Video In to AXI4-Stream v1.0

Product Guide

PG043 April 24, 2012

XILINX®

XILINX®

Table of Contents

Introduction

The Xilinx LogiCORE™ IP Video In to AXI4-Stream core is designed to interface from a video source (clocked parallel video data with synchronization signals - active video with either syncs, blanks or both) to the AXI4-Stream Video Protocol Interface. This core works in conjunction with the timing detector portion of the Xilinx Video Timing Controller (VTC) core. This core provides a bridge between a video input and video processing cores with AXI4-Stream Video Protocol interfaces.

Features

- Video (clocked parallel video data with synchronization signals - active video with either syncs, blanks or both) input
- AXI4-Stream Video Protocol interface for output
- Interface to Xilinx Video Timing Controller core for video timing generation
- Handles asynchronous clock boundary crossing between video clock domain and AXI4-Stream clock domain
- Selectable FIFO depth from 64 -8192 locations
- Selectable input data width of 8-64 bits

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Zynq-7000, Artix-7, Virtex®-7, Kintex®-7, Virtex-6, Spartan®-6
Supported User Interfaces	AXI4-Stream(2)
Resources	See Table 2-1.
Provided with Core	
Documentation	Product Guide
Design Files	Verilog Source Code
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Models	Verilog Source Code
Tested Design Tools	
Design Entry Tools	ISE 14.1
Simulation(3)	Mentor Graphics ModelSim, Xilinx® ISim
Synthesis Tools	Xilinx Synthesis Technology (XST)
Support	
Provided by Xilinx, Inc.	

1. For a complete listing of supported devices, see the release notes for this core.
2. Video protocol as defined in the *Video IP: AXI Feature Adoption* section of UG761 AXI Reference Guide.
3. For the supported versions of the tools, see the ISE Design Suite 14: Release Notes Guide.

Overview

Many Xilinx video processing cores utilize the AXI-4 Stream video protocol to transfer video between cores. Between systems, video is commonly transmitted with explicit blanking and sync signals for horizontal and vertical timing, and a data valid signal. DVI is an example of such a transmission mode. The Video In to AXI4-Stream core converts incoming video with explicit sync and timing to the AXI4-Stream Video protocol in order to interface to Xilinx video processing cores that use this protocol.

The Video In to AXI-4 Stream core accepts video inputs. For the purposes of this document, video is defined as parallel video data along with a pixel clock and one of the following sets of timing signals:

- Vsync, Hsync, and DE
- Vblank, Hblank, and DE
- Vsync, Hsync, Vbank, Hblank, and DE

Any of these sets of signals is sufficient for the operation of the Video In to AXI4-Stream core. The particular choice is important to the VTC detector, so the generation of the VTC core should specify which set of timing signals will be used. The output side of the core is an AXI4-Stream interface in master mode. This interface consists of parallel video data, `tdata`, handshaking signals `tvalid` and `tready`, and two flags, `tlast` and `tuser` which serve to identify certain pixels in the video stream. The flag `tlast` designates the last valid pixel of each line, and is also known as the end of line (EOL). The flag `tuser` designates the first valid pixel of a frame, and is known as start of frame (SOF). These two flags are necessary to identify pixel locations on the AXI4 stream bus because there are no sync or blank signals. Only active pixel are carried on the bus. The *Video IP: AXI Feature Adoption* section of the UG761 AXI Reference Guide describes the video over AXI4 Stream Video protocol in detail.

A Video In to AXI4-Stream core with a video timing generator block diagram is shown in Figure 1-1

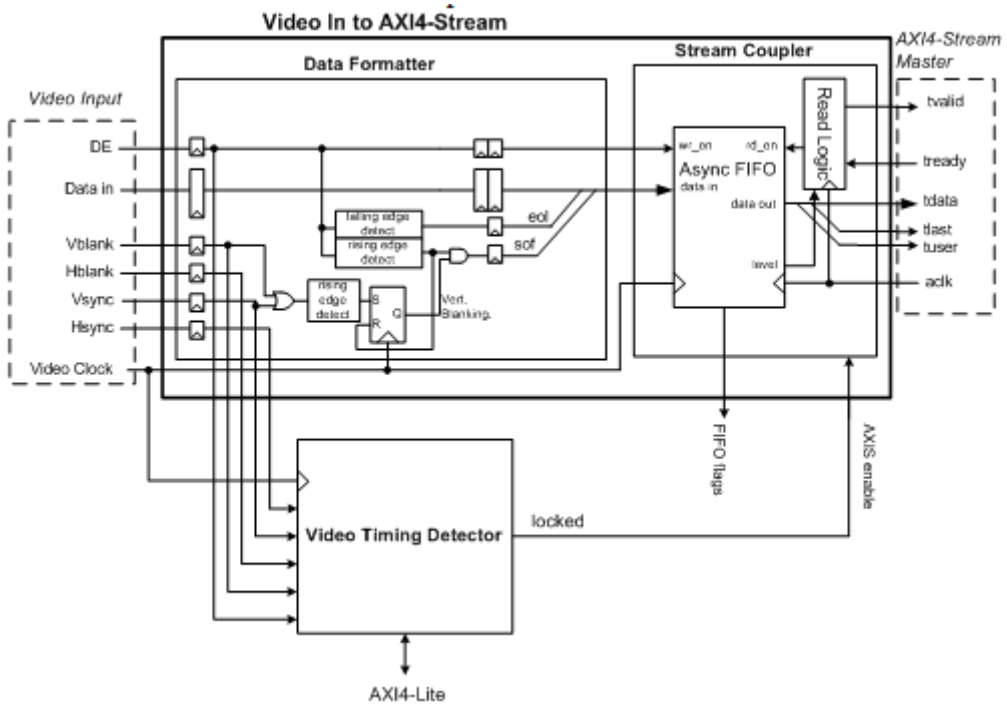

Figure 1-1: **Block Diagram of Video In to AXI4-Stream Core and Usage with the Video Timing Controller**

The core is designed to be used in parallel with the detector functionality of the Video Timing Controller (VTC). The video timing detector detects the line standard of the incoming video, and makes the detected timing values, such as the number of active pixels per line and the number of active lines available to video processing cores downstream of the Video In to AXI4-Stream core via an AXI4-Lite interface. It is recommended to connect the "locked" status output of video timing detector to the `axis_enable` input of the Video In to AXI4-Stream core in order to inhibit the AXI4-Stream bus when the video input is missing or unstable.

Feature Summary

The Video In to AXI4-Stream core converts a video input, consisting of parallel video data, video syncs, blanks and data enable, to an AXI4-Stream master bus that follows the AXI4-Stream Video protocol. The core provides a pass thru of all timing signals for the Xilinx video timing controller, although the signals for the Video timing Controller are not required to pass through the Video In to AXI4 Stream core.

The core handles the asynchronous clock boundary crossing between the video clock domain and the AXI4-Stream clock domain. The data width is selectable from 8 to 64 bits,

depending on the number of components required for the video format, and the number of bits per component. There is an input FIFO with selectable depth from 32 to 8192 locations.

Applications

- Video input to AXI4-Stream Video Protocol interface for parallel, clocked video sources
  - DVI
  - HDMI
  - Image Sensors
  - Other clocked, parallel video sources

Licensing

The Video In to AXI4-Stream core is provided at no cost with the Xilinx tools. Use of it is covered by the standard software license, and there are no further licensing requirements needed to use it in a design. Source code for this core is provided to enable customers to modify the core to meet their own unique requirements if necessary. Use of this core or any portions of this core is only allowed on a Xilinx device.

Product Specification

Standards Compliance

The Video In to AXI4-Stream core is compliant with the AXI4-Stream Video Protocol and AXI4-Lite interconnect standards. Refer to the *Video IP: AXI Feature Adoption* section of the UG761 AXI Reference Guide for additional information.

Performance

The following sections detail the performance characteristics of the Video In to AXI4-Stream core.

Maximum Frequencies

This section contains typical clock frequencies for the target devices. The maximum achievable clock frequency can vary. The maximum achievable clock frequency and all resource counts can be affected by other tool options, additional logic in the FPGA device, using a different version of Xilinx tools and other factors. Refer to in Table 2-1 through Table 2-6 for device-specific information.

Latency

When the downstream processing block on the AXI4-Stream bus can take data at the pixel rate or faster, the typical latency through the Video In to AXI4-Stream core is 6 cycles of vid_in_clk + 3 cycles of aclk.

If the downstream block takes pixels at a slower rate, the FIFO will be used to even out the mismatch in the input and output rates over the course of lines and frames. This storage of pixels in the FIFO will add to the latency and will vary according to the data flow in and out of the core.

Throughput

The average data rates of active pixels on the AXI4-Stream interface matches the average rate of active pixels in on the Video bus. However, the clock rates of the input and output need not match. Furthermore, since the AXI4-Stream bus does not carry blank pixels, the clock rate can be lower than the video clock rate and still have sufficient bandwidth to meet the average rate requirement. Additional FIFO depth is required in order to smooth the mismatch in instantaneous rates. Both the input video pixel clock (Fvclk) and the rate of the AXI4-Stream Clock (Faclk) is limited by the overall Fmax.

If Faclk is equal to or greater than Fvclk, only the minimum buffer size (32 locations) is required. This assumes that the cores connected downstream of the Video In to AXI4-Stream core can sink data at the full video rate. e.g. The downstream core can accept data in a virtually continuous stream with gaps occurring only following EOL, and each line consecutively with line gaps only preceding SOF. In this scenario, the FIFO will go empty after the EOL on each line.

If Faclk is less than Fvclk, additional buffering is required. The FIFO must be large enough to handle the differential in the rate that pixels are coming in on the video clock, and the slower rate that they can go out on the AXI4-Stream bus using aclk. For aclk frequencies above the line average but below that of vclk, the input FIFO depth must be:

FIFO depth min =32 + Active Pixels * Fvclk/Faclk

If the downstream processing core accepts data at a lower rate than the `aclk`, additional buffering is required in an amount sufficient to prevent the FIFO from overflowing during the course of a frame.

Resource Utilization

The information presented in Table 2-1 through Table 2-6 is a guide to the resource utilization and maximum clock frequency of the Video In to AXI4-Stream core for Virtex-7, Kintex-7, Artix-7, Zynq-7000, Virtex-6, and Spartan-6 FPGA families. The design was tested using ISE® v14.1 tools with default tool options for characterization data.

Note: Data width in the following tables refers to the aggregate data width of all the video components into and out of the core. For example, RGB data with 8-bits per component has a data width of 24.

Table 2-1: **Spartan-6**

Data Width	FIFO Depth	LUTs	FFs	RAM 16/8	Fmax (MHz)
8	32	83	104	0/0	223
24	1024	147	161	1/1	209
64	8192	204	264	33/0	129

Table 2-2: **Virtex-7**

Data Width	FIFO Depth	LUTs	FFs	RAM 36/18	Fmax (MHz)
8	32	83	104	0/0	337
24	1024	120	161	1/0	309
64	8192	221	262	16/1	309

Table 2-3: **Virtex-6**

Data Width	FIFO Depth	LUTs	FFs	RAM 36/18	Fmax (MHz)
8	32	79	104	0/0	303
24	1024	120	161	1/0	236
64	8192	209	262	16/1	186

Table 2-4: **Kintex-7**

Data Width	FIFO Depth	LUTs	FFs	RAM 36/18	Fmax (MHz)
8	32	79	104	0/0	357
24	1024	119	161	1/0	320
64	8192	208	262	16/1	320

Table 2-5: **Artix-7**

Data Width	FIFO Depth	LUTs	FFs	RAM 36/18	Fmax (MHz)
8	32	80	104	0/0	265
24	1024	113	161	1/0	222
64	8192	208	262	16/1	222

Table 2-6: **Zynq-7000**

Data Width	FIFO Depth	LUTs	FFs	RAM 36/18	Fmax (MHz)
8	32	74	104	0/0	261
24	1024	111	161	1/0	211
64	8192	209	262	16/1	211

Core Interfaces and Register Space

Port Descriptions

The Video In to AXI4-Stream core uses industry standard control and data interfaces to connect to other system components. The following sections describe the various interfaces available with the core. Figure 2-1 illustrates an I/O diagram of the Video In to AXI4-Stream

core. Not all of the timing signals are required by this core, however it also passes these signals out to a Xilinx Video Timing Controller which, depending on its configuration may require certain signal. Therefore all timing signals are present. For the Video In to AXI4 Stream core, the data enable is always required. Also, either a vertical sync or a vertical blank input is required.

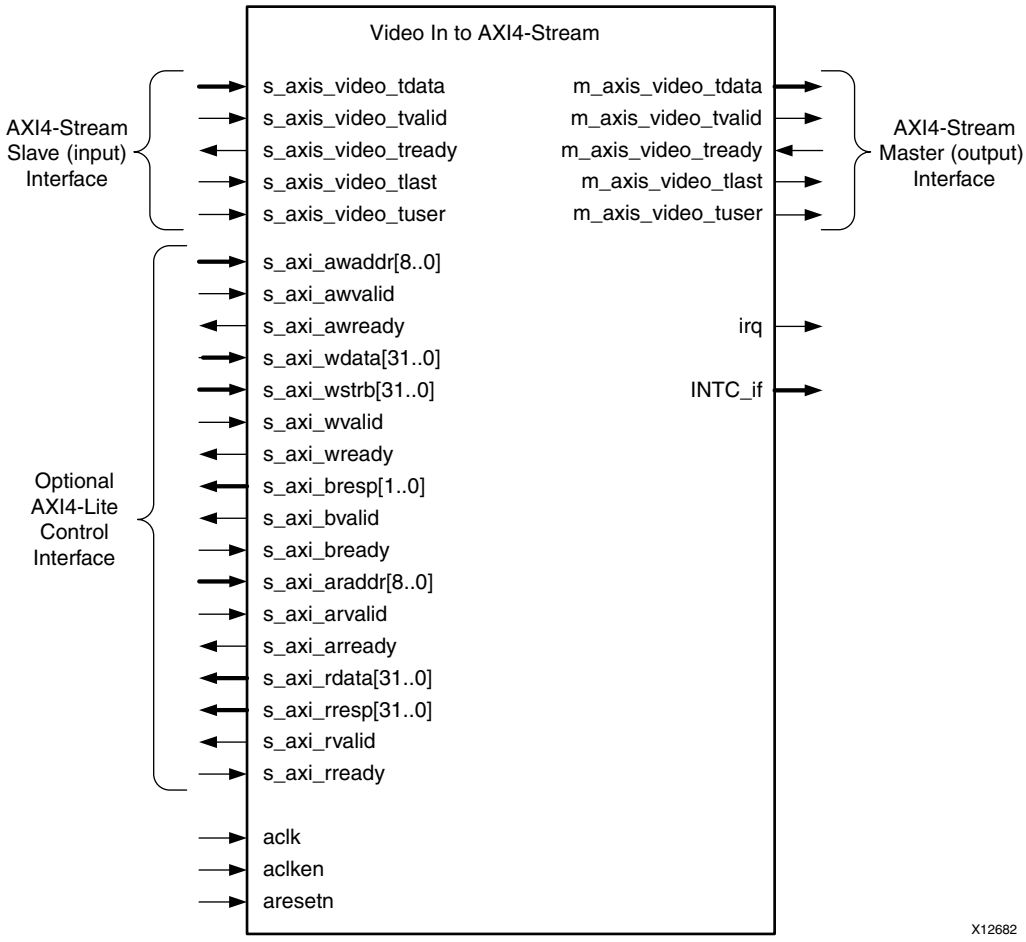

Figure 2-1: **Video In to AXI4-Stream Core Top-Level Signaling Interface**

Common Interface

Table 2-7: **Port Name I/O Width Description**

Signal Name	Direction	Width	Description
rst	In	1	Core reset. Active High
wr_error	Out	1	Active HIGH FIFO write error flag. 1 = FIFO write was attempted when FIFO was full.

Table 2-7: **Port Name I/O Width Description**

Signal Name	Direction	Width	Description
empty	Out	1	Active HIGH FIFO empty flag. 1 = FIFO read was attempted when FIFO was empty. Due to EOL flushing, this flag will be asserted at the end of every line during normal operation.
axis_enable	In	1	Enable the AXI4-Stream bus. 1 = Enable AXI4-Stream bus to operate. 0 = Inhibit AXI4-Stream operation by forcing m_axis_video_tdata LOW.

Video Timing Pass-Through Outputs

Table 2-8: **Port Name I/O Width Description**

Signal Name	Direction	Width	Description
vtd_vsync	Out	1	Vertical synch video timing signal.
vtd_hsync	Out	1	Horizontal synch video timing signal.
vtd_vblank	Out	1	Vertical blank video timing signal.
vtd_hblank	Out	1	Horizontal blank video timing signal.
vtd_active_video	Out	1	Active video flag. 1 = active video, 0 = blanked video

Video Input Interface

The Video In to AXI4-Stream core receives standard video data using the Video Input interface and transmits data using AXI4-Stream interfaces that implement a video protocol as defined in the *AXI Reference Guide (UG761)*, Video IP: AXI Feature Adoption section.

Table 2-9: **Port Name I/O Width Description**

Signal Name	Direction	Width	Description
vid_in_clk	In	1	Video input clock
video_de	In	1	Video data enable. 1 = active video, 0 = blanked video
vid_vsync	In	1	Vertical synch video timing signal. Active High
vid_hsync	In	1	Horizontal synch video timing signal. Active High
vid_vblank	In	1	Vertical blank video timing signal. Active High
vid_hblank	In	1	Horizontal blank video timing signal. Active High
video_data	In	8-64	Parallel video input data.

AXI4-Stream Interface

AXI4-Stream Signal Names and Descriptions

Table 2-10 describes the AXI4-Stream signal names and descriptions.

Table 2-10: **AXI4-Stream Data Interface Signal Descriptions**

Signal Name	Direction	Width	Description
m_axis_video_tdata	Out	8 to 64	Output Video Data
m_axis_video_tvalid	Out	1	Output Valid
m_axis_video_tready	In	1	Output Ready
m_axis_video_tuser	Out	1	Output Video Start Of Frame
m_axis_video_tlast	Out	1	Output Video End Of Line
ACLK	In	1	Clock
ACLKEN	In	1	Clock Enable
ARESETn	In	1	Active low synchronous

The ACLK, ACLKEN and ARESETn signals are shared between the core, the AXI4-Stream data interfaces, and AXI4-Lite control interfaces in the system.

ACLK

The AXI4-Stream must be synchronous to the clock signal ACLK. AXI4-Stream signals are sampled on the rising edge of ACLK. AXI4-Stream output signal changes occur after the rising edge of ACLK.

ACLKEN

The ACLKEN pin is an active-high, synchronous clock-enable input pertaining the AXI4-Stream interface. Setting ACLKEN low (de-asserted) halts the operation of the AXI4-Stream Bus despite rising edges on the ACLK pin. Internal states are maintained, and output signal levels are held until ACLKEN is asserted again. When ACLKEN is de-asserted, core AXI4-Stream inputs are not sampled, except ARESETn, which supersedes ACLKEN.

ARESETn

The ARESETn pin is an active-low, synchronous reset input. ARESETn supersedes ACLKEN, and when set to 0, the core resets even if ACLKEN is de-asserted.

Video Data

The AXI4-Stream interface specification restricts TDATA widths to integer multiples of 8 bits. Therefore, in the case of data with 10 or 12 bits, data must be padded with zeros on

the MSB to form a N*8-bit wide vector before connecting to m_axis_video_tdata. Padding does not affect the size of the core.

Similarly, data on the Video in to AXI-4 Stream output m_axis_video_tdata is packed and padded to multiples of 8 bits as necessary. Figure 2-10 shows an example of this for 12-bit RGB data. Zero padding the most significant bits is only necessary for 10, 12, and 14 bit wide data.

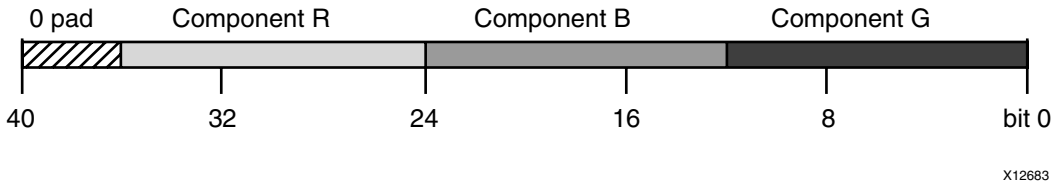

Figure 2-2: **RGB Data Encoding on m_axis_video_tdata**

READY/VALID Handshake

A valid transfer occurs whenever READY, VALID, ACLKEN, and ARESETn are high at the rising edge of ACLK. During valid transfers, DATA only carries active video data. Blank periods and ancillary data packets are not transferred via the AXI4-Stream Video protocol.

Guidelines on Driving m_axis_video_tready

The m_axis_video_tready signal may be asserted before, during or after the cycle in which the Video in to AXI4-Stream core asserted m_axis_video_tvalid. The assertion of m_axis_video_tready may be dependent on the value of m_axis_video_tvalid. A slave that can immediately accept data qualified by m_axis_video_tvalid, should pre-assert its m_axis_video_tready signal until data is received. Alternatively, m_axis_video_tready can be registered and driven the cycle following VALID assertion. It is recommended that the AXI4-Stream slave should drive READY independently, or pre-assert READY to minimize latency.

Start of Frame Signal - m_axis_video_tuser

The Start-Of-Frame (SOF) signal, physically transmitted over the AXI4-Stream tuser0 signal, marks the first pixel of a video frame. The SOF pulse is 1 valid transaction wide, and must coincide with the first pixel of the frame. SOF serves as a frame synchronization signal, which allows downstream cores to re-initialize, and detect the first pixel of a frame. The SOF signal may be asserted an arbitrary number of aclk cycles before the first pixel value is presented on tdata, as long as a tvalid is not asserted.

End of Line Signal - m_axis_video_tlast

The End-Of-Line signal, physically transmitted over the AXI4-Stream tlast signal, marks the last pixel of a line. The EOL pulse is 1 valid transaction wide, and must coincide with the last pixel of a scan-line, as seen in Figure 2-3.

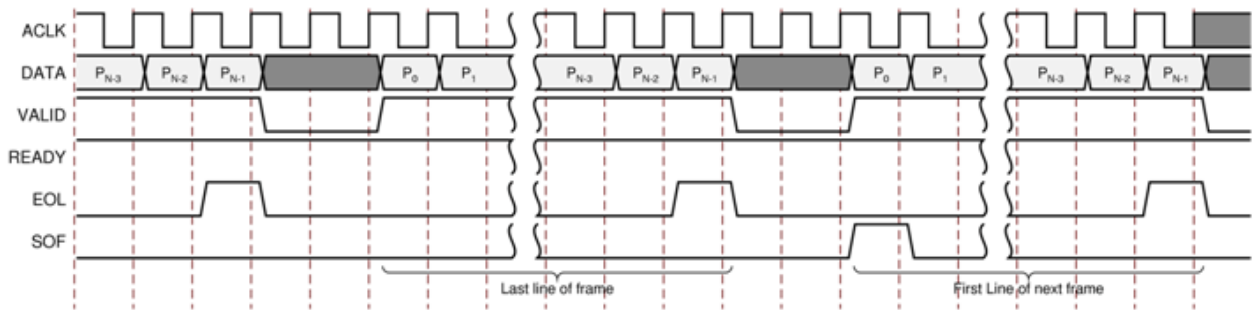

Figure 2-3: **Use of EOL and SOF Signals**

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core.

Graphical User Interface (GUI)

The Xilinx Video In to AXI4-Stream core is easily configured to meet the developer's specific needs through the CORE Generator™ GUI. This section provides a quick reference to parameters that can be configured at generation time.

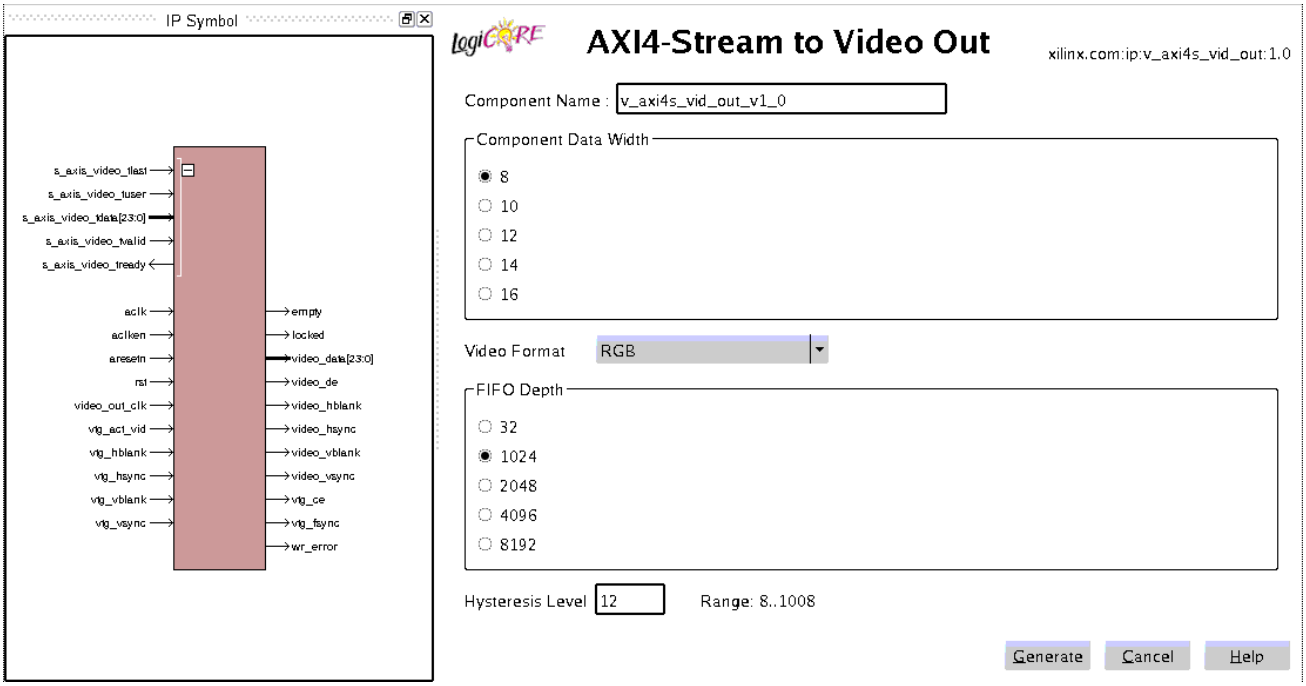

Figure 3-1: **Video in to AXI-4 Stream CORE Generator GUI**

The GUI displays a representation of the IP symbol on the left side, and the parameter assignments on the right side, which are described as follows:

- **Component Name:** The component name is used as the base name of output files generated for the module. Names must begin with a letter and must be composed from characters: a to z, 0 to 9 and "_".

- **Component Data Width:** Specifies the bit width of input samples. This is used in conjunction with the Video Format to determine the width of the input video bus, `vid_data`, and the AXI4-Stream data bus, `m_axis_video_tdata`.
- **Video Format**: Specifies the video format used. The video formats are specified in the *Video IP: AXI Feature Adoption* section of the UG761 AXI Reference Guide.
  The format selected determines the number of components used. The number of components (1-4) is multiplied by the component width to determine the width of the video data bus, `v_data`. In turn, this width is rounded up to the nearest factor of 8 to determine the width of the AXI4-Stream data bus, `m_axis_video_tdata`. For example, if the component width is 14 and the Video Format is RGB (3 components), the `v_data` will be 42 bits wide and `m_axis_video_tdata` will be 48 bits.
- **FIFO Depth**: Specifies the number of locations in the input FIFO. The options for FIFO depth are 32, 1024, 2048, 4096, and 8192.
- **Hysteresis Level**: Defines the "Cushion" level of the frame buffer. i.e. the number of locations that are considered the minimum fill level for FIFO operation to start. This number must be at least 16 less than the depth of the FIFO.

Designing with the Core

General Design Guidelines

The video inputs of the Video In to AXI4 Stream core should be connected to the input video source, for example, a DVI interface chip that produces parallel video data, and timing signals. Not all of the timing signals are required by this core, however it also passes these signals out to a Xilinx Video Timing Controller which, depending on its configuration may require certain timing signals. The set of timing signals used should be those required by the VTC detector. See the PG016 Video Timing Controller Product Guide for more details. For the Video In to AXI4 Stream core, the data enable is always required. Also, either a vertical sync or a vertical blank input is required.

The main output of the core is a master AXI-4 Stream bus, that connects to downstream video processing blocks as shown in Figure 4-1. The master and slave interfaces share a common clock, reset and clock enable.

As shown in Figure 4-1, the Video In to AXI4 Stream Core is generally used in conjunction with the Video Timing Controller which detects the video timing parameter used by downstream processing blocks.

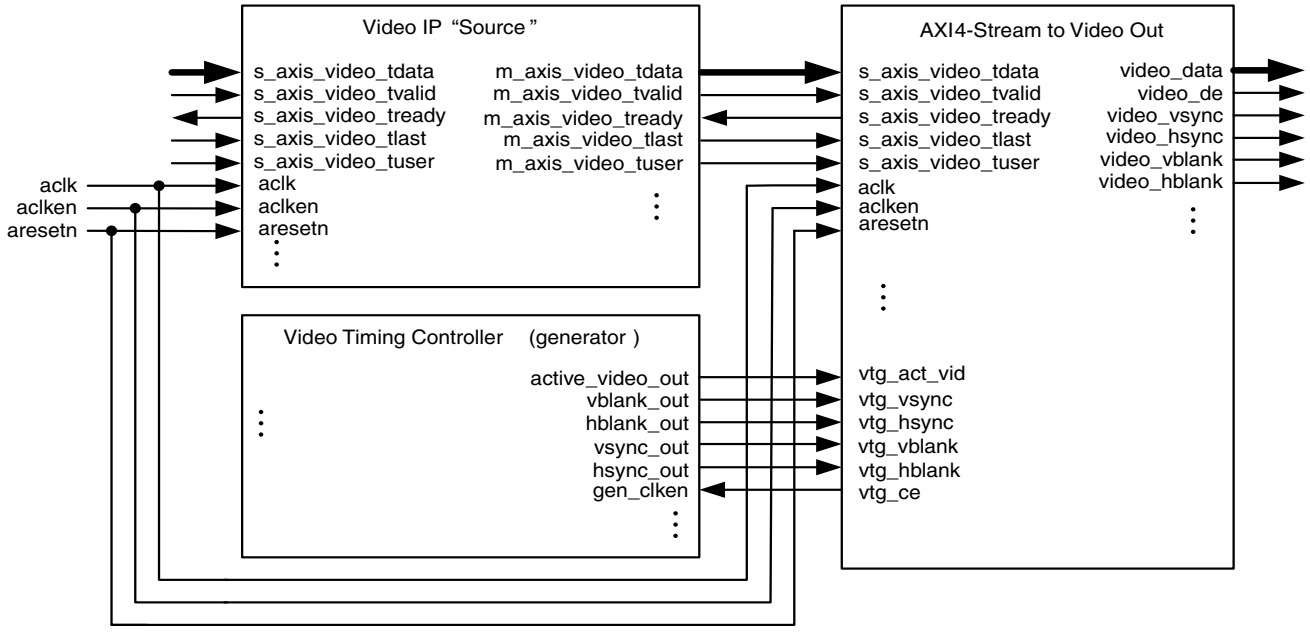

Figure 4-1: **Example of ACLK Routing and AXI4-Stream Interconnect**

Clocking

There are two clocks used in this core.

- Video input pixel clock
- AXI4-Stream clock

The video input clock corresponds to the video line standard used on the input. It is part of the video line standard and is used by both the Video In to AXI4-Stream core and by the corresponding Video Timing Controller core that is used to detect video timing.

The AXI4-Stream clock (aclk) is part of the AXI4-Stream bus. To minimize buffering requirements, this clock should be of equal or higher frequency than the video input clock. This clock can be slower than the video input clock, in which case, additional buffering is required to store pixels so that lines can be input at the burst rate of the video clock. This is discussed in the Buffer Requirements section. At a minimum, the aclk frequency must be higher than the average pixel rate.

Resets

There are two external resets provided: rst, which resets the entire core, and aresetn, which resets the AXI4-Stream interface. Both resets cause the FIFO to be reset. When asserted, the reset should be held for least two clock periods of the lowest frequency clock.

System Consideration

Buffer Requirements

The FIFO depth is selectable via the GUI when the core is generated. The buffering requirement for the asynchronous FIFO depends mainly on the relative frequency of the AXI4-Stream clock (`aclk`) to the video clock (`vid_in_clk`) frequency, and the line standard being used.

If the frequency of the AXI4-Stream clock (Faclk) is equal to or greater than the frequency of the Video input pixel clock (Fvclk), only the minimum buffer size (32 locations) is required. This assumes that the cores connected downstream of the Video In to AXI4-Stream core can sink data at the full video rate. e.g. The downstream core can accept data in a virtually continuous stream with gaps occurring only following EOL, and each line consecutively with line gaps only preceding SOF. In this scenario, the FIFO will go empty after the EOL on each line.

If Faclk is less than Fvclk, additional buffering is required. The FIFO must store enough pixels to supply pixels continuously throughout the active line. Additionally, due to phasing requirements, the horizontal active period on the output will overlap the effective blanking period of pixels coming in from the AXI4-Stream bus. This means that the input FIFO must also be large enough to provide output pixels continuously during this time.

For AXI4-Stream clock frequencies above the line average but below that of video input pixel clock, the minimum FIFO initial fill level must be:

FIFO depth min. = 32+ Active Pixels * Fvclk/Faclk

If the downstream processing core accepts data at a lower rate than the AXI4-Stream clock, Additional buffering is required in an amount sufficient to prevent the FIFO from overflowing during the course of a frame.

Module Descriptions

Figure 1-1 shows a block diagram of the Video In to AXI4-Stream core. The video connections are on the left, and the AXI4-Stream interface is on the right. There are two main blocks, the data formatter and the stream coupler. The stream coupler contains an Asynchronous FIFO. These two blocks are described in detail below:

Data Formatter

The data formatter derives the EOL and SOF flags required to transmit AXI4-Stream Video protocol. It also controls writing of the FIFO in the stream coupler. The flags are generated by looking at the edges of the data enable (DE) signal. Since only active pixels are carried on AXI4-Stream, the FIFO is only written when active pixels are present. There are input registers on all inputs to minimize input loading. The EOL flag is timed to be coincident with the last pixel before the falling edge of DE. Video data is delayed so that the falling edge of DE can be detected, and the EOL flag asserted coincident with the proper pixel as it is written to the FIFO. Similarly, the SOF flag is created based on the rising edge of DE, however it additionally requires knowledge of the vertical timing to identify the first line. This is done using the logical or OR vsync and vblank. The falling edge of either of these indicates that the input video is in the vertical blanking period. This enables SOF generation, and the SOF flag is asserted at the next rising edge of DE; the first valid pixel of the field.

The rising edge of DE also resets the vertical blanking flip-flop. Figure 4-2 shows the timing of outputs and internal signals relative to the inputs. The de_1 and de_2 signals are delayed versions of de. The outputs are highlighted in bold.

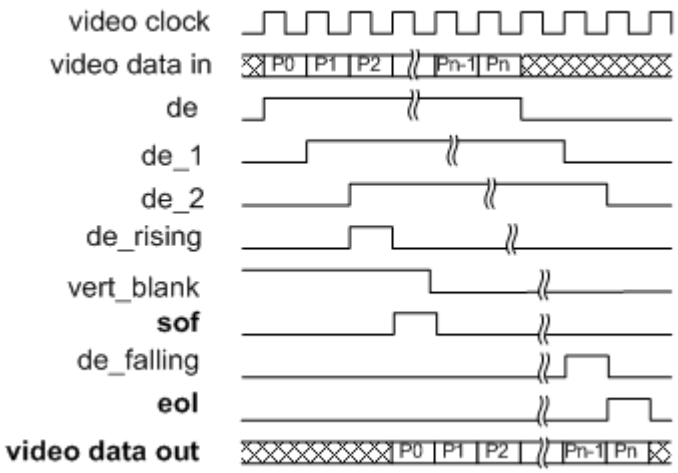

Figure 4-2: **Data Formatter Timing Diagram**

Stream Coupler

The Stream Coupler block consists mainly of an asynchronous FIFO and write logic for the input side of the FIFO. Reading of the FIFO is controlled by the Output Synchronizer. The FIFO serves two primary purposes:

1. Clock domain crossing
2. Buffering of data between AXI4-Stream input and video output.

The buffering requirements are dependent on the ratio of the AXI4-Stream clock rate to the video clock rate. This is described more in Buffer Requirements.

Asynchronous FIFO

The crossing of clock domains dictates that this be an asynchronous FIFO. The FIFO designed for this core has two important distinguishing features:

1. It has status flags and a fill level output in both clock domains.
2. An "invalid" (`read_error`) flag is produced in parallel with output data for the use case of reads when the FIFO is empty. It also has pointer inhibiting logic to prevent pointer crossings on underflow and overflow.

This asynchronous nature of the FIFO presents challenges for the fill level indicators and the flags. The chief risk is that pointer values could glitch when being sampled from one clock domain to another. Thus, there must be pointer synchronization across clock domains, and there must be a handshaking protocol to insure that all pointer updates are seen even when the clock rate in the two domains are radically different. Figure 4-3 is a block diagram of the asynchronous FIFO. Note the synchronizing logic and the handshaking between clock domains. The size of the FIFO is set in the GUI when the core is generated.

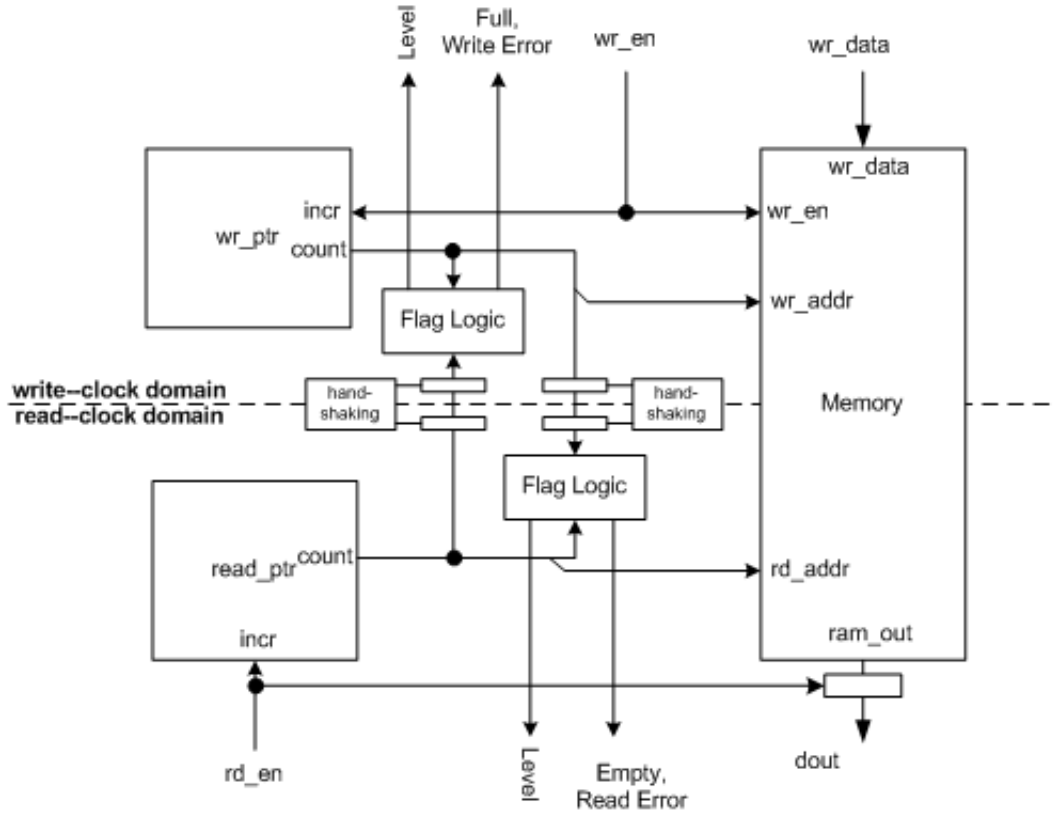

Figure 4-3: **Block Diagram of Bridge Core Asynchronous FIFO**

Synchronized Gray codes are commonly used in asynchronous FIFOs to eliminate the problems of synchronizing multiple counter bits changing on the same clock edge. However, instead of Gray code pointers, the FIFO for the Video In to AXI4-Stream core uses binary pointers synchronized via handshaking. The main reason is that calculating the fill level, which is used integrally in the read logic, is simple with binary pointers but impractical with gray code pointers.

Clock Domain Crossing of Pointers

The synchronization and handshaking for pointers is shown in detail in Figure 4-4. The first two register delays are required to resolve metastability. The third is to insure that the register has time to take the data before the handshaking is returned. Otherwise if the clock in one domain were several times faster than the other, there is a chance that the handshaking could be returned, before the pointer register is updated in the slower clock domain.

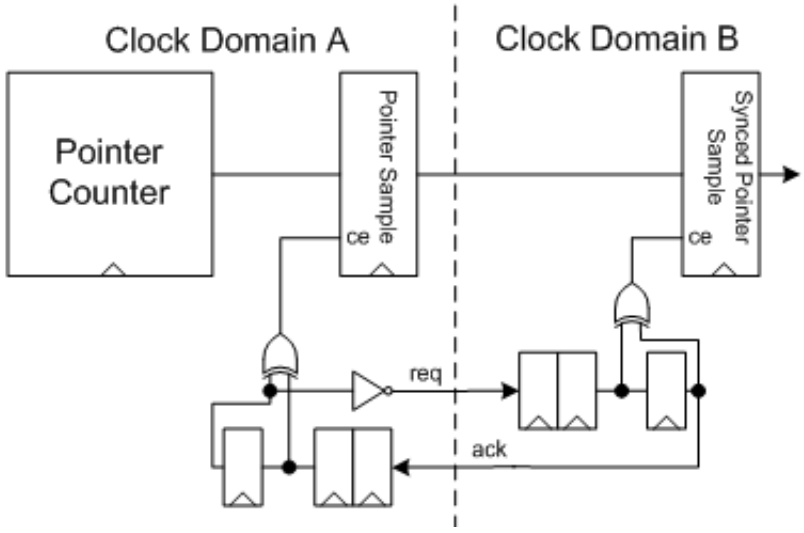

Figure 4-4: **Synchronization and Handshaking for Clock Domain Crossing of Pointers**

The extra delay for turning around the handshaking signal insures that data will be transferred reliably no matter what the relative clock rates. i.e. if Freqa > 2Freqb each "req"is still guaranteed to update the sync pointer in domain B. By the same token, if Freqb > 2Freqa, each ack is guaranteed to update the pointer sample in domain A. Figure 4-5 shows an example of pointer synchronization between clock domains, and the handshaking scheme shown in Figure 4-4. This handshaking scheme, utilizes two states: request and acknowledge. The "request" state is when Req and Ack are not equal and "acknowledge" is when they are equal.

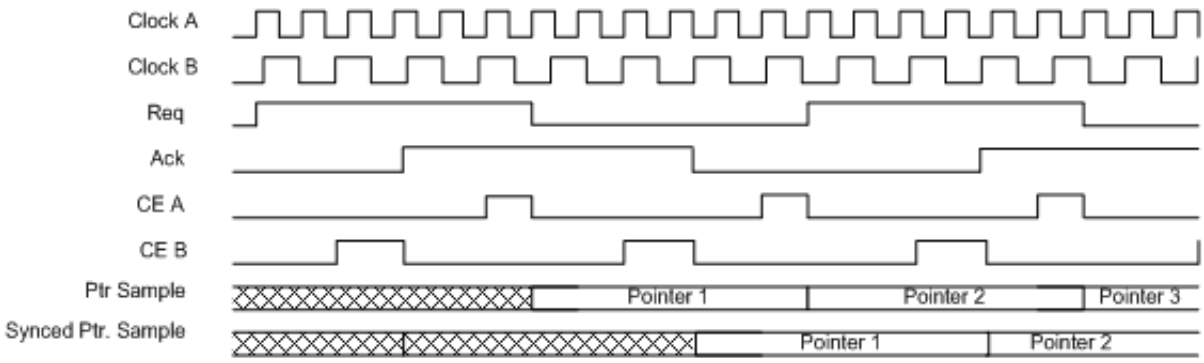

Figure 4-5: **Waveform Diagram of Handshaking and Clock Domain Crossing of Pointers**

This sample and hold method with handshaking delays the capture of the pointers by several clock edges in each domain but the pointer transfers are always reliable and glitch free. The latency of the pointers causes pessimism in the level outputs, and the flags. That is, the empty flag will persist in the read domain for several clocks after a write has occurred, etc. Also, the level output will not necessarily be monotonic. This is taken into

account in the design of the bridges by providing a small "cushion" or minimum fill level in the FIFOs so that the pointer pessimism does not cause artifacts.

Underflow Prevention

In addition to the steps for synchronizing flags, extra care must be taken in handling the empty condition. It is not enough to signal an empty condition, because "reads" of the FIFO will not stop on empty. Additional read operations must be performed to get the EOL to the output of the FIFO. This happens routinely at the end of every line, and the FIFO must not lose any data. That is, underflow is not allowed. When new data is eventually written to the FIFO, the read must pick up with the first new valid pixel. To do this, the read pointer is inhibited when the FIFO is empty.

The empty flag asserts coincident with the last available location being clocked into the output register. The read pointer, however is not advanced to match the write pointer but remains pointing at the last valid pixel that was read. Subsequent reads, when empty is asserted, will cause the read_error flag to be asserted, flagging the pixel from the FIFO as invalid. The data output, however will not change.

Thus, when a read occurs to an empty FIFO, the invalid flag (read_error) is set, and the read pointer does not increment. In this way, the EOL can be advanced through the pipe by a series of reads on the empty FIFO. With each read, an invalid pixel backfills the advancing EOL but the downstream logic can distinguish these from valid pixels.

Pointer Format

It is important to provide an accurate level, and to distinguish between full and empty conditions when the read and write pointers are equal. This is done by having extra "revolution" bits on the pointers in addition to the address bits as shown in Figure 4-6.

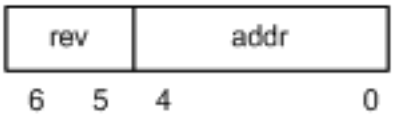

Figure 4-6: **Pointer Format for a 32 Location FIFO**

At the cost of a couple of extra bits in the pointers, this allows the level calculations and flags to be unambiguously determined in full and empty conditions.

Read Logic

The function of the read logic is to control the handshaking for the AXI4-Stream bus and to provide pixels to this bus as rapidly as possible. In general, the strategy for AXI4-Stream is downstream-greedy. That is, downstream modules take pixels as soon as they are available and there is buffer space to accommodate them. Since the Video In to AXI4-Stream core is, by definition, at the front of the pipeline, it strives to empty its FIFO as fast as possible.

The Read Logic controls the tvalid handshaking signal based on the level and flags from the FIFO, and the tready signal returned from the downstream module. Whenever data is available in the FIFO, tvalid is asserted. When tready is returned active, the FIFO is read and the new pixel is again denoted by the tvalid being active. Thus, the tvalid is asserted except when there is no valid data available from the FIFO. The FIFO will only begin filling if the downstream core cannot accept data as fast as it is coming in from the video bus. This will happen, for example, if the AXI4-Stream clock, aclk, is slower than the video clock. In this case, during the active portion of each line, pixels will be coming into the FIFO faster than they can be sent out on the AXI4-Stream bus. Thus the FIFO will begin to fill. Usually the FIFO will empty at the end of the active line when the downstream core is still taking pixels, but the incoming video data is in the horizontal blanking period.

At the end of each line, the EOL must be flushed through from the FIFO to the output registers. This is done so that the downstream core can access the complete line without having to wait through the horizontal blanking period for new pixels to push the EOL out.

This flushing requirement presents a challenge since this occurs during horizontal blanking, meaning that no data is coming into the FIFO. It requires generation of invalid pixels to flush out the valid pixels. Since no inactive pixels are allowed on the AXI4-Stream bus, these invalid pixels must be swallowed by the core prior to the output.

Flushing of the EOL is accomplished by reading from the FIFO whenever it is not empty or when it is empty and the last pixel has the EOF (tlast) flag set (assuming AXI4-Stream bus is not applying backpressure). In other words, read the FIFO until it is empty, and beyond when the EOL is present. When the FIFO is empty, and the last pixel has been read (the EOL pixel) the FIFO will mark subsequent pixels as invalid, but the EOL will continue to propagate to the output. Valid pixels get sent out on the AXI-4 Stream bus, invalid pixels do not. In this way, invalid pixels are swallowed before they get to the AXI-4 Stream bus.

Constraining the Core

Required Constraints

The only constraints required are clock frequency constraints for the video clock, `video_in_clk`, and the AXI4-Stream clock, `aclk`. Paths between the two clock domains should be constrained as false paths, which should be ignored during timing analysis.

For example, the following UCF code could be used to constrain the video clock and the AXI4-Stream Clock for 150MHz operation:

```
NET vid_in_clk TNM_NET = vid_in_clk;
TIMESPEC TS_vid_in_clk = PERIOD vid_in_clk 150 MHz HIGH 50%;
NET aclk TNM_NET = aclk;
TIMESPEC TS_aclk = PERIOD aclk 150 MHz HIGH 50%;

TIMESPEC TS_vid_in_clk_to_aclk = FROM vid_in_clk TO aclk  TIG;
TIMESPEC TS_aclk_to_vid_in_clk = FROM aclk TO vid_in_clk  TIG;
```

Device, Package, and Speed Grade Selections

There are no device, package, or speed grade requirements for this core. This core has not been characterized for use in low power devices.

Clock Frequencies

The pixel clock frequency is the required frequency for this core. See Maximum Frequencies in Chapter 2.

Clock Management

There are two clock domains for the Video In to AXI4-Stream core. The clock crossing boundary is handled by the FIFO, and a handshaking system for passing pointers between domains.

Clock Placement

There are no specific Clock placement requirements for this core.

Banking

There are no specific Banking rules for this core.

Transceiver Placement

There are no Transceiver Placement requirements for this core.

I/O Standard and Placement

There are no specific I/O standards and placement requirements for this core.

Detailed Example Design

No example design is available at the time for the LogiCORE IP Video In to AXI4-Stream v1.0 core.

Appendix A

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

http://www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf.

For a comprehensive listing of Video and Imaging application notes, white papers, reference designs and related IP cores, see the Video and Imaging Resources page at:

http://www.xilinx.com/esp/video/refdes_listing.htm#ref_des.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

These documents provide supplemental material useful with this user guide:

1. UG761 AXI Reference Guide

Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

See the IP Release Notes Guide (XTP025) for more information on this core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Resolved Issues
- Known Issues

Ordering Information

The Video In to AXI4-Stream 1.0 core is provided under the Xilinx Core License Agreement and can be generated using the Xilinx® CORE Generator™ system. The CORE Generator system is shipped with Xilinx ISE® Design Suite software.

Contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE IP modules and software. Information about additional Xilinx LogiCORE IP modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
4/24/2012	1.0	Initial Xilinx release of core.

Notice of Disclaimer