Key Design Features

- Synthesizable, technology independent IP Core for FPGA, ASIC and SoC
- Supplied as human readable VHDL (or Verilog) source code
- Fully asynchronous 24-bit RGB video inputs
  (Option to support YCbCr video formats if required)
- 24-bit RGB video outputs synchronized to the system clock
- Generates clean and progressive output video without combing or tearing
- Excellent vertical resolution and much better than intra-line interpolation methods
- Supports three different deinterlacing modes including: Classic weave, inter-line interpolation and interpolation adapted for motion between fields
- Supports all interlaced video formats such as: 480i, 576i, 1080i etc. All modes are real-time programmable
- Integrated frame buffer dynamically skips and repeats frames in order to adapt to the desired input and output frame rates
- Diagnostic flags asserted in the event of an input or output buffer overflow
- Simple generic memory interface suitable for SDRAM, DDR, DDR2, DDR3 etc.
- Fully pipelined architecture with simple flow-control. Compatible with all other Zipcores video IP, AXI4-stream and Avalon-ST
- Supports 200MHz+ operation on basic FPGA platforms

Applications

- Studio-quality video de-interlacing
- Conversion of 'legacy' SDTV formats to HDTV video formats
- Generating progressive RGB video via inexpensive PAL/NTSC decoder chips
- Digital TV set-top boxes. Industrial imaging. Automotive, home and personal media solutions

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
deint_mode	Deinterlacing mode selection	integer	0: WEAVE 1: INTERP 2: MA
line_width	Width of linestores in pixels	integer	2^4 < pixels < 2^{16}
log2_line_width	Log2 of linestore width	integer	log2(line_width)
field_polarity	Swaps the polarity of the input field	boolean	True/False

Block Diagram

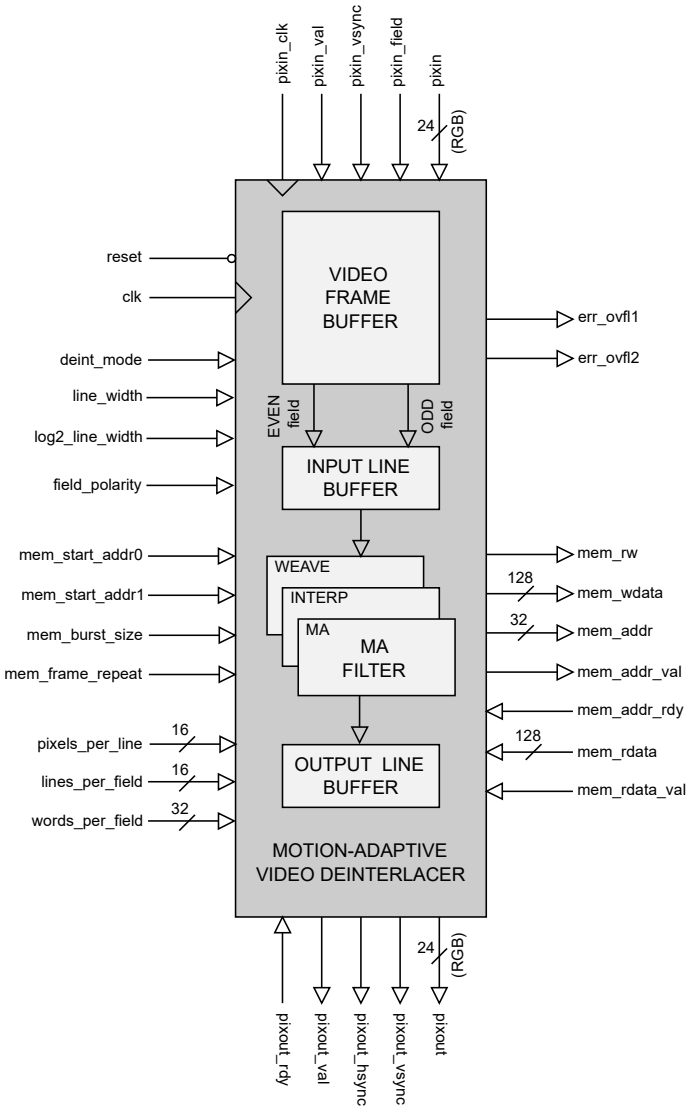

Figure 1: Basic video deinterlacer architecture

Generic Parameters cont...

Generic name	*Description*	*Type*	*Valid range*
mem_start_addr0	Start address of frame buffer 0 in memory	integer	≥ 0 (128-bit aligned)
mem_start_addr1	Start address of frame buffer 1 in memory	integer	≥ 0 (128-bit aligned)
mem_burst_size	Size of memory read/write burst (in 128-bit words)	integer	2, 4, 8, 16, 32 or 64
mem_frame_repeat	Enable/disable frame repeat mode	boolean	True/False

Pin-out Description

SYSTEM SIGNALS

Pin name	I/O	Description	Active state
clk	in	Synchronous system clock	rising edge
reset	in	Asynchronous reset	low
err_ovfl1	out	Input overflow error (signifies insufficient input memory B/W)	high
err_ovfl2	out	Output overflow error (signifies insufficient output memory B/W)	high
pixels_per_line [15:0]=	in	Number of pixels per input line	data
lines_per_field [15:0]	in	Number of lines per input field	data
words_per_field [31:0]	in	Number of 128-bit words per field Calculated as (pixels_per_line * lines_per_field * 24) /128 (Must be a whole number)	data

ASYNCHRONOUS INPUT VIDEO INTERFACE (INTERLACED)

Pin name	I/O	Description	Active state
pixin_clk	in	Input pixel clock	rising edge
pixin [23:0]	in	24-bit RGB pixel in	data
pixin_field	in	Input field number (Coincident with first pixel of a new input field)	0: even 1: odd
pixin_vsync	in	Vertical sync in (Coincident with first pixel of a new input field)	high
pixin_val	in	Input pixel valid	high

SYNCHRONOUS OUTPUT VIDEO INTERFACE (PROGRESSIVE)

Pin name	I/O	Description	Active state
pixout [23:0]	out	24-bit RGB pixel out	data
pixout_vsync	out	Vertical sync out (Coincident with first pixel of a new output frame)	high
pixout_hsync	out	Horizontal sync out (Coincident with first pixel of a new output line)	high
pixout_val	out	Output pixel valid	high
pixout_rdy	in	Ready to accept output pixel (handshake signal)	high

GENERIC 128-BIT MEMORY INTERFACE

Pin name	I/O	Description	Active state
mem_rw	out	Memory read / write flag	0: write 1: read
mem_wdata [127:0]	out	Memory write data	data
mem_addr [31:0]	out	Memory read / write address (128-bit aligned)	data
mem_addr_val	out	Memory request valid	high
mem_addr_rdy	in	Ready to accept memory request (handshake signal)	high
mem_rdata[127:0]	in	Memory read data	data
mem_rata_val	in	Memory read data valid	data

General Description

The DEINTERLACER_MA IP Core is a studio quality 24-bit RGB video deinterlacer capable of generating progressive output video at any resolution up to 2^{16} x 2^{16} pixels. The design is fully programmable and supports any desired interlaced video format.

The design allows for three possible deinterlacing schemes. These are: weave, bilinear interpolation or motion-adaptive interpolation. The weave approach applies no filtering and may be useful to obtain a 'raw' interlaced format for subsequent processing. The other two methods are classed as 'inter-field' interpolation methods as spatial filtering is performed between both odd and even fields to achieve a clean and progressive output. The relative merits and disadvantages of each scheme are discussed further into the document.

The deinterlacer core features a fully integrated video frame buffer. This buffer is completely 'elastic' and will dynamically skip and/or repeat frames depending on the input and output frame rates. All frame buffer management is handled internally with the provision of a simple memory interface for storing odd and even fields off-chip. The memory interface is 128-bits wide and is completely generic[1]. All memory transfers are sequential bursts of N x 128-bit words and may be adapted for connection to a variety of memory types such as SDRAM, DDR2 or DDR3.

The input video interface is asynchronous to the system clock. Input pixels are sampled on the rising clock-edge of *pixin_clk* with the signals *pixin_field* and *pixin_vsync* identifying the field number and the first pixel of each field. All signals are qualified by *pixin_val* asserted high.

Output pixels are synchronous with the system clock and are generated in accordance with a simple valid-ready streaming protocol. The output pixels and sync flags are transferred at the deinterlacer outputs on a rising clock-edge when *pixout_val* and *pixout_rdy* are both active high. If required, the application circuit may assert *pixout_rdy* low to stall the flow of output pixels.

The basic architecture of the motion-adaptive deinterlacer is shown in Figure 1.

1 Other memory word widths are available on request. We can also provide physical interfaces with your chosen memory technology. Please contact Zipcores for more information.

Pixels per line, lines per field and words_per_field

The programmable parameters *pixels_per_line* and *lines_per_field* define the format of the interlaced video input[2]. As an example, these values would be set as '720' and '240' if the input video format was digitized NTSC at 720 x 480 resolution (480i).

The width of the linestores must be sufficient to hold a complete line of interlaced video and the width should be set to the nearest power of 2. For example, if *pixels_per_line* is set to '720', then *line_width* should be set to '1024' and *log2_line_width* should be set to '10'.

The *words_per_field* parameter defines the total number of 128-bit words in a complete field. This must be a whole number. Note that when changing any of the programmable parameters, the deinterlacer must be reset for a least one system clock cycle before normal operation resumes.

Memory interface parameters

The memory interface parameters should be set according to the input video format. These parameters define both the physical memory map of the frame buffer and the way the frame buffer is accessed by the deinterlacer core.

Figure 2 shows a memory map and the relationship between the generic parameters *mem_start_addr0, mem_start_addr1* and *words_per_field*. The size of physical memory must be large enough to buffer both the odd and even fields as shown otherwise a memory conflict will occur.

In addition, it is important that the parameter *mem_burst_size* is set correctly to ensure that each burst is a whole number of sequential 128-bit bursts. In particular, the calculation (words_per_field/mem_burst_size) must result in a whole number.

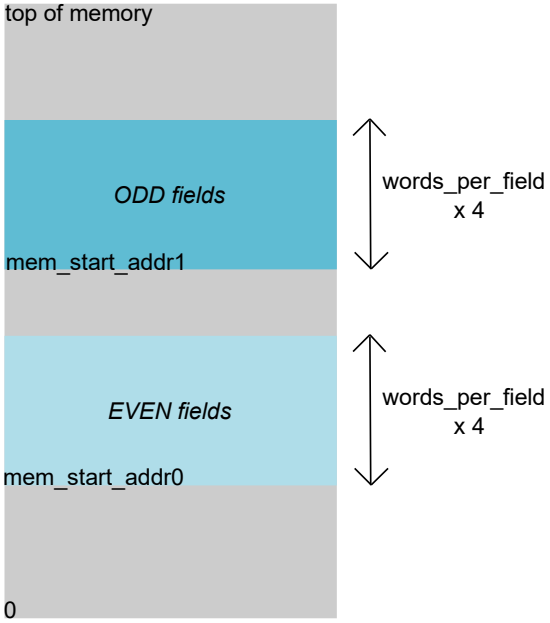

Figure 2: Frame buffer memory map

The following tables list the generic parameters for some common interlaced video formats.

1080i COMMON INTERLACED FORMATS

Mode	Pixels per line	Lines per field	Words per field	Mem burst size
1920 x 1080i	1920	540	194400	32
1440 x 1080i	1440	540	145800	8
1280 x 1080i	1280	540	129600	64

576i COMMON INTERLACED FORMATS

Mode	Pixels per line	Lines per field	Words per field	Mem burst size
1024 x 576i	1024	288	55296	64
960 x 576i	960	288	51840	64
768 x 576i	768	288	41472	64
720 x 576i	720	288	38880	32
704 x 576i	704	288	38016	64
544 x 576i	544	288	29376	64
480 x 576i	480	288	25920	64

480i COMMON INTERLACED FORMATS

Mode	Pixels per line	Lines per field	Words per field	Mem burst size
960 x 480i	960	240	43200	64
864 x 480i	864	240	38880	32
720 x 480i	720	240	32400	16
704 x 480i	704	240	31680	64
640 x 480i	640	240	28800	64
544 x 480i	544	240	24480	32
528 x 480i	528	240	23760	16
480 x 480i	480	240	21600	32
352 x 480i	352	240	15840	32

As a general rule, choosing the maximum burst size will result in the best possible synchronous memory performance due to reduced page-break cost.

2 Auto-detect of the interlaced input video format is an optional extra. Please contact Zipcores for more details.

Deinterlacing modes

The generic parameter *deint_mode* selects one of three possible deinterlacing schemes. These are WEAVE, INTERPOLATE or MOTION-ADAPTIVE. The following table outlines the basic characteristics of each scheme.

Deint_mode	***Description and properties***
0: WEAVE	Classic interleave approach. Odd and even lines are interleaved sequentially to generate a full frame of video. Gives excellent results for static images with the best possible vertical resolution. Moving video exhibits tearing or combing between fields. This mode is useful if raw interlaced video is required for subsequent processing or if the output video is static or slow moving e.g. electronic billboards, menus, etc. Results in the smallest hardware implementation size.
1: INTERP	This method uses bi-linear filtering between odd and even fields to generate a smooth interpolated image. It does tend to soften the image a little but the incidence of combing or tearing between fields is much less noticeable. Results in a medium size hardware implementation that is slightly larger than the weave approach.
2: MA	Most complex algorithm. Uses a 5x5 filter window to calculate motion vectors between odd and even lines. The spacial filtering is modified depending on the calculated vectors. Generates the best image quality with crisp, sharp edges and negligible combing artifacts. Results in the largest hardware implementation size.

Figure 3 demonstrates the visual effect of each deinterlacing mode on a moving ball in a video snapshot. Image (a) represents the original interlaced source image. Image (b) is the same image after inter-field interpolation. Image (c) shows the result after full motion-adaptive interpolation.

The most marked difference can be observed between the white spots on the ball. In the weave case, the characteristic combing is quite prominent. However, in the motion-adaptive case, the edges of each spot are quite well defined. The bilinear interpolated case gives a result somewhere between the two extremes.

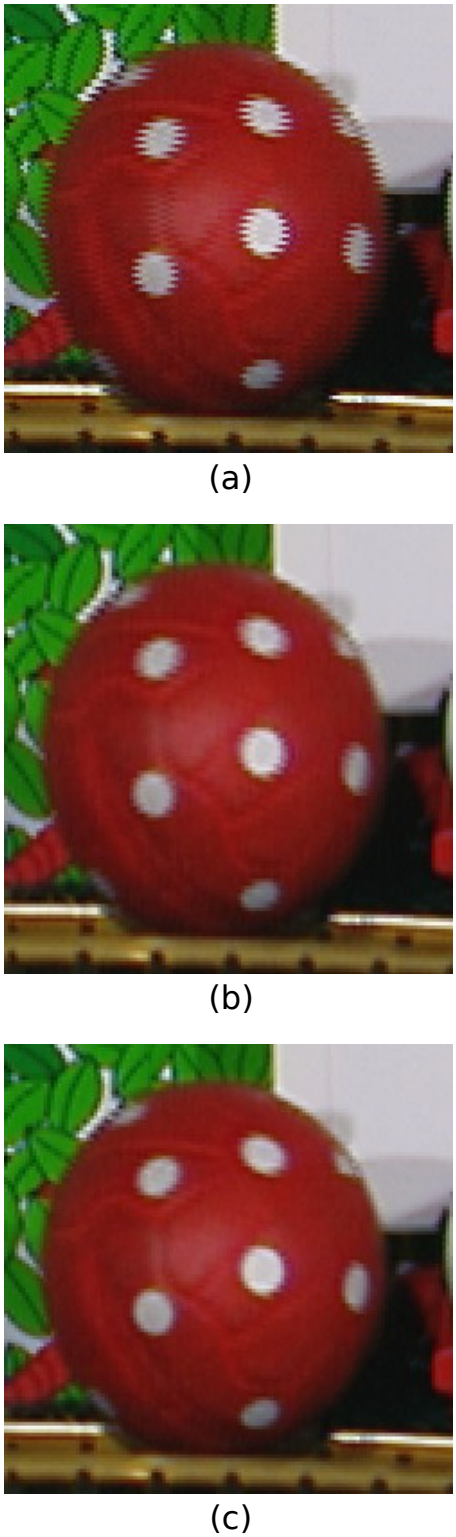

Figure 3: Visual effect of different deinterlacing modes: (a) Weave, (b) Interpolate & (c) Motion-adaptive

Buffer overflow conditions

If the input pixel data rate becomes too high for the internal frame buffer to tolerate, the input pixel FIFOs will overflow and the signal *err_ovfl1* will be asserted high. This happens when the instantaneous pixel-rate exceeds the maximum write bandwidth available. To prevent this condition, it is recommended that the system clock frequency (*clk*) is greater than the input pixel clock frequency (*pixin_clk*).

Likewise, if the output FIFOs overflow, the signal *err_ovfl2* will be asserted high. The output FIFOs have enough buffering to accommodate four 'in-flight' read memory bursts for a maximum burst size of 64. For this reason, the memory read latency must not exceed 256 system clock cycles. If a very high memory read latency is expected, then please contact Zipcores and the amount of internal buffering can be adjusted accordingly.

Note that if an overflow condition occurs, the only way to recover is to assert a system reset. After reset, the system will re-sync to the incoming video stream and normal operation will resume.

Functional Timing

Asynchronous input video interface

Figure 4 shows the signalling at the input to the deinterlacer at the start of a new field. The first line of a new field begins with *pixin_vsync* asserted high together with the first pixel. When *pixin_val* is de-asserted then input pixel is ignored. The signal *pixin_field* is a flag that identifies whether the input field is odd or even. This flag is only sampled at the start of a new field when *pixin_vsync* and *pixin_val* are high.

(Note: The polarity of the *pixin_field* flag can be changed using the *field_polarity* generic. This means that an ODD field can be interchanged for an EVEN field and vice-versa depending on the True/False setting. If the field polarity is set incorrectly then it will result in a poor quality image).

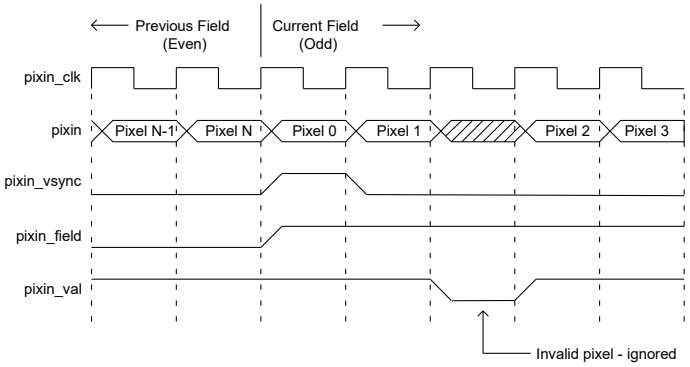

Figure 4: Input video interface timing

Synchronous output video interface

Output pixels and syncs are transferred out of the deinterlacer on the rising clock-edge of *clk* when *pixin_val* and *pixin_rdy* are both high. If *pixin_rdy* is held low, then the output is stalled and the internal frame-buffer will buffer input pixels (or whole frames) until *pixin_rdy* is asserted high again.

Figure 5 shows the output video timing at the start of a new output frame. Both *pixin_vsync* and *pixin_hsync* are asserted high with the first pixel of a new frame.

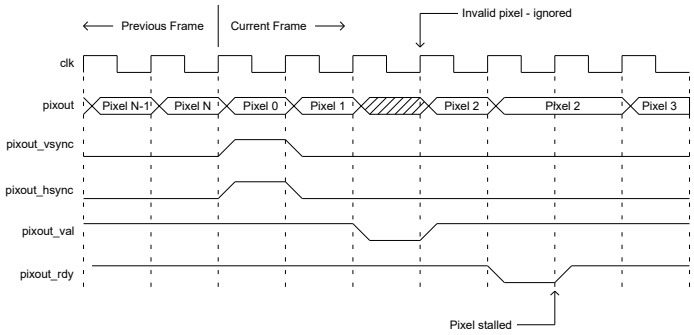

Figure 5: Output video interface timing - start of new output frame

Figure 6 demonstrates the timing at the start of a new line. A new line begins with *pixin_hsync* coincident with the first pixel. The signal *pixin_vsync* is held low.

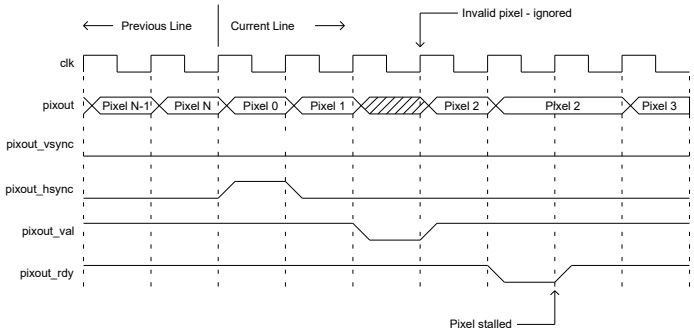

Figure 6: Output video timing - start of new output line

Generic 128-bit memory interface

Figure 7 shows a series of write bursts to memory. In this particular example, the parameter *mem_burst_size* has been set to 4. Each memory burst is a block write of 4 words[3]. The addresses are guaranteed to be sequential within a burst. Between bursts, the *mem_addr_valid* signal is de-asserted for one cycle.

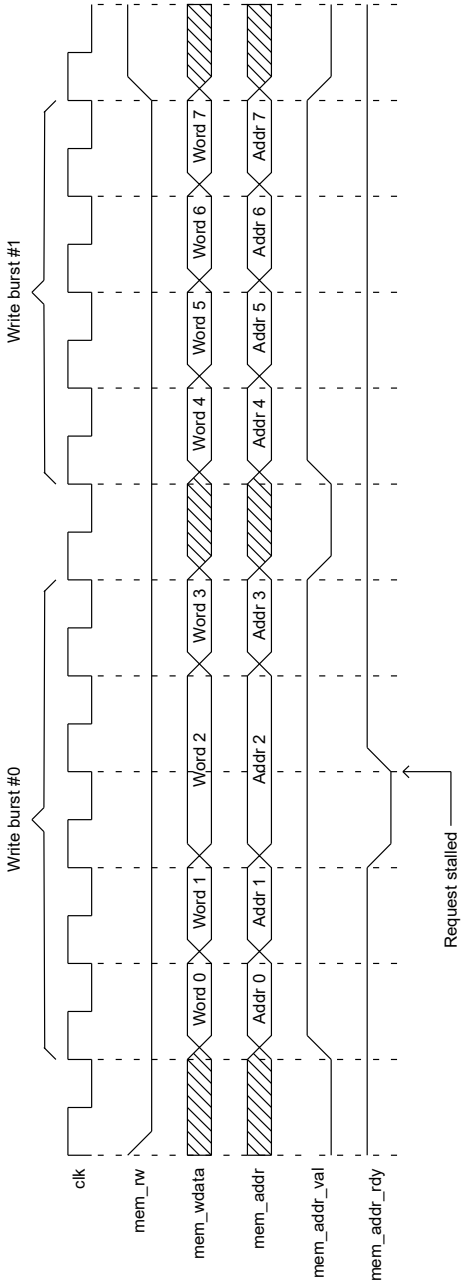

Figure 7: Memory write burst timing (burst size of 4)

The timing is very similar for a read burst. Figure 8 shows single read burst and corresponding read data returned from memory.

The memory interface is also compatible with many third party tools. Examples include those provided by Xilinx® (ISE/Vivado) and Altera® memory interface generator IP (Quartus).

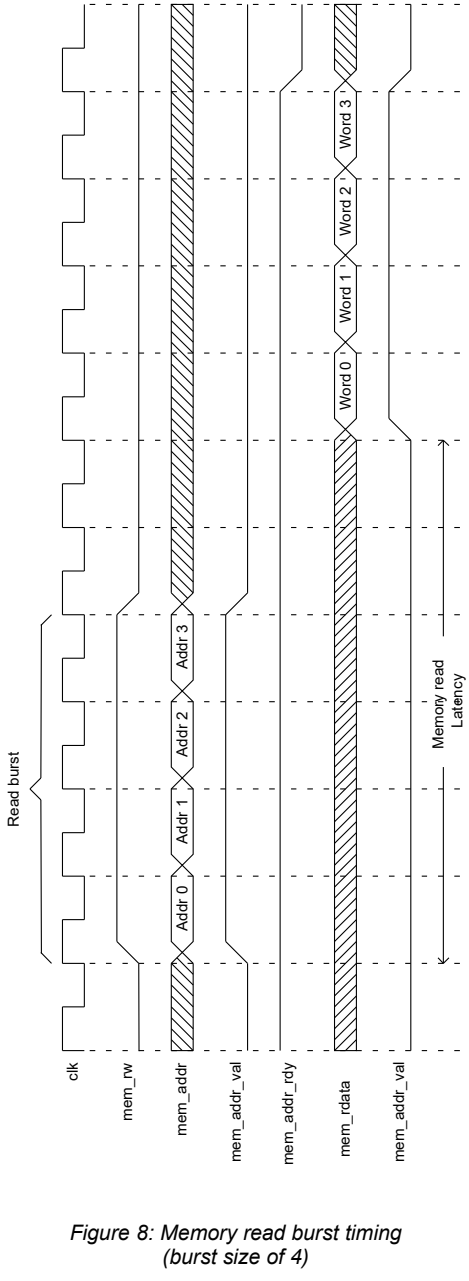

Figure 8: Memory read burst timing (burst size of 4)

3 A larger burst size is advised for synchronous memory types to reduce page-breaks. A burst size of 4 is shown for illustration only.

Source File Description

All 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 video file
deint_file_reader_ma.vhd	Reads text-based source video file
mem_model_pack.vhd	Memory model functions
ram_model.vhd	Single port memory model
mem_model_1Mx128bit.vhd	Large 1Mx128 memory model
pipeline_reg.vhd	Pipeline register element
pipeline_shovel.vhd	Pipeline register 'shovel' element
fifo_sync_bit.vhd	Generic 1-bit FIFO
fifo_sync_ram.vhd	Generic RAM-based FIFO
fifo_sync_reg.vhd	RAM-based FIFO internal register
vid_in_reg.vhd	Video input register
vid_out_reg.vhd	Video output register
vid_async_fifo.vhd	Asynchronous pixel FIFO
vid_sync_fifo.vhd	Synchronous pixel FIFO
vid_sync_fifo_reg.vhd	Sync FIFO internal register
ram_dp_w_r.vhd	Dual port RAM component
vid_align_frame.vhd	Aligns pixels to the start of frame
vid_pack128.vhd	Pixel packer
pack_16_to_32.vhd	16-bit to 32-bit packer
pack_24_to_32.vhd	24-bit to 32-bit packer
pack_32_to_32.vhd	32-bit to 32-bit packer
pack_32_to_128.vhd	32-bit to 128-bit packer
vid_frame_fifo.vhd	Main frame-FIFO controller
vid_mem_write.vhd	Memory write burst controller
vid_mem_read.vhd	Memory read burst controller
vid_mem_arb.vhd	Memory R/W arbiter
vid_unpack128.vhd	Pixel unpacker
unpack_32_to_16.vhd	32-bit to 16-bit unpacker
unpack_32_to_24.vhd	32-bit to 24-bit unpacker
unpack_32_to_32.vhd	32-bit to 32-bit unpacker
unpack_128_to_32.vhd	128-bit to 32-bit unpacker
vid_sync_regen.vhd	Video sync generator
vid_frame_buffer.vhd	Video frame buffer
vid_mem_arb_dual.vhd	Memory R/W arbiter (even/odd fields)
deint_field_mux.vhd	Input field multiplexer
deint_line_buffer.vhd	Line buffer for even/odd fields
deint_filter_ma.vhd	Deinterlacer pixel filter
deinterlacer_ma.vhd	Top-level deinterlacer component
deinterlacer_ma_bench.vhd	Top-level test bench component

Functional Testing

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

The VHDL testbench instantiates the deinterlacer component and the user may modify the generic parameters in accordance with the desired interlaced video format and the desired filtering scheme. In the example provided, the input format has been set to 720x480i and the deinterlacing mode set to '2' for full motion-adaptive.

The component 'deint_file_reader_ma.vhd' reads the input source video for the simulation. This component reads a text-based file which contains the RGB pixel data and sync information. The text file is called *video_in.txt* and should be placed in the top-level simulation directory.

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

```
1  0  1  00 11 22  # pixel 0, line 0, start of field 0
1  0  0  33 44 55  # pixel 1
1  0  0  66 77 88  # pixel 2
1  0  0  99 00 11  # pixel 3
.
.
1  1  1  00 11 22  # pixel 0, line 0, start of field 1
1  1  0  33 44 55  # pixel 1
1  1  0  66 77 88  # pixel 2
1  1  0  99 00 11  # pixel 3
.
.
etc..
```

In this example, the first line of of the *video_in.txt* file asserts the input signals pixin_val = 1, pixin_field = 0, pixin_vsync = 1 and pixin = 0x001122.

The simulation must be run for at least 50 ms during which time an output text file called *video_out.txt* will be generated. This file contains a sequential list of 24-bit output pixels. Figure 9 shows the resulting output frame generated by the test.

Figure 9: Output frame from test bench example

Development Board Testing

The deinterlacer IP core was fully tested using a live PAL (576i) and NTSC (480i) video source to review the subjective image quality for the different deinterlacing schemes. The hardware setup included the Zipcores HD-Video development board[4] with a Samsung DVD player providing the CVBS video source. The deinterlacer IP Core was implemented using the Sparan6 FPGA on the devboard together with some basic IP for decoding the BT.656 stream and generating the correct video output timing for the progressive output video. Figure 10 shows a basic block diagram of the hardware setup.

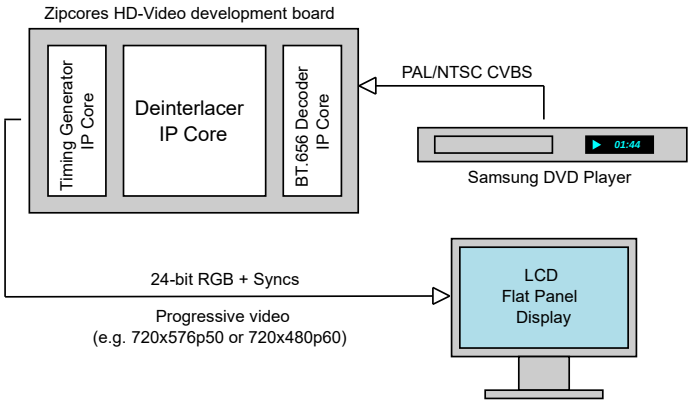

Figure 10: Block diagram of the test setup

Figure 11 is a photo of the hardware arrangement showing the Zipcores development board with CVBS video input and the LCD display. Different live video streams were used to review the subjective image quality.

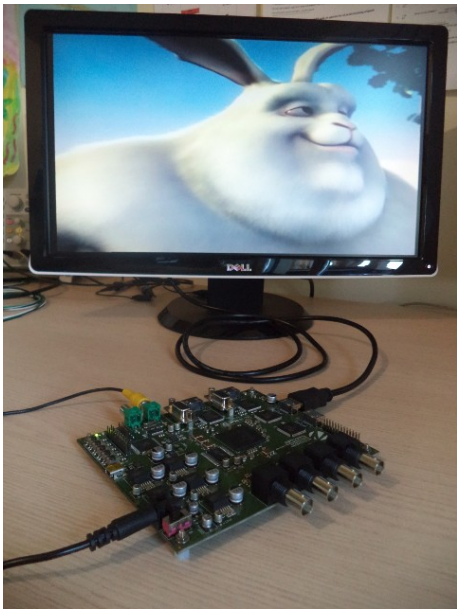

Figure 11: Photo of the test setup

As expected, it was found that the weave approach gave the best possible vertical resolution for static and slow-moving video sequences.

Bilinear interpolation gave good all round performance with no significant combing artefacts. The resultant image did appear a little 'softer' in the vertical dimension compared to weave.

Overall, the best performance was given by the motion-adaptive approach. No combing or tearing was evident in a range of fast moving sequences. Edge definition and contrast were much improved over the simple bilinear case.

All three schemes exhibited good stability with no 'bob' or vibration evident between adjacent interlaced lines.

Synthesis and Implementation

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

- deinterlacer_ma.vhd
  - deint_field_mux.vhd
  - deint_line_buffer.vhd
  - deint_filter_ma.vhd
  - fifo_sync_ram.vhd
    - ram_dp_w_r.vhd
    - fifo_sync_reg.vhd
  - vid_mem_arb_dual.vhd
    - pipeline_shovel.vhd
    - fifo_sync_bit.vhd
  - vid_frame_buffer.vhd
    - vid_in_reg.vhd
    - vid_async_fifo.vhd
    - vid_align_frame.vhd
    - vid_pack128.vhd
      - pack_16_to_32.vhd
      - pack_24_to_32.vhd
      - pack_32_to_32.vhd
      - pack_32_to_128.vhd
    - vid_sync_fifo.vhd
      - ram_dp_w_r.vhd
      - vid_sync_fifo_reg.vhd
    - vid_frame_fifo.vhd
      - vid_mem_write.vhd
      - vid_mem_read.vhd
    - vid_mem_arb.vhd
      - pipeline_reg.vhd
    - vid_sync_fifo.vhd
      - ram_dp_w_r.vhd
      - vid_sync_fifo_reg.vhd
    - vid_unpack128.vhd
      - unpack_32_to_16.vhd
      - unpack_32_to_24.vhd
      - unpack_32_to_32.vhd
      - unpack_128_to_32.vhd
    - vid_sync_regen.vhd
    - vid_out_reg.vhd
      - pipeline_reg.vhd

The VHDL IP core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx® 7-series FPGAs. Synthesis results for other FPGAs and technologies can be provided on request. Note that choosing the WEAVE deinterlacing mode results in the smallest and fastest implementation. The MA mode results in the largest implementation size. Careful attention must be made to the width of the linestores as this will effect the amount of RAM resource used.

4 See: http://www.zipcores.com/hd-video-development-board.html