1) How do the pixin_vsync and pixin_hsync relate to the VSYNC and HSYNC video timing signals at the DAC?

The signals *pixin_vsync* and *pixin_hsync* should not be confused with true video timing signals as such - i.e. the VSYNC and HSYNC at a video DAC for instance. Their purpose is to identify the first pixel in a frame and the first pixel in a line only.

The signal *pixin_vsync* is coincident with the first pixel of an input frame. The signal *pixin_hsync* is coincident with the first pixel of an input line. Likewise, the signals *pixout_vsync* and *pixout_hsync* are coincident with the first pixels of an output frame or line. Figure 1. shows the simplified timing waveforms.

If it helps, the pixin_vsync and pixin_hsync signals can be considered as two extra bits of sideband information attached to the pixel. Also remember that, like the pixel, the sync signals are qualified by the valid. When valid is low then the syncs are ignored.

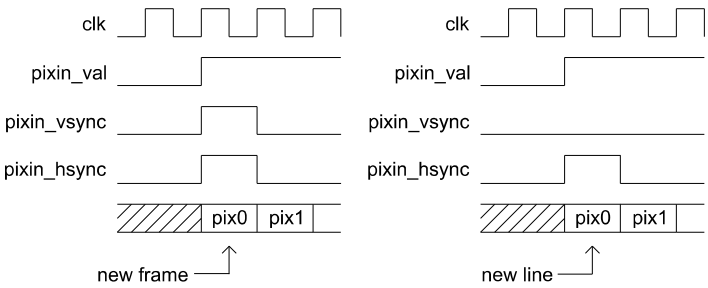

Figure 1: pixin_vsync and pixin_hsync timing

Essentially, the video scaler uses a high-speed data-streaming interface. It only processes active pixels and does not waste bandwidth by processing 'redundant' periods of horizontal or vertical blanking.

2) Why doesn't the video scaler generate output pixels immediately?

After reset, the scaler will discard input pixels until the first input frame is found. This will be when *pixin_vsync* and *pixin_val* are both high. From this point onwards, the scaling operation begins.

Depending on the scaler type, it may take up to 2 lines of input video before any output video is generated. This is due to the internal line buffers being pre-filled with pixels.

3) How do the video scalers generate pixels and lines at the start and at the end of each line or frame?

At the start of a new frame, the video scaler filter taps in x and y are empty (or at least contain invalid pixels). For this reason the scalers treat the pixels differently at the frame boundaries.

At the beginning of a new line or frame then the first pixel or line is duplicated in the filter taps. At the end of a line or frame then the last pixel or line is replicated. In this way, it ensures that the output pixels at the frame borders remain uncorrupted and true to the original source image.

4) The output video looks corrupted. I think I'm doing something wrong ...

The only way that the output video can be corrupted is if pixels are lost or repeated at the scaler interfaces. This can happen when the valid-ready protocol is used incorrectly. Pixels and syncs are sampled at the input (or output) of the scaler on a rising clock-edge when valid and ready are both high. It's important to wire up valid and ready correctly on both scaler interfaces - irrespective of whether down-scaling or up-scaling. Failure to do so will result in corrupted video. It may be easier to understand valid-ready signalling in terms of FIFO nomenclature. See FAQ 5. for an example wiring diagram.

5) How can I interface the video scaler to an input FIFO or output FIFO? Which Xilinx or Altera FIFOs should I use?

It's very simple to connect a FIFO to the input or output of our video scalers. Figure 2. shows an example wiring diagram.

On the input side of the scaler, the *pixin_val* input should be wired to the 'not empty' flag of the FIFO. The *pixin_rdy* output should be wired to the read-enable port of the FIFO. On the output side, The *pixout_val* signal should be wired to the write-enable port of the FIFO. The *pixout_rdy* signal should be wired to the 'not full' flag. The sync signals should be bundled together with the pixel.

It is important to note that in order for the flow-control to work correctly, the 'empty' and read flags of the FIFO should have zero latency. That is, the flags should immediately reflect the state of the FIFO. For instance, if the empty flag is updated one cycle late, then an input pixel may be incorrectly read. Likewise, if the full flag is updated one cycle late, then an output pixel could be lost.

Xilinx® and Altera® offer FIFO solutions in their IP libraries. For compatibility with the video scalers, the Xilinx FIFO should be generated with the FIRST-WORD-FALL-THROUGH option (FWFT). In the Altera case, the SHOW-AHEAD-SYNC-FIFO-MODE should be selected.

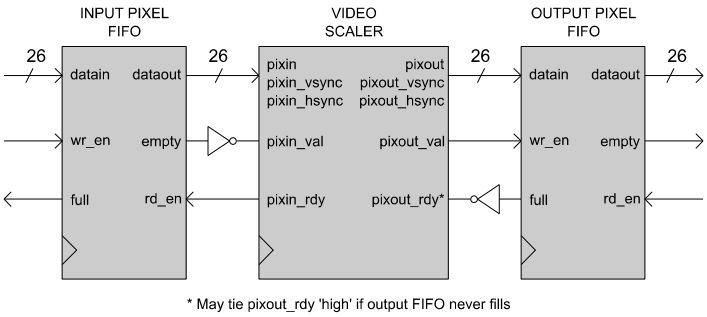

Figure 2: Video scaler FIFO wiring

6) I've sent one complete frame of video to the scaler but I don't get one complete frame out. Am I doing something wrong?

For every complete input video frame the scaler will produce one complete *scaled* output video frame. If this behaviour is not observed, then there are a couple of common things to check. One possibility is that the valid-ready protocol at the interfaces is not being observed correctly. This will result in pixels being lost at the interfaces. The other is incorrect scaler parameters. Always ensure that the number of pixels per line and lines per frame have been specified correctly for the chosen scale factor.

7) Can I multiplex different video sources into the same scaler?

Yes, this is easily done - as long as the video input sources are multiplexed cleanly between frames. The circuit should be designed so that *pixin_val*, *pixin_vsync* and *pixin_data* are swapped between each source at the end of each complete input frame. This is often easiest to do during vertical blanking periods. Also make sure that the scaling parameters are correct for each respective source. Figure 3. shows a possible arrangement for scaling multiple video sources with the same video scaler.

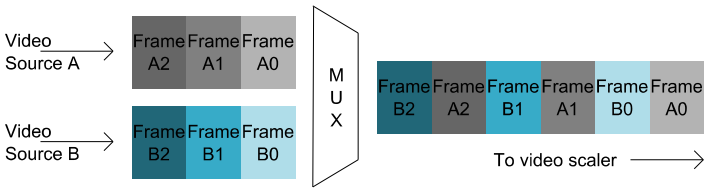

Figure 3: Multiplexing video streams into the video scaler

8) There are long periods when pixin_rdy goes low – why is this?

There are a number of situations where this might happen. The first instance is at the start of a new frame. In this case, there is a delay as the first pixel taps are filled both horizontally and vertically (see FAQ 2). When up-scaling, *pixin_rdy* will go low due to the change in ratio.

For optimum performance, it is essential that the input frame size is correct for the given scaling parameters. In addition, the user must ensure that the input size frame is at least 16x16 pixels – otherwise the scaler will fail.

9) How do I recover from a corrupted input video frame?

The safest way to recover is to assert a system reset (active low) for at least 1 system clock cycle. As the system reset is asynchronous, make sure that the signal is clocked though a couple registers to avoid possible removal time issues. Once reset, the video scaler will then re-align to the next start of frame and continue operation as normal.

10) Can I use the video scaler without a frame buffer?

Yes, it's possible to use the video scaler without a frame buffer. However, the design must ensure that the frame rate in is *exactly* the same as the frame rate out. If this is not the case, then the input and output video will become out of sync and eventually the system will fail.

In order to maintain the correct relationship between input and output frame rates, then the input pixel clock and the output pixel clock must have a fixed relationship. Most FPGAs have PLL resources that easily allow multiple clocks to be generated from a single source.

For instance, consider the case where a VGA input source at 60Hz frame rate is scaled-up to XGA at 60Hz. The VGA pixel clock is 25MHz and the output XGA pixel clock is 65MHz. A PLL may be used with the ratio 13/5 to generate the 65MHz pixel clock from the 25MHz clock source. In this way, both input and output pixel clocks are related and the input and output video will not become out of sync.

In addition, when using the scaler without a frame buffer, make sure that there is enough input buffering for a few lines of video (see FAQ 2). Input/output buffering is normally implemented using a FIFO arrangement similar to that shown in FAQ 5.

11) Can I change the scale parameters on-the-fly?

The scale parameters can be changed on a frame-by-frame basis if needed. The only requirement is that the user must assert the system reset after the parameters are changed. This allows the scaler to re-synchronize to the new parameters and lock to the next start of frame.

Generally it's convenient to change the parameters and toggle the reset during vertical blanking when there are no active pixels. This will ensure uninterrupted operation.

12) Is the scaler IP Core compatible with all FPGA and ASIC vendors? Can you give me timing and area figures for a specific technology?

The video scalers are provided as generic VHDL (or Verilog) source code that is compatible with all major FPGA vendors and technologies. Specific timing and area figures can be provided on request.

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	17/08/2009
1.1	Added 6,7	22/01/2010
1.2	Added 8,9	09/03/2010
1.3	Added 10. Updated logos and fixed typos	23/08/2011
1.4	Added 11. Some minor corrections and additions to text	21/01/2013
1.5	Added 12. Modified 11 in keeping with new code changes.	20/05/2013