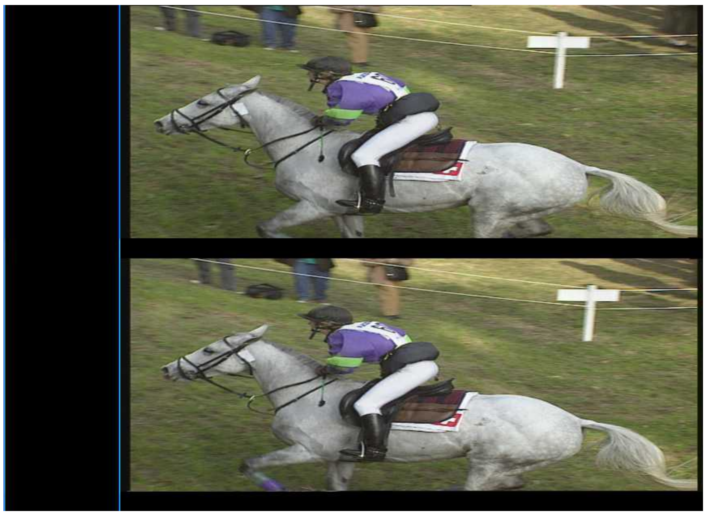

Figure 4: Output image from the simulation showing the separate fields in the BT.656 encoded video

Synthesis

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

- bt_656_encoder.vhd
  - bt_656_enc_sof.vhd
  - bt_656_enc_csc.vhd
    - pipeline_reg.vhd
  - bt_656_enc_422.vhd
    - pipeline_reg.vhd
  - bt_656_enc_pack.vhd
  - fifo_sync.vhd
    - pipeline_reg.vhd
  - bt_656_enc.vhd
  - bt_656_enc_unpack.vhd

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx® Virtex 6 and Spartan 6 FPGA devices. Synthesis results for other FPGAs 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 *mode* parameter set to '0' for 576i (PAL) video. The resource usage is specified after Place and Route.

VIRTEX 6

Resource type	***Quantity used***
Slice register	129
Slice LUT	204
Block RAM	2
DSP48	12
Occupied Slices	115
Clock frequency (approx)	300 MHz

SPARTAN 6

Resource type	***Quantity used***
Slice register	129
Slice LUT	219
Block RAM	4
DSP48	12
Occupied Slices	91
Clock frequency (approx)	160 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	12/03/2013
1.1	Added full valid-ready flow-control and better colour-space conversion	25/02/2014