DMA Transfer at Higher Pixel Rates

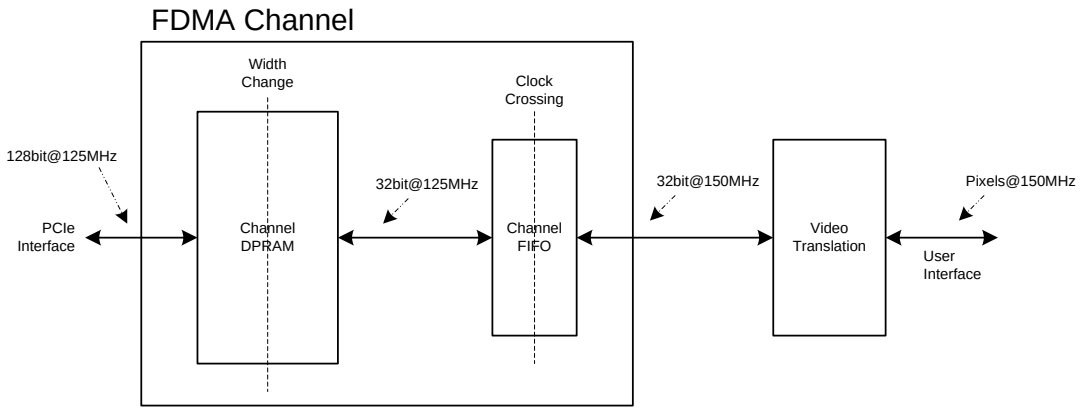
The OmniTek Multi-Channel Streaming DMA Controller supports the transfer of data streams such as those associated with video across a PCI Express (PCIe) interface through its FIFO-based FDMA channels.

The Controller is available as an IP block for implementation on a range of 28nm, 40nm and 60nm Altera and Xilinx FPGAs. The FPGAs all feature built-in PCIe hard IP blocks but operate at different speeds which in turn affects the speed of the PCIe interface clock and the bandwidth of the built-in transceivers.

These factors have a strong effect on the capacity of the FPGA to deliver video with the result that, in some cases, the FPGA becomes a bottleneck because it is unable to keep up with the video input.

It is, however, possible to configure the DMA controller and other elements of your video system to double its throughput – as explained in this Application Note.

The Basic Architecture



The Basic Problems

The process of transferring video over a DMA channel works very well in many cases but it is open to issues both with channel width and with the speed of data transfer.

Channel Width

Most video is transferred over an FDMA channel that is configured to be 32bits wide. Prior to this transfer – and outside the main hierarchy of the DMA controller – the video is repacked to make good use of the channel width. For example, 4:2:2 10bit sources which in their native form require a 20bit wide interface are typically repacked so that three pixels occupy two 32bit words. This 'width translation' works for most video formats but some video formats exceed the bandwidth of a 32bit-wide DMA channel.

Relatively Slow PCIe Interface

The internal architecture of the DMA channel means that, within the channel, user data is first passed onto the PCIe clock domain then width translation is performed, ready for the data to be passed to the PCIe Hard IP block.

Where the PCIe interface clock is slower than the video pixel clock, the DMA channel becomes a bottleneck. The above diagram shows how this can happen with a 125MHz PCIe interface.

This issue commonly arises when trying to use the DMA controller to transfer 30bit 4:4:4 1080p60 on an older-style 40nm or 60nm device.

The Solution…

The solution to both these issues is to use a 64bit wide DMA channel. This resolves the width issue by doubling the number of bits available to pack the pixel data into. It removes the potential for bottlenecks by doubling the available bandwidth.

The option to specify a channel as 64 bits wide has therefore been added to the DMA Controller's Configuration GUI. This option is available for both input and output FDMA channels as the issue affects both directions.

But Additional Requirements

The use of one or more 64bit DMA channel does however place some additional requirements on the system design.

1. The system design **must** accommodate the FIFO at the heart of the FDMA channel.
2. An even number of 32bit words must be transferred.

The reason for this is that data is passed into the FIFO at the configured channel width. So in a 64bit channel, an even number of 32bit words must be transferred because otherwise a single 32bit word will be left in the DPRAM, waiting for the rest of a 64bit word to arrive before it can be passed into the FIFO.

In general, this should not cause any issues for FDMA input (or capture) channels as all common video formats have an even number of 32bit words as pixel data. However adding an Avalon-ST Video control packet to the video data for host-to-card traffic can cause the total amount of data needing to be transferred to become an odd number of 32bit words.

To work around this issue, the following special packing for Avalon-ST Video headers must be used when the 64bit interface is selected. *Note:* The details of this packing are only of concern to users writing their own software application. The supplied DMA drivers are automatically informed of the width of each DMA channel and know which type of header definition to use in response.

Note: The entry '**skipped**' in the following packing details indicates that the Video Translation Block will discard the marked word when it assembles the Video bus.

10 bit – 4:2:2 or 4:4:4 on 64bit interface

31 24	23 20	19 14	13 10	9 4	3 0	
	0		0		F	Word 0
skipped						Word 1
	W[7:4]		W[11:8]		W[15:12]	Word 2
	H[11:8]		H[15:12]		W[3:0]	Word 3
	I		H [3:0]		H [7:4]	Word 4
	0		0		0	Word 5

8 bit – 4:2:2 on 64bit interface

31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0	
skipped					0		F	Word 0
	W[3:0]		W[7:4]		W[11:8]		W[15:12]	Word 1
	H [3:0]		H [7:4]		H[11:8]		H[15:12]	Word 2
	0		0		0		I	Word 3

8 bit – 4:4:4 on 64bit interface

31 20	19 16	15 12	11 8	7 4	3 0	
	0		0		F	Word 0
skipped						Word 1
	W[7:4]		W[11:8]		W[15:12]	Word 2
	H[11:8]		H[15:12]		W[3:0]	Word 3
	I		H [3:0]		H [7:4]	Word 4
	0		0		0	Word 5

8 bit – 4:4:4:4 on 64bit interface

31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0	
	0		0		0		F	Word 0
skipped								Word 1
	W[3:0]		W[7:4]		W[11:8]		W[15:12]	Word 2
	H [3:0]		H [7:4]		H[11:8]		H[15:12]	Word 3
	0		0		0		I	Word 4
	0		0		0		0	Word 5