LogiCORE IP DisplayPort v3.2

Product Guide

PG064 July 25, 2012

Table of Contents

SECTION I: SUMMARY

IP Facts

Chapter 1: Overview

Feature Summary. 11
Unsupported Features. 12
Licensing and Ordering Information 12

Chapter 2: Product Specification

Standards 13
Performance. 13
Port Descriptions 14
Register Space 21
Sink Core. 31

Chapter 3: Designing with the Core

Source Overview 44
Sink Overview. 61
Source Core Architecture. 67
Sink Core Architecture 77
Clocking. 85
Resets 86

SECTION II: VIVADO DESIGN SUITE

Chapter 4: Customizing and Generating the Core

GUI 88
Parameterization 91
Output Generation. 92

Chapter 5: Constraining the Core

Board Layout 93
Required Constraints 93
Device, Package, and Speed Grade Selections 94
Clock Frequencies 94
Clock Management 94
Clock Placement 94
Banking 94
Transceiver Placement 94
I/O Standard and Placement 95

Chapter 6: Detailed Example Design

Directory and File Contents 97
Example Design 99
Demonstration Test Bench 100
Implementation 101

SECTION III: ISE DESIGN SUITE

Chapter 7: Customizing and Generating the Core

GUI 103
Output Generation 106
Source Core Parameterization 106
Sink Core Parameterization 107

Chapter 8: Constraining the Core

Board Layout 108
Required Constraints 108
Device, Package, and Speed Grade Selections 109
Clock Frequencies 109
Clock Management 109
Clock Placement 110
Transceiver Placement 110
I/O Standard and Placement 110

Chapter 9: Detailed Example Design

Directory and File Contents 112
Directory and File Contents 112
Example Design 118

Demonstration Test Bench 120
Implementation 121
Simulation 121
Messages and Warnings 122

SECTION IV: APPENDICES

Appendix A: Verification, Compliance, and Interoperability

Simulation 124
Hardware Testing 124

Appendix B: Migrating

Functionality Changes 125

Appendix C: Additional Resources

Xilinx Resources 126
References 126
Technical Support 127
Revision History 127
Notice of Disclaimer 127

SECTION I: SUMMARY

IP Facts

Overview

Product Specification

Designing with the Core

Introduction

The Xilinx LogiCORE™ IP DisplayPort™ interconnect protocol is designed for transmission and reception of serial-digital video for consumer and professional displays. DisplayPort is a high-speed serial interface standard supported by PC chipsets, GPU's and display controllers, HDTV and monitors from industry leaders.

This protocol replaces VGA, DVI and HDMI™ outside and LVDS inside the box for higher resolution, higher frame rate and color bit depth display.

Features

- Source (TX) and Sink (RX) Controllers
- Designed to VESA DisplayPort Standard v1.1a and v1.2
  - For a 5.4 Gb/s link rate, a high performance 7 series FPGA is required with speed grade -2 or -3
  - DisplayPort v1.2 is supported with 7 series devices
- 1, 2 or 4 lanes at 1.62, 2.7 or 5.4 Gb/s
- One, two or four pixel-wide video interface supporting up to a 4k x 2k monitor resolution
- RGB and YCbCr color space, up to 16 bits per color
- Auto lane rate and width negotiation
- I2C over a 1 Mb/s AUX channel
- Secondary channel audio support (two channels)
- With additional license, supports DisplayPort Audio Support (two channels with SPDIF). See product page for details.

LogiCORE IP Facts Table

Core Specifics				
Supported Device Family[1]	Artix™-7, Virtex®-7, Kintex™-7, Virtex-6, Spartan®-6			
Supported User Interfaces	Native Video, AXI4-Stream, AXI4-Lite			
Resources Used				
	I/O (to pins)	LUTs	FFs	Block RAMs
Sink	12	~7000	~5400	0
Source	13	~6500	~5100	0
Provided with Core				
Example Design	Simple RTL Source Policy Maker RTL Sink Policy Maker RTL EDID ROM, RTL I2C Controller			
Test Bench	Verilog and VHDL			
Constraints File	ISE: UCF Vivado: XDC Full Timing Constraints and Transceiver Physical Constraints			
Simulation Model	Verilog and VHDL Wrapper			
Supported S/W Driver[2]	N/A			
Tested Design Flows[3]				
Design Entry	Vivado™ Design Suite v2012.2 ISE™ Design Suite v14.2			
Simulation	Mentor Graphics ModelSim			
Synthesis	Xilinx Synthesis Technology (XST) Xilinx ISIM Vivado Synthesis			
Support				
Provided by Xilinx @ www.xilinx.com/support				

Notes:

1. For a complete listing of supported devices, see the release notes for this core.
2. Standalone driver details can be found in the EDK or SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from //wiki.xilinx.com.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

This chapter contains an overview of the core as well as details about applications, licensing, and standards.

Source Core

The Source core moves a video stream from a standardized main link through a complete DisplayPort Link Layer, and onto High-Speed Serial I/O for transport to a Sink device.

Main Link

The Main Link for the Source core interfaces to a user-driven stream of video data. Using horizontal and vertical sync signals for framing, this user interface matches the industry standard for display controllers and plugs into existing video streams. The user can specify one or two pixel-wide data through a register field. The user can also specify the number of bits per pixel as well as colorspace (RGB or YCbCr or YOnly). In addition, it’s possible to specify one, two or four pixel-wide data through a register configuration and provide an accompanying video clock that operates between 13.5 and 150 MHz.

The Source core is responsible for managing the video data and preparing it for transmission over the high-speed serial I/O. It performs the required operations for the Link and Physical Layers of the *DisplayPort Standard v1.1a* or *v1.2*, based on protocol selection.

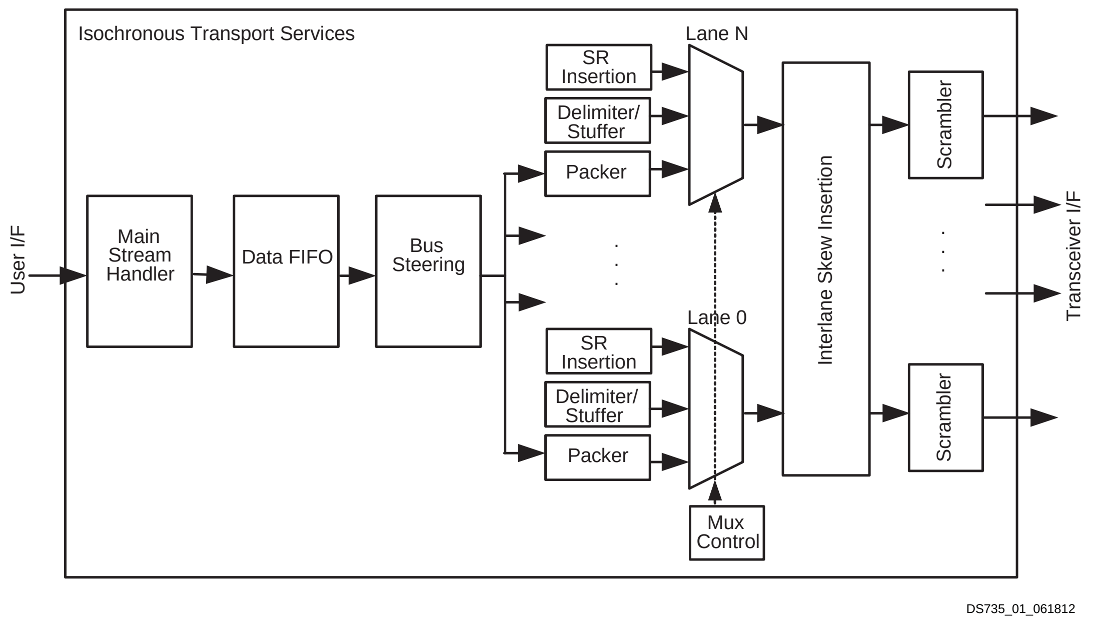


Figure 1-1: **Source Main Link Datapath**

Secondary Channel

The current version of the DisplayPort IP supports 2-channel Audio. An SPDIF controller is generated when the Audio option is enabled (additional license required). Secondary Channel features from the Displayport v1.1a specification are supported.

The DisplayPort Audio IP core is offered in a modules to provide flexibility and freedom to modify the system as needed. As shown in Figure 1-2, the Audio interface to the DisplayPort core is defined using an AXI4-Stream interface to improve system design and IP integration.

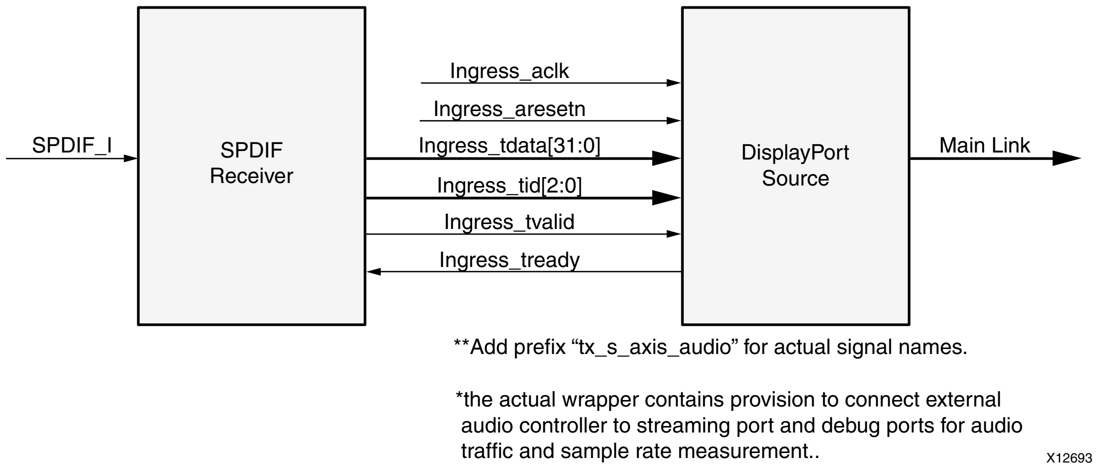

SPDIF_I
SPDIF Receiver
Ingress_aclk
Ingress_aresetn
Ingress_tdata[31:0]
Ingress_tid[2:0]
Ingress_tvalid
Ingress_tready
DisplayPort Source
Main Link
**Add prefix "tx_s_axis_audio" for actual signal names.
*the actual wrapper contains provision to connect external audio controller to streaming port and debug ports for audio traffic and sample rate measurement..
X12693

Figure 1-2: **Audio Data Interface of DisplayPort Source System**

SPDIF is used as the default controller for the DisplayPort Source, and AXI-SPDIF is shipped with the DisplayPort core and delivered in the example design. This system allows access to the AXI4-Stream interface. See the *AMBA AXI4-Stream Specification* for interface timing.

The SPDIF controller as a receiver receives audio samples from the SPDIF line and stores them in an internal buffer. 32-bit AXI TDATA is formatted according as follows:

Control Bits + 24-bit Audio Sample + Preamble

See PG045, *LogiCORE IP SPDIF Product Guide* for more details.

The ingress channel buffer in the DisplayPort core will accept data from the SPDIF controller based on buffer availability and audio control programming. A valid transfer takes place when `tready` and `tvalid` are asserted as described in the AXI4-Stream protocol. The ingress channel buffer acts as a holding buffer.

The DisplayPort Source has a fixed secondary packet length [Header = 4 Bytes + 4 Parity Bytes, Payload = 32 Sample Bytes + 8 Parity Bytes]. In a 1-2 channel transmission, the Source accumulates eight audio samples in the internal channel buffer, and then sends the packet to main link. In a 3-8 channel transmission, the Source waits for at least one sample in all internal channel buffers, and then sends the packet to main link.

Host Interface

The core can be configured through the AMBA® AXI4-Lite processor interface. The registers are mapped as packed 32-bit values from the perspective of the interface.

AUX Channel

The AUX Channel provides peer information between source and sink endpoints. The core is also designed to facilitate I2C communication over this link.

High-Speed Serial I/O

The user can specify up to four lanes through the Xilinx CORE Generator™ GUI. Though more lanes can be selected, the actual number in use is determined by a negotiation procedure between endpoints. The instantiations of the transceivers have been brought to the top and provided to the user for greater visibility.

Sink Core

The Sink device accepts incoming serial data streams from the High-Speed Serial I/O, properly reconstructs the original data to an appropriate format, and provides a video stream in a standard format on the main link. It performs required operations for the Link and Physical Layers of the *DisplayPort Standard v1.1a* and *v1.2* based on protocol selection.

Main Link

The Main Link for the Sink Core drives a stream of video data toward the user. Using horizontal and vertical sync signals for framing, this user interface matches the industry standard for display controllers and plugs in to existing video streams with little effort. Though the core provides data and control signaling, the user is still expected to supply an appropriate clock. This clock can be generated with the use of M and N values provided by the core. Alternatively, the user might want to generate a clock by other means. The core's underflow protection allows the user to use a fast clock to transfer data into a frame buffer.

The user can specify one, two, or four pixel-wide data through a register field. The bit width and format is determined from the Main Stream Attributes, which are provided as register fields.

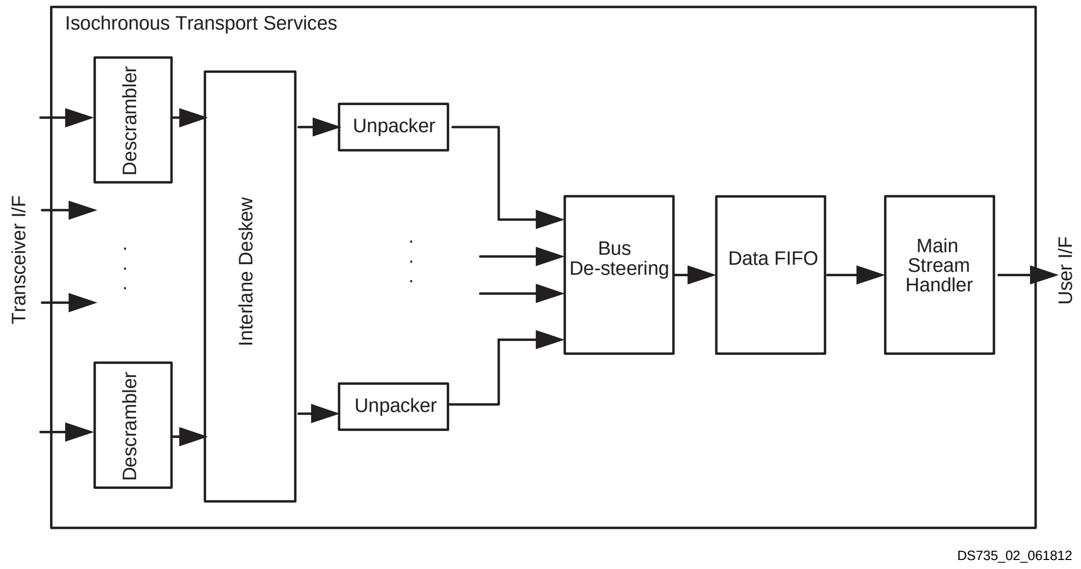


Figure 1-3: **Sink Main Link Datapath**

Secondary Channel

The current version of the DisplayPort core supports two-channel Audio. The SPDIF controller is generated when the Audio option is enabled. Secondary Channel features from the Displayport v1.1a specification are supported. DisplayPort Audio IP core is offered in modules to provide flexibility to modify the system as needed.

As shown in Figure 1-4, Audio interface to DisplayPort is defined using AXI4-Stream interface.

SPDIF is used as default controller for DisplayPort sink and SPDIF is shipped along with DisplayPort IP and delivered in example design. User will have access to AXI-Streaming interface. See the *AMBA AXI-Streaming Specification* for interface timing.

Audio data and secondary packets are received from the main link and stored in internal buffers of DisplayPort Sink core. The AXI4-Stream interface of DisplayPort transfers audio sample along with control bits to SPDIF transmitter and AXI4-Stream slave has to accept it immediately. In other words, DisplayPort Sink should never be back pressured.

SPDIF transmitter sends out samples as per SPDIF protocol format. Typically SPDIF PHY is a differential device and hence user should create a differential signal and make proper connections to PHY at system level.

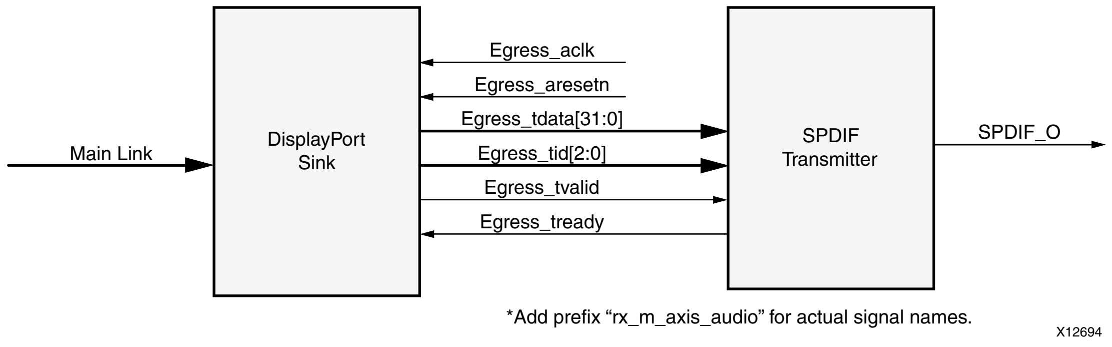


Figure 1-4: **Audio Data Interface of DisplayPort Sink System**

Host Interface

The core can be configured through the AMBA AXI4-Lite processor interface. The registers are mapped as packed 32-bit values from the perspective of the interface.

AUX Channel

The AUX Channel provides peer information between source and sink endpoints. The core is also designed to facilitate I2C communication over this link.

High-Speed Serial I/O

The user can specify up to four lanes through a pre-synthesis directive. Though more lanes can be selected, the actual number in use is determined by a negotiation procedure between endpoints. The instantiations of the transceivers have been brought to the top and provided to the user for greater visibility.

Feature Summary

Xilinx DisplayPort IP offers both Source (TX) and Sink (RX) functionality for high performance video, such as 4Kx2K resolution. The IP is certified to be used in source application and known to work well between Xilinx (TX) and Xilinx (RX) devices. In use cases, where the source device is a non-Xilinx device the use case needs to be reviewed with Xilinx and get it qualified.

DisplayPort IP offers auto lane rate and width negotiation, X1,X2, or X4 lanes at 1.62, 2.7 or 5.4G based on AXI4-Lite interface and has secondary Audio with S/PDIF controller option available. It offers vendor specific DPCD. IP also comes with example design and test bench.

Unsupported Features

- The automated test feature is not supported.
- If the source is a non-Xilinx device connecting to a Sink IP implemented in a Xilinx device, qualification by Xilinx is required before using it in production.
- Audio (3-8 channel) is not supported. Audio-specific updates of the DisplayPort v1.2 specification are not supported.
- The current implementation supports audio functions as described in Displayport 1.1a spec. New packets of Displayport 1.2 are not supported.
- Multi Stream Transport is not supported.
- FAUX is not supported.
- Bridging Function is not supported. The control registers required for bridging functionality are not included in the DisplayPort Configuration Data.

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core License Agreement. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability of Xilinx LogiCORE IP.

For more information about licensing for the core, see the DisplayPort product page.

CAUTION! *Users attempting to use the Audio feature without a license will not see an error until implementation, at which point tools will generate an error stating that an SPDIF and/or Reed Solomon Decoder license is not found.*

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

The Xilinx LogiCORE™ IP DisplayPort™ interconnect protocol is designed for transmission and reception of serial-digital video for consumer and professional displays. DisplayPort is a high-speed serial interface standard supported by PC chipsets, GPU's and display controllers, HDTV and monitors from industry leaders and major silicon manufacturers.

Standards

The IP described by this document is designed to be compatible with *DisplayPort Standard*, v1.1a and *DisplayPort Standard*, v1.2. For silicon status, please check with Xilinx.

While the functional cores each include an I2C compatible interface, the design does not provide a fully compliant implementation. Specifically, the I2C interface sections do not support multiple bus masters and bus arbitration.

This core supports a two-channel SPDIF controller along with DisplayPort Standard v1.1a Audio logic targeted for two channels.

Performance

This section contains details about the performance of this core.

Maximum Frequencies

The core uses six clock domains:

- `lnk_clk`. Most of the core operates in this domain. `lnk_clk` is directly related to link rate:

  `lnk_clk = link_rate`/20.

- `vid_clk`. This is the primary user interface clock. It runs as fast as 135 MHz, which enables a screen resolution of 2560x1600 when using two-wide pixels. See Selecting the Pixel Interface in Chapter 3 for more information on how to select the appropriate pixel interface.

- `s_axi_aclk`. This is the processor domain. The AUX clock domain is derived from this domain, but requires no additional constraints.
- `aud_clk`. This is the audio interface clock. The frequency will be equal to 512 x audio sample rate.
- `spdif_sample_clk`. This is used by the SPDIF receiver to sample incoming traffic. This clock should be ≥ 512 x audio sample rate.
- `aud_axis_aclk`. This clock is used by the Audio streaming interface. This clock should be ≥ 512 x audio sample rate.

Table 2-1 shows the clock ranges.

Table 2-1: **Clock Ranges**

Clock Domain	Min	Max	Description
lnk_clk	81 MHz	270 MHz(1)	Link clock
vid_clk	13.5 MHz	150 MHz	Video clock
s_axi_aclk	25 MHz	135 MHz	Host processor clock
aud_clk	16 MHz	100 MHz	Audio Clock (512 * Audio Sample Rate)
spdif_sample_clk	16 MHz	100 MHz	≥ Audio Clock
aud_axis_aclk	16 MHz	100 MHz	≥ Audio Clock

1. Valid for devices which support HBR2. HBR link rate will run at 135MHz.

Port Descriptions

Table 2-2: **Source Core I/O Signals**

Signal Name[a]	Direction From Core	Description
DisplayPort Processor Interface		
s_axi_aclk	Input	AXI Bus Clock.
s_axi_aresetn	Input	AXI Reset. Active-Low.
s_axi_awaddr[31:0]	Input	Write Address.
s_axi_awprot[2:0]	Input	Protection type.
s_axi_awvalid	Input	Write address valid.
s_axi_awready	Output	Write address ready.
s_axi_wdata[31:0]	Input	Write data bus.
s_axi_wstrb[3:0]	Input	Write strobes.
s_axi_wvalid	Input	Write valid.
s_axi_wready	Output	Write ready.
s_axi_bresp[1:0]	Output	Write response.

Table 2-2: **Source Core I/O Signals *(Cont'd)***

Signal Name[a]	Direction From Core	Description
s_axi_bvalid	Output	Write response valid.
s_axi_bready	Input	Response ready.
s_axi_araddr[31:0]	Input	Read address.
s_axi_arprot[2:0]	Input	Protection type.
s_axi_arvalid	Input	Read address valid.
s_axi_arready	Output	Read address ready.
s_axi_rdata[31:0]	Output	Read data.
s_axi_rresp[1:0]	Output	Read response.
s_axi_rvalid	Output	Read valid.
s_axi_rready	Input	Read ready.
axi_int	Output	AXI interrupt out.
User Data Interface		
tx_vid_clk	Input	User data video clock.
tx_vid_vsync	Input	Vertical sync pulse.
tx_vid_hsync	Input	Horizontal sync pulse.
tx_vid_oddeven	Input	Odd/even field select.
tx_vid_enable	Input	User data video enable.
tx_vid_pixel0[47:0]	Input	Video data.
tx_vid_pixel1[47:0]	Input	Video data.
tx_vid_pixel2[47:0]	Input	Video data.
tx_vid_pixel3[47:0]	Input	Video data.
tx_vid_rst	Input	User video reset.
Main Link Interface		
lnk_clk_p	Input	Differential clock input from pin.
lnk_clk_n	Input	Differential clock input from pin.
lnk_clk_81_p	Input	Differential link clock, 81 MHz. Must be placed on the MGTREVCLKP pin. Valid for Virtex-6 and Spartan-6 device families.
lnk_clk_81_n	Input	Differential link clock, 81 MHz. Must be placed on the MGTREVCLKN pin. Valid for Virtex-6 and Spartan-6 device families.
lnk_clk	Output	Reference clock for the FPGA fabric.
lnk_tx_lane_p[3:0]	Output	High-speed lane serial data.
lnk_tx_lane_n[3:0]	Output	High-speed lane serial data.

Table 2-2: **Source Core I/O Signals *(Cont'd)***

Signal Name[a]	Direction From Core	Description
AUX Channel Interface		
aux_tx_channel_in_p	Input	Positive polarity of the AUX Manchester-II data.
aux_tx_channel_in_n	Input	Negative polarity of the AUX Manchester-II data.
aux_tx_channel_out_p	Output	Positive polarity of the AUX Manchester-II data.
aux_tx_channel_out_n	Output	Negative polarity of the AUX Manchester-II data.
aux_tx_io_p	Input/Output	Positive Polarity AUX Manchester-II data (used for Spartan-6 devices)
aux_tx_io_n	Input/Output	Negative Polarity AUX Manchester-II data (used for Spartan-6 devices)
HPD Interface		
tx_hpd	Input	Hot Plug Detect.
SPDIF Audio Processor Interface		
aud_s_axi_aclk	input	AXI Bus Clock
aud_s_axi_aresetn	input	AXI Reset. Active-Low.
aud_s_axi_awaddr[31:0]	input	Write Address.
aud_s_axi_awprot[2:0],	input	Protection type.
aud_s_axi_awvalid	Input	Write address valid.
aud_s_axi_awready	Output	Write address ready.
aud_s_axi_wdata[31:0]	Input	Write data bus.
aud_s_axi_wstrb[3:0]	Input	Write strobes.
aud_s_axi_wvalid	Input	Write valid.
aud_s_axi_wready	Output	Write ready.
aud_s_axi_bresp[1:0]	Output	Write response.
aud_s_axi_bvalid	Output	Write response valid
aud_s_axi_bready	Input	Response ready.
aud_s_axi_araddr[31:0]	Input	Read address.
aud_s_axi_arprot[2:0]	Input	Protection type.
aud_s_axi_arvalid	Input	Read address valid.
aud_s_axi_arready	Output	Read address ready.
aud_s_axi_rdata[31:0]	Output	Read data.
aud_s_axi_rresp[1:0]	Output	Read response.
aud_s_axi_rvalid	Output	Read valid.
aud_s_axi_rready	Input	Read ready.

Table 2-2: **Source Core I/O Signals *(Cont'd)***

Signal Name[a]	Direction From Core	Description
aud_axi_int	Output	AXI interrupt out.
	SPDIF Interface	
spdif_in	Input	SPDIF channel input.
	Audio Clock Interface	
aud_clk	Input	Audio sample clock (512 * fs). fs= sampling frequency.
aud_rst	Input	Audio Interface Reset (Active-High).
aud_axis_aclk	Input	Audio streaming interface clock (greater than or equal to 512 * fs)
aud_axis_aresetn	Input	Audio Streaming Interface Reset (Active-Low).
spdif_sample_clk	Input	SPDIF Controller sampling clock. Should be greater than or equal to 512*fs.

a. Signal names beginning with `s_` or `m_` denote slave and master interfaces respectively.

Table 2-3: **Sink Core I/O Signals**

Signal Name[a]	Direction From Core	Description
	DisplayPort Processor Interface	
s_axi_aclk	Input	AXI Bus Clock .
s_axi_aresetn	Input	AXI Reset. Active-Low.
s_axi_awaddr[31:0]	Input	Write Address.
s_axi_awprot[2:0]	Input	Protection type.
s_axi_awvalid	Input	Write address valid.
s_axi_awready	Output	Write address ready.
s_axi_wdata[31:0]	Input	Write data bus.
s_axi_wstrb[3:0]	Input	Write strobes.
s_axi_wvalid	Input	Write valid.
s_axi_wready	Output	Write ready.
s_axi_bresp[1:0]	Output	Write response.
s_axi_bvalid	Output	Write response valid.
s_axi_bready	Input	Response ready.
s_axi_araddr[31:0]	Input	Read address.
s_axi_arprot[2:0]	Input	Protection type.
s_axi_arvalid	Input	Read address valid.
s_axi_arready	Output	Read address ready.
s_axi_rdata[31:0]	Output	Read data.

Table 2-3: **Sink Core I/O Signals *(Cont'd)***

Signal Name[a]	Direction From Core	Description
s_axi_rresp[1:0]	Output	Read reponse.
s_axi_rvalid	Output	Read valid.
s_axi_rready	Input	Read ready.
axi_int	Output	AXI interrupt out.
User Data Interface		
rx_vid_clk	Input	User data video clock.
rx_vid_vsync	Output	Vertical sync pulse.
rx_vid_hsync	Output	Horizontal sync pulse.
rx_vid_oddeven	Output	Odd/even field select.
rx_vid_enable	Output	User data video enable.
rx_vid_pixel0[47:0]	Output	Video data.
rx_vid_pixel1[47:0]	Output	Video data.
rx_vid_pixel2[47:0]	Output	Video data.
rx_vid_pixel3[47:0]	Output	Video data.
rx_vid_rst	Input	User video reset.
Main Link Interface		
lnk_clk	Output	Reference clock for the FPGA fabric.
lnk_clk_p	Input	Differential clock input from pin.
lnk_clk_n	Input	Differential clock input from pin.
lnk_clk_81_p	Input	Differential link clock, 81 MHz. Must be placed on the MGTREVCLKP pin. Valid for Virtex-6 and Spartan-6 device families.
lnk_clk_81_n	Input	Differential link clock, 81 MHz. Must be placed on the MGTREVCLKN pin. Valid for Virtex-6 and Spartan-6 device families.
lnk_rx_lane_p[3:0]	Input	High-speed lane serial data.
lnk_rx_lane_n[3:0]	Input	High-speed lane serial data.
lnk_m_vid[23:0]	Output	M-value for clock generation.
lnk_n_vid[23:0]	Output	N-value for clock generation.
lnk_m_aud[23:0]	Output	M-value for audio clock generation.
lnk_n_aud[23:0]	Output	N-Value for audio clock generation.
AUX Channel Interface		
aux_rx_channel_in_p	Input	Positive polarity of the AUX Manchester-II data.
aux_rx_channel_in_n	Input	Negative polarity of the AUX Manchester-II data.

Table 2-3: **Sink Core I/O Signals *(Cont'd)***

Signal Name[a]	Direction From Core	Description
aux_rx_channel_out_p	Output	Positive polarity of the AUX Manchester-II data.
aux_rx_channel_out_n	Output	Negative polarity of the AUX Manchester-II data.
aux_rx_io_p	Input/Output	Positive Polarity AUX Manchester-II data (used for Spartan-6 devices).
aux_rx_io_n	Input/Output	Negative Polarity AUX Manchester-II data (used for Spartan-6 devices).
I2C Interface		
i2c_sda_in	Input	I2C serial data in.
i2c_sda_enable_n	Output	I2C data out enable. Active-Low.
i2c_scl_in	Input	I2C serial clock in.
i2c_scl_enable_n	Output	I2C serial clock output enable. Active-Low.
HPD Interface		
rx_hpd	Output	Hot Plug Detect.
SPDIF Audio Processor Interface		
aud_s_axi_aclk	Input	AXI Bus Clock.
aud_s_axi_aresetn	Input	AXI Reset. Active-Low.
aud_s_axi_awaddr[31:0]	Input	Write Address.
aud_s_axi_awprot[2:0]	Input	Protection type.
aud_s_axi_awvalid	Input	Write address valid.
aud_s_axi_awready	Output	Write address ready.
aud_s_axi_wdata[31:0]	Input	Write data bus.
aud_s_axi_wstrb[3:0]	Input	Write strobes.
aud_s_axi_wvalid	Input	Write valid.
aud_s_axi_wready	Output	Write ready.
aud_s_axi_bresp[1:0]	Output	Write response.
aud_s_axi_bvalid	Output	Write response valid
aud_s_axi_bready	Input	Response ready.
aud_s_axi_araddr[31:0]	Input	Read address.
aud_s_axi_arprot[2:0]	Input	Protection type.
aud_s_axi_arvalid	Input	Read address valid.
aud_s_axi_arready	Output	Read address ready.
aud_s_axi_rdata[31:0]	Output	Read data.
aud_s_axi_rresp[1:0]	Output	Read response.
aud_s_axi_rvalid	Output	Read valid.

Table 2-3: **Sink Core I/O Signals *(Cont'd)***

Signal Name[a]	Direction From Core	Description
aud_s_axi_rready	Input	Read ready.
aud_axi_int	Output	AXI interrupt out. AXI interrupt out of SPDIF controller.
Audio Clock Interface		
aud_clk	Input	Audio sample clock (512 * fs). fs= sampling frequency.
aud_rst	Input	Audio Interface Reset (Active-High).
aud_axis_aclk	Input	Audio streaming interface clock (greater than or equal to 512 * fs)
aud_axis_aresetn	Input	Audio Streaming Interface Reset (Active-Low).
SPDIF Interface		
spdif_out	Output	SPDIF channel output

a. Signal names beginning with s_ or m_ denote slave and master interfaces respectively.

Audio Streaming Signals

The DisplayPort Source Audio streaming signals are listed in Table 2-4.

Table 2-4: **DisplayPort Source Audio Interface**

S. No	Name	Direction	Description
1	tx_s_axis_audio_ingress_aclk	Input	AXI Streaming Clock
2	tx_s_axis_audio_ingress_aresetn	Input	Active LOW reset
3	tx_s_axis_audio_ingress_tdata [31:0]	Input	Streaming data input. • [3:0] – PR (Preamble Code) ◦ 4'b0001 -> Subframe1 / start of audio block ◦ 4'b0010 -> Subframe 1 ◦ 4'b0011 -> Subframe 2 • [27:4] – Audio Sample Word • [28] – V (Validity Bit) • [29] – U (User Bit) • [30] – C (Channel Status) • [31] – P (Parity)
4	tx_s_axis_audio_ingress_tid [2:0]	Input	Audio channel ID. Range [0:7]
5	tx_s_axis_audio_ingress_tvalid	Input	Valid indicator for audio data from master.
6	tx_s_axis_audio_ingress_tready	Output	Ready indicator from DisplayPort source.

The DisplayPort Sink Audio streaming definition is listed in Table 2-5.

Table 2-5: **DisplayPort Sink Audio Interface**

S.No	Name	Direction	Description
1	rx_m_axis_audio_egress_aclk	Input	AXI Streaming Clock
2	rx_m_axis_audio_egress_aresetn	Input	Active-Low reset
3	rx_m_axis_audio_egress_tdata [31:0]	Output	Streaming data output. • [3:0] – PR (Preamble Code) ◦ 4'b0001 -> Subframe1 / start of audio block ◦ 4'b0010 -> Subframe 1 ◦ 4'b0011 -> Subframe 2 • [27:4] – Audio Sample Word • [28] – V (Validity Bit) • [29] – U (User Bit) • [30] – C (Channel Status) • [31] – P (Parity)
4	rx_m_axis_audio_egress_tid [2:0]	Output	Audio channel ID. Range [0:7]
5	rx_m_axis_audio_egress_tvalid	Output	Valid indicator for audio data from master.
6	rx_m_axis_audio_egress_tready	Input	Ready indicator from external streaming module.

Register Space

Source Core

The DisplayPort Configuration Data is implemented as a set of distributed registers which may be read or written from the AXI4-Lite interface. These registers are considered to be synchronous to the AXI4-Lite domain and asynchronous to all others.

For parameters that may change while being read from the configuration space, two scenarios may exist. In the case of single bits, the data may be read without concern as either the new value or the old value will be read as valid data. In the case of multiple bit fields, a lock bit may be used to prevent the status values from being updated while the read is occurring. For multi-bit configuration data, a toggle bit will be used indicating that the local values in the functional core should be updated.

Any bits not specified in Table 2-6 are considered reserved and will return '0' upon read.

Table 2-6: **DisplayPort Source Core Configuration Space**

Offset	R/W	Definition
Link Configuration Field		
0x000	RW	LINK_BW_SET. Main link bandwidth setting. The register uses the same values as those supported by the DPCD register of the same name in the sink device. • [7:0] - LINK_BW_SET: Sets the value of the main link bandwidth for the sink device. ◦ 0x06 = 1.62 Gbps ◦ 0x0A = 2.7 Gbps ◦ 0x14 = 5.4 Gbps (7 series family with protocol version 1.2 only)
0x004	RW	LANE_COUNT_SET. Sets the number of lanes that will be used by the source in transmitting data. • [4:0] - Set to 1, 2, or 4
0x008	RW	ENHANCED_FRAME_EN • [0] -Set to '1' by the source to enable the enhanced framing symbol sequence.
0x00C	RW	TRAINING_PATTERN_SET. Sets the link training mode. • [1:0] - Set the link training pattern according to the two bit code. ◦ 00 = Training off ◦ 01 = Training pattern 1, used for clock recovery ◦ 10 = Training pattern 2, used for channel equalization ◦ 11 = Training pattern 3, used for channel equalization for cores with DisplayPort v1.2.

0x010	RW	LINK_QUAL_PATTERN_SET. Transmit the link quality pattern.
		• [1:0] - Enable transmission of the link quality test patterns.
		  ◦ 00 = Link quality test pattern not transmitted
		  ◦ 01 = D10.2 test pattern (unscrambled) transmitted
		  ◦ 10 = Symbol Error Rate measurement pattern
		  ◦ 11 = PRBS7 transmitted

0x014	RW	SCRAMBLING_DISABLE. Set to '1' when the transmitter has disabled the scrambler and transmits all symbols. • [0] - Disable scrambling.
0x018	RW	DOWNSPREAD_CTRL. Down-spreading control. • [0] -Set to '1' to enable a 0.5% spreading of the clock or '0' for none.
0x01C	WO	SOFTWARE_RESET. Reads will return zeros. • - [0] - Soft Video Reset: When set, video logic will be reset.
Core Enables		
0x080	RW	TRANSMITTER_ENABLE. Enable the basic operations of the transmitter. • [0] - When set to '0', all lanes of the main link will output stuffing symbols.
0x084	RW	MAIN_STREAM_ENABLE. Enable the transmission of main link video information. • [0] - When set to '0', the active lanes of the DisplayPort transmitter will output only VB-ID information with the NoVideo flag set to '1'. ***Note:*** Main stream enable/disable functionality is gated by the VSYNC input. The values written in the register are applied at the video frame boundary only.

Table 2-6: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x088	RW	SECONDARY_STREAM_ENABLE. Enable the transmission of secondary link information. • [0] - A value of '0' in this register disables the secondary stream.
0x0C0	WO	FORCE_SCRAMBLER_RESET. Reads from this register always return 0x0. • [0] - '1' forces a scrambler reset.
Core ID		
0x0F8	RO	VERSION_REGISTER. For displayport_v3_2, VERSION REGISTER will be 32'h03_02_0_0_00. • [31:24] - Core major version. • 23:16] - Core minor version. • [15:12] - Core version revision. • [11:8] - Core Patch details. • [7:0] - Internal revision.
0x0FC	RO	CORE_ID. Returns the unique identification code of the core and the current revision level. • [31:24] - DisplayPort protocol major version • [23:16] - DisplayPort protocol minor version • [15:8] - DisplayPort protocol revision • [7:0] ◦ 0x00: Transmit ◦ 0x01: Receive The CORE_ID values for the various protocols and cores are: • DisplayPort v1.1a protocol with a Transmit core: 32'h01_01_0a_00 • DisplayPort v1.2 protocol with a Transmit core: 32'h01_02_00_00
AUX Channel Interface		

0x100	RW	AUX_COMMAND_REGISTER. Initiates AUX channel commands of the specified length.
		• [12] - Address only transfer enable. When this bit is set to 1, the source will initiate Address only transfers (STOP will be sent after the command).
		• [11:8] - AUX Channel Command.
		  ◦ 0x8 = AUX Write
		  ◦ 0x9 = AUX Read
		  ◦ 0x0 = IC Write
		  ◦ 0x4 = IC Write MOT
		  ◦ 0x1 = IC Read
		  ◦ 0x5 = IC Read MOT
		  ◦ 0x2 = IC Write Status
		• [3:0] - Specifies the number of bytes to transfer with the current command. The range of the register is 0 to 15 indicating between 1 and 16 bytes of data.

0x104	WO	AUX_WRITE_FIFO. FIFO containing up to 16 bytes of write data for the current AUX channel command. • [7:0] - AUX Channel byte data.
0x108	RW	AUX_ADDRESS. Specifies the address for the current AUX channel command. • [19:0] - Twenty bit address for the start of the AUX Channel burst.

Table 2-6: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x10C	RW	AUX_CLOCK_DIVIDER. Contains the clock divider value for generating the internal 1MHz clock from the AXI4-Lite host interface clock. The clock divider register provides integer division only and does not support fractional AXI4-Lite clock rates (for example, set to 75 for a 75 MHz AXI4-Lite clock). • [7:0] - Clock divider value.
0x110	RC	TX_USER_FIFO_OVERFLOW. Indicates an overflow in the user FIFO. The event may occur if the video rate does not match the TU size programming. • [0] - FIFO_OVERFLOW_FLAG: A '1' indicates that the internal FIFO has detected an overflow condition. This bit clears upon read.
0x130	RO	INTERRUPT_SIGNAL_STATE. Contains the raw signal values for those conditions which may cause an interrupt. • [3] - REPLY_TIMEOUT: A '1' indicates that a reply timeout has occurred. • [2] - REPLY_STATE: A'1' indicates that a reply is currently being received. • [1] - REQUEST_STATE: A'1' indicates that a request is currently being sent. • [0] - HPD_STATE: Contains the raw state of the HPD pin on the DisplayPort connector.
0x134	RO	AUX_REPLY_DATA. Maps to the internal FIFO which contains up to 16 bytes of information received during the AUX channel reply. Reply data is read from the FIFO starting with byte 0. The number of bytes in the FIFO corresponds to the number of bytes requested. • [7:0] - AUX reply data
0x138	RO	AUX_REPLY_CODE. Reply code received from the most recent AUX Channel request. The AUX Reply Code corresponds to the code from the DisplayPort specification. ***Note:*** The core will not retry any commands that were Deferred or Not Acknowledged. • [1:0] ◦ 00 = AUX ACK ◦ 01 = AUX NACK ◦ 10 = AUX DEFER • [3:2] ◦ 00 = I2C ACK ◦ 01 = I2C NACK ◦ 10 = I2C DEFER
0x13C	RW	AUX_REPLY_COUNT. Provides an internal counter of the number of AUX reply transactions received on the AUX Channel. Writing to this register clears the count. • [7:0] - Current reply count.

Table 2-6: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x140	RC	INTERRUPT_STATUS. Source core interrupt status register. A read from this register clears all values. • [5] - EXT_PKT_TXD: Extended packet is transmitted and controller is ready to accept new packet. • [4] - HPD_PULSE_DETECTED: A pulse on the HPD line was detected. The duration of the pulse can be determined by reading 0x150. • [3] - REPLY_TIMEOUT: A reply timeout has occurred. • [2] - REPLY_RECEIVED: An AUX reply transaction has been detected. • [1] - HPD_EVENT: The core has detected the presence of the HPD signal. This interrupt asserts immediately after the detection of HPD and after the loss of HPD for 2 msec. • [0] - HPD_IRQ: An IRQ framed with the proper timing on the HPD signal has been detected.
0x144	RW	INTERRUPT_MASK. Masks the specified interrupt sources from asserting the axi_init signal. When set to a 1, the specified interrupt source is masked. This register resets to all 1s at power up. The respective MASK bit controls the assertion of axi_int only and does not affect events updated in the INTERRUPT_STATUS register. • [5] - EXT_PKT_TXD: Mask Extended Packet Transmitted interrupt. • [4] - HPD_PULSE_DETECTED: Mask HPD Pulse interrupt. • [3] - REPLY_TIMEOUT: Mask reply timeout interrupt. • [2] - REPLY_RECEIVED: Mask reply received interrupt. • [1] - HPD_EVENT: Mask HPD event interrupt. • [0] - HPD_IRQ: Mask HPD IRQ interrupt.
0x148	RO	REPLY_DATA_COUNT. Returns the total number of data bytes actually received during a transaction. This register does not use the length byte of the transaction header. • [4:0] - Total number of data bytes received during the reply phase of the AUX transaction.
0x14C	RO	REPLY_STATUS • [15:12] - RESERVED • [11:4] - REPLY_STATUS_STATE: Internal AUX reply state machine status bits. • [3] - REPLY_ERROR: When set to a '1', the AUX reply logic has detected an error in the reply to the most recent AUX transaction. • [2] - REQUEST_IN_PROGRESS: The AUX transaction request controller sets this bit to a '1' while actively transmitting a request on the AUX serial bus. The bit is set to '0' when the AUX transaction request controller is idle. • [1] - REPLY_IN_PROGRESS: The AUX reply detection logic sets this bit to a '1' while receiving a reply on the AUX serial bus. The bit is '0' otherwise. • [0] - REPLY_RECEIVED: This bit is set to '0' when the AUX request controller begins sending bits on the AUX serial bus. The AUX reply controller sets this bit to '1' when a complete and valid reply transaction has been received.
0x150	RO	HPD_DURATION • [15:0] - Duration of the HPD pulse in microseconds.
Main Stream Attributes (Refer to the DisplayPort specification for more details [Ref 1].)		
0x180	RW	MAIN_STREAM_HTOTAL. Specifies the total number of clocks in the horizontal framing period for the main stream video signal. • [15:0] - Horizontal line length total in clocks.

Table 2-6: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x184	RW	MAIN_STREAM_VTOTAL. Provides the total number of lines in the main stream video frame. • [15:0] - Total number of lines per video frame.
0x188	RW	MAIN_STREAM_POLARITY. Provides the polarity values for the video sync signals. • [1] - VSYNC_POLARITY: Polarity of the vertical sync pulse. • [0] - HSYNC_POLARITY: Polarity of the horizontal sync pulse.
0x18C	RW	MAIN_STREAM_HSWIDTH. Sets the width of the horizontal sync pulse. • [14:0] - Horizontal sync width in clock cycles.
0x190	RW	MAIN_STREAM_VSWIDTH. Sets the width of the vertical sync pulse. • [14:0] - Width of the vertical sync in lines.
0x194	RW	MAIN_STREAM_HRES. Horizontal resolution of the main stream video source. • [15:0] - Number of active pixels per line of the main stream video.
0x198	RW	MAIN_STREAM_VRES. Vertical resolution of the main stream video source. • [15:0] - Number of active lines of video in the main stream video source.
0x19C	RW	MAIN_STREAM_HSTART. Number of clocks between the leading edge of the horizontal sync and the start of active data. • [15:0] - Horizontal start clock count.
0x1A0	RW	MAIN_STREAM_VSTART. Number of lines between the leading edge of the vertical sync and the first line of active data. • [15:0] - Vertical start line count.
0x1A4	RW	MAIN_STREAM_MISC0. Miscellaneous stream attributes. • [7:0] - Implements the attribute information contained in the DisplayPort MISC0 register described in section 2.2.4 of the standard. • [0] -Synchronous Clock. • [2:1] - Component Format. • [3] - Dynamic Range. • [4] - YCbCr Colorimetry. • [7:5] - Bit depth per color/component.
0x1A8	RW	MAIN_STREAM_MISC1. Miscellaneous stream attributes. • [7:0] - Implements the attribute information contained in the DisplayPort MISC1 register described in section 2.2.4 of the standard. • [0] - Interlaced vertical total even. • [2:1] - Stereo video attribute. • [7:3] - Reserved.
0x1AC	RW	M-VID. M value for the video stream as computed by the source core. If synchronous clocking mode is used, this register must be written with the M value. • [23:0] - Unsigned value computed in the asynchronous clock mode.
0x1B0	RW	TRANSFER_UNIT_SIZE. Sets the size of a transfer unit in the framing logic On reset, transfer size is set to 64. • [6:0] - This number should be in the range of 32 to 64 and is set to a fixed value that depends on the inbound video mode. Note that bit 0 cannot be written (the transfer unit size is always even).

Table 2-6: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x1B4	RW	N-VID. N value for the video stream as computed by the source core. If synchronous clocking mode is used, this register must be written with the N value. • [23:0] - Unsigned value computed in the asynchronous clock mode.
0x1B8	RW	USER_PIXEL_WIDTH. Selects the width of the user data input port. • [2:0]: ◦ 1 = Single pixel wide interface ◦ 2 = Dual pixel wide interface ◦ 4 = Quad pixel wide interface
0x1BC	RW	USER_DATA_COUNT_PER_LANE. This register is used to translate the number of pixels per line to the native internal 16-bit datapath. If (HRES * bits per pixel) is divisible by 16, then word_per_line = ((HRES * bits per pixel)/16) Else word_per_line = (INT((HRES * bits per pixel)/16))+1 **For single-lane design**: Set USER_DATA_COUNT_PER_LANE = words_per_line - 1 **For 2-lane design:** If words_per_line is divisble by 2, then Set USER_DATA_COUNT_PER_LANE = words_per_line - 2 Else Set USER_DATA_COUNT_PER_LANE = words_per_line + MOD(words_per_line,2) - 2 **For 4-lane design:** If words_per_line is divisble by 4, then Set USER_DATA_COUNT_PER_LANE = words_per_line - 4 Else Set USER_DATA_COUNT_PER_LANE = words_per_line + MOD(words_per_line,4) - 4
0x1C0	RW	MAIN_STREAM_INTERLACED. Informs the DisplayPort transmitter main link that the source video is interlaced. By setting this bit to a '1', the core will set the appropriate fields in the VBID value and Main Stream Attributes. This bit must be set to a '1' for the proper transmission of interlaced sources. • [0] - Set to a '1' when transmitting interlaced images.
0x1C4	RW	MIN_BYTES_PER_TU: Programs source to use MIN number of bytes per transfer unit. The calculation should be done based on the DisplayPort specification. • [7:0] - Set the value to INT((LINK_BW/VIDEO_BW)*TRANSFER_UNIT_SIZE)
0x1C8	RW	FRAC_BYTES_PER_TU: Calculating MIN bytes per TU will often not be a whole number. This register is used to hold the fractional component. • [9:0] - The fraction part of ((LINK_BW/VIDEO_BW)*TRANSFER_UNIT_SIZE) scaled by 1000 is programmed in this register.

Table 2-6: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x1cc	RW	INIT_WAIT: This register defines the number of initial wait cycles at the start of a new line by the Framing logic. This allows enough data to be buffered in the input FIFO. If (MIN_BYTES_PER_TU <= 4) • [7:0] - Set INIT_WAIT to 64 Else • [7:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)
PHY Configuration Status		
0x200	RW	PHY_RESET. Reset the transmitter PHY. • [0] - Set to '1' to hold the PHY in reset. Clear to release.
0x210	RW	PHY_PRE-EMPHASIS_LANE_0. Set the pre-emphasis level for lane 0 of the DisplayPort link. • [2:0] - Controls the pre-emphasis level for lane 0 of the transmitter. Up to eight levels are supported for a wide variety of possible PHY implementations. The mapping of the four levels supported by the DisplayPort standard to the eight levels indicated here is implementation specific.
0x214	RW	PHY_PRE-EMPHASIS_LANE_1. Bit definition identical to that of PHY_PREEMPHASIS_LANE_0.
0x218	RW	PHY_PRE-EMPHASIS_LANE_2. Bit definition identical to that of PHY_PREEMPHASIS_LANE_0.
0x21C	RW	PHY_PRE-EMPHASIS_LANE_3. Bit definition identical to that of PHY_PREEMPHASIS_LANE_0.
0x220	RW	PHY_VOLTAGE_DIFF_LANE_0. Controls the differential voltage swing for lane 0 of the DisplayPort link. • [2:0] - Supports up to eight levels of voltage swing for a wide variety of PHY implementations. The mapping of the four levels supported by the DisplayPort specification to the eight levels indicated here is implementation specific.
0x224	RW	PHY_VOLTAGE_DIFF_LANE_1. Bit definition identical to that of PHY_PREEMPHASIS_LANE_0.
0x228	RW	PHY_VOLTAGE_DIFF_LANE_2. Bit definition identical to that of PHY_PREEMPHASIS_LANE_0.
0x22C	RW	PHY_VOLTAGE_DIFF_LANE_3. Bit definition identical to that of PHY_PREEMPHASIS_LANE_0.
0x230	RW	TRANSMIT_PRBS7. Enable the pseudo random bit sequence 7 pattern transmission for link quality assessment. • [0] - A'1' in this bit enables the transmission of the sequence.
0x234	RW	PHY_CLOCK_SELECT. Instructs the PHY PLL to generate the proper clock frequency for the required link rate. • [2:0] ◦ 0x05 = 5.40 Gb/s link ◦ 0x03 = 2.70 Gb/s link ◦ 0x01 = 1.62 Gb/s link

Table 2-6: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x238	RW	TX_PHY_POWER_DOWN [3:0]. Control PHY Power down. One bit per lane. When set to 1, moves the GT to power down mode.
0x23c	RW	PHY_PRECURSOR_LANE_0. Set the pre-cursor level for lane 0 of the DisplayPort link (non-Spartan-6 devices). • [4:0] - Controls the pre-cursor level for lane 0 of the transmitter. The mapping of the four levels supported by the DisplayPort standard to the 32 levels indicated here is implementation specific.
0x240	RW	PHY_PRECURSOR_LANE_1. Bit definition identical to that of PHY_PRECURSOR_LANE_0.
0x244	RW	PHY_PRECURSOR_LANE_2. Bit definition identical to that of PHY_PRECURSOR_LANE_0.
0x248	RW	PHY_PRECURSOR_LANE_3. Bit definition identical to that of PHY_PRECURSOR_LANE_0.
0x24c	RW	PHY_POSTCURSOR_LANE_0. Set the post-cursor level for lane 0 of the DisplayPort link (non-Spartan-6 devices). • [4:0] - Controls the post-cursor level for lane 0 of the transmitter. The mapping of the four levels supported by the DisplayPort standard to the 32 levels indicated here is implementation specific.
0x250	RW	PHY_POSTCURSOR_LANE_1. Bit definition identical to that of PHY_POSTCURSOR_LANE_0.
0x254	RW	PHY_POSTCURSOR_LANE_2. Bit definition identical to that of PHY_POSTCURSOR_LANE_0.
0x258	RW	PHY_POSTCURSOR_LANE_3. Bit definition identical to that of PHY_POSTCURSOR_LANE_0.
0x280	RO	PHY_STATUS. Provides the current status from the PHY. • [1:0] - Reset done for lanes 0 and 1. • [3:2] - Reset done for lanes 2 and 3. • [4] - PLL for lanes 0 and 1 locked. • [5] - PLL for lanes 2 and 3 locked. • [6] - FPGA fabric clock PLL locked. • [15:7] - Unused, read as 0. • [17:16] - Transmitter buffer status, lane 0. • [19:18] - Transmitter error, lane 0. • [21:20]- Transmitter buffer status, lane 1. • [23:22] - Transmitter error, lane 1. • [25:24] - Transmitter buffer status, lane 2. • [27:26] - Transmitter error, lane 2. • [29:28] - Transmitter buffer status, lane 3. • [31:30] - Transmitter error, lane 3.

DisplayPort Audio

The DisplayPort Audio registers are listed in Table 2-7.

Table 2-7: **DisplayPort Audio Registers**

Offset	R/W	Definition
0x300	R/W	TX_AUDIO_CONTROL. Enables audio stream packets in main link and provides buffer control. • [0]: Audio Enable
0x304	R/W	TX_AUDIO_CHANNELS. Used to input active channel count. Transmitter collects audio samples based on this information. • [2:0] Channel Count
0x308	Write Only	TX_AUDIO_INFO_DATA. [31:0] Word formatted as per CEA 861-C Info Frame. Total of eight words should be written in following order: • 1st word – ◦ [7:0] = HB0 ◦ [15:8] = HB1 ◦ [23:16] = HB2 ◦ [31:24] = HB3 • 2nd word – DB3,DB2,DB1,DB0 . . • 8th word –DB27,DB26,DB25,DB24 The data bytes DB1...DBN of CEA Info frame are mapped as DB0-DBN-1. No protection is provided for wrong operations by software.
0x328	R/W	TX_AUDIO_MAUD. M value of audio stream as computed by transmitter. • [23:0] = Unsigned value computed when audio clock and link clock are synchronous.
0x32C	R/W	TX_AUDIO_NAUD. N value of audio stream as computed by transmitter. • [23:0] = Unsigned value computed when audio clock and link clock are synchronous.
0x330 - 0x350	WO	TX_AUDIO_EXT_DATA. [31:0] = Word formatted as per Extension packet described in protocol specification. Extended packet is fixed to 32 Bytes length. The controller has buffer space for only one extended packet. A total of nine words should be written in following order: • 1st word - ◦ [7:0] = HB0 ◦ [15:8] = HB1 ◦ [23:16] = HB2 ◦ [31:24] = HB3 • 2nd word - DB3,DB2,DB1,DB0 ... • 9th word -DB31,DB30,DB29,DB28 See the DisplayPort specification for HB* definition. No protection is provided for wrong operations by software. This is a key-hole memory. So, nine writes to this address space is required.

Sink Core

The DisplayPort Configuration Data is implemented as a set of distributed registers which may be read or written from the AXI4-Lite interface. These registers are considered to be synchronous to the AXI4-Lite domain and asynchronous to all others.

For parameters that may change while being read from the configuration space, two scenarios may exist. In the case of single bits, the data may be read without concern as either the new value or the old value will be read as valid data. In the case of multiple bit fields, a lock bit may be maintained to prevent the status values from being updated while the read is occurring. For multi-bit configuration data, a toggle bit will be used indicating that the local values in the functional core should be updated.

Any bits not specified in Table 2-8 are to be considered reserved and will return '0' upon read.

Table 2-8: **DisplayPort Sink Core Configuration Space**

Offset	R/W	Definition
Receiver Core Configuration		
0x000	RW	LINK_ENABLE. Enable the receiver • 1 - Enables the receiver core. Asserts the HPD signal when set.
0x004	RW	AUX_CLOCK_DIVIDER. Contains the clock divider value for generating the internal 1 MHz clock from the AXI4-Lite host interface clock. The clock divider register provides integer division only and does not support fractional AXI4-Lite clock rates (for example, set to 75 for a 75 MHz AXI4-Lite clock). • 7:0 - Clock divider value.
0x00C	RW	DTG_ENABLE. Enables the display timing generator in the user interface. • 0 - DTG_ENABLE: Set to '1' to enable the timing generator. The DTG should be disabled when the core detects the no-video pattern on the link.
0x010	RW	USER_PIXEL_WIDTH. Configures the number of pixels output through the user data interface. The Sink controller programs the pixel width to the active lane count (default). User can override this by writing a new value to this register. • 2:0 ◦ 1 = Single pixel wide interface. ◦ 2 = Dual pixel output mode. Valid for designs with 2 or 4 lanes. ◦ 4 = Quad pixel output mode. Valid for designs with 4 lanes only.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x014	RW	INTERRUPT_MASK. Masks the specified interrupt sources from asserting the axi_init signal. When set to a '1', the specified interrupt source is masked. This register resets to all 1s at power up. • 9 - EXT_PKT_RXD: Set to '1' when extension packet is received. • 8 - INFO_PKT_RXD: Set to '1' when info packet is received. • 6 - VIDEO: Set to '1' when valid video frame is detected on main link. Video interrupt is set after a delay of eight video frames following a valid scrambler reset character. • 4 - TRAINING_LOST: Training has been lost on active lanes. • 3 - VERTICAL_BLANKING: Start of the vertical blanking interval. • 2 - NO_VIDEO: The no-video condition has been detected after active video received. • 1 - POWER_STATE: Power state change, DPCD register value 0x00600. • 0 - MODE_CHANGE: Resolution change, as detected from the MSA fields.
0x018	RW	MISC_CONTROL. Allows the host to instruct the receiver to pass the MSA values through unfiltered. • 0 - USE_FILTERED_MSA: When set to '0', this bit disables the filter on the MSA values received by the core. When set to '1', two matching values must be detected for each field of the MSA values before the associated register is updated internally. • 1 - When set to '1', the long I2C write data transfers are responded to using DEFER instead of Partial ACKs. • 2 - When set to '1', I2C DEFERs will be sent as AUX DEFERs to the source device.
0x01C	WO	SOFTWARE_RESET_REGISTER. • 0 - Soft Video Reset: When set, video logic will be reset. Reads will return zeros.
AUX Channel Status		
0x020	RO	AUX_REQUEST_IN_PROGRESS. Indicates the receipt of an AUX Channel request • 0 - A'1' indicates a request is in progress.
0x024	RO	REQUEST_ERROR_COUNT. Provides a running total of errors detected on inbound AUX Channel requests. • 7:0 - Error count, a write to register address 0x28 clears this counter.
0x028	RO	REQUEST_COUNT. Provides a running total of the number of AUX requests received. • 7:0 - Total AUX request count, a write to register 0x28 clears this counter.
0x02C	WO	HPD_INTERRUPT. Instructs the receiver core to assert an interrupt to the transmitter using the HPD signal. A read from this register always returns 0x0. • 0 - Set to '1' to send the interrupt through the HPD signal. The HPD signal is brought low for 750 us to indicate to the source that an interrupt has been requested.
0x030	RO	REQUEST_CLOCK_WIDTH. Width of the recovered AUX clock from the most recent request. • 9:0 - Indicates the number of AXI_CLK cycles between sequential rising edges during the SYNC period of the most recent AUX request.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x034	RO	REQUEST_COMMAND. Provides the most recent AUX command received. • 3:0 - Provides the command field of the most recently received AUX request.
0x038	RO	REQUEST_ADDRESS. Contains the address field of the most recent AUX request. • 19:0 - The twenty-bit address field from the most recent AUX request transaction is placed in this register. For I2C over AUX transactions, the address range will be limited to the seven LSBs.
0x03C	RO	REQUEST_LENGTH. The length of the most recent AUX request is written to this register. The length of the AUX request is the value of this register plus one. • 3:0 - Contains the length of the AUX request. Transaction lengths from 1 to 16 bytes are supported. For address only transactions, the value of this register will be 0.
0x040	RC	INTERRUPT_CAUSE. Indicates the cause of a pending host interrupt. A read from this register clears all values. • 9 - EXT_PKT_RXD: Set to '1' when extension packet is received. • 8 - INFO_PKT_RXD: Set to '1' when info packet is received. • 6 - VIDEO: Set to '1' when a valid video frame is detected on main link. • 5 - Reserved • 4 - TRAINING_LOST: This interrupt is set when the receiver has been trained and subsequently loses clock recovery, symbol lock or inter-lane alignment. • 3 - VERTICAL_BLANKING: This interrupt is set at the start of the vertical blanking interval as indicated by the VerticalBlanking_Flag in the VB-ID field of the received stream. • 2 - NO_VIDEO: the receiver has detected the no-video flags in the VBID field after active video has been received. • 1 - POWER_STATE: The transmitter has requested a change in the current power state of the receiver core. • 0 - VIDEO_MODE_CHANGE: A change has been detected in the current video mode transmitted on the DisplayPort link as indicated by the MSA fields. The horizontal and vertical resolution parameters are monitored for changes.
0x050	RW	HSYNC_WIDTH. The display timing generator control logic outputs a fixed length, active-High pulse for the horizontal sync. The timing of this pulse may be controlled by setting this register appropriately. The default value of this register is 0x0f0f. • [15:8] - HSYNC_FRONT_PORCH: Defines the number of video clock cycles to place between the last pixel of active data and the start of the horizontal sync pulse. • [7:0] - HSYNC_PULSE_WIDTH: Specifies the number of clock cycles the horizontal sync pulse is asserted. The vid_hsync signal will be high for the specified number of clock cycles.
0x060	RW	FAST_I2C_DIVIDER. Fast I2C mode clock divider value. Set this value to (AXI4-Lite clock frequency/10) - 1. Valid only for DPCD 1.2.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
DPCD Fields		
0x084	RW	LOCAL_EDID_VIDEO. Indicates the presence of EDID information for the video stream. • 0 - Set to '1' to indicate to the transmitter through the DPCD registers that the receiver supports local EDID information.
0x088	RW	LOCAL_EDID_AUDIO. Indicates the presence of EDID information for the audio stream. • 0 - Set to '1' to indicate to the transmitter through the DPCD registers that the receiver supports local EDID information
0x08C	RW	REMOTE_COMMAND. General byte for passing remote information to the transmitter. • 7:0 - Remote data byte.
0x090	RW	DEVICE_SERVICE_IRQ. Indicates DPCD DEVICE_SERVICE_IRQ_VECTOR state. • 0 - Set to '1' to indicate a new command. Indicates a new command present in the REMOTE_COMMAND register. A Write of 0x1 to this register sets the DPCD register DEVICE_SERVICE_IRQ_VECTOR (0x201), REMOTE_CONTROL_PENDING bit. A write of 0x0 to this register has no effect. Refer to DPCD register section of the specification for more details. Reads from this register reflect the state of DPCD register. • 1 - Reflects SINK_SPECIFIC_IRQ state of DPCD 0x201 register.
0x094	RW	VIDEO_UNSUPPORTED. DPCD register bit to inform the transmitter that video data is not supported. • 0 - Set to '1' when video data is not supported.
0x098	RW	AUDIO_UNSUPPORTED. DPCD register bit to inform the transmitter that audio data is not supported • 0 - Set to '1' when audio data is not supported.
0x09c	RW	Override LINK_BW_SET. This register can be used to override LINK_BW_SET in the DPCD register set. Register 0x0b8 (apb_direct_dpcd_access) must be set to '1' to override DPCD values. • 4:0 - Link rate override value for DisplayPort v1.2 protocol designs • 3:0 - Link rate override value for DisplayPort v1.1a protocol designs ◦ 0x6 - 1.62 G ◦ 0xA - 2.7 G ◦ 0x14 - 5.4 G
0x0A0	RW	Override LANE_COUNT_SET. This register can be used to override LANE_COUNT_SET in the DPCD register set. Register 0x0b8 (apb_direct_dpcd_access) must be set to '1' to override DPCD values. • 4:0 - Lane count override value (1, 2 or 4 lanes • 6 - TPS3_SUPPORTED: Capability override for DisplayPort v1.2 protocol designs only. Reserved for v1.1a protocol. • 7 - ENHANCED_FRAME_CAP: Capability override

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x0A4	RW	Override TRAINING_PATTERN_SET. This register can be used to override TRAINING_PATTERN_SET in the DPCD register set. Register 0x0b8 (apb_direct_dpcd_access) must be set to '1' to override DPCD values. • 1:0 - TRAINING_PATTERN_SELECT Override • 3:2 - LINK_QUAL_PATTERN_SET Override for DisplayPort v1.1a only. • 4 - RECOVERED_CLOCK_OUT_EN Override • 5 - SCRAMBLING_DISABLE Override • 7:6 - SYMBOL ERROR COUNT SEL Override
0x0A8	RW	Override TRAINING_LANE0_SET. This register can be used to override TRAINING_LANE0_SET in the DPCD register set. Register 0x0b8 (apb_direct_dpcd_access) must be set to '1' to override DPCD values. • 1:0 - VOLTAGE SWING SET override • 2 - MAX_SWING_REACHED override • 4:3 - PRE-EMPHASIS_SET override • 5 - MAX_PRE-EMPHASIS_REACHED override • 7:6 - Reserved
0x0AC	RW	Override TRAINING_LANE1_SET. This register can be used to override TRAINING_LANE1_SET in the DPCD register set. Register 0x0b8 (apb_direct_dpcd_access) must be set to '1' to override DPCD values. Same as Override TRAINING_LANE0_SET.
0x0B0	RW	Override TRAINING_LANE2_SET. This register can be used to override TRAINING_LANE2_SET in the DPCD register set. Register 0x0b8 (apb_direct_dpcd_access) must be set to '1' to override DPCD values. Same as Override TRAINING_LANE0_SET.
0x0B4	RW	Override TRAINING_LANE3_SET. This register can be used to override TRAINING_LANE3_SET in the DPCD register set. Register 0x0b8 (apb_direct_dpcd_access) must be set to '1' to override DPCD values. Same as Override TRAINING_LANE0_SET.
0x0B8 *	RW	Override DPCD Control Register. Setting this register to 0x1 enables AXI/APB write access to DPCD capability structure.
0x0BC	RW	Override DPCD DOWNSPREAD control field. Register 0x0B8 must be set to '1' to override DPCD values. • 0 - MAX_DOWNSPREAD Override
0x0C0	RW	Override DPCD LINK_QUAL_LANE0_SET field for DPCD1.2 version only. Register 0x0B8 must be set to '1' to override DPCD values. • 2:0 - LINK_QUAL_LANE0_SET override
0x0C4	RW	Override DPCD LINK_QUAL_LANE1_SET field for DPCD1.2 version only. Register 0x0B8 must be set to 1 to override DPCD values. • 2:0 - LINK_QUAL_LANE1_SET override
0x0C8	RW	Override DPCD LINK_QUAL_LANE2_SET field for DPCD1.2 version only. Register 0x0B8 must be set to '1' to override DPCD values. • 2:0 - LINK_QUAL_LANE2_SET override
0x0CC	RW	Override DPCD LINK_QUAL_LANE3_SET field for DPCD1.2 version only. Register 0x0B8 must be set to '1' to override DPCD values. • 2:0 - LINK_QUAL_LANE3_SET override

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x0E0	RW	GUID word 0. Allows the user to setup GUID if required from host interface. Valid for DPCD1.2 version only. • [31:0] - Lower 4 bytes of GUID DPCD field
0x0E4	RW	GUID word 1. Allows the user to setup GUID if required from host interface. Valid for DPCD1.2 version only. • [31:0] - Bytes 4 to 7 of GUID DPCD field
0x0E8	RW	GUID word 2. Allows the user to setup GUID if required from host interface. Valid for DPCD1.2 version only. • [31:0] - Bytes 8 to 11 of GUID DPCD field
0x0EC	RW	GUID word 3. Allows the user to setup GUID if required from host interface. Valid for DPCD1.2 version only. • [31:0] - Bytes 12 to 15 of GUID DPCD field
0x0F0	RW	GUID Override. [0]: When set to 0x1, the GUID field of the DPCD reflects the data written in GUID Words 0 to 3. Valid for DPCD1.2 version only. When this register is set to 0x1, GUID field of DPCD becomes read only and source-aux writes are NACK-ed.
Core ID		
0x0F8	RO	VERSION Register. For displayport_v3_2, VERSION REGISTER will be 32'h03_02_0_0_00. • 31:24 - Core major version • 23:16 - Core minor version • 15:12 - Core version revision • 11:8 - Core Patch details • 7:0 - Internal revision
0x0FC	RO	CORE_ID. Returns the unique identification code of the core and the current revision level. • 31:24 - DisplayPort protocol major version • 23:16 - DisplayPort protocol minor version • 15:8 - DisplayPort protocol revision • 7:0 - Core mode of operation ◦ 0x00: Transmit ◦ 0x01: Receive Depending on the protocol and core used, the CORE_ID values are as follows: • DisplayPort v1.1a, Receive core: 32'h01_01_0a_01 • DisplayPort v1.2, Receive core: 32'h01_02_00_0
0x09C	RW	CFG_LINK_RATE. Advanced option to program required Link Rate that reflects in DPCD capabilities.
0x0A0	RW	CFG_LANE_COUNT. Advanced option to program required Lane Count that reflects in DPCD capabilities.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x110	RO	USER_FIFO_OVERFLOW. This status bit indicates an overflow of the user data FIFO of pixel data. This event may occur if the input pixel clock is not fast enough to support the current DisplayPort link width and link speed. • [0] - FIFO_OVERFLOW_FLAG: A '1' indicates that the internal FIFO has detected an overflow condition. This bit clears upon read.
0x114	RO	USER_VSYNC_STATE. Provides a mechanism for the host processor to monitor the state of the video data path. This bit is set when vsync is asserted. • [0] - state of the vertical sync pulse.
PHY Configuration and Status		
0x200	RW	PHY_RESET. Controls the reset to the PHY section of the DisplayPort receiver core. At power up, this register has a value of 0x3. • 1:0 - When set a value of 0x3, the receiver PHY will be held in reset. This value must be set to a value of 0 before the receiver core will function properly.
0x208	RO	PHY_STATUS. Provides status for the receiver core PHY. • 1:0 - Reset done for lanes 0 and 1 (Tile 0). • 3:2 - Reset done for lanes 2 and 3 (Tile 1). • 4 - PLL for lanes 0 and 1 locked (Tile 0). • 5 - PLL for lanes 2 and 3 locked (Tile 1). • 6 - FPGA fabric clock PLL locked. • 7 - Receiver Clock locked. • 9:8 - PRBS error, lanes 0 and 1. • 11:10 - PRBS error, lanes 2 and 3. • 13:12 - RX voltage low, lanes 0 and 1. • 15:14 - RX voltage low, lanes 2 and 3. • 16 - Lane alignment, lane 0. • 17- Lane alignment, lane 1. • 18- Lane alignment, lane 2. • 19- Lane alignment, lane 3. • 20 - Symbol lock, lane 0. • 21- Symbol lock, lane 1. • 22- Symbol lock, lane 2. • 23- Symbol lock, lane 3. • 25:24 - RX buffer status, lane 0. • 27:26 - RX buffer status, lane 1. • 29:28 - RX buffer status, lane 2. • 31:30 - RX buffer status, lane 3.
0x20C	RW	RX_PHY_ELEC_RESET_ENABLE . This register controls the reset of the PHY when electrical idle is detected. The electrical idle condition happens when the link is not connected. This logic is enabled for Spartan-6 devices only. • [0] - RX_PHY_ELEC_RESET_ENABLE: Set to '1', to reset the PHY module when electrical idle condition is detected.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x210	RW	RX_PHY_POWER_DOWN. These bits allow the receiver core to conditionally power down specific lanes of the PHY if supported for a particular technology implementation. These bits should be written only after the training process has been completed and the link is stable. • [3] - LANE_3_POWER_DOWN: Set o a '1' to power down the PHY for lane 3. • [2] - LANE_2_POWER_DOWN: Set to a '1' to power down the PHY for lane 2. • [1] - LANE_1_POWER_DOWN: Set to a '1' to power down the PHY for lane 1. • [0] - LANE_0_POWER_DOWN: Set to a '1' to power down the PHY for lane 0.
0x214	RW	MIN_VOLTAGE_SWING. Some DisplayPort implementations require the transmitter to set a minimum voltage swing during training before the link can be reliably established. This register is used to set a minimum value which must be met in the TRAINING_LANEX_SET DPCD registers. The internal training logic will force training to fail until this value is met. • [1:0] - The minimum voltage swing setting matches the values defined in the DisplayPort specification for the TRAINING_LANEX_SET register.
Displayport Audio		
12'h300	RW	RX_AUDIO_CONTROL. This register enables audio stream packets in main link. • [0] - Audio Enable
12'h304	RO	RX_AUDIO_INFO_DATA [31:0] Word formatted as per CEA 861-C Info Frame. Total of eight words should be read. • 1st word: [7:0] = HB0 [15:8] = HB1 [23:16] = HB2 [31:24] = HB3 • 2nd word - DB3,DB2,DB1,DB0 . . • 8th word -DB27,DB26,DB25,DB24 The data bytes DB1...DBN of CEA Info frame are mapped as DB0-DBN-1.Info frame data is copied into these registers (read only).
12'h324	RO	RX_AUDIO_MAUD. M value of audio stream as decoded from Audio time stamp packet by the sink (read only). • [31:24] - Reserved • [23:0] - MAUD
12'h328	RO	RX_AUDIO_NAUD. N value of audio stream as decoded from Audio time stamp packet by the sink (read only). • [31:24] - Reserved • [23:0] - NAUD

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
12'h32C	RO	RX_AUDIO_STATUS. • [9] - Extension Packet Received. Resets automatically after all words (9) are read. Blocks new packet until host reads the data. • [8:3] - Reserved. • [2:1] - RS Decoder Error Counter. Used for debugging purpose. • [0] - Info Packet Received. Resets automatically after all info words (eight) are read. Blocks new packet until host reads the data.
12'h330- 12'h350	RO	RX_AUDIO_EXT_DATA • [31:0] - Word formatted as per extension packet described in protocol specification. Packet length is fixed to 32 bytes in Sink controller. User should convey this information to Source using the vendor fields and ensure proper packet size transmission is done by the Source controller. Total of nine words should be read. • 1st word - [7:0] = HB0 [15:8] = HB1 [23:16] = HB2 [31:24] = HB3 • 2nd word - DB3,DB2,DB1,DB0 . . • 9th word -DB31,DB30,DB29,DB28 Extension packet data is copied into these registers (read only). This is a key-hole memory. So, nine reads from this address space is required.
DPCD Configuration Space: Refer to the DisplayPort 1.1a Specification for detailed descriptions of these registers.		
0x400	RO	DPCD_LINK_BW_SET. Link bandwidth setting. • 7:0 - Set to 0x0A when the link is configured for 2.8 Gbps or 0x06 when configured for 1.62 Gbps.
0x404	RO	DPCD_LANE_COUNT_SET. Number of lanes enabled by the transmitter. • 4:0 - Contains the number of lanes that are currently enabled by the attached transmitter. Valid values fall in the range of 1-4.
0x408	RO	DPCD_ENHANCED_FRAME_EN. Indicates that the transmitter has enabled the enhanced framing symbol mode. • 0 - Set to '1' when enhanced framing mode is enabled.
0x40C	RO	DPCD_TRAINING_PATTERN_SET. Current value of the training pattern registers. • 1:0 - TRAINING_PATTERN_SET: Set the link training pattern according to the two bit code: ◦ 00 = Training not in progress ◦ 01 = Training pattern 1 ◦ 10 = Training pattern 2 ◦ 11 = RESERVED

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x410	RO	DPCD_LINK_QUALITY_PATTERN_SET. Current value of the link quality pattern field of the DPCD training pattern register. • 1:0 - transmitter is sending the link quality pattern: ◦ 00 = Link quality test pattern not transmitted ◦ 01 = D10.2 test pattern (unscrambled) transmitted ◦ 10 = Symbol Error Rate measurement pattern ◦ 11 = PRBS7 transmitted
0x414	RO	DPCD_RECOVERED_CLOCK_OUT_EN. Value of the output clock enable field of the DPCD training pattern register. • 0 - Set to '1' to output the recovered receiver clock on the test port.
0x418	RO	DPCD_SCRAMBLING_DISABLE. Value of the scrambling disable field of the DPCD training pattern register. • 0 - Set to '1' when the transmitter has disabled the scrambler and transmits all symbols.
0x41C	RO	DPCD_SYMBOL_ERROR_COUNT_SELECT. Current value of the symbol error count select field of the DPCD training pattern register. • 1:0 - SYMBOL_ERROR_COUNT_SEL: ◦ 00 = Disparity error and illegal symbol error ◦ 01 = Disparity error ◦ 10 = Illegal symbol error ◦ 11 = Reserved
0x420	RO	DPCD_TRAINING_LANE_0_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. • 1:0 - VOLTAGE_SWING_SET ◦ 00 = Training Pattern 1 with voltage swing level 0 ◦ 01 = Training Pattern 1 with voltage swing level 1 ◦ 10 = Training Pattern 1 with voltage swing level 2 ◦ 11 = Training Pattern 1 with voltage swing level 3 • 2 - MAX_SWING_REACHED: Set to '1' when the maximum driven current setting is reached. • 4:3 - PRE-EMPHASIS_SET ◦ 00 = Training Pattern 2 without pre-emphasis ◦ 01 = Training Pattern 2 with pre-emphasis level 1 ◦ 10 = Training Pattern 2 with pre-emphasis level 2 ◦ 11 = Training Pattern 2 with pre-emphasis level 3 • 5 - MAX_PRE-EMPHASIS_REACHED: Set to '1' when the maximum pre-emphasis setting is reached.
0x424	RO	DPCD_TRAINING_LANE_1_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. The fields of this register are identical to DPCD_TRAINING_LANE_0_SET.
0x428	RO	DPCD_TRAINING_LANE_2_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. The fields of this register are identical to DPCD_TRAINING_LANE_0_SET.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x42C	RO	DPCD_TRAINING_LANE_3_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. The fields of this register are identical to DPCD_TRAINING_LANE_0_SET.
0x430	RO	DPCD_DOWNSPREAD_CONTROL. The transmitter uses this bit to inform the receiver core that downspreading has been enabled. • 0 - SPREAD_AMP: Set to '1' for 0.5% spreading or '0' for none.
0x434	RO	DPCD_MAIN_LINK_CHANNEL_CODING_SET. 8B/10B encoding can be disabled by the transmitter through this register bit. • 0 - Set to '0' to disable 8B/10B channel coding. The default is '1'.

0x438	RO	DPCD_SET_POWER_STATE. Power state requested by the source core. On reset, power state is set to power down mode.
		• 1:0 - requested power state
		  ◦ 00 = Reserved
		  ◦ 01 = state D0, normal operation
		  ◦ 10 = state D3, power down mode
		  ◦ 11 = Reserved

0x43C	RO	DPCD_LANE01_STATUS. Value of the lane 0 and lane 1 training status registers. • 6 - LANE_1_SYMBOL_LOCKED • 5 - LANE_1_CHANNEL_EQ_DONE • 4 - LANE_1_CLOCK_RECOVERY_DONE • 2 - LANE_0_SYMBOL_LOCKED • 1 - LANE_0_CHANNEL_EQ_DONE • 0 - LANE_0_CLOCK_RECOVERY_DONE
0x440	RO	DPCD_LANE23_STATUS. Value of the lane 2 and lane 3 training status registers. • 6 - LANE_3_SYMBOL_LOCKED • 5 - LANE_3_CHANNEL_EQ_DONE • 4 - LANE_3_CLOCK_RECOVERY_DONE • 2 - LANE_2_SYMBOL_LOCKED • 1 - LANE_2_CHANNEL_EQ_DONE • 0 - LANE_2_CLOCK_RECOVERY_DONE
0x444	RO	SOURCE_OUI_VALUE. Value of the Organizationally Unique Identifier (OUI) as written by the transmitter via the DPCD register AUX transaction. • 23:0 - Contains the value of the OUI set by the transmitter. This value may be used by the host policy maker to enable special functions across the link.
0x448	RC/RO	SYM_ERR_CNT01. Reports symbol error counter of lanes 0 and 1. • [32] = Lane 1 error count valid. This bit get cleared when this registered is read. • [30:16] = Lane 1 error count. • [15] = Lane 0 error count valid. This bit get cleared when this registered is read. • [14:0] = Lane 0 error count.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x44C	RC/RO	SYM_ERR_CNT23. Reports symbol error counter of lanes 2 and 3. • [32] = Lane 3 error count valid. This bit get cleared when this registered is read. • [30:16] = Lane 3 error count. • [15] = Lane 2 error count valid. This bit get cleared when this registered is read. • [14:0] = Lane 2 error count.
MSA Values		
0x500	RO	MSA_HRES. The horizontal resolution detected in the Main Stream Attributes. • 15:0 - Represents the number of pixels in a line of video.
0x504	RO	MSA_HSPOL. Horizontal sync polarity. • 0 - Indicates the polarity of the horizontal sync as requested by the transmitter.
0x508	RO	MSA_HSWIDTH. Specifies the width of the horizontal sync pulse. • 14:0 - Specifies the width of the horizontal sync in terms of the recovered video clock.
0x50C	RO	MSA_HSTART. This main stream attribute is the number of clock cycles between the leading edge of the horizontal sync and the first cycle of active data. • 15:0 - Number of blanking cycles before active data.
0x510	RO	MSA_HTOTAL. Tells the receiver core how many video clock cycles will occur between leading edges of the horizontal sync pulse. • 15:0 - Total number of video clocks in a line of data.
0x514	RO	MSA_VHEIGHT. Total number of active video lines in a frame of video. • 15:0 - The vertical resolution of the received video.
0x518	RO	MSA_VSPOL. Specifies the vertical sync polarity requested by the transmitter. • 0 - A value of '1' in this register indicates an active-High vertical sync, and a '0' indicates an active-Low vertical sync.
0x51C	RO	MSA_VSWIDTH. The transmitter uses this value to specify the width of the vertical sync pulse in lines. • 14:0 - Specifies the number of lines between the leading and trailing edges of the vertical sync pulse.
0x520	RO	MSA_VSTART. This main stream attribute specifies the number of lines between the leading edge of the vertical sync pulse and the first line of active data. • 15:0 - Number of blanking lines before the start of active data.
0x524	RO	MSA_VTOTAL. Total number of lines between sequential leading edges of the vertical sync pulse. • 15:0 - The total number of lines per video frame is contained in this value.

Table 2-8: **DisplayPort Sink Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x528	RO	MSA_MISC0. Contains the value of the MISC0 attribute data. • 7:5 - COLOR_DEPTH: Number of bits per color/component. • 4 - YCbCR_COLOR: Set to 1 (ITU-R BT709-5) or 0 (ITU-R BT601-5). • 3 - DYNAMIC_RANGE: Set to 1 (CEA range) or 0 (VESA range). • 2:1 - COMPONENT_FORMAT: ∘ 00 = RGB ∘ 01 = YCbCr 4:2:2 ∘ 10 = YCbCr 4:4:4 ∘ 11 = Reserved 0 - CLOCK_MODE: ∘ 0 = Synchronous clock mode ∘ 1 = Asynchronous clock mode
0x52C	RO	MSA_MISC1. Contains the value of the MISC1 attribute data. • 7:3 - RESERVED: These bits are always set to 0. • 2:1 - STEREO_VIDEO: Used only when stereo video sources are being transmitted. See the *DisplayPort Specification v1.1a* section 2.24 for more information. • 0 - INTERLACED_EVEN: A '1' indicates that the number of lines per frame is an even number.
0x530	RO	MSA_MVID. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_NVID registers. • 23:0 - MVID: Value of the clock recovery M value.
0x534	RO	MSA_NVID. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_MVID registers. • 23:0 - NVID: Value of the clock recovery N value.
0x538	RO	MSA_VBID. The most recently received VB-ID value is contained in this register. • 7:0 - VBID: See Table 2-3 (p44) in the *DisplayPort Specification v1.1a* for more information.
Vendor Specific DPCD		
0xE00-0xEFC	RW	SOURCE_DEVICE_SPECIFIC_FIELD. User access to Source specific field of DPCD address space. AXI accesses are all word-based (32 bits). • 0xE00 - 0xE02 : Read Only (IEEE OUI Value Programmed by Source) • 0xE03 - 0xEFF : Write/Read
0xF00-0xFFC	RW	SINK_DEVICE_SPECIFIC_FIELD. User access to Sink specific field of DPCD address space. AXI accesses are all word-based (32 bits). • 0xF00 - 0xF02 : Read Only (IEEE OUI Value from GUI) • 0xF03 - 0xFFF : Write/Read

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

Source Overview

Main Link Setup and Management

This section is intended to elaborate on and act as a companion to the link training procedure, described in section 3.5.1.3 of the *DisplayPort Standard v1.1a*.

For the user's convenience, the DisplayPort Source core comes with two example controller designs. The first is a simple RTL-based state machine that may be used to quickly demonstrate the proper startup procedure. This is provided because simulating the full Policy Maker example design requires many hours of simulation to complete. The RTL-based state machine should only be used for simulation, as a and for establishing a quick link with the Xilinx Sink core. This controller is not expected to interoperate with other standard products.

The second controller is a netlist-based, fully-functional Policy Maker. As defined by the *VESA DisplayPort Standard* specification [Ref 1], the Link Policy Maker manages the link and is responsible for keeping the link synchronized. This includes link discovery, initialization, and maintenance. The Stream Policy Maker manages the transport initialization and maintenance of the isochronous stream by controlling sequences of actions by the underlying hardware. These functions are supplied through this netlist, which can be used in many standard designs without tuning. While the netlist is ideal for hardware applications, long modeling times means it is not optimal for simulation.

For users requiring more capability and tuning, the full Link Policy Maker example design is available as full C source code to the purchasers of the DisplayPort core. The Policy Maker sets up and maintains the link with varying levels of interaction by the user. For users who decide to use the provided software, this section may be treated as reference. For more information on acquiring the Policy Maker source code, see http://www.xilinx.com/products/ipcenter/EF-DI-DISPLAYPORT.htm.

Regardless of whether the provided Policy Maker is used, Xilinx advises all users of the source core to use a MicroBlaze™ processor or similar embedded processor to properly initialize and maintain the link. The tasks encompassed in the Link and Stream Policy Makers are likely too complicated to be efficiently managed by a hardware-based state machine.

Link Training

The link training commands are passed from the DPCD register block to the link training function. When set into the link training mode, the functional data path is blocked and the link training controller issues the specified pattern. Care must be taken to place the Sink device in the proper link training mode before the source state machine enters a training state. Otherwise, unpredictable results may occur.

Figure 3-1 shows the flow diagram for link training.

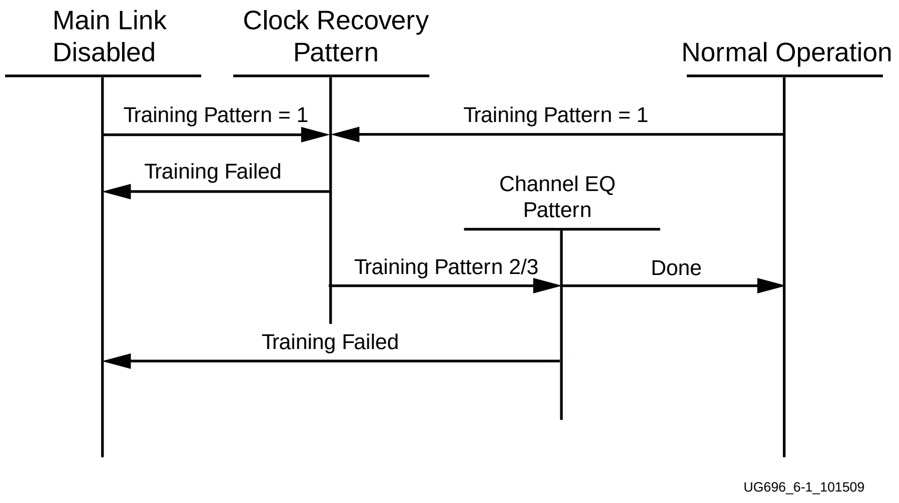


Figure 3-1: **Link Training States**

Source Core Setup and Initialization

The following text contains the procedural tasks required to achieve link communication. See the description of the DPCD in the *DisplayPort Standard v1.1a*.

Source Core Setup

1. Place the PHY into reset.
    - PHY_RESET = 0x01
2. Disable the transmitter.
    - TRANSMITTER_ENABLE = 0x00
3. Set the clock divider.
    - AUX_CLOCK_DIVIDER = (see register description for proper value)
4. Set DisplayPort clock speed.

- PHY_CLOCK_SELECT = desired link speed

5. Bring the PHY out of reset.
   - PHY_RESET = 0x00
6. Wait for the PHY to be ready.
   - (PHY_STATUS & 0x3F) == 0x3F
7. Enable the transmitter.
   - TRANSMITTER_ENABLE = 0x01
8. (Optional) Turn on the interrupt mask for HPD.
   - INTERRUPT_MASK = 0x00

Note: At this point, the source core is initialized and ready to use. The link policy maker should be monitoring the status of HPD and taking appropriate action for connect / disconnect events or HPD interrupt pulses.

Upon HPD Assertion

1. Read the DPCD capabilities fields out of the sink device (0x00000 - 0x0000B) via the AUX channel.
2. Determine values for lane count, link speed, enhanced framing mode, downspread control and main link channel code based on each link partners' capability and needs.
3. Write the configuration parameters to the link configuration field (0x00100 - 0x00101) of the DPCD via the AUX channel.

Note: Some sink devices' DPCD capability fields are unreliable. Many source devices start with the maximum transmitter capabilities and scale back as necessary to find a configuration the sink device can handle. This could be an advisable strategy instead of relying on DPCD values.

4. Equivalently, write the appropriate values to the Source core's local configuration space.
   a. LANE_COUNT_SET
   b. LINK_BW_SET
   c. ENHANCED_FRAME_EN
   d. PHY_CLOCK_SELECT

Training Pattern 1 Procedure (Clock Recovery)

1. Turn off scrambling and set training pattern 1 in the source via direct register writes.
   - SCRAMBLING_DISABLE = 0x01
   - TRAINING_PATTERN_SET = 0x01
2. Turn off scrambling and set training pattern 1 in the sink DPCD (0x00102 - 0x00106) via the AUX channel.

3. Wait 100 us before reading status registers for all active lanes (0x00202 - 0x00203) via the AUX channel.
4. If clock recovery failed, check for voltage swing or preemphasis level increase requests (0x00206 -0x00207) and react accordingly.
    - Run this loop up to five times. If after five iterations this has not succeeded, reduce link speed if at high speed and try again. If already at low speed, training fails.

Training Pattern 2 Procedure (Symbol Recovery, Interlane Alignment)

1. Turn off scrambling and set training pattern 2 in the source via direct register writes.
    - SCRAMBLING_DISABLE = 0x01
    - TRAINING_PATTERN_SET = 0x02
2. Turn off scrambling and set training pattern 2 in the sink DPCD (0x00102 - 0x00106) via the AUX channel.
3. Wait 400 us then read status registers for all active lanes (0x00202 - 0x00203) via the AUX channel.
4. Check the channel equalization, symbol lock, and interlane alignment status bits for all active lanes (0x00204) via the AUX channel.
5. If any of these bits are not set, check for voltage swing or preemphasis level increase requests (0x00206 -0x00207) and react accordingly.
6. Run this loop up to five times. If after five iterations this has not succeeded, reduce link speed if at high speed and Return to the instructions for Training Pattern 1. If already at low speed, training fails.
7. Signal the end of training by enabling scrambling and setting training pattern to 0x00 in the sink device (0x00102) via the AUX channel.
8. On the source side, re-enable scrambling and turn of training.
    - TRAINING_PATTERN_SET = 0x00
    - SCRAMBLING_DISABLE = 0x00

At this point, training has completed.

Note: Training pattern 3 replaces training pattern 2 for 5.4 G link rate devices. See the DisplayPort v1.2 specification for details.

Enabling Main Link Video

Main link video should not be enabled until a proper video source has been provided to the source core. Typically the source device will want to read the EDID from the attached sink device to determine its capabilities, most importantly its preferred resolution and other resolutions that it supports should the preferred mode not be available. Once a resolution has been determined, set the Main Stream Attributes in the source core (0x180 - 0x1B0). Enable the main stream (0x084) only when a reliable video source is available.

Note: The scrambler/de-scrambler must be reset after enabling the main link video. Before starting to transmit video, the source must initialize the scrambler and the link partner's de-scrambler. This is done by forcing a scrambler reset (0x0c0) before the main link is enabled.

Accessing the Link Partner

The DisplayPort core is configured through the AXI4-Lite host interface. The host processor interface uses the DisplayPort AUX Channel to read the register space of the attached sink device and determines the capabilities of the link. Accessing DPCD and EDID information from the Sink is done by writing and reading from register space 0x100 through 0x144. (For information on the DPCD register space, refer to the VESA DisplayPort Standard.)

Before any AUX channel operation may be completed, the user must first set the proper clock divide value in 0x10C. This must be done only one time after a reset. The value held in this register should be equal to the frequency of `s_axi_aclk`. So, if `s_axi_aclk` runs at 135 MHz, the value of this register should be 135 ('h87). This register is required to apply a proper divide function for the AUX channel sample clock, which must operate at 1 MHz.

The act of writing to the AUX_COMMAND initiates the AUX event. Once an AUX request transaction is started, the host should not write to any of the control registers until the REPLY_RECEIVED bit is set to '1,' indicating that the sink has returned a response.

AUX Write Transaction

An AUX write transaction is initiated by setting up the AUX_ADDRESS, and writing the data to the AUX_WRITE_FIFO followed by a write to the AUX_COMMAND register with the code 0x08. Writing the command register begins the AUX channel transaction. The host should wait until either a reply received event or reply timeout event is detected. These events are detected by reading INTERRUPT_STATUS registers (either in ISR or polling mode).

When the reply is detected, the host should read the AUX_REPLY_CODE register and look for the code 0x00 indicating that the AUX channel has successfully acknowledged the transaction.

Figure 3-2 shows a flow of an AUX write transaction.

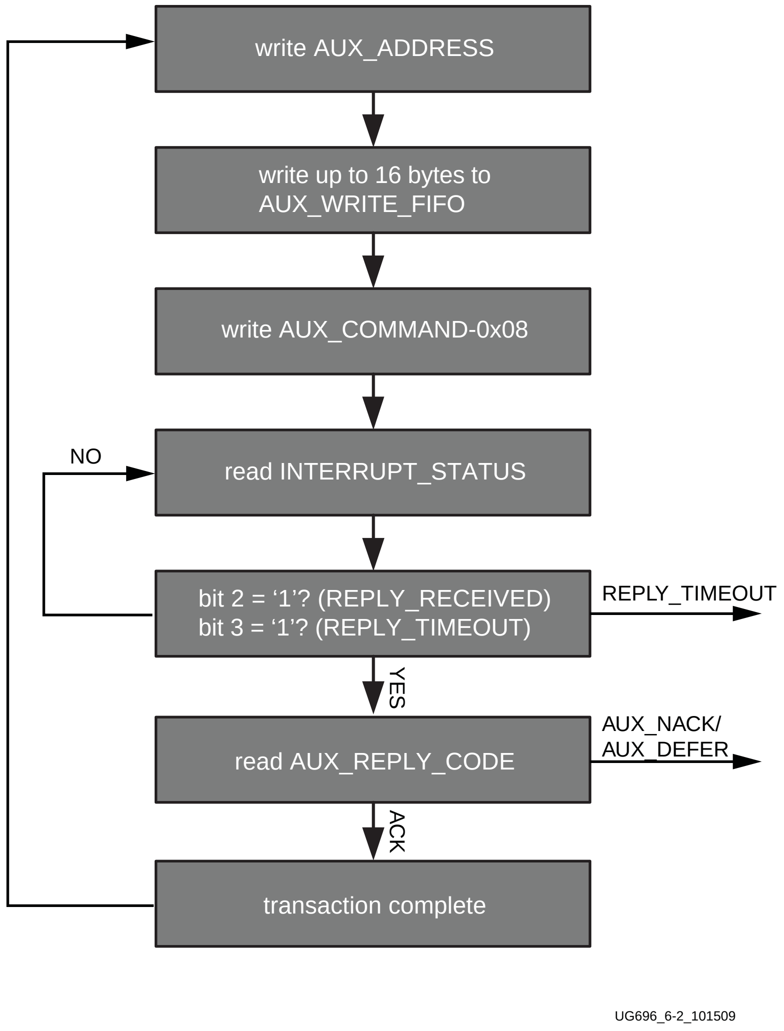


Figure 3-2: **AUX Write Transaction**

AUX Read Transaction

The AUX read transaction is prepared by writing the transaction address to the AUX_ADDRESS register. Once set, the command and the number of bytes to read are written to the AUX_COMMAND register. After initiating the transfer, the host should wait for an interrupt or poll the INTERRUPT_STATUS register to determine when a reply is received.

When the REPLY_RECEIVED signal is detected, the host may then read the requested data bytes from the AUX_REPLY_DATA register. This register provides a single address interface to a byte FIFO which is 16 elements deep. Reading from this register automatically advances the internal read pointers for the next access.

Figure 3-3 shows a flow of an AUX read transaction.

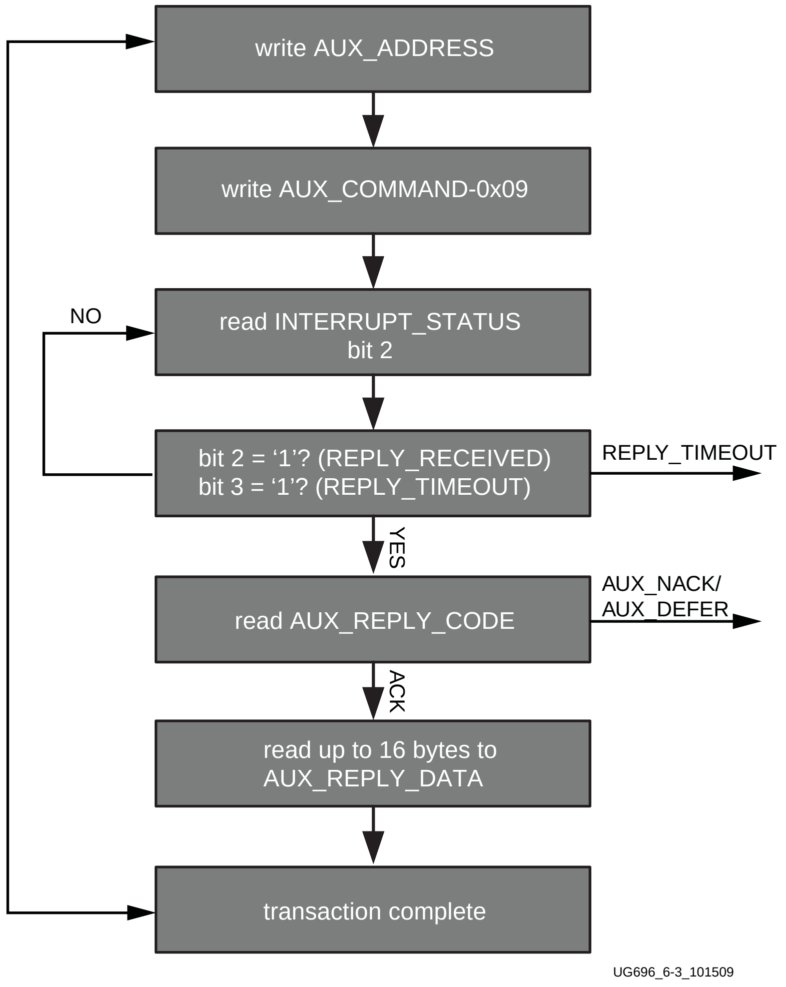


Figure 3-3: **AUX Read Transaction**

Commanded I2C Transactions

The core supports a special AUX channel command intended to make I2C over AUX transactions faster and easier to perform. In this case, the host will bypass the external I2C master/slave interface and initiate the command by directly writing to the register set.

The sequence for performing these transactions is exactly the same as a native AUX channel transaction with a change to the command written to the AUX_COMMAND register. The supported I2C commands are summarized in Table 3-1.

Table 3-1: **I2C over AUX Commands**

AUX_COMMAND[11:8]	Command
0x0	IIC Write
0x4	IIC Write MOT

Table 3-1: **I2C over AUX Commands**

AUX_COMMAND[11:8]	Command
0x1	IIC Read
0x5	IIC Read MOT
0x6	IIC Write Status with MOT
0x2	IIC Write Status

By using a combination of these commands, the host may emulate an I2C transaction.

Figure 3-4 shows the flow of commanded I2C transactions.

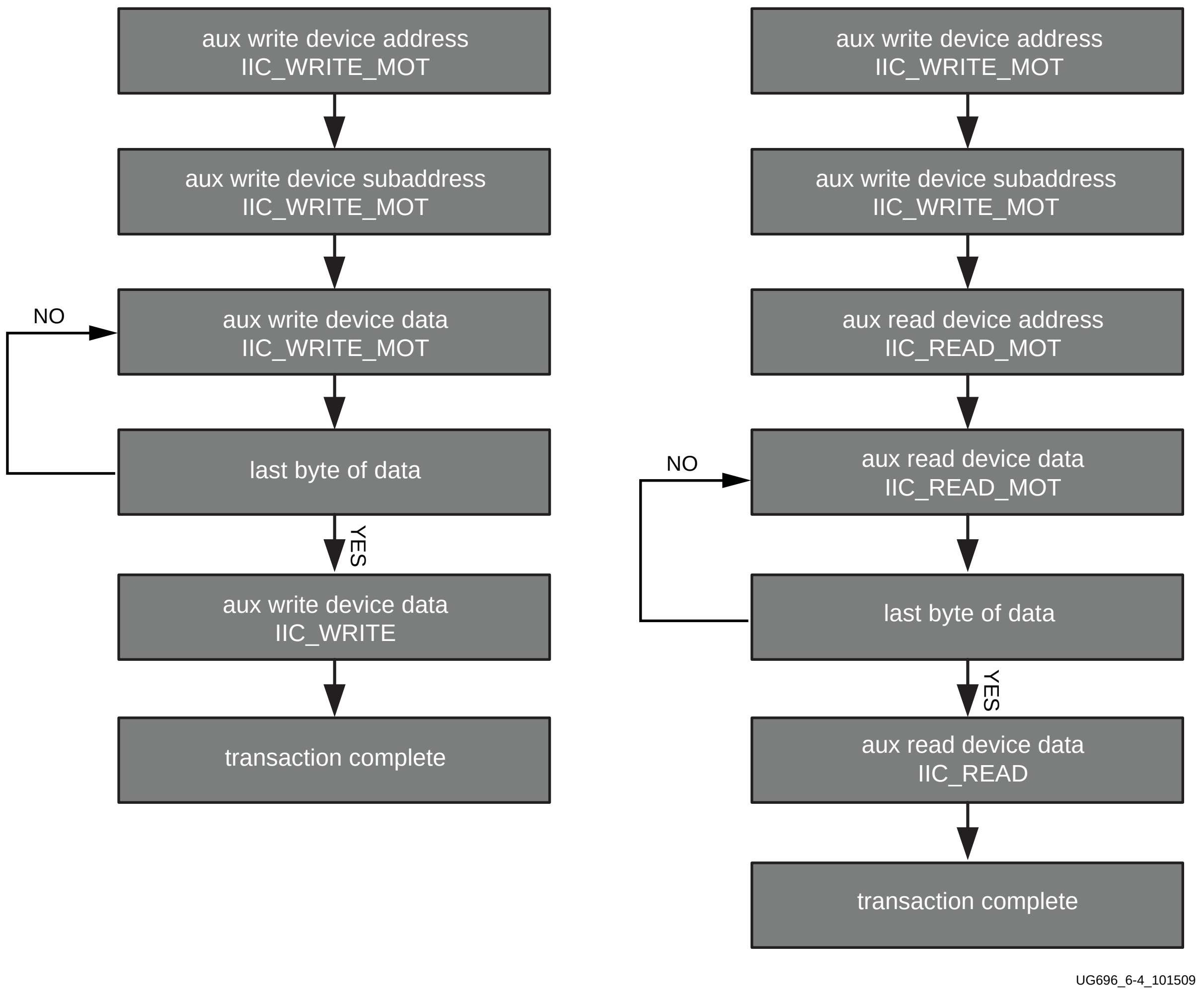


Figure 3-4: **Commanded I2C Device Transactions, Write (Left) and Read (Right)**

Since I2C transactions may be significantly slower than AUX channel transactions, the host should be prepared to receive multiple AUX_DEFER reply codes during the execution of the above state machines.

The AUX-I2C commands are as follows:

- MOT Definition:
  - Middle Of Transaction bit in the command field.
  - This controls the stop condition on the I2C slave.
  - For a transaction with MOT set to 1, the I2C bus is not STOPPED, but left to remain the previous state.
  - For a transaction with MOT set to 0, the I2C bus is forced to IDLE at the end of the current command or in special Abort cases.
- Partial ACK:
  - For I2C write transactions, the Sink core can respond with a partial ACK (ACK response followed by the number of bytes written to I2C slave).

Special AUX commands include:

- Write Address Only and Read Address Only: These commands do not have any length field transmitted over the AUX channel. The intent of these commands are to:
  - Send address and RD/WR information to I2C slave. No Data is transferred.
  - End previously active transaction, either normally or through an abort.

  The Address Only Write and Read commands are generated from the source by using bit [12] of the command register with command as I2C WRITE/READ.
- Write Status: This command does not have any length information. The intent of the command is to identify the number of bytes of data that have been written to an I2C slave when a Partial ACK or Defer response is received by the source on a AUX-I2C write.

  The Write status command is generated from the source by using bit [12] of the command register with command as I2C WRITE STATUS.

Generation AUX transactions are described in Table 3-2.

Table 3-2: **Generation AUX Transactions**

Transaction	AUX Transaction	I2C Transaction	Usage	Sequence
Write Address only with MOT = 1	START -> CMD -> ADDRESS -> STOP	START -> DEVICE_ADDR -> WR -> ACK/NACK	Setup I2C slave for Write to address defined	1. Write AUX Address register(0x108) with device address. 2. Issue command to transmit transaction by writing into AUX command register (0x100). Bit [12] must be set to 1.
Read Address only with MOT = 1	START -> CMD -> ADDRESS -> STOP	START -> DEVICE_ADDR -> RD -> ACK/NACK	Setup I2C slave for Read to address defined.	1. Write AUX Address register with device address. 2. Issue command to transmit transaction by writing into AUX command register. Bit [12] must be set to 1.
Write / Read Address only with MOT = 0	START -> ADDRESS -> STOP	STOP	To stop the I2C slave, used as Abort or normal stop.	1. Write AUX Address register (0x108) with device address. 2. Issue command to transmit transaction by writing into AUX command register (0x100). Bit [12] must be set to 1.
Write with MOT = 1	START -> CMD -> ADDRESS -> LENGTH -> D0 to DN -> STOP	I2C bus is IDLE or New device address START -> START/RS -> DEVICE_ADDR -> WR -> ACK/NACK -> DATA0 -> ACK/NACK to DATAN -> ACK/NACK I2C bus is in Write state and the same device address DATA0 -> ACK/NACK to DATAN -> ACK/NACK	Setup I2C slave write data.	1. Write AUX Address register(0x108) with device address. 2. Write the data to be transmitted into AUX write FIFO register (0x104). 3. Issue write command and data length to transmit transaction by writing into AUX command register (0x100). Bits [3:0] represent length field.

Table 3-2: **Generation AUX Transactions *(Cont'd)***

Transaction	AUX Transaction	I2C Transaction	Usage	Sequence
Write with MOT = 0	START -> CMD -> ADDRESS -> LENGTH -> D0 to DN -> STOP	I2C bus is IDLE or Different I2C device address START -> START/RS -> DEVICE_ADDR -> WR -> ACK/NACK -> DATA0 -> ACK/NACK to DATAN -> ACK/NACK -> STOP I2C bus is in Write state and the same I2C device address DATA0 -> ACK/NACK to DATAN -> ACK/NACK -> STOP	Setup I2C slave write data and stop the I2C bus after the current transaction.	1. Write AUX Address register (0x108) with device address. 2. Write the data to be transmitted into AUX write FIFO register (0x104). 3. Issue write command and data length to transmit transaction by writing into AUX command register (0x100). Bits [3:0] represent length field.
Read with MOT = 1	START -> CMD -> ADDRESS -> LENGTH -> STOP	I2C bus is IDLE or Different I2C device address START -> START/RS -> DEVICE_ADDR -> RD -> ACK/NACK -> DATA0 -> ACK/NACK to DATAN -> ACK/NACK I2C bus is in Write state and the same I2C device address DATA0 -> ACK/NACK to DATAN -> ACK/NACK	Setup I2C slave read data.	1. Write AUX Address register (0x108) with device address. 2. Issue read command and data length to transmit transaction by writing into AUX command register (0x100). Bits [3:0] represent the length field.

Table 3-2: **Generation AUX Transactions *(Cont'd)***

Transaction	AUX Transaction	I2C Transaction	Usage	Sequence
Read with MOT = 0	START -> CMD -> ADDRESS -> LENGTH -> D0 to DN -> STOP	I2C bus is IDLE or Different I2C device address START -> START/RS -> DEVICE_ADDR -> RD -> ACK/NACK -> DATA0 -> ACK/NACK to DATAN -> ACK/NACK -> STOP I2C bus is in Write state and the same I2C device address DATA0 -> ACK/NACK to DATAN -> ACK/NACK -> STOP	Setup I2C slave read data and stop the I2C bus after the current transaction.	1. Write AUX Address register (0x108) with device address. 2. Issue read command and data length to transmit transaction by writing into AUX command register (0x100). Bits [3:0] represent the length field.
Write Status with MOT = 1	START -> CMD -> ADDRESS -> STOP	No transaction	Status of previous write command that was deferred or partially ACKED.	1. Write AUX Address register (0x108) with device address. 2. Issue status update command to transmit transaction by writing into AUX command register (0x100). Bit [12] must be set to 1.
Write Status with MOT = 0	START -> CMD -> ADDRESS -> STOP	Force a STOP and the end of write burst	Status of previous write command that was deferred or partially ACKED. MOT = 0 will ensure the bus returns to IDLE at the end of the burst.	1. Write AUX Address register (0x108) with device address. 2. Issue status update command to transmit transaction by writing into AUX command register (0x100). Bit [12] must be set to 1.

Handling I2C Read Defers:

- The Sink core could issue a DEFER response for a burst read to I2C. The following are the actions that can be taken by the Source core.

- Issue the same command (previously issued read, with same device address and length) and wait for response. The Sink core on completion of the read from I2C (after multiple defers) should respond with read data.
- Abort the current read using:
  - Read to a different I2C slave
  - Write command
  - Address-only Read or write with MOT = 0.

Handling I2C Write Partial ACK:

- The sink could issue a partial ACK response for a burst Write to I2C. The following are the actions that can be taken by the Source core:
  - Use the Write status command to poll the transfers happening to the I2C. On successful completion, the sink should issue an NACK response to these requests while intermediate ones will get partial ACK.
  - Issue the same command (previously issued with the same device address, length and data) and wait for response. On completion of the write to I2C (after multiple partial ACK), the Sink core should respond with an ACK.
  - Abort the current Write using:
    - Write to a different I2C slave
    - Read command
    - Address-only Read or Write with MOT = 0.

Handling I2C Write Defer:

- The Sink core could issue a Defer response for a burst write to I2C. The following are the actions that can be taken by the Source core:
  - Use the Write status command to poll the transfers happening to the I2C. On successful completion, the Sink core should issue an ACK response to these request while intermediate ones will get a partial ACK.
  - Issue the same command (previously issued with the same device address, length and data) and wait for response. The Sink core on completion of the write to I2C (after multiple Defers) should respond with an ACK.
  - Abort the current Write using:
    - Write to a different I2C slave
    - Read command
    - Address only Read or Write with MOT = 0.

Transmitter Clock Generation

The transmitter clocking architecture supports both the asynchronous and synchronous clocking modes included in the *DisplayPort Standard v1.1a*. The clocking mode is selected by way of the Stream Clock Mode register (MAIN_STREAM_MISC0 bit[0]). When set to '1', the link and stream clock are synchronous, in which case the MVid and NVid values are a constant. In synchronous clock mode, the source core uses the MVid and NVid register values programmed by the host processor via the AXI4-Lite interface.

When the Stream Clock Mode register is set to '0', asynchronous clock mode is enabled and the relationship between MVid and NVid is not fixed. In this mode, the source core will transmit a fixed value for NVid and the MVid value provided as a part of the clocking interface.

Figure 3-5 shows a block diagram of the transmitter clock generation process.

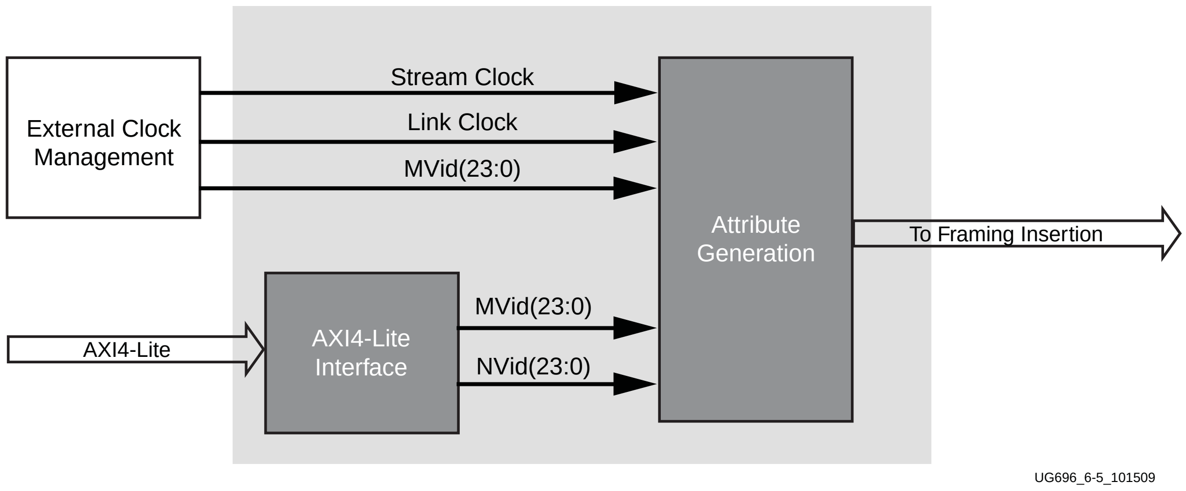


Figure 3-5: **Transmitter Clock Generation**

Hot Plug Detection

The Source device must debounce the incoming HPD signal by sampling the value at an interval greater than 250 microseconds. For a pulse width between 500 microseconds and 1 millisecond, the Sink device has requested an interrupt. The interrupt is passed to the host processor through the AXI4-Lite interface.

Should the HPD signal remain low for greater than 2 milliseconds, this indicates that the sink device has been disconnected and the link should be shut down. This condition is also passed through the AXI4-Lite interface as an interrupt. The host processor must properly determine the cause of the interrupt by reading the appropriate DPCD registers and take the appropriate action.

HPD Event Handling

HPD signaling has three use cases:

- Connection event defined as HPD_EVENT is detected, and the state of the HPD is "1".
- Disconnection event defined as HPD_EVENT is detected, and the state of the HPD is "0".
- HPD IRQ event as captured in the INTERRUPT_STATUS register bit "0".

Figure 3-6 shows the source core state and basic actions to be taken based on HPD events.

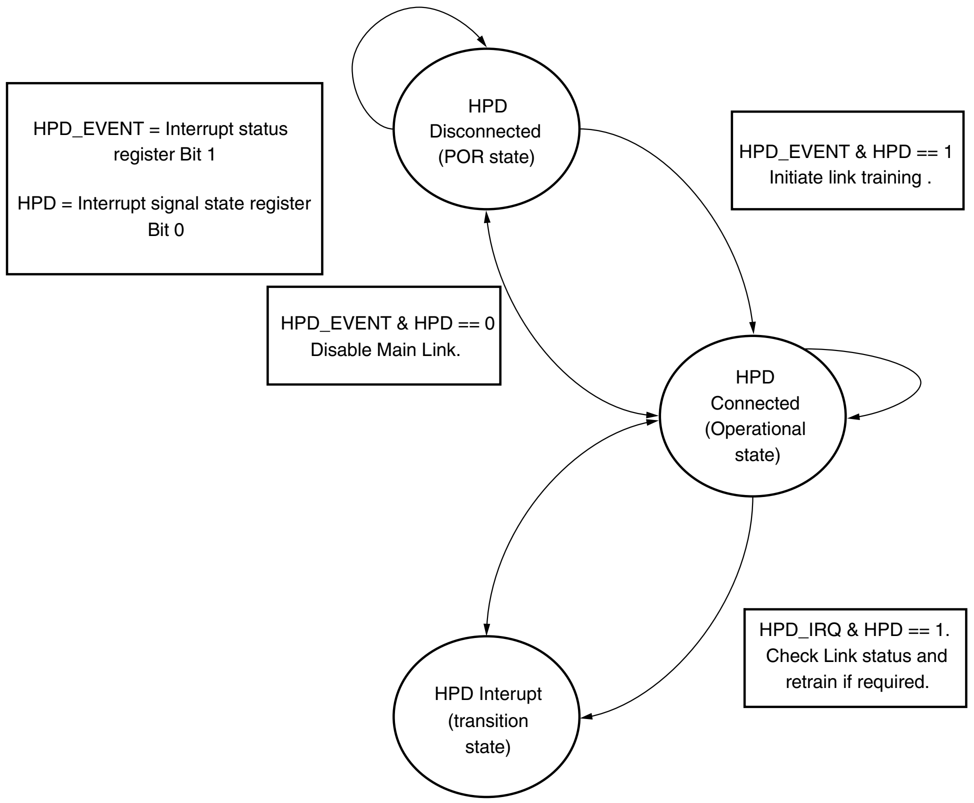

HPD_EVENT = Interrupt status register Bit 1
HPD = Interrupt signal state register Bit 0
HPD Disconnected (POR state)
HPD_EVENT & HPD == 1 Initiate link training .
HPD_EVENT & HPD == 0 Disable Main Link.
HPD Connected (Operational state)
HPD_IRQ & HPD == 1. Check Link status and retrain if required.
HPD Interupt (transition state)

Figure 3-6: **HPD Event Handling in Source Core**

Secondary Channel Operation

This section describes the procedural tasks required to achieve audio communication.

Programming SPDIF Receiver

1. Reset the SPDIF Receiver by writing 0x000A to SPDIF Soft Reset Register (Base address of SPDIF Receiver + 0x40). When there is a change in video/audio parameters, it is recommended to follow this step.

2. Enable Audio reception by writing 0x0001 to SPDIF Control Register (Base address of SPDIF Receiver + 0x44).
3. Read SPDIF - Channel Status Register (Base address of SPDIF Receiver + 0x4C) Bit [24:27] of channel status gives the sampling frequency information for 32k, 44.1k and 48k frequencies, Table 3-3.

Table 3-3: **Sampling Frequency Bits**

Bit [24:27]	Sampling Frequency
0000	44.1k
0100	48k
1100	32k

Based on the incoming audio rate, adjust the aud_clk generator to 512*fs frequency.

There are Receive FIFO Full, Receive FIFO Empty, start of block, BMC error, and Preamble error interrupts through single interrupt out signal in the SPDIF Receiver. See PG045, *LogiCORE IP SPDIF Product Guide* for details about enabling these interrupts.

Programming DisplayPort Source

1. Disable Audio by writing 0x00 to TX_AUDIO_CONTROL register. The disable bit will also flush the buffers in DisplayPort Source and set MUTE bit in VB-ID. When there is a change in video/audio parameters, it is recommended to follow this step.
2. Write Audio Info Frame (Based on your requirement. This may be optional for some systems.). Audio Info Frame consists of 8 writes. The order of write transactions are important and follow the steps mentioned in the Table 2-7.
3. Write Channel Count to TX_AUDIO_CHANNELS register (the value is actual count -1).
4. If the system is using synchronous clocking then write MAUD and NAUD values TX_AUDIO_MAUD and TX_AUDIO_NAUD registers.
5. Enable Audio by writing 0x01 to TX_AUDIO_CONTROL register. Ensure all steps of SPDIF are completed before enabling DisplayPort Audio.

Info Packet Management

The core provides an option to program a single Info packet. The packet will be transmitted to Sink once per every video frame or 8192 cycles.

To change an Info packet during transmission, follow these steps:

1. Disable Audio (Since new info packet means new audio configuration). The disable audio will also flush internal audio buffers.
2. Follow steps mentioned in Programming DisplayPort Source.

Extension Packet Management

A single packet buffer is provided for the extension packet. If the extension packet is available in the buffer, the packet is transmitted as soon as there is availability in the secondary channel. The packet length is FIXED to eight words (32 bytes).

Use the following steps to write an extended packet in the DisplayPort Source controller:

1. Write nine words (as required) into TX_AUDIO_EXT_DATA buffer.
2. Wait for EXT_PKT_TXD interrupt.
3. Write new packet (follow step 1).

Audio Clocking (Recommendation)

The system should have a clock generator (preferably programmable) to generate 512 X fs (Audio Sample Rate) clock frequency. This clock is used by SPDIF Controller to stream data using AXI4-Stream interface. The same clock (aud_clk) is used by DisplayPort Source device to calculate MAUD and NAUD when running in asynchronous clocking mode.

SPDIF Sampling clock (aud_axis_aclk) is used by controller to extract data from bi-phase stream. The requirement for this clock is that its frequency is greater than 512 X fs (audio sample frequency). Typically this clock is set to a high frequency such as 100 MHz to recover all rates starting from 32 KHz to 192 KHz.

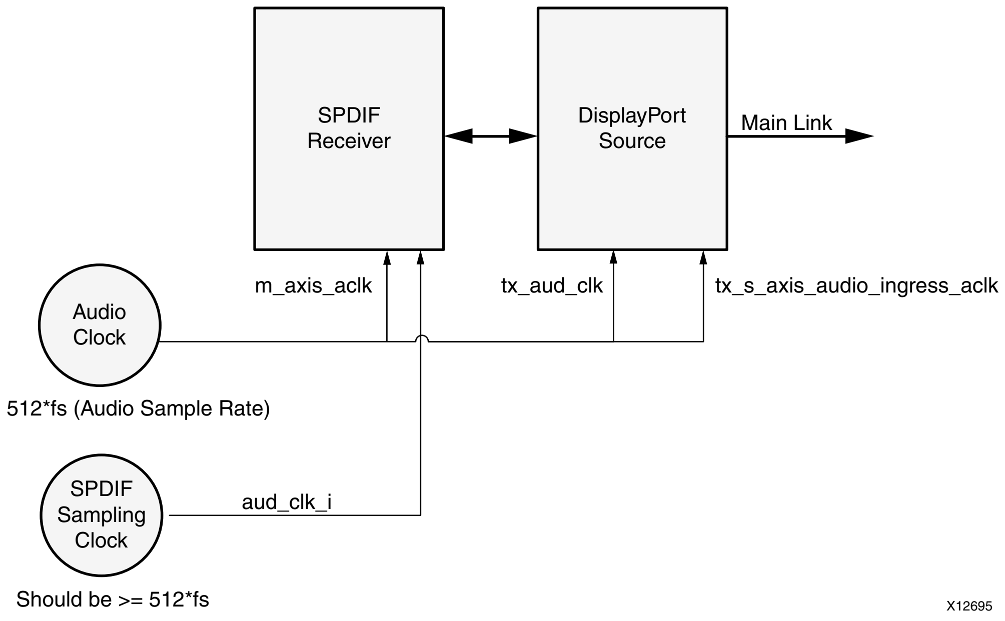


Figure 3-7: **Source: Audio Clocking**

Sink Overview

The Sink core requires a series of initialization steps before it begins receiving video. These steps include bringing up the Physical Interface (PHY) and setting the internal registers for the proper management of the AUX channel interface, as described in Figure 3-8. The Sink policy maker in the example design provides the basic steps for initialization.

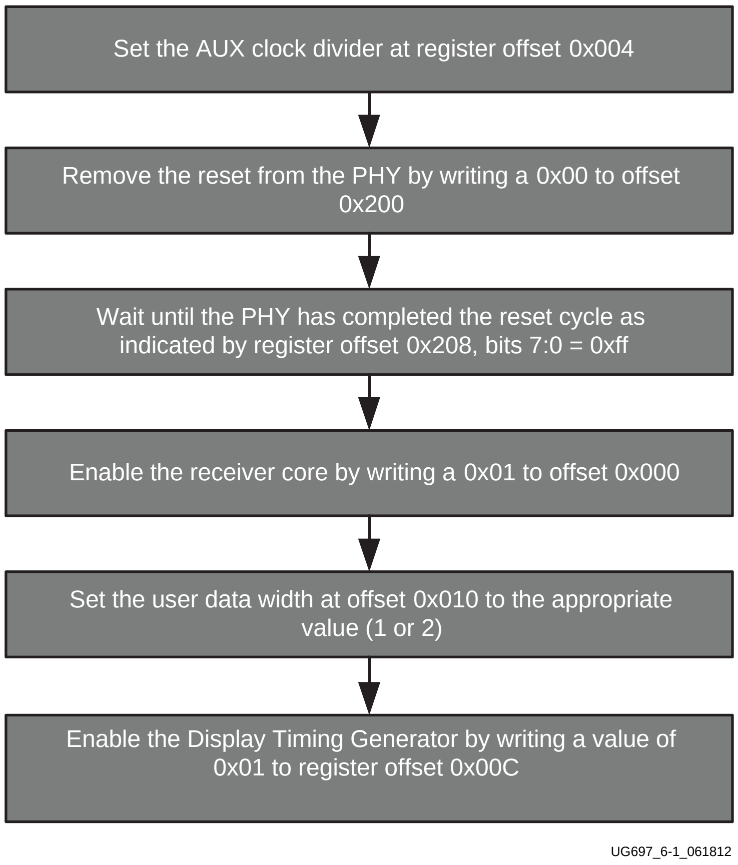

Set the AUX clock divider at register offset 0x004
Remove the reset from the PHY by writing a 0x00 to offset 0x200
Wait until the PHY has completed the reset cycle as indicated by register offset 0x208, bits 7:0 = 0xff
Enable the receiver core by writing a 0x01 to offset 0x000
Set the user data width at offset 0x010 to the appropriate value (1 or 2)
Enable the Display Timing Generator by writing a value of 0x01 to register offset 0x00C
UG697_6-1_061812

Figure 3-8: **Receiver Core Initialization**

The DisplayPort link Hot Plug Detect signal is tied directly to the state of the receiver core enable bit. Until the core is enabled, the receiver will not respond to any AUX transactions or main link video input.

While the Display Timing Generator may be enabled at any time, it is recommended to keep the DTG disabled until the receiver core policy maker detects the start of active video. This condition can be detected initially through the assertion of the MODE_INTERRUPT which will detect the change in the vertical and horizontal resolution values.

Upon receipt of the interrupt, the receiver policy maker should verify the values of the Main Stream Attributes (offset 0x500-0x530) to ensure that the requested video mode is within the range supported by the sink device. If these values are within range, the Display Timing

Generator should be enabled to begin passing valid video frames through the user data interface.

Link Training

The link training commands are passed from the DPCD register block to the link training function. When set into the link training mode, the functional data path is blocked, and the link training controller monitors the PHY and detects the specified pattern. Care must be taken to place the Sink core into the proper link training mode before the source begins sending the training pattern. Otherwise, unpredictable results may occur.

The link training process is specified in section 3.5.1.3 of the *DisplayPort Specification v1.1a*.

Figure 3-9 shows the flow diagram for link training.

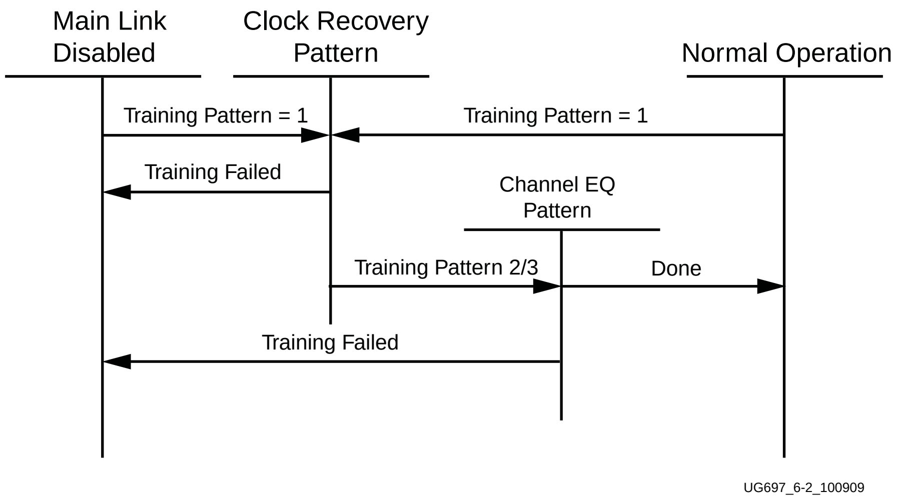


Figure 3-9: **Link Training States**

Receiver Clock Generation

The receiver core requires the generation of a video stream clock for transmitting the recovered image data over the user data interface. Data fields within the Main Stream Attributes (M and N values) provide the information by which an accurate stream clock may be reconstructed. The receiver core places this information on dedicated signals and provides an update flag to signal a change in these values. Alternatively, the user may use a fast clock to pull data from the User Data Interface and push it into a frame buffer.

Figure 3-10 shows how to use the M and N values from the core to generate a clock. See section 2.2.3 of the *DisplayPort Standard v1.1a* for more details.

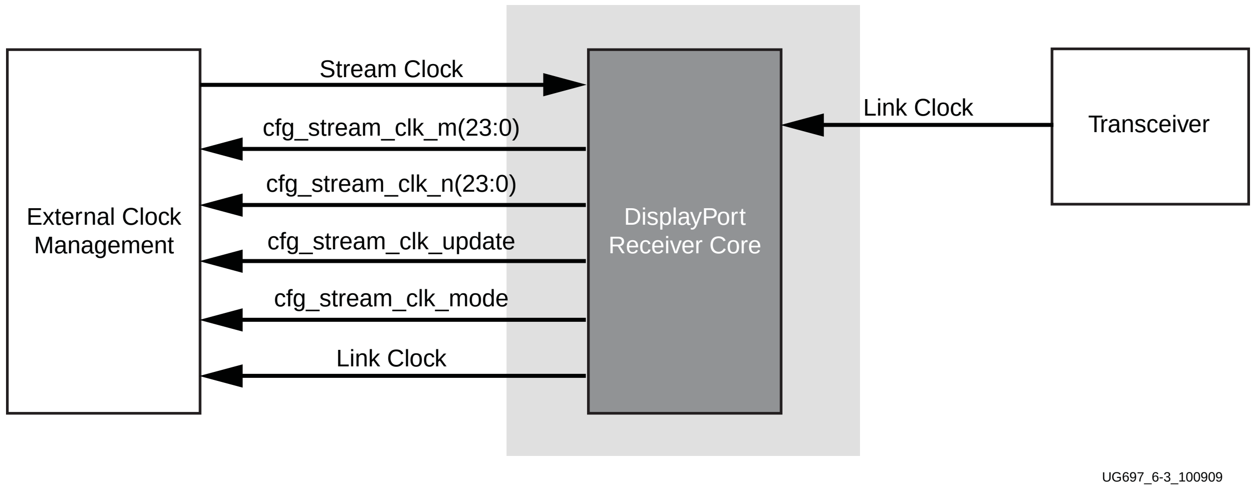


Figure 3-10: **Receiver Clock Generation**

Common Event Detection

In certain applications, the detection of some events may be required. This section describes how to detect these events.

Transition from Video to No Video

In the course of operation, the source core may stop sending video, as detected by the NO_VIDEO interrupt. During this time, the user should not rely on any MSA values.

Transition from No Video to Video

The transmission of video after a NO_VIDEO pattern can be detected by the VERTICAL_BLANKING interrupt. Upon the reception of a VERTICAL_BLANKING interrupt, if disabled, the user may then reenable the display timing generator.

Mode Change

A mode change can be detected by the MODE_CHANGE interrupt. The user must either read the new MSA values from register space or use the dedicated ports provided on the Main Link in order to properly frame the video data.

Cable is Unplugged, Lost Training

When a cable becomes unplugged or training is lost for any other reason, the TRAINING_LOST interrupt will occur. At that point, video data and MSA values should not be relied on.

Once the cable becomes plugged in again, no action is required from the user; the core will properly reset itself and apply HPD.

Link is Trained

The user may determine that the core is properly training by reading from the PHY_STATUS register and observing lane alignment and symbol lock on all active lanes. Additionally, it is advisable to ensure the PLL is locked and reset is complete, also part of the PHY_STATUS register.

Audio Management

This section contains the procedural tasks required to achieve audio communication.

Programming DisplayPort Sink

1. Disable Audio by writing 0x00 to RX_AUDIO_CONTROL register. The disable bit also flushes the buffers in DisplayPort Sink. When there is a change in video/audio parameters, it is recommended to follow this step.
2. Enable Audio by writing 0x01 to RX_AUDIO_CONTROL register.
3. For reading Info Packet, poll the RX_AUDIO_STATUS[0] register, and when asserted, read all eight words.
4. MAUD and NAUD are available as output ports and also in registers. Use these values per the design's clocking structure. For example, in software a poll routine can be used to detect a change and trigger a PLL-M & N value programming.

Programming SPDIF Transmitter

1. Reset SPDIF Receiver by writing 0x0A to SPDIF Soft Reset Register (Base address of SPDIF Transmitter + 0x40). When there is a change in video/audio parameters, it is recommended to follow this step.
2. Enable Audio transmission and set the audio clock divisor to "0001" i.e. '8' to generate the SPDIF signal with the FS sampling rate (User should give a Audio clock of 512*FS, i.e. 8* bit rate, bit rate is 64*FS) by writing 0x0005 to SPDIF Control Register (Base address of SPDIF Transmitter + 0x44).

There are TX FIFO full and TX FIFO empty interrupts generation through single interrupt out signal in SPDIF Transmitter. See PG045, L*ogiCORE IP SPDIF Product Guide,* for details about enabling of these interrupts.

Reading Info/Ext Packet

These packets can be read using poll mode or interrupt mode.

Poll Mode

1. Read RX_AUDIO_STATUS register until Info/Ext packet bit is set.
2. Based on Info/Ext bit setting, read respective buffers immediately. New packets get dropped if buffer is not read.
3. The status bit automatically gets cleared after reading packet.

Interrupt Mode

1. Ensure EXT_PKT_RXD/INFO_PKT_RXD interrupt is enabled by setting proper mask.
2. Wait for interrupt, Read interrupt cause register to check if EXT_PKT_RXD or INFO_PKT_RXD is set.
3. Based on interrupt status, read packet from appropriate buffer immediately.

Audio Clocking (Recommended)

DisplayPort Sink device will receive MAUD and NAUD values from the upstream source device. These values are accessible to the system through the output ports and registers.

The system should have a clock generator (preferably programmable) to generate 512 X fs (Audio Sample Rate) clock frequency based on MAUD and NAUD values. This clock is used by SPDIF transmitter to send data to the SPDIF link.

The AXI4-Stream clock does not need to be related to the Audio clock.

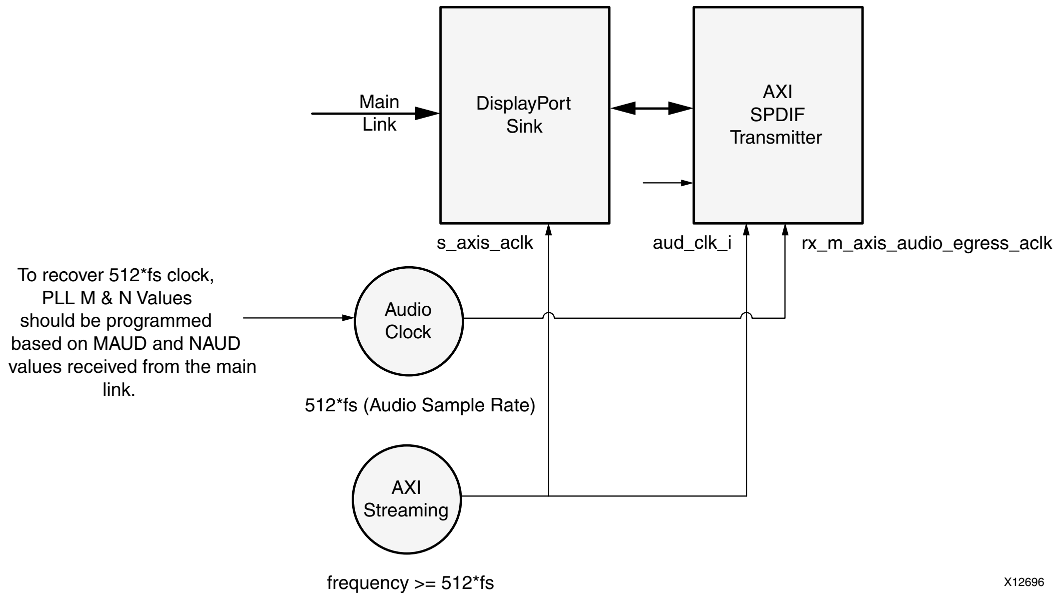

Main Link
DisplayPort Sink
AXI SPDIF Transmitter
s_axis_aclk
aud_clk_i
rx_m_axis_audio_egress_aclk
To recover 512*fs clock, PLL M & N Values should be programmed based on MAUD and NAUD values received from the main link.
Audio Clock
512*fs (Audio Sample Rate)
AXI Streaming
frequency >= 512*fs
X12696

Figure 3-11: **Sink: Audio Clocking**

Source Core Architecture

This chapter provides an overview of the DisplayPort Source core architecture. The DisplayPort core is a full-featured soft IP core, incorporating all necessary logic to properly communicate on this high-speed standard. The Source core supports transmission of high-definition video from a standard-format main link onto up to four lanes of High-Speed Serial I/O.

Module Architecture

The Source core is partitioned into four major blocks, as shown in Figure 3-12:

- **Main Link.** Provides for the delivery of the primary video stream.
- **Secondary Link.** Integrates the delivery of audio information into the Main Link blanking period.
- **AUX Channel.** Establishes the dedicated source to sink communication channel.

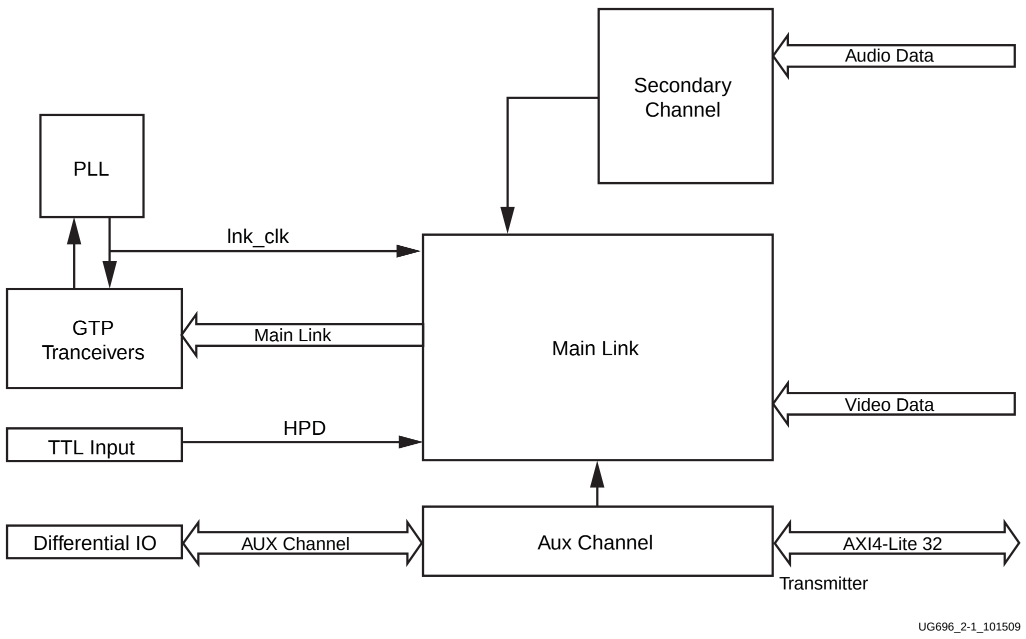


Figure 3-12: **Source Core Top Level**

Source Core Interfaces

General Signals

Table 3-4 describes the General Use signals.

Table 3-4: **General Use Signal Descriptions**

Signal Name	Type	Description
lnk_clk	Output	Link clock for fabric

User Data Interface

Table 3-5 describes the User Data Interface signals.

Table 3-5: **User Data Interface Signal Descriptions**

Signal Name	Type	Description
tx_vid_clk	Input	User video data clock. Input clock rates up to 150 MHz are supported.
tx_vid_rst	Input	User video reset.
tx_vid_vsync	Input	Active high vertical sync pulse. The width is set by the source transmitter.
tx_vid_hsync	Input	Active high horizontal sync pulse. The width is set by the source transmitter.
tx_vid_oddeven	Input	Indicates an odd '1' or even '0' field polarity
tx_vid_enable	Input	Video data valid. Both input pixels are qualified with a single enable. ***Note:*** vid_enable may not toggle during a scan line.
tx_vid_pixel0[47:0]	Input	Video pixel data.
tx_vid_pixel1[47:0]	Input	Video pixel data. This pixel will not be valid if: • It has not been programmed to be valid in the configuration space. • The link has been configured to operate at only one lane.
tx_vid_pixel2[47:0]	Input	Video pixel data 2. This pixel will not be valid if: • It has not been programmed to be valid in the configuration space. • The link has been configured to operate at only one or two lanes.
tx_vid_pixel3[47:0]	Input	Video pixel data 3. This pixel will not be valid if: • It has not been programmed to be valid in the configuration space. • The link has been configured to operate at only one or two lanes.

The primary interface for user image data has been modeled on the industry standard for display timing controller signals. The port list consists of video timing information encoded in a vertical and horizontal sync pulse and data valid indicator. These single bit control lines frame the active data and provide flow control for the streaming video.

Vertical timing is framed using the vertical sync pulse which indicates the end of frame N-1 and the beginning of frame N. The vertical back porch is defined as the number of horizontal sync pulses between the end of the vertical sync pulse and the first line containing active pixel data. The vertical front porch is defined as the number of horizontal sync pulses between the last line of active pixel data and the start of the vertical sync pulse. When combined with the vertical back porch and the vertical sync pulse width, these parameters form what is commonly known as the vertical blanking interval.

At the trailing edge of each vertical sync pulse, the user data interface will reset key elements of the image data path. This provides for a robust user interface that recovers from any kind of interface error in one vertical interval or less.

Figure 3-13 shows the typical signalling of a full frame of data.

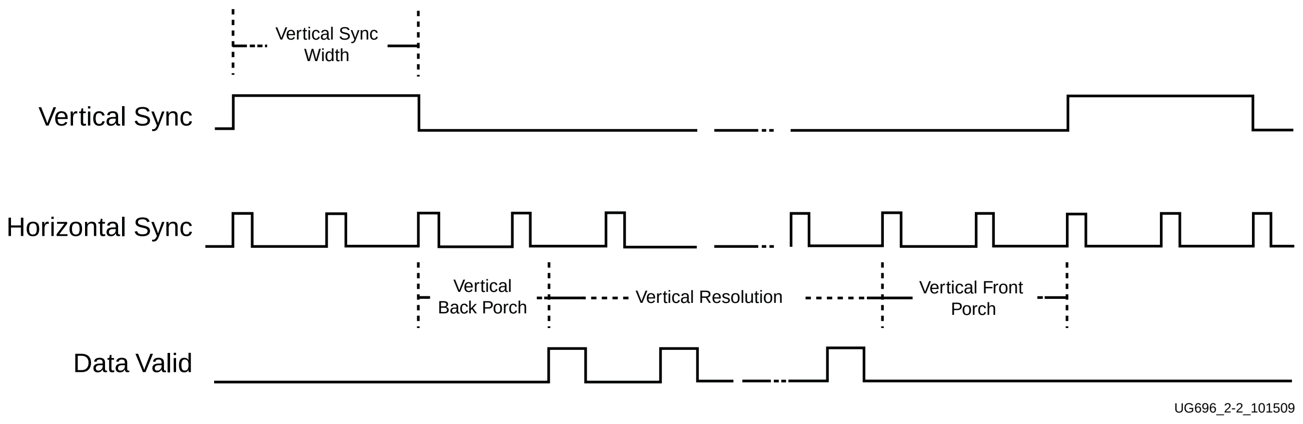


Figure 3-13: **User Interface Vertical Timing**

Similarly, the horizontal timing information is defined by a front porch, back porch, and pulse width. The porch values are defined as the number of clocks between the horizontal sync pulse and the start or end of active data. Pixel data is only accepted into the image data interface when the data valid flag is active-High, as shown in Figure 3-14.

Note that the data valid signal must remain asserted for the duration of a scan line. Dropping the valid signal may result in improper operation.

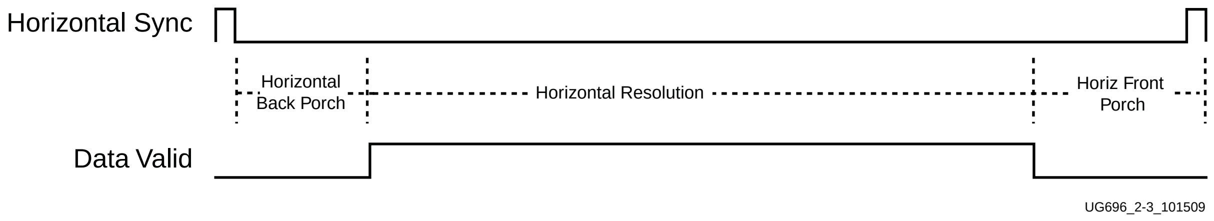


Figure 3-14: **User Interface Horizontal Timing**

In the two dimensional image plane, these control signals frame a rectangular region of active pixel data within the total frame size. This relationship of the total frame size to the active frame size is shown in Figure 3-15.

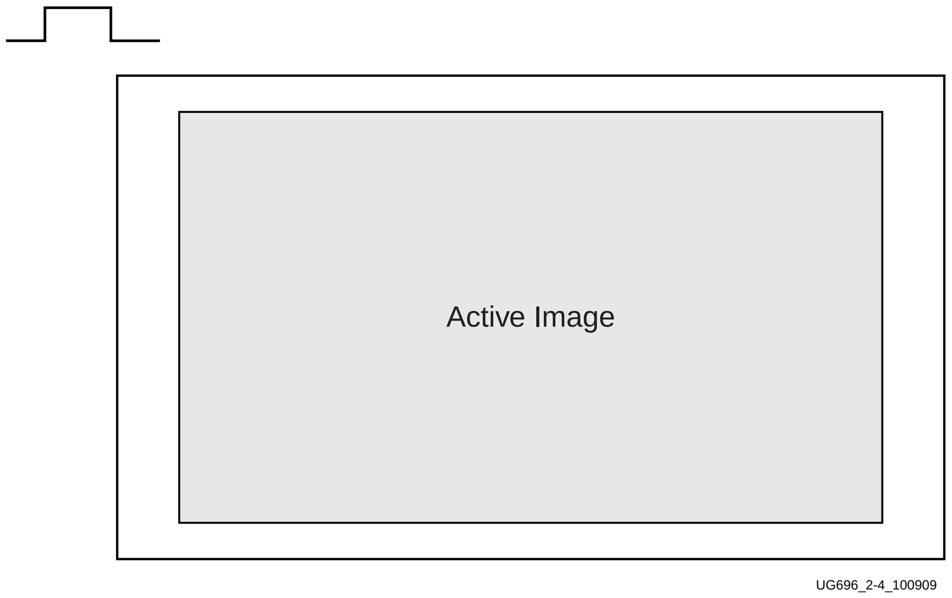


Figure 3-15: **Active Image Data**

The User Data Interface can accept one or two pixels per clock cycle. The vid_pixel width is always 48 bits, regardless of if all bits are used. For pixel mappings that do not require all 48 bits, the convention used for this core is to occupy the MSB bits first and leave the lower bits either untied or driven to zero. Table 3-6 provides the proper mapping for all supported data formats.

Table 3-6: **Pixel Mapping for the User Data Interface**

Format	BPC/BPP	R	G	B	Cr	Y	Cb	Cr/Cb	Y
RGB	6/18	[47:42]	[31:26]	[15:10]					
RGB	8/24	[47:40]	[31:24]	[15:8]					
RGB	10/30	[47:38]	[31:22]	[15:6]					

Table 3-6: **Pixel Mapping for the User Data Interface *(Cont'd)***

Format	BPC/BPP	R	G	B	Cr	Y	Cb	Cr/Cb	Y
RGB	12/36	[47:36]	[31:20]	[15:4]					
RGB	16/48	[47:32]	[31:16]	[15:0]					
YCbCr444	6/18				[47:42]	[31:26]	[15:10]		
YCbCr444	8/24				[47:40]	[31:24]	[15:8]		
YCbCr444	10/30				[47:38]	[31:22]	[15:6]		
YCbCr444	12/36				[47:36]	[31:20]	[15:4]		
YCbCr444	16/48				[47:32]	[31:16]	[15:0]		
YCbCr422	8/16							[47:40]	[31:24]
YCbCr422	10/20							[47:38]	[31:22]
YCbCr422	12/24							[47:36]	[31:20]
YCbCr422	16/32							[47:32]	[31:16]
YONLY	8/8								[47:40]
YONLY	10/10								[47:38]
YONLY	12/12								[47:36]
YONLY	16/16								[47:32]

Notes:

For a YCrCb 4:2:2, the input follows YCr, YCb, YCr, YCb and so on. This means Cr and Cb are mapped to the same bits on the video input ports of the Source core.

Selecting the Pixel Interface

The Pixel clock is supported up to 150 MHz, and it is very difficult to meet timing above this frequency. However, users do have the option of selecting a 1, 2 or 4 pixel video interface.

To determine the necessary pixel interface to support a specific resolution, it is important to know the active resolution and blanking information.

For example:

To support an active resolution of 2560x1600@60, there are two possible blanking formats: Normal Blanking and Reduced Blanking, as defied by the VESA specification.

2560x1600@60 + Blanking = 3504x1658@60

Requires a Pixel clock of 348.58 MHz

2560x1600@60 + Reduced Blanking = 2720x1646@60

Requires a Pixel clock of 268.63 MHz

Assuming a pixel clock of 150MHz and a 2 (Dual) Pixel interface:

2560x1600@60 + Blanking = 3504x1658@60 = 348.58 MHz

348.58 MHz / 2 = 172.28 MHz

2560x1600@60 + Reduced Blanking = 2720x1646@60 = 268.63 MHz

268.63 MHz / 2 = 134.31 MHz

With a 2 Pixel interface, the DisplayPort IP can support 2560x1600 only if there is a Reduced Blanking input. If full Blanking support is needed, then a 4 Pixel interface should be used.

Host Processor Interface

Table 3-7 describes the Host Processor Interface signals.

Table 3-7: **Host Processor Interface Signal Descriptions for DisplayPort**

Signal Name	Type	Description
s_axi_aclk	Input	AXI Bus Clock.
s_axi_aresetn	Input	AXI Reset. Active-Low.
s_axi_awaddr[31:0]	Input	Write Address
s_axi_awprot[2:0]	Input	Protection type.
s_axi_awvalid	Input	Write address valid.
s_axi_awready	Output	Write address ready.
s_axi_wdata[31:0]	Input	Write data bus.
s_axi_wstrb[3:0]	Input	Write strobes.
s_axi_wvalid	Input	Write valid.
s_axi_wready	Output	Write ready.
s_axi_bresp[1:0]	Output	Write response.
s_axi_bvalid	Output	Write response valid.
s_axi_bready	Input	Response ready.
s_axi_araddr[31:0]	Input	Read address.
s_axi_arprot[2:0]	Input	Protection type.
s_axi_arvalid	Input	Read address valid.
s_axi_arready	Output	Read address ready.
s_axi_rdata[31:0]	Output	Read data.
s_axi_rresp[1:0]	Output	Read response.
s_axi_rvalid	Output	Read valid.
s_axi_rready	Input	Read ready.
axi_int	Output	AXI interrupt out.

The host processor bus uses an AMBA AXI4-Lite interface, which was selected because of its simplicity. The processor bus allows for single reads and writes to configuration space. See Source Core in Chapter 2 for full address mapping.

Additionally, the host processor interface is the gateway for initiating and maintaining the main link. This is done through Link and Device services, which include EDID and DPCD reads. Main link initiation concludes with a Link Training sequence, which is also started through this interface. Refer to Link Training as well as the VESA specification[Ref 1] for more information about the initiation sequence.

The core comes with an example design policy maker in C source code. For users who do not have specific needs to control or tune the core, this is an ideal resource.

The host processor interface allows the user to program SPDIF control registers, as described in Table 3-8.

Table 3-8: **Host Processor Interface Signal Descriptions for Audio**

SIgnal Name	Type	Description
aud_s_axi_aclk	Input	AXI Bus Clock.
aud_s_axi_aresetn	Input	AXI Reset. Active-Low.
aud_s_axi_awaddr[31:0]	Input	Write Address.
aud_s_axi_awprot[2:0]	Input	Protection type.
aud_s_axi_awvalid	Input	Write address valid.
aud_s_axi_awready	Output	Write address ready.
aud_s_axi_wdata[31:0]	Input	Write data bus.
aud_s_axi_wstrb[3:0]	Input	Write strobes.
aud_s_axi_wvalid	Input	Write valid.
aud_s_axi_wready	Output	Write ready.
aud_s_axi_bresp[1:0]	Output	Write response.
aud_s_axi_bvalid	Output	Write response valid
aud_s_axi_bready	Input	Response ready.
aud_s_axi_araddr[31:0]	Input	Read Address.
aud_s_axi_arprot[2:0]	Input	Protection type.
aud_s_axi_arvalid	Input	Read address valid.
aud_s_axi_arready	Output	Read address ready.
aud_s_axi_rdata[31:0]	Output	Read data.
aud_s_axi_rresp[1:0]	Output	Read response.
aud_s_axi_rvalid	Output	Read valid.
aud_s_axi_rready	Input	Read ready.
aud_axi_int	Output	AXI interrupt out.

AXI4-Lite Read and Write Cycles

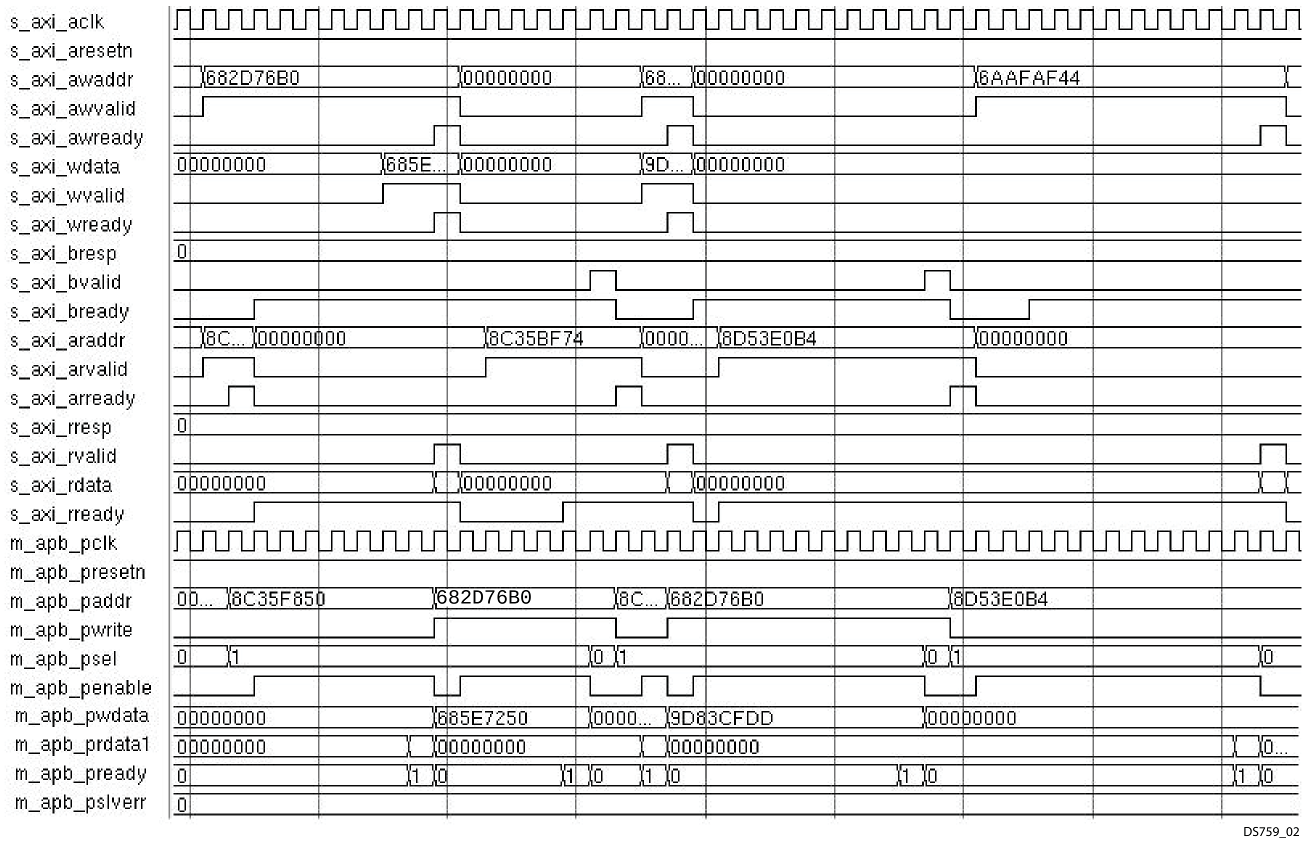


Figure 3-16: **AXI4-Lite Read and Write Cycles**

The AXI4-Lite write transfer begins with the address, write signal, and write data set to their proper values on the first rising edge of the clock. The first clock cycle of the transfer is called the SETUP cycle. On the second rising edge of the clock, the enable signal is asserted and the ENABLE cycle is entered. The address, data, and control signals all remain valid through both cycles of the transfer. The transfer completes on the following rising edge of the clock, as shown in Figure 3-16.

The AXI4-Lite read transfer begins with the SETUP cycle on the first rising edge of the clock with the address and control signals at their proper values. As with the write transfer, the enable signal is asserted on the next rising edge marking the beginning of the ENABLE cycle. The slave peripheral must provide data during this cycle. The read data is sampled on the next rising edge of the clock at the end of the ENABLE cycle. This transfer is shown in Figure 3-16.

Transceiver Interface

Table 3-9 describes the Transceiver Interface signals

Table 3-9: **Transceiver Interface Signal Descriptions**

Signal Name	Type	Description
lnk_clk_p	Input	Differential link clock input. Must be placed on the MGTREFCLKP pin.
lnk_clk_n	Input	Differential link clock input. Must be placed on the MGTREFCLKN pin.
lnk_clk_81_p	Input	Differential link clock, 81MHz. Must be placed on the MGTREVCLKP pin. Valid for Virtex-6 and Spartan-6 device families.
lnk_clk_81_n	Input	Differential link clock, 81MHz. Must be placed on the MGTREVCLKN pin. Valid for Virtex-6 and Spartan-6 device families.
lnk_tx_lane_p[LANE_COUNT-1:0]	Output	High-speed differential data output.
lnk_tx_lane_n[LANE_COUNT-1:0]	Output	High-speed differential data output.

The transceivers have been pulled out of the core and are provided as instances in the top-level wrapper. The user may choose up to four high-speed lanes. Despite the number of lanes that have been chosen, the negotiation process is handled by a policy maker, which may elect for fewer number of in-use lanes. Additionally, the core supports both 2.7 Gbps and 1.62 Gbps operation. The negotiation process also determines the actual line rate.

The user must provide the appropriate reference clock on the lnk_clk_p/nports. These ports must be physically located on the appropriate MGTREFCLK pins. Additionally, the user must physically locate the `lnk_tx_lane` ports to the appropriate pins. To find the appropriate placement locations, refer to the transceiver user guide for the FPGA family used (References).

For Virtex-6 and Spartan®-6 FPGAs, there is not a common reference clock between the 1.62 Gbps and 2.7 Gbps link rates. If both rates are desired, the user must provide both an 81 MHz and 135 MHz reference clock on the board and supply them to the appropriate MGTREFCLK pins. Proper clock switching is provided within the PHY wrapper file. For more information on Spartan-6 transceivers, see the *Spartan-6 FPGA GTP Transceiver User Guide* [Ref 3] and for Virtex-6, see the *Virtex-6 FPGA GTX Transceiver Advance Product Specification* [Ref 5].

For 7 series FPGAs, a common reference clock of 108 MHz is needed for 1.62, 2.7 and 5.4 Gbps link rates.

The transceivers have been tuned for optimal communication. The constraints related to transceiver tuning have been placed directly in the RTL instance. Users may want to review these values and make sure they are fully aware of their functions.

AUX Channel Interface/HPD Interface

Table 3-10 describes the AUX Channel Interface/HPD Interface signals.

Table 3-10: **AUX Channel Interface Signal Descriptions**

Signal Name	Type	Description
aux_tx_channel_in_p	Input	Differential signal for AUX channel communication.
aux_tx_channel_in_n	Input	Differential signal for AUX channel communication.
aux_tx_channel_out_p	Output	Differential signal for AUX channel communication.
aux_tx_channel_out_p	Output	Differential signal for AUX channel communication.
aux_tx_io_p	Input/ Output	Positive Polarity AUX Manchester-II data (used for Spartan-6 devices).
aux_tx_io_n	Input/ Output	Negative Polarity AUX Manchester-II data (used for Spartan-6 devices).
tx_hpd	Input	Hot plug detect. ***Note:*** The DisplayPort core requires HPD IO to be 3.3v. If a 2.5V IO standard is being used, a 3.3V level shifter should be added to the HPD signal on the board.

The AUX channel is used for link and device communication between source and sink devices. The AUX channel uses Manchester-II Coding and requires a 1 MHz (or a multiple of 1 MHz) clock source. The AXI4-Lite clock is used to run the internal operations of the AUX Channel logic. As a result, using the bus interface clock in this way restricts the AXI4-Lite clock frequency to an integer multiple of 1 MHz.

Tie these ports to general IO pins and use the LVDS drive standard. For Spartan-6 devices supporting the DisplayPort IO standard, these pins may be combined to use the dedicated DisplayPort drive standard.

Audio Interface

Table 3-11 describes the Audio Clock Interface signals.

Table 3-11: **Audio Clock Interface Signals**

Signal	Direction	Description
aud_clk	Input	Audio sample clock (512 * fs). fs= sampling frequency.
aud_rst	Input	Audio Interface Reset (Active-High).
aud_axis_aclk	Input	Audio streaming interface clock (greater than or equal to 512 * fs)
aud_axis_aresetn	Input	Audio Streaming Interface Reset (Active-Low).
spdif_sample_clk	Input	SPDIF Controller sampling clock. Should be greater than or equal to 512*fs.

The SPDIF input is sampled by SPDIF receiver, and audio samples are transferred to the Displayport Audio engine through the AXI4-Stream interface.

Table 3-12: **SPDIF Interface Signals**

Signal	Direction	Description
spdif_in	Input	SPDIF Channel Input

Sink Core Architecture

The Sink core is partitioned into the following four major blocks

- **Main Link.** Provides for the delivery of the primary video stream.
- **Secondary Link.** Provides the delivery of audio information from the blanking period of the video stream to an AXI4-Stream (SPDIF) interface.
- **AUX Channel.** Establishes the dedicated source to sink communication channel.
- **DPCD.** Contains the set of Display Port Configuration Data, which is used to establish the operating parameters of each core.

Figure 3-17 shows a top-level diagram of the Sink core.

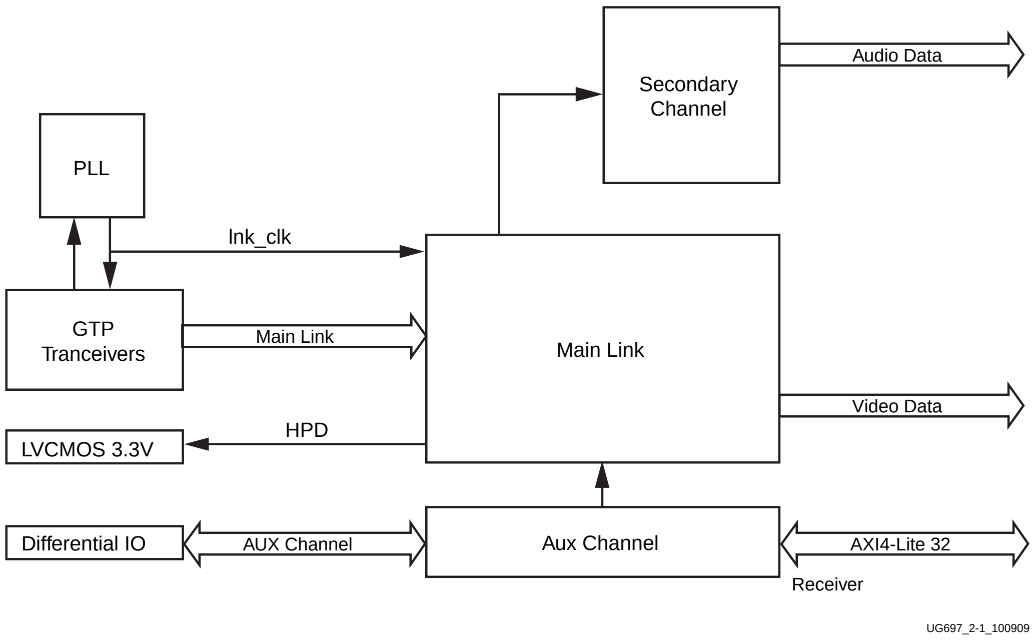


Figure 3-17: **Sink Core Top Level**

Sink Core Interfaces

General Signals

Table 3-13 describes the General Use signals.

Table 3-13: **General Use Signal Descriptions**

Signal Name	Type	Description
lnk_clk	Output	Link clock for pixel clock generation
lnk_m_vid[23:0]	Output	Video time stamp
lnk_n_vid[23:0]	Output	Video time stamp
lnk_n_aud[23:0]	Output	N-value for audio clock generation.

User Data Interface

Table 3-14 describes the User Data Interface signals.

Table 3-14: **User Data Interface Signal Descriptions**

Signal Name	Type	Description
rx_vid_clk	Input	User video data clock. Input clock rates up to 135MHz are supported.
rx_vid_rst	Input	User video reset.
rx_vid_vsync	Output	Active high vertical sync pulse. The width is set by the source device.
rx_vid_hsync	Output	Active high horizontal sync pulse. The width is set by the source device. The vid_hsync signal only asserts to indicate when to start a new line.
rx_vid_oddeven	Output	Indicates an odd (1) or even (0) field polarity.
rx_vid_enable	Output	Video Data Valid.
rx_vid_pixel0[47:0]	Output	Video pixel data 0.
rx_vid_pixel1[47:0]	Output	Video pixel data 1. This pixel will not be valid if: • It has not been programmed to be valid in the configuration space, or • The link has been configured to operate at only one lane

Table 3-14: **User Data Interface Signal Descriptions *(Cont'd)***

Signal Name	Type	Description
rx_vid_pixel2[47:0]	Output	Video pixel data 2. This pixel will not be valid if: • It has not been programmed to be valid in the configuration space, or • The link has been configured to operate at only one lane or two lanes
rx_vid_pixel3[47:0]	Output	Video pixel data 3. This pixel will not be valid if: • It has not been programmed to be valid in the configuration space, or • The link has been configured to operate at only one lane or two lanes

The primary interface for user image data has been modeled on the industry standard for display timing controller signals. The port list consists of video timing information encoded in a vertical and horizontal sync pulse and data valid indicator. These single-bit control lines frame the active data and provide flow control for the streaming video.

Vertical timing is framed using the vertical sync pulse, which indicates the end of frame N-1 and the beginning of frame N. The vertical back porch is defined as the number of horizontal sync pulses between the end of the vertical sync pulse and the first line containing active pixel data. The vertical front porch is defined as the number of horizontal sync pulses between the last line of active pixel data and the start of the vertical sync pulse. When combined with the vertical back porch and the vertical sync pulse width, these parameters form what is commonly known as the vertical blanking interval.

At the trailing edge of each vertical sync pulse, the User Data Interface will reset key elements of the image data path. This provides for a robust user interface that recovers from any kind of interface error in one vertical interval or less.

The user has the option to use the resolved M and N values from the stream to generate a clock, or to use a sufficiently-fast clock and pipe the data into a line buffer. Xilinx recommends using a fast clock and ignoring the M and N values unless the user can be certain of the source of these values. Unlike the Source Core, when using a fast clock, the data valid signal may toggle within a scan line. Figure 3-18 shows the typical signalling of a full frame of data.

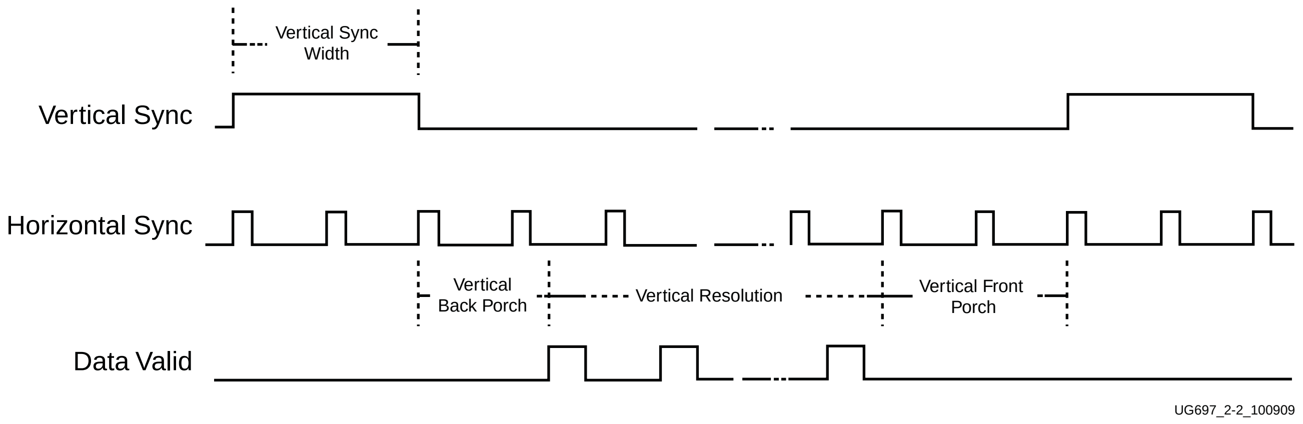


Figure 3-18: **User Interface Vertical Timing**

The horizontal timing information is defined by a front porch, back porch, and pulse width. The porch values are defined as the number of clocks between the horizontal sync pulse and the start or end of active data. Pixel data is only accepted into the image data interface when the data valid flag is active-High. Figure 3-19 is an enlarged version of Figure 3-18, giving more detail on a single scan line. The horizontal sync pulse should be used as a line advance signal. Use the rising edge of this signal to increment the line count. Note that Data Valid may toggle if using a fast clock.

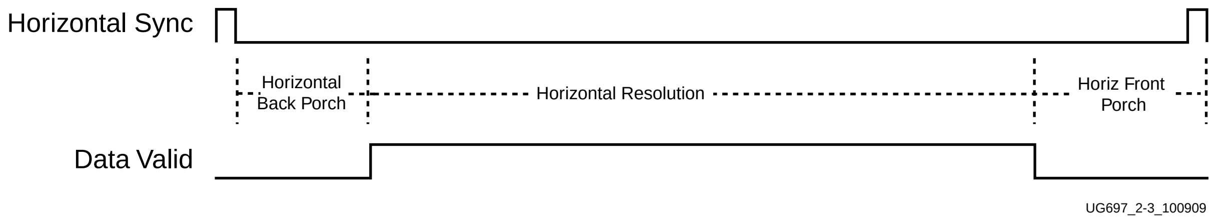


Figure 3-19: **User Interface Horizontal Timing**

In the two dimensional image plane, these control signals frame a rectangular region of active pixel data within the total frame size. This relationship of the total frame size to the active frame size is shown in Figure 3-20.

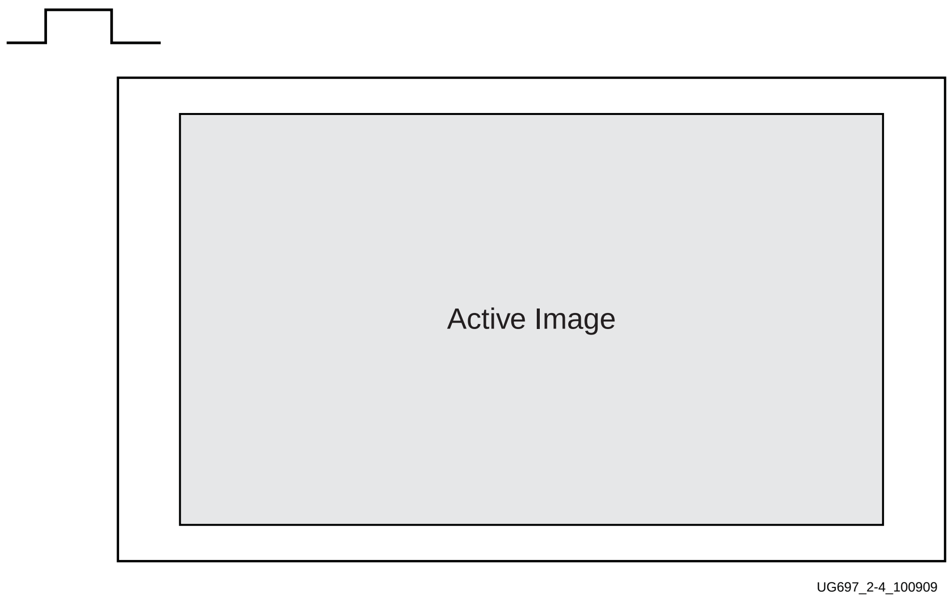


Figure 3-20: **Active Image Data**

The User Data Interface can accept one or two pixels per clock cycle. The second pixel is active only when USER_PIXEL_WIDTH is set and the negotiated number of lanes is greater than one.

The vid_pixel width is always 48 bits, regardless of if all bits are used. For pixel mappings that do not require all 48 bits, the convention used for this core is to occupy the MSB bits first and leave the lower bits either untied or driven to zero. Table 3-15 provides the proper mapping for all supported data formats.

Table 3-15: **Pixel Mapping for the User Data Interface**

Format	BPC/BPP	R	G	B	Cr	Y	Cb	Cr/Cb	Y
RGB	6/18	[47:42]	[31:26]	[15:10]					
RGB	8/24	[47:40]	[31:24]	[15:8]					
RGB	10/30	[47:38]	[31:22]	[15:6]					
RGB	12/36	[47:36]	[31:20]	[15:4]					
RGB	16/48	[47:32]	[31:16]	[15:0]					
YCbCr444	6/18				[47:42]	[31:26]	[15:10]		
YCbCr444	8/24				[47:40]	[31:24]	[15:8]		
YCbCr444	10/30				[47:38]	[31:22]	[15:6]		
YCbCr444	12/36				[47:36]	[31:20]	[15:4]		
YCbCr444	16/48				[47:32]	[31:16]	[15:0]		

Table 3-15: **Pixel Mapping for the User Data Interface *(Cont'd)***

Format	BPC/BPP	R	G	B	Cr	Y	Cb	Cr/Cb	Y
YCbCr422	8/16							[47:40]	[31:24]
YCbCr422	10/20							[47:38]	[31:22]
YCbCr422	12/24							[47:36]	[31:20]
YCbCr422	16/32							[47:32]	[31:16]
YONLY	8/8								[47:40]
YONLY	10/10								[47:38]
YONLY	12/12								[47:36]
YONLY	16/16								[47:32]

Notes:

For a YCrCb 4:2:2, the output pixel follows YCr, YCb, YCr, YCb and so on. This means Cr and Cb are mapped to the same bits on the video output ports of the Sink core.

Host Processor Interface

The host processor bus uses an AXI4-Lite interface, which was selected because of its simplicity. The processor bus allows for single reads and writes to the configuration space. See Chapter 2, Register Space for address mapping.

Use the Sink core's Host Processor Interface to enable and set up the core. This interface may also be used to check the status of training.

AXI4-Lite Read and Write Cycles

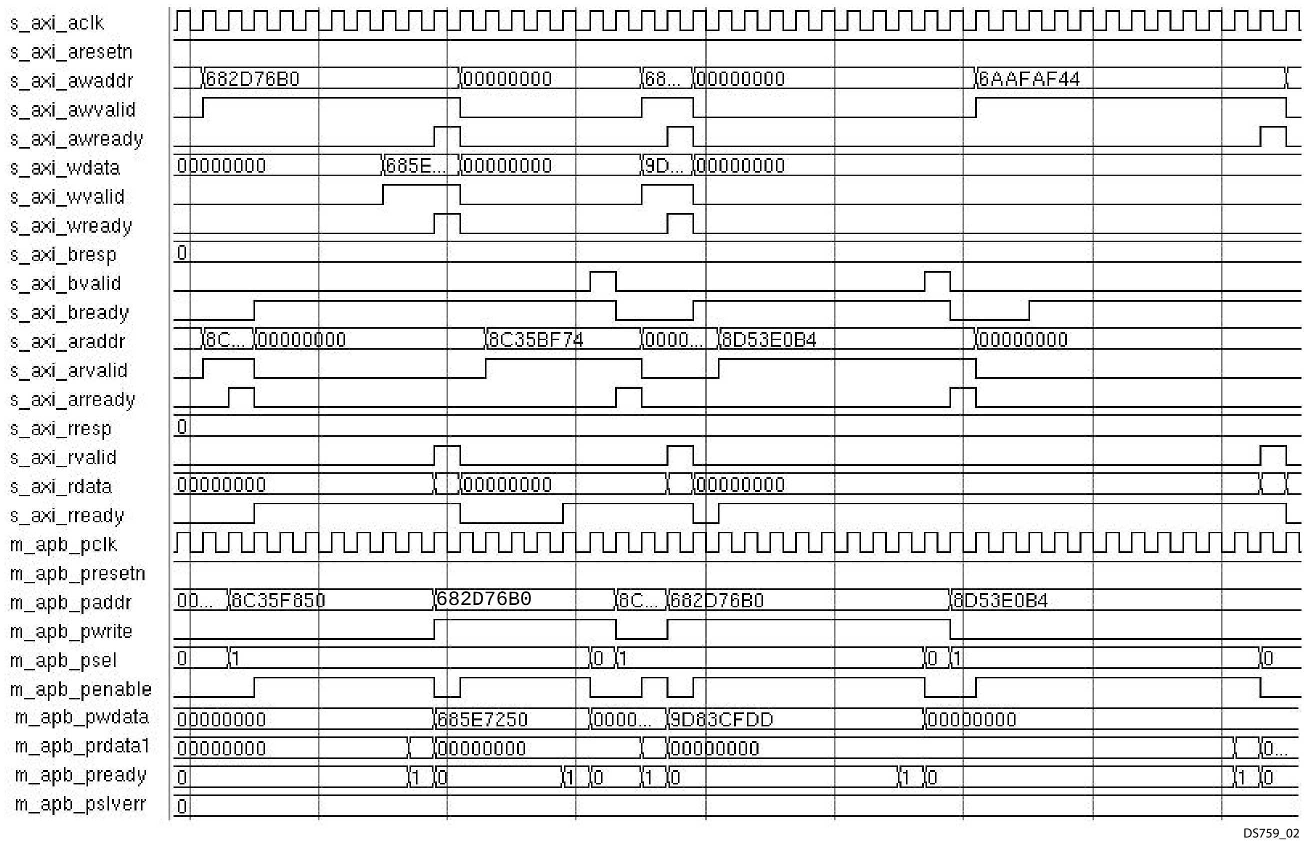


Figure 3-21: **AXI4-Lite Read and Write Cycles**

The AXI4-Lite write transfer begins with the address, write signal, and write data set to their proper values on the first rising edge of the clock. The first clock cycle of the transfer is called the SETUP cycle. On the second rising edge of the clock, the enable signal is asserted and the ENABLE cycle is entered. The address, data, and control signals all remain valid through both cycles of the transfer. The transfer completes on the following rising edge of the clock, as shown in Figure 3-21.

The AXI4-Lite read transfer begins with the SETUP cycle on the first rising edge of the clock with the address and control signals at their proper values. As with the write transfer, the enable signal is asserted on the next rising edge marking the beginning of the ENABLE cycle. The slave peripheral must provide data during this cycle. The read data is sampled on the next rising edge of the clock at the end of the ENABLE cycle. This transfer is shown in Figure 3-21.

Transceiver Interface

The transceivers have been pulled out of the core and are provided as instances in the top-level wrapper. The user may choose up to four high-speed lanes. Despite the number of lanes that have been chosen, the core automatically handles the negotiation process, which may result in a fewer number of in-use lanes. The negotiation process also determines the actual line rate.

The user must provide the appropriate reference clock on the `lnk_clk_p/n` ports. These ports must be physically located on the appropriate MGTREFCLK pins. Additionally, the user must physically locate the `lnk_tx_lane` ports to the appropriate pins. To find the appropriate placement locations, refer to the transceiver user guide for the FPGA family used (References).

For Spartan®-6 FPGAs, there is not a common reference clock between the 1.62 Gbps and 2.7 Gbps lane rates. If both rates are desired, the user must provide both an 81 MHz and 135 MHz reference clock on the board and supply them to the appropriate MGTREFCLK pins. Proper clock switching is provided within the PHY wrapper file. For more information on Spartan-6 FPGA transceivers, see the *Spartan-6 FPGA GTP Transceiver User Guide* [Ref 3].

For 7 series FPGAs, a common reference clock of 108 MHz is needed for 1.62, 2.7 and 5.4 Gbps link rates.

The transceivers have been tuned for optimal communication. The constraints related to transceiver tuning have been placed directly in the RTL instance. Users may want to review these values and make sure they are fully aware of their functions.

AUX Channel Interface/HPD Interface

AUX Channel Services are provided through a dedicated differential pair in the PHY layer. The data operates at a frequency of 1 Mbps with all data Manchester-II encoded. The functional independence of the AUX Channel allows for a design which is independent of the main link with the exception of the DisplayPort Configuration Data (DPCD). All DPCD registers are considered to be asynchronous to the link clock. Where necessary, synchronization stages will be used to properly sample the data in the main link design.

The AXI4-Lite clock is used to run the internal operations of the AUX Channel logic. In addition, the AXI4-Lite clock is used to derive the data rate of the Manchester-II encoded transmit and reply data. Using the bus interface clock in this way restricts the AXI4-Lite clock frequency to an integer multiple of 1 MHz. This restriction is required in order to generate the Manchester-II codes at the frequency of 1 Mbps.

Tie these ports to general IO pins and use the LVDS drive standard. For Spartan-6 devices, these pins may be combined to use the dedicated DISPLAY_PORT drive standard.

DisplayPort Configuration Data

The DisplayPort Configuration Data is implemented as a set of registers which may be read or written from the AXI4-Lite interface. While these registers are not technically part of the AUX Channel interface, they are integrated here for access via the AXI4-Lite bus interface. These registers are considered to be synchronous to the AXI4-Lite domain and asynchronous to all others.

For parameters that may change while being read from the configuration space, two scenarios may exist. In the case of single bits, the data may be read without concern as either the new value or the old value will be read as valid data. In the case of multiple bit fields, a lock bit may be maintained to prevent the status values from being updated while the read is occurring. For multi-bit configuration data, a toggle bit will be used indicating that the local values in the functional core should be updated.

I2C Interface

Note: This is a pass-through interface. The expectation is for the controller to be built outside of the core. An example is in the example design.

The Source core enables the I2C protocol over the AUX channel. For direct access via I2C and as an alternative to the host processor bus, use this dedicated interface.

Figure 3-22 shows an example I2C Transaction.

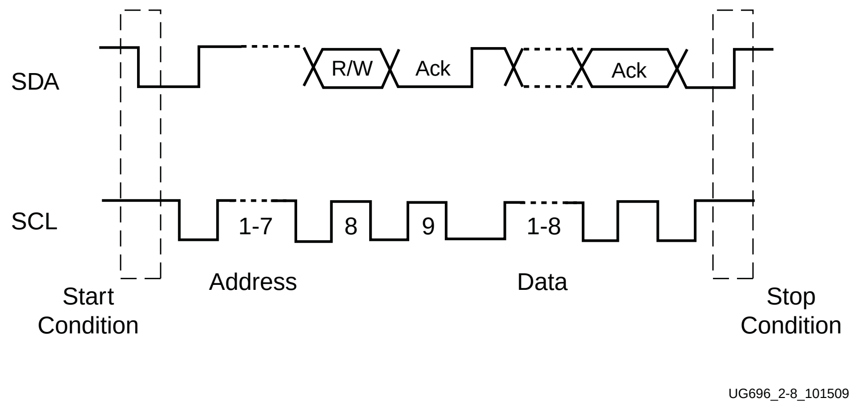


Figure 3-22: **I2C Transaction**

Audio Interface

Audio is received from the DisplayPort link and transferred to the SPDIF controller using the AXI4-Stream interface. Audio data is converted to the SPDIF format and then transmitted at the required audio rate.

Clocking

The core uses six clock domains:

- **lnk_clk.** Most of the core operates in this domain. This domain is based off of the lnk_clk_p/n. When the lanes are running at 2.7 Gbps, lnk_clk will operate at 135 MHz. When the lanes are running at 1.62 Gbps, lnk_clk will operate at 81 MHz.

- **vid_clk.** This is the primary user interface clock. It has been tested to run as fast as 135 MHz, which accommodates to a screen resolution of 2560x1600 when using two-wide pixels. See Selecting the Pixel Interface in Chapter 5 for more information on how to select the appropriate pixel interface.
- **s_axi_aclk.** This is the processor domain. It has been tested to run as fast as 135 MHz. The AUX clock domain is derived from this domain, but requires no additional constraints.
- **aud_clk**. This is the audio interface clock. The frequency will be equal to 512 x audio sample rate.
- **spdif_sample_clk**. This is used by SPDIF receiver to sample incoming traffic. This clock should be >= 512 x audio sample rate.
- **aud_axis_aclk**. This clock is used by the Audio streaming interface. This clock should be >= 512 x audio sample rate.

Resets

Resets for the Source and Sink cores of the DisplayPort solution are as follows:

- Source Core Resets
  - **s_axi_aresetn**. AXI Reset. Active-Low.
  - **tx_vid_rst**. User video reset.
  - **aud_s_axi_aresetn**. AXI Reset. Active-Low.
  - **aud_rst**. Audio Interface Reset. Active-High.
  - **aud_axis_aresetn**. Audio Streaming Interface Reset. Active-Low.
- Sink Core Resets
  - **s_axi_aresetn**. AXI Reset. Active-Low.
  - **rx_vid_rst**. User video reset.
  - **aud_s_axi_aresetn**. AXI Reset. Active-Low.
  - **aud_rst**. Audio Interface Reset. Active-High.
  - **aud_axis_aresetn**. Audio Streaming Interface Reset. Active-Low.

SECTION II: VIVADO DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Detailed Example Design

Customizing and Generating the Core

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

The Source (TX) and Sink (RX) core are generated independently through the Xilinx Vivado software using a graphical user interface (GUI).

This chapter describes the GUI options used to generate and customize the cores. The Source and Sink cores are generated independently, and the user may choose to generate only one or both cores.

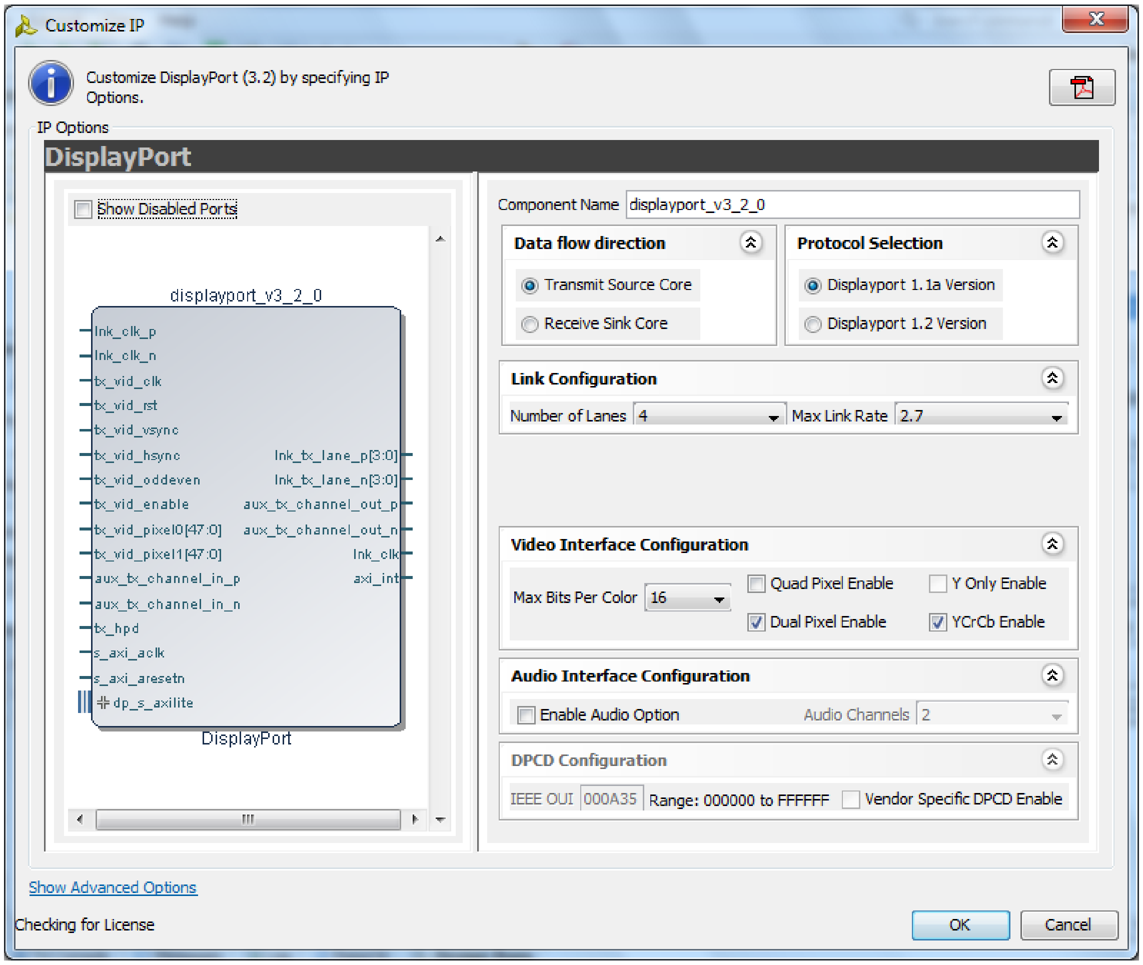
For assistance with starting and using the Xilinx Vivado software, see the Xilinx Vivado Design Suite documentation.

GUI

This section describes the Xilinx Vivado software configuration screen, provided to configure the DisplayPort design.

Main Screen

Figure 4-1 shows the DisplayPort VIVADO main configuration screen. Descriptions of the GUI options on this screen are provided in the following text.



Component Name: displayport_v3_2_0

Data flow direction: [✓ Transmit Source Core] Receive Sink Core

Protocol Selection: [✓ Displayport 1.1a Version] Displayport 1.2 Version

Link Configuration: Number of Lanes 4, Max Link Rate 2.7

Video Interface Configuration: Max Bits Per Color 16; Quad Pixel Enable ☐; Y Only Enable ☐; Dual Pixel Enable ✓; YCrCb Enable ✓

Audio Interface Configuration: Enable Audio Option ☐; Audio Channels 2

DPCD Configuration: IEEE OUI 000A35 Range: 000000 to FFFFFF; Vendor Specific DPCD Enable ☐

Figure 4-1: **Vivado Main Screen**

Component Name

The Component Name is used as the name of the top-level wrapper file for the core. The underlying netlist still retains its original name. Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9, and "_". The name displayport_v3_2 is used as internal module name and should not be used for the component name. The default is displayport_v3_2_0.

Data Flow Direction

Select either the Sink (RX) or Source (TX) core with the Data Flow Direction radio button. If both directions are desired, the user must generate both a TX and RX core separately and combine these with the supplied wrapper files.

Protocol Selection

Select the protocol version for which the core is to be generated. DisplayPort v1.2 is supported on 7 series FPGA families and above.

Number of Lanes

Choose 1, 2, or 4 maximum lanes. Choose fewer lanes for a more optimized design. More lanes allow for higher overall bandwidth and higher resolutions

Max Link Rate

Select the maximum link rate to be supported by the design.

Max Bits Per Color

Choose the maximum bits per color that the core supports. Six bits per color is supported by default.

Quad Pixel Enable

Select this check box to enable the four pixel-wide video interface. The quad pixel interface option is available for four-lane designs.

Dual Pixel Enable

Select this check box to enable the two pixel-wide video interface. The dual pixel interface option is available for two- and four-lane designs.

Y Only Enable

Select this check box to enable Y-Only color space logic. This option is available on cores using DisplayPort Standard v1.2 only.

YCRCB Enable

Select this check box to enable YCRCB-4:2:2 color space.

Enable Audio Option

Select this check box to enable generation of the core with two-channel audio support. A SPDIF core of the correct direction is also generated when this option is selected.

IEEE OUI

This Receiver Sink core option allows the user to preset the OUI register value before synthesis generation. The value defaults to Xilinx's OUI.

Vendor Specific DPCD Enable

The Receive Sink core check box to enable use of a vendor-specific DPCD area. Select this check box to enable this logic.

Parameterization

This section contains details about parameterization of the Source and Sink cores.

Source Core Parameterization

The user may specify a number of options through the Vivado tool, which will determine the presence of certain functions. It is advisable to disable any feature that is not needed in order to reduce resource utilization. Table 4-1 describes the parameterizable options.

Table 4-1: **Parameterizable Options**

Parameter	Default Value	Description
LANE_SUPPORT	4	{1, 2, 4} Indicates the maximum number of lanes to be supported for transmission. Note that unused lane support hardware will be removed from the design.
LINK_RATE	2.7	{1.62, 2.7, 5.4} Indicates the maximum link rate in Gb/s supported by the design.
SECONDARY_SUPPORT	0	Enables secondary channel logic to send Audio packets.
AUDIO_CHANNELS	2	Current version of IP supports 2-channel audio. The value is hard coded.
PROTOCOL_SELECTION	0	Protocol selection: • 0: DisplayPort 1.1a • 1: DisplayPort 1.2
MAX_BITS_PER_COLOR	8	{8, 10, 12, 16} Sets maximum bits-per-color support and optimizes IP accordingly.
QUAD_PIXEL_ENABLE	0	Enables support of quad-pixel video interface.
DUAL_PIXEL_ENABLE	1	Enables support of dual-pixel video interface.
YCRCB_ENABLE	1	Enables YCrCb 4:2:2 colorimetry support.
YONLY_ENABLE	0	Enables Y-Only colorimetry support

Sink Core Parameterization

The user may specify a number of options through the VIVADO tool, which will determine the presence of certain functions. Note that it is advisable to disable any feature that is not needed in order to reduce resource utilization. Table 4-2 shows the parametrization options.

Table 4-2: **Parameterizable Options**

Parameter	Default Value	Description
LANE_SUPPORT	4	{1, 2, 4} Indicates the maximum number of lanes to be supported for transmission. Note that unused lane support hardware will be removed from the design.
LINK RATE	2.7	{1.62, 2.7, 5.4 } Indicates the maximum link rate in Gb/s supported by the design.
SECONDARY_SUPPORT	0	Enables secondary channel logic to send Audio packets.
AUDIO_CHANNELS	2	Current version of IP supports 2-channel audio. The value is hard-coded.
PROTOCOL_SELECTION	0	Protocol selection: • 0: DisplayPort v1.1a • 1: DisplayPort v1.2
MAX_BITS_PER_COLOR	8	Sets maximum bits per color support and optimizes IP accordingly.
QUAD_PIXEL_ENABLE	0	Enables support of quad-pixel video interface.
DUAL_PIXEL_ENABLE	1	Enables support of dual-pixel video interface.
YCRCB_ENABLE	1	Enables YCrCb 4:2:2 colorimetry support.
YONLY_ENABLE	0	Enabled Y-Only colorimetry support.
IEEE_OUI	24'h000A35	{24-bit value} Indicates the user's OUI value
VENDOR_SPECIFIC	0	Enables DPCD space of vendor-specific fields in the Sink core.

Output Generation

The output files generated from the Xilinx Vivado software are placed in the project directory. For a full description of the generated files, see Directory and File Contents in Chapter 6.

Constraining the Core

This chapter defines the constraint requirements of the DisplayPort core. An example user constraints file (XDC) is provided in the implementation directory, which implements the constraints defined in this chapter.

When a Kintex®-7 is selected as the target device, the XDC will be generated for an XC7K325T-FFG900-2 device as an example. The example designs and XDCs can be retargeted for other devices. Information is provided in this chapter to indicate which constraints to modify when targeting devices other than those shown in the example designs.

Board Layout

For board layout concerns, refer to the VESA DisplayPort Standard specification [Ref 1]. For layout of the high-speed I/O lanes, refer to the appropriate section of the relative transceiver user guide. See References in Appendix C. Special consideration must be made for the AUX channel signals. Xilinx requires unidirectional LVDS signaling for Virtex-6 FPGAs. See I/O Standard and Placement.

Required Constraints

To operate the core at the highest performance rating, the following constraints must be present. Prorate these numbers if slower performance is desired.

```
create_clock -name lnk_clk_p -period 9.259 [get_ports lnk_clk_p]
create_clock -name axi_aclk  -period 20    [get_pins -hier *s_axi_aclk]
create_clock -name vid_clk   -period 7.4   [get_pins -hier *tx_vid_clk]
#-----------------------------------------------------------------
# Generated clocks
#-----------------------------------------------------------------
create_generated_clock -name lnk_clk -source [get_ports lnk_clk_p] -multiply_by 5
-divide_by 4 [get_pins -of_objects [get_cells -hier *ref_clk_out_bufg] -filter
{direction == out}]
#-----------------------------------------------------------------
# cross clock constraints
#-----------------------------------------------------------------
set_false_path -from [get_clocks axi_aclk] -to [get_clocks vid_clk]
set_false_path -from [get_clocks vid_clk]  -to [get_clocks axi_aclk]
set_false_path -from [get_clocks axi_aclk] -to [get_clocks lnk_clk]
set_false_path -from [get_clocks lnk_clk]  -to [get_clocks axi_aclk]
set_false_path -from [get_clocks vid_clk]  -to [get_clocks lnk_clk]
set_false_path -from [get_clocks lnk_clk]  -to [get_clocks vid_clk]
```

Device, Package, and Speed Grade Selections

Supported devices are listed in IP Facts, page 6.

Clock Frequencies

See Maximum Frequencies in Chapter 2 for more details about clock frequencies.

Clock Management

See Clocking in Chapter 3 for details about clock management.

Clock Placement

There are no clock placement constraints.

Banking

There are no banking constraints.

Transceiver Placement

Placement of the GT is board specific. For designs that target certain parts and families, the GT placement is set in the constraints file.

I/O Standard and Placement

This section contains details about I/O constraints.

AUX Channel

The VESA DisplayPort Standard [Ref 1] describes the AUX channel as a bidirectional LVDS signal. For 7 series designs, the core is a unidirectional LVDS or (LVDS25) and requires two pin pairs. The output AUX signal is 3-state controlled. Design the board to combine these signals outside of the FPGA.

For Kintex-7 and Artix-7 devices supporting HR IO banks, use the following constraints:

```
set_property IOSTANDARD LVDS_25   [get_ports aux_channel_in_p]
set_property IOSTANDARD LVDS_25   [get_ports aux_channel_in_n]
set_property IOSTANDARD LVDS_25   [get_ports aux_channel_out_p]
set_property IOSTANDARD LVDS_25   [get_ports aux_channel_out_n]
```

For Virtex-7 devices supporting HP IO banks, use the following constraints:

```
set_property IOSTANDARD LVDS   [get_ports aux_channel_in_p]
set_property IOSTANDARD LVDS   [get_ports aux_channel_in_n]
set_property IOSTANDARD LVDS   [get_ports aux_channel_out_p]
set_property IOSTANDARD LVDS   [get_ports aux_channel_out_n]
```

HPD

The HPD signal can operate in either a 3.3V or 2.5V I/O bank. By definition in the specification, it is a 3.3V signal. However, it is not uncommon to combine this signal with the AUX signals.

For Kintex-7 and Artix-7 devices supporting HR IO banks, use the following constraints:

```
set_property IOSTANDARD LVCMOS18  [get_ports hpd];
```

For Virtex-7 devices supporting HP IO banks, use the following constraints:

```
set_property IOSTANDARD LVCMOS25  [get_ports hpd];
```

High-Speed I/O

The four high-speed lanes operate in the LVDS (LVDS25) IO standard.

For Kintex-7 and Artix-7 devices supporting HR IO banks, use the following constraints:

```
set_property IOSTANDARD LVDS_25   [get_ports lnk_tx_lane_p]
set_property IOSTANDARD LVDS_25   [get_ports lnk_tx_lane_n]
```

For Virtex-7 devices supporting HP IO banks, use the following constraints:

```
set_property IOSTANDARD LVDS      [get_ports lnk_rx_lane_p]
set_property IOSTANDARD LVDS      [get_ports lnk_rx_lane_n]
```

Detailed Example Design

This chapter provides detailed information about the example design, including a description of files and the directory structure generated by the Xilinx Vivado tool, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

Directory and File Contents

The output files generated from the Xilinx Vivado software are placed in the project directory. The file output list may include some or all of the following files.

- <project directory>
  Top-level project directory of the Vivado tools.
  - <project directory>/<Project_name.data>
    Contains constraints and file set details.
  - <project directory>/<Project_name.src>
    Contains the sources like XCI, XDC, and TCL files as well as document files.
  - <project directory>/example_project
    Contains the source files necessary to create the DisplayPort example design.
    - <displayport_component name>.sim/
      A sub directory, `sim_1`, contains compilation scripts and RTL files of the IP in encrypted format.
    - <displayport_component name>.src/
      Contains a copy of core directory files `<project directory>/<Project_name.src>` and example design level constraints.
    - <displayport_component name>.data/
      Contains a copy core directory files `<project directory>/<Project_name.data>`.

<project directory>/<Project_name.src>/sources_1/ip/displayport_v3_2_0/

It contains templates for instantiation the core, sub-cores, example design, synth, XML, XDC and the XCI files.

Table 6-1: **Project Directory**

Name	Description
<project_dir>	
<displayport_component_name>.xci	Log file from Vivado software describing the options were used to generate the DisplayPort core. An XCI file is generated by the Vivado software for each core created in the current project directory. An XCI file can also be used as an input to the Vivado software.
<displayport_component_name>.xml	Contains the IPXACT representation of the IP.
<displayport_component_name>_ex.tcl	TCL script file used to generate DisplayPort Core.
<displayport_component_name>.veo <displayport_component_name>.vho	HDL template for instantiating the DisplayPort core.

<displayport_component name>/example_design

This directory contains the source files necessary to create the example design.

Table 6-2: **example_design Directory**

Name	Description
example_design	
<displayport_component_name>._exdes.v	Example design top-level file
<displayport_component_name>_exdes.xdc	Constraint file to the DisplayPort core.
<displayport_component_name>_[tx|rx]_fsm_cntrl.v	FSM control to the example design.
<displayport_component_name>_defines_[tx|rx].v	Defines to transmitter core

<displayport_component name>/implement

This directory contains the Synplify files.

Table 6-3: implement Directory

Name	Description
implement	
implement_synplify.bat	Batch file to create synplify project
synplify.prj	Synplify project file

<displayport_component name>/simulation

This directory contains the simulation model.

Table 6-4: **simulation Directory**

Name	Description
simulation	
displayport_v3_2_tb.v	This file contains the top-level DisplayPort simulation model. It instantiates the example design source files, as well as the core simulation netlist.

<displayport_component name>/simulation/functional

This directory contains simulation scripts and a waveform file.

Table 6-5: **functional Directory**

Name	Description
/simulation/functional	
simulate_mti.do	A ModelSim macro file that compiles the example design sources and the structural simulation models, then runs the functional simulation to completion.
simulate_ncsim.sh	An NCSim macro file that compiles the example design sources and the structural simulation models, and then runs the functional simulation to completion.
wave_mti.do	Sets up the wave file by organizing signals by interface.

Example Design

The following files describe the top-level example design for the DisplayPort cores.

```
<project_dir>/<displayport_component_name>/example_design/<component_name>_exdes.v
```

The top-level example design adds flip-flops to the user data interface. This allows the entire design to be synthesized and implemented in a target device to provide post place-and-route gate-level simulation.

Policy Maker

The following files describe the policy maker design for the DisplayPort cores:

Sink Core

```
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_rx_fsm_cntrl.v
```

Source Core

```
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_tx_fsm_cntrl.v
```

Each policy maker design contains a state machine, which connects to the processor interface. An instruction set has been stored in RAM, which may be modified as the user sees fit. The basic instruction set provided demonstrates the rudimentary procedure for setting up the cores.

A MicroBlaze™ processor-based Policy Maker design is available free of charge to purchasers of the DisplayPort core. Users may interchange the state machine with the software version, which is full of features and compliant to the specification.

EDID ROM

These fully functional Sink-only files demonstrate how to connect an EDID to the core.

```
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_iic_edid_rom.vhd

<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_iic_rom.vhd
```

Additionally, this EDID may be used in hardware. Adjust the register values as needed.

Demonstration Test Bench

The demonstration test bench is a simple Verilog program to exercise the example design and the cores. The following files describe the demonstration test bench.

Sink Core

```
<project_dir>/<displayport_component_name>/simulation/displayport_rx_v3_2_tb.v
```

The sink demonstration test bench performs the following tasks:

- Generates input clock signals
- Applies a reset to the example design
- Sets the lane count of the Sink core to 4 through the AUX channel
- Sets the bandwidth of the Sink core to 2.7 Gbps through the AUX channel
- Alerts the Sink core that training is beginning
- Sends training patterns 1 and 2 across the high-speed lanes
- Sets the power state value through the AUX channel

Source Core

```
<project_dir>/<displayport_component_name>/simulation/displayport_v3_2_tb.v
```

The source demonstration test bench performs the following tasks:

- Generates input clock signals
- Applies a reset to the example design
- Asserts HPD to the Source core
- Responds to AUX channel requests
- Drives video data on the user data interface

Implementation

The implementation script is a shell script that processes the example design through the Xilinx tool flow. It is located at:

```
<projectdirectory>/<Project_name.src>/sources_1/ip/< displayport_component_name >/
<displayport_component_name>.tcl
```

SECTION III: ISE DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Detailed Example Design

Customizing and Generating the Core

This chapter includes information on using Xilinx tools to customize and generate the core. The DisplayPort design consists of a Source (TX) and a Sink (RX) core. These cores are generated independently through the Xilinx CORE Generator software using a graphical user interface (GUI).

This chapter describes the GUI options used to generate and customize the cores. The Source and Sink cores are generated independently, and the user may choose to generate only one or both cores.

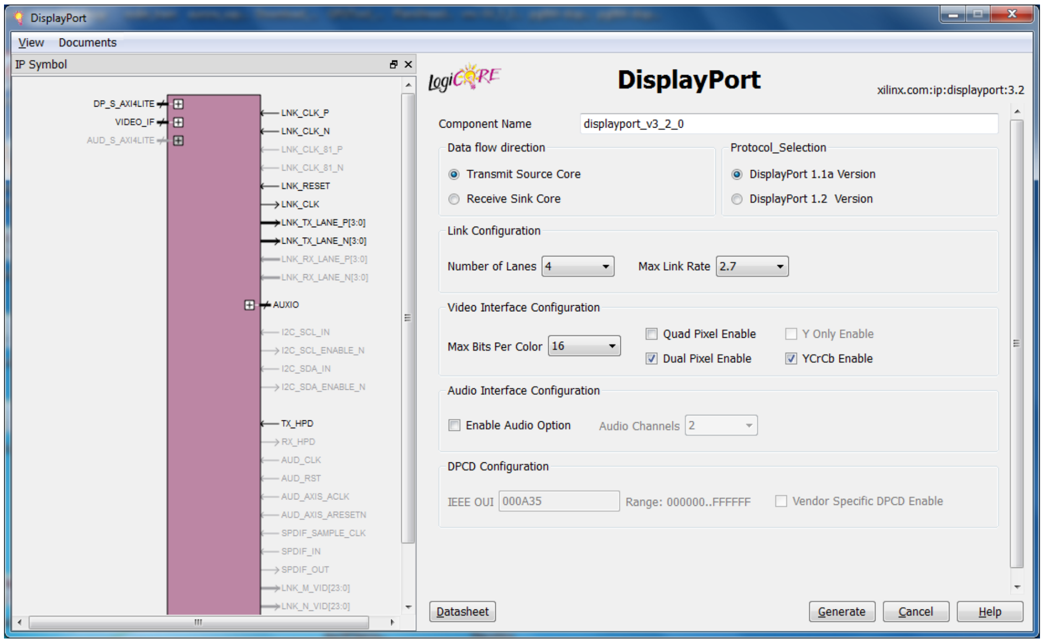
For assistance with starting and using the Xilinx CORE Generator software, see the documentation supplied with the Xilinx ISE software, including the Xilinx CORE Generator product page at: www.xilinx.com/tools/coregen.htm.

GUI

This section describes the Xilinx CORE Generator software configuration screen, provided to configure the DisplayPort design.

Main Screen

Figure 7-1 shows the DisplayPort CORE Generator main configuration screen. Descriptions of the GUI options on this screen are provided in the following text.



Component Name: displayport_v3_2_0

Data flow direction: [✓ Transmit Source Core] Receive Sink Core

Protocol_Selection: [✓ DisplayPort 1.1a Version] DisplayPort 1.2 Version

Link Configuration: Number of Lanes 4; Max Link Rate 2.7

Video Interface Configuration: Max Bits Per Color 16; ☐ Quad Pixel Enable; ☑ Dual Pixel Enable; ☐ Y Only Enable; ☑ YCrCb Enable

Audio Interface Configuration: ☐ Enable Audio Option; Audio Channels 2

DPCD Configuration: IEEE OUI 000A35; Range: 000000..FFFFFF; ☐ Vendor Specific DPCD Enable

Figure 7-1: **Main Configuration Screen**

Component Name

The Component Name is used as the name of the top-level wrapper file for the core. The underlying netlist still retains its original name. Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9, and "_". The name displayport_v3_2 is used as internal module name and should not be used for the component name. The default is displayport_v3_2_0.

Data Flow Direction

Select either the Sink (RX) or Source (TX) core with the Data Flow Direction radio button. If both directions are desired, the user must generate both a TX and RX core separately and combine these with the supplied wrapper files.

Protocol Selection

Select the protocol version for which the core is to be generated. DisplayPort v1.2 is supported on 7 series FPGA families and above.

Number of Lanes

Choose 1, 2, or 4 maximum lanes. Choose fewer lanes for a more optimized design. More lanes allow for higher overall bandwidth and higher resolutions.

Max Link Rate

Select the maximum link rate to be supported by the design.

Max Bits Per Color

Choose the maximum bits per color that the core supports. Six bits per color is supported by default.

Quad Pixel Enable

Select this check box to enable the four pixel-wide video interface. The quad pixel interface option is available for four-lane designs.

Dual Pixel Enable

Select this check box to enable the two pixel-wide video interface. The dual pixel interface option is available for two- and four-lane designs.

Y Only Enable

Select this check box to enable Y-Only color space logic. This option is available on cores using DisplayPort Standard v1.2 only.

YCRCB Enable

Select this check box to enable YCRCB-4:2:2 color space.

Enable Audio Option

Select this check box to enable generation of the core with two-channel audio support. A SPDIF core of the correct direction is also generated when this option is selected.

IEEE OUI

This Receiver Sink core option allows the user to preset the OUI register value before synthesis generation. The value defaults to Xilinx's OUI.

Vendor Specific DPCD Enable

The Receive Sink core check box to enable use of a vendor-specific DPCD area. Select this check box to enable this logic.

Output Generation

The output files generated from the Xilinx CORE Generator software are placed in the project directory. For a full description of the generated files, see Directory and File Contents in Chapter 9.

Source Core Parameterization

The user may specify a number of options through the CORE Generator tool, which will determine the presence of certain functions. Note that it is advisable to disable any feature that is not needed in order to reduce resource utilization. Table 7-1 described the parameterizable options.

Table 7-1: **Parameterizable Options**

Parameter	Default Value	Description
LANE_SUPPORT	4	{1, 2, 4} Indicates the maximum number of lanes to be supported for transmission. Note that unused lane support hardware will be removed from the design.
LINK_RATE	2.7	{1.62, 2.7, 5.4} Indicates the maximum link rate in Gb/s supported by the design.
SECONDARY_SUPPORT	0	Enables secondary channel logic to send Audio packets.
AUDIO_CHANNELS	2	Current version of IP supports 2-channel audio. The value is hard coded.
PROTOCOL_SELECTION	0	Protocol selection: • 0: DisplayPort 1.1a • 1: DisplayPort 1.2
MAX_BITS_PER_COLOR	8	{8, 10, 12, 16} Sets maximum bits-per-color support and optimizes IP accordingly.
QUAD_PIXEL_ENABLE	0	Enables support of quad-pixel video interface.
DUAL_PIXEL_ENABLE	1	Enables support of dual-pixel video interface.
YCRCB_ENABLE	1	Enables YCrCb 4:2:2 colorimetry support.
YONLY_ENABLE	0	Enables Y-Only colorimetry support

Sink Core Parameterization

The user may specify a number of options through the CORE Generator tool, which will determine the presence of certain functions. Note that it is advisable to disable any feature that is not needed in order to reduce resource utilization. Table 7-2 shows the parametrization options.

Table 7-2: **Parameterizable Options**

Parameter	Default Value	Description
LANE_SUPPORT	4	{1, 2, 4} Indicates the maximum number of lanes to be supported for transmission. Note that unused lane support hardware will be removed from the design.
LINK RATE	2.7	{1.62, 2.7, 5.4 } Indicates the maximum link rate in Gb/s supported by the design.
SECONDARY_SUPPORT	0	Enables secondary channel logic to send Audio packets.
AUDIO_CHANNELS	2	Current version of IP supports 2-channel audio. The value is hard-coded.
PROTOCOL_SELECTION	0	Protocol selection: • 0: DisplayPort v1.1a • 1: DisplayPort v1.2
MAX_BITS_PER_COLOR	8	Sets maximum bits per color support and optimizes IP accordingly.
QUAD_PIXEL_ENABLE	0	Enables support of quad-pixel video interface.
DUAL_PIXEL_ENABLE	1	Enables support of dual-pixel video interface.
YCRCB_ENABLE	1	Enables YCrCb 4:2:2 colorimetry support.
YONLY_ENABLE	0	Enabled Y-Only colorimetry support.
IEEE_OUI	24'h000A35	{24-bit value} Indicates the user's OUI value
VENDOR_SPECIFIC	0	Enables DPCD space of vendor-specific fields in the Sink core.

Constraining the Core

This chapter defines the constraint requirements of the DisplayPort core. An example user constraints file (UCF) is provided in the implementation directory, which implements the constraints defined in this chapter.

When a Spartan®-6 is selected as the target device, the UCF will be generated for an XC6SLX150T-FGG676-3 device as an example. The example designs and UCFs can be retargeted for other devices.

Information is provided in this chapter to indicate which constraints to modify when targeting devices other than those shown in the example designs.

Board Layout

For board layout concerns, refer to the *VESA DisplayPort Standard* specification [Ref 1]. For layout of the high-speed I/O lanes, refer to the appropriate section of the relative transceiver user guide. See References in Appendix C.

Special consideration must be made for the AUX channel signals. Xilinx requires unidirectional LVDS signaling for Virtex-6 FPGAs. See I/O Standard and Placement.

Required Constraints

To operate the core at the highest performance rating, the following constraints must be present. Prorate these numbers if slower performance is desired.

```
NET "s_axi_aclk" TNM_NET = s_axi_aclk;
TIMESPEC TS_s_axi_clk = PERIOD “s_axi_aclk” 7.408 ns HIGH 50 %;

NET "lnk_clk" TNM_NET = lnk_clk;
TIMESPEC TS_lnk_clk = PERIOD "lnk_clk" 7.408 ns HIGH 50 %;

NET "vid_clk" TNM_NET = vid_clk;
TIMESPEC TS_vid_clk = PERIOD "vid_clk" 7.408 ns HIGH 50 %;

# As per required sample rate. The below example shows for 192 KHz rate.
NET "spdif_sample_clk" TNM_NET = spdif_sample_clk;
```

```
TIMESPEC TS_spdif_sample_clk = PERIOD "spdif_sample_clk" 10 ns HIGH 50 %;

NET "aud_axis_aclk" TNM_NET = aud_axis_aclk;
TIMESPEC TS_aud_axis_aclk = PERIOD "aud_axis_aclk" 10 ns HIGH 50 %;

NET "aud_clk" TNM_NET = aud_clk;
TIMESPEC TS_ aud_clk = PERIOD “aud_clk” 10 ns HIGH 50 %;
```

Device, Package, and Speed Grade Selections

Supported devices are listed in IP Facts, page 6.

Clock Frequencies

See Maximum Frequencies in Chapter 2 for details about clock frequencies.

Clock Management

The core uses six clock domains:

- **lnk_clk.** Most of the core operates in this domain. This domain is based off of the lnk_clk_p/n. When the lanes are running at 2.7 Gbps, lnk_clk will operate at 135 MHz. When the lanes are running at 1.62 Gbps, lnk_clk will operate at 81 MHz.
- **vid_clk.** This is the primary user interface clock. It has been tested to run as fast as 135 MHz, which accommodates to a screen resolution of 2560x1600 when using two-wide pixels. See Selecting the Pixel Interface in Chapter 3 for more information on how to select the appropriate pixel interface.
- **s_axi_aclk.** This is the processor domain. It has been tested to run as fast as 135 MHz. The AUX clock domain is derived from this domain, but requires no additional constraints.
- **aud_clk**. This is the audio interface clock. The frequency will be equal to 512 x audio sample rate.
- **spdif_sample_clk**. This is used by SPDIF receiver to sample incoming traffic. This clock should be >= 512 x audio sample rate.
- **aud_axis_aclk**. This clock is used by the Audio streaming interface. This clock should be >= 512 x audio sample rate.

Table 8-1 shows the clock ranges.

Table 8-1: **Clock Ranges**

Clock Domain	Min	Max	Description
lnk_clk	81 MHz	135 MHz	Link clock
vid_clk	13.5 MHz	150 MHz	Video clock
s_axi_aclk	25 MHz	135 MHz	Host processor clock
aud_clk	16 MHz	100 MHz	Audio Clock (512 * Audio Sample Rate)
spdif_sample_clk	16 MHz	100 MHz	>= Audio Clock
aud_axis_aclk	16 MHz	100 MHz	>= Audio Clock

Clock Placement

The reference clocks for the GT are board specific. The pins locations of the reference clock is inferred by transceiver placement. With certain family/part based designs, the locations of the reference clocks are explicitly specified in the constraints.

Transceiver Placement

Placement of the GT is board specific. For designs that target certain parts and families, the GT placement is set in the constraints file.

I/O Standard and Placement

AUX Channel

The *VESA DisplayPort Standard* [Ref 1] describes the AUX channel as a bidirectional LVDS signal. For Virtex-6 FPGAs, the core has been designed as unidirectional LVDS_25, requiring two pin pairs. The output AUX signal is 3-state controlled. The board should be designed to combine these signals external to the FPGA. The UCF provides the following constraints for AUX:

```
NET "aux_rx_out_channel_p" IOSTANDARD = "LVDS_25";
NET "aux_rx_out_channel_n" IOSTANDARD = "LVDS_25";
NET "aux_rx_in_channel_p"   IOSTANDARD = "LVDS_25";
NET "aux_rx_in_channel_n"   IOSTANDARD = "LVDS_25";
```

Spartan-6 FPGAs offer an I/O standard explicitly for DisplayPort (called Display_Port). This is a bidirectional standard. The user can, but is not required to, combine the unidirectional pins and use this standard. The BUFDS instances are provided in the PHY wrapper file. In order to support the Display_Port standard, change the UCF to the following constraints:

```
NET "aux_rx_channel_p"      IOSTANDARD = "DISPLAY_PORT";
NET "aux_rx_channel_n"      IOSTANDARD = "DISPLAY_PORT";
```

Note: Do not use a bidirectional BLVDS or LVDS I/O standard.

HPD

The HPD signal can operate in either a 3.3V or 2.5V I/O bank. By definition in the specification, it is a 3.3V signal. However, it is not uncommon to combine this signal with the AUX signals. The UCF provides the following constraint:

```
NET "hpd" IOSTANDARD = "LVCMOS33";
```

For 2.5V operation:

```
NET "hpd" IOSTANDARD = "LVCMOS25";
```

High-Speed I/O

The four high-speed lanes operate in the LVDS_25 IO standard and should not be changed:

```
NET "lnk_rx_lane_p<0>" IOSTANDARD = "LVDS_25";
NET "lnk_rx_lane_p<1>" IOSTANDARD = "LVDS_25";
NET "lnk_rx_lane_p<2>" IOSTANDARD = "LVDS_25";
NET "lnk_rx_lane_p<3>" IOSTANDARD = "LVDS_25";
NET "lnk_rx_lane_n<0>" IOSTANDARD = "LVDS_25";
NET "lnk_rx_lane_n<1>" IOSTANDARD = "LVDS_25";
NET "lnk_rx_lane_n<2>" IOSTANDARD = "LVDS_25";
NET "lnk_rx_lane_n<3>" IOSTANDARD = "LVDS_25";
```

Detailed Example Design

This chapter details information about the example design delivered with the core.

Directory and File Contents

This chapter provides detailed information about the example design, including a description of files and the directory structure generated by the Xilinx CORE Generator, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

Directory and File Contents

The output files generated from the Xilinx CORE Generator software are placed in the project directory. The file output list may include some or all of the following files.

Directory Hierarchy

- **<project directory>**
  Top-level project directory for the CORE Generator software
  - <project directory>/<displayport_component name>
    Contains the DisplayPort release notes text file and netlists.
    - <displayport_component name>/doc
      Contains the DisplayPort solution PDF documentation.
    - <displayport_component name>/example_design
      Contains the source files necessary to create the DisplayPort example design.
    - <displayport_component name>/implement
      Contains the supporting files for synthesis and implementation of the DisplayPort example design.
      - **<displayport_component name>/implement/results**
        Contains the implementation results that are created when the implement scripts are run.
    - <displayport_component name>/simulation

Contains the test bench and other supporting source files used to create the DisplayPort simulation model.

<displayport_component name>/simulation/functional

Contains the scripts and define files for simulating the DisplayPort example design in ModelSim.

File Details

<project directory>

This is the top-level file. It contains templates for instantiation the core and the XCO file.

Table 9-1: **Project Directory**

Name	Description
<project_dir>	
<displayport_component_name>.xco	Log file from CORE Generator software describing which options were used to generate the DisplayPort core. An XCO file is generated by the CORE Generator software for each core that it creates in the current project directory. An XCO file can also be used as an input to the CORE Generator software.
<displayport_component_name>_flist.txt	A text file listing all of the output files produced when the customized DisplayPort core was generated in the CORE Generator software.
<displayport_component_name>.veo <displayport_component_name>.vho	The HDL template for instantiating the DisplayPort core.
<displayport_component_name>.v	Top level module of the generated DisplayPort core.
<displayport_component_name>.xise	ISE project to load the example design files into the ISE tool.

<project directory>/<displayport_component name>

This directory contains the netlist and structural simulation model.

Table 9-2: **Component Name Directory**

Name	Description
<project_dir>/<displayport_component_name>	
displayport_readme.txt	The DisplayPort core release notes text file.
<displayport_component_name>.ngc	The netlist for the DisplayPort core.
<displayport_component_name>.v[hd]	The structural simulation model for the DisplayPort core. It is used for functionally simulating the core.

Table 9-2: **Component Name Directory *(Cont'd)***

Name	Description
<spdif_rx_top>.ngc	SPDIF Receiver core netlist delivered when Audio is enabled for a Source Core.
<spdif_rx_top>.v[hd]	The structural simulation model for the SPDIF Receiver core netlist delivered when Audio is enabled for a Source Core.
<spdif_tx_top>.ngc	SPDIF Transmitter core netlist delivered when Audio is enabled for a Sink Core.
<spdif_tx_top>.v[hd]	The structural simulation model for the SPDIF Transmitter core netlist delivered when Audio is enabled for a Sink Core.

<displayport_component name>/doc

This directory contains the appropriate user guide and data sheet.

Table 9-3: **Doc Directory**

Name	Description
<project_dir>/<displayport_component_name>/doc	
displayport_ds802.pdf	The DisplayPort Data Sheet.
displayport_ug767.pdf	The DisplayPort User Guide.

<displayport_component name>/example_design

This directory contains the top-level wrapper files and a core controller. For Sink cores, this directory also includes an example EDID.

Table 9-4: **Example Design Directory**

Name	Description
<project_dir>/<displayport_component_name>/example_design	
<displayport_component_name>_common_pkg.vhd	Sink-only. Used with the example EDID design.
<displayport_component_name>_iic_edid_rom.vhd <displayport_component_name>_edid_iic.v edid_mem.list	Sink-only. EDID source code.
<displayport_component_name>_iic_rom.vhd	Sink-only. I2C controller.
<displayport_component_name>_rx_fsm_cntrl.v	Sink-only. Controller that autonomously sets up the Sink core for operation.
<displayport_component_name>_defines.v	Included file used throughout the example design.

Table 9-4: **Example Design Directory *(Cont'd)***

Name	Description
<displayport_component_name>_tx_fsm_cntrl.v	Source-only. Controller that is useful for implementing and simulating the example design. This is not a suitable policy maker for a full design.
<displayport_component_name>_exdes.v	Top-level example design. This wrapper pulls in all of the other core files in this directory.
<displayport_component_name>_exdes.ucf	Implementation constraints file. Contains period constraints as well as example pin placements.

<displayport_component name>/implement

This directory contains synthesis and implementation scripts as well as a constraints file.

Table 9-5: **Implementation Directory**

Name	Description
<project_dir>/<displayport_component_name>/implement	
implement.sh	Linux shell script that processes the example design through the Xilinx tool flow.
xst.prj	XST project file for the example design; it lists all of the source files to be synthesized.
implement.bat	PC script that processes the example design through the Xilinx tool flow.
xst.scr	XST script file for the example design that is used to synthesize the core; it is called from the implement script (implement.sh).
<displayport_component_name>.lso	Library search order file. Required for XST.
implement_synplify.sh	Linux shell script that processes the example design through the Synplify Synthesis and Xilinx tool flow.
implement_synplify.bat	PC script that processes the example design through the Synplify Synthesis and Xilinx tool flow.
synplify.prj	Synpilfy project file for example design.

<displayport_component name>/implement/results

This directory contains the implementation results that are created when the implement scripts are run.

<displayport_component name>/simulation

This directory contains a simple test bench.

Table 9-6: **Simulation Directory**

Name	Description
<project_dir>/<displayport_component_name>/simulation	
<displayport_component_name>_tb.v	This file contains the top-level DisplayPort simulation model. It instantiates the example design source files, as well as the core simulation netlist.

<displayport_component name>/simulation/functional

This directory contains simulation scripts and a waveform file.

Table 9-7: **Functional Directory**

Name	Description
<project_dir>/<displayport_component_name>/simulation/functional	
simulate_mti.do	A ModelSim macro file that compiles the example design sources and the structural simulation models, then runs the functional simulation to completion.
simulate_mti.sh simulate_mti.bat	Linux and PC simulation scripts for Mentor Graphics ModelSim.
simulate_isim.sh simulate_isim.bat simcmds.tcl	Linux and PC ISim simulator scripts and source TCL file.
simulate_ncsim.sh	An NCSim macro file that compiles the example design sources and the structural simulation models, and then runs the functional simulation to completion.
wave_ncsim.sv	NCSIM wave and run file.
wave_mti.do	Sets up the wave file by organizing signals by interface.

<displayport_component name>/src

This directory contains GTP instantiations as well as core wrapper files. GT wrappers are generated based on the device selection by the user. The PHY module is organized using the GENERATE statement to include a wrapper instance for all families.

Table 9-8: **Source Directory**

Name	Description
<project_dir>/<displayport_component_name>/src	
<displayport_v3_2>.v	Top-level module that instantiates the Link, PHY and optional SPDIF components.
<displayport_component_name>_txlink_top.v <displayport_component_name>_rxlink_top.v	Port list template for the core Link layer netlist of the source and sink.
<displayport_component_name>_tx_defs.v	Source-only. Defines the file and is used by the PHY.
<displayport_component_name>_rx_dpcd_defs.v <displayport_component_name>_rx_defs.v	Sink-only. Defines the file and is used by the PHY.
<displayport_component_name>_rx_phy.v	Sink-only. PHY wrapper. Contains Virtex®-6 and Spartan®-6 FPGA receive-only GTP instances as well as DCM instances.
<displayport_component_name>_tx_phy.v	Source-only. PHY wrapper. Contains Virtex-6 and Spartan-6 FPGA transmit-only GTP instances.
<displayport_component_name>_txrx.v	Top-level module to be used when combining the Source and Sink cores for a bridging application.
<displayport_component_name>_txrx_phy.v	PHY wrapper. Contains Virtex-6 and Spartan-6 FPGA bi-directional GT instances. Use this instance when you want to use the GT for both Sink and Source cores.
<displayport_component_name>_s6_gt_tile.v	Spartan-6 GTP instance. Generated from the wizard and used in the Spartan-6 wrapper module.
<displayport_component_name>_s6_gt_wrapper.v	Spartan-6 GTP wrapper generated from the wizard. This module is used in the PHY modules.
<displayport_component_name>_v6_gtx_wrapper_gtx.v	Virtex-6 GTX instance from the wizard. This module is used in the Virtex-6 wrapper module.
<displayport_component_name>_v6_gtx_wrapper.v	Virtex-6 GTX wrapper generated from the wizard. This module is used in the PHY modules.

Table 9-8: **Source Directory *(Cont'd)***

Name	Description
<displayport_component_name>_gt_7_series_wrapper_1_gt.v <displayport_component_name>_gt_7_series_wrapper_2_gt.v <displayport_component_name>_gt_7_series_wrapper_4_gt.v <displayport_component_name>_gt_a7_wrapper_1_gt.v <displayport_component_name>_gt_a7_wrapper_2_gt.v <displayport_component_name>_gt_a7_wrapper_4_gt.v	Kintex-7, Artix-7, and Virtex-7 device GT instance from wizard for 1-, 2- and 4 -lane designs respectively. This module is used in the PHY modules.
<displayport_component_name>_gt_7_series_wrapper_1.v <displayport_component_name>_gt_7_series_wrapper_2.v <displayport_component_name>_gt_7_series_wrapper_4.v <displayport_component_name>_gt_a7_wrapper_1.v <displayport_component_name>_gt_a7_wrapper_2.v <displayport_component_name>_gt_a7_wrapper_4.v	Kintex-7, Artix-7, and Virtex-7 GT wrapper from wizard for 1-, 2- and 4- lane designs respectively. This module is used in the PHY modules

Example Design

Policy Maker System Example Reference Design

Xilinx provides both Source and Sink DisplayPort controllers. The RTL-based Sink Policy Maker is suitable for core bring up and link maintenance. It enables the core, asserts Hot Plug Detect, and enables the Display Timing Generator. For users who desire finer tuning of the link, this Policy Maker is provided as open source.

Two Source Policy Maker options are available to the purchaser of the DisplayPort core. The first is a simple RTL-based controller. Use this version for a quick simulation or fast training against the Xilinx Sink DisplayPort core in hardware. As with the Sink version, this is provided as open source for user tuning. This version should not be used in applications that require compliance to the VESA specification.

For full compliance, Xilinx provides a second netlist-based Source Policy Maker. This version is ideal for hardware bring up and productization. It is based off of a MicroBlaze™ design running C code, which is also available to the purchaser of this core. For information about acquiring the source code, contact your local Xilinx sales office.

XAPP493, *Implementing a DisplayPort Source Policy Maker Using a MicroBlaze Embedded Processor*, describes the implementation of a DisplayPort Source Policy Maker targeted specifically for the Spartan-6 FPGA Consumer Video Kit (CVK). XAPP593, *Displayport Sink Reference Design*, describes the implementation of a DisplayPort Source/Sink Policy Maker targeted specifically for the Spartan-6 FPGA Consumer Video Kit (CVK1.0). Both documents can be found on xilinx.com.

Top Level Example Design

The following files describe the top-level example design for the DisplayPort cores.

```
<project_dir>/<displayport_component_name>/example_design/<component_name>_exdes.v
```

The top-level example design adds flip-flops to the user data interface. This allows the entire design to be synthesized and implemented in a target device to provide post place-and-route gate-level *simulation.*

Policy Maker

The following files describe the policy maker design for the DisplayPort cores:

Sink Core

```
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_rx_fsm_cntrl.v
```

Source Core

```
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_tx_fsm_cntrl.v
```

Each policy maker design contains a state machine, which connects to the processor interface. An instruction set has been stored in RAM, which may be modified as the user sees fit. The basic instruction set provided demonstrates the rudimentary procedure for setting up the cores.

A MicroBlaze™ processor-based Policy Maker design is available free of charge to purchasers of the DisplayPort core. Users may interchange the state machine with the software version, which is full of features and designed to the specification.

EDID ROM

These fully functional Sink-only files demonstrate how to connect an EDID to the core.

```
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_iic_edid_rom.vhd
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_iic_rom.vhd
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_edid_iic.v
<project_dir>/<displayport_component_name>/example_design/
<displayport_component_name>_edid_mem.list
```

Additionally, this EDID may be used in hardware. Adjust the register values as needed.

Demonstration Test Bench

The demonstration test bench is a simple Verilog program to exercise the example design and the cores. The following files describe the demonstration test bench.

Sink Core

```
<project_dir>/<displayport_component_name>/simulation/
<displayport_component_name>_tb.v
```

The sink demonstration test bench performs the following tasks:

- Generates input clock signals
- Applies a reset to the example design
- Sets the lane count of the Sink core through the AUX channel
- Sets the bandwidth of the Sink core through the AUX channel
- Alerts the Sink core that training is beginning
- Sends training patterns 1 and 2 across the high-speed lanes
- Sets the power state value through the AUX channel

Source Core

```
<project_dir>/<displayport_component_name>/simulation/
<displayport_component_name>_tb.v
```

The source demonstration test bench performs the following tasks:

- Generates input clock signals
- Applies a reset to the example design
- Asserts HPD to the Source core
- Responds to AUX channel requests
- Drives video data on the user data interface

Implementation

The implementation script is a shell script that processes the example design through the Xilinx tool flow. It is located at:

```
<project_dir>/<displayport_component_name>/implement/implement.sh
```

The PC implementation script issues equivalent instructions and is located at :

```
<project_dir>/<displayport_component_name>/implement/implement.bat
```

The implement script performs the following steps:

- Synthesizes the HDL example design files using XST
- Runs Ngdbuild to consolidate the core netlist and the example design netlist into the NGD file containing the entire design
- Maps the design to the target technology
- Place-and-routes the design on the target device
- Performs static timing analysis on the routed design using Timing Analyzer (TRCE)
- Generates a bitstream
- Enables Netgen to run on the routed design to generate a VHDL or Verilog netlist (as appropriate for the Design Entry project setting) and timing information in the form of SDF files

The Xilinx tool flow generates several output and report files. These are saved in the following directory which is created by the implement script:

```
<project_dir>/<displayport_component_name>/implement/results
```

Simulation

Xilinx provides both a ModelSim, ISim, and NCSim script that automates the simulation of the test bench. They are available from:

```
<project_dir>/<displayport_component_name>/simulation/functional/
```

The test script performs the following tasks:

- Compiles the structural UniSim simulation model
- Compiles HDL Example Design source code
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds signals of interest (`wave_mti.do`, or `wave_ncsim.sv`)
- Runs the simulation to completion

SECTION IV: APPENDICES

Verification, Compliance, and Interoperability

Migrating

Debugging

Additional Resources

Verification, Compliance, and Interoperability

The DisplayPort cores have been verified with functional simulation and extensive hardware testing for v1.1a and some hardware testing for v1.2. The Source core on the Spartan- 6 device CVK1.0 board is compliant to the DisplayPort v1.1a specification. The Sink core is not VESA complaint due to a known jitter tolerance issue.

Simulation

A parameterizable transaction-based test bench was used to test the core. Broad protocol and implementation-specific coverage were used to fully verify the cores. The tests included the following:

- Full I2C operation over the AUX channel
- Bandwidth and performance tests
- Main link stress tests
- Processor interface register read and write accesses
- Scramble/Descramble quality checks
- Video and Audio data integrity checks

Hardware Testing

The DisplayPort cores have been validated using a Spartan-6 FPGA Consumer Video Kit (CVK 1.0) from TED. The hardware has been tested against Quantum Data's 882 Test Instrument (DisplayPort v1.1a). The cores were also tested against established, certified products for interoperability.

Migrating

This appendix describes migrating from older versions of the IP (v2.3) to the current IP release (v3.1 or 3.2).

Note: For details on migration from the ISE Design Suite to the Vivado Design Suite, see UG911, *Vivado Design Suite Migration Methodology Guide*.

Functionality Changes

The following optional features were changed:

- Optional audio support was added. For more details see the Audio Management in Chapter 3.
- Optional vendor specific DPCD area was added in Sink core. See Sink Core in Chapter 2 for details.

Source Core: Programming Model Changes

- Programming value of Register USER_DATA_COUNT_PER_LANE (0x1BC) was changed.
- PHY configuration registers from address 0x23C to 0x258 added to program POST and PRE-CURSOR values of GT.
- CORE_ID (0x0FC) register values have been updated. New VERSION_ID (0x0F8) Register added.
- TX_USER_FIFO_OVERFLOW (0x110) added to indicate FIFO status.

Sink Core: Programming Model Changes

- Within DTG_ENABLE (0x00c), the DMA_MODE programming bit functionality was removed. By default, the Sink core works on a higher-rate video clock. The timing mode is no longer supported.
- The MSA_DELAY_BYPASS (0x018) register has been renamed to MISC_CONTROL. The functionality of the bits remain the same.
- The CORE_ID (0x0FC) register values have been updated. New VERSION_ID (0x0F8) register added.

Additional Resources

Xilinx Resources

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

www.xilinx.com/support.

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

www.xilinx.com/company/terms.htm.

References

These documents provide supplemental material useful with this user guide:

1. VESA *DisplayPort Standard v1.1a*, January 11, 2008.
2. VESA *DisplayPort Standard v1.2*, December 22, 2009.
3. UG386, *Spartan-6 FPGA GTP Transceivers User Guide*.
4. UG371, *Virtex-6 FPGA GTH Transceivers User Guide*.
5. UG366, *Virtex-6 FPGA GTX Transceivers Advance Product Specification*.
6. Spartan-6 FPGA Consumer Video Kit
7. XAPP493, *Implementing a DisplayPort Source Policy Maker Using a MicroBlaze Embedded Processor.*
8. High-bandwidth Digital Content Protection System v1.3 Amendment for DisplayPort, v1.0
9. AMBA AXI Protocol, v2.0
10. UG476, *7 Series FPGAs GTX Transceivers User Guide*
11. UG761, *Xilinx 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

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/25/12	1.0	Initial Xilinx release as a Product Guide. Replaces DS802, *LogiCORE IP DisplayPort Data Sheet* and UG767, *LogiCORE IP DisplayPort User Guide*. Added Control PHY Power Down register. Added Artix-7 device support.

Notice of Disclaimer