DisplayPort TX Subsystem v2.0

Product Guide

Vivado Design Suite

PG199 July 14, 2017

XILINX
ALL PROGRAMMABLE™

Table of Contents

Appendix B: Application Software Development

Appendix C: Additional Resources and Legal Notices

Introduction

DisplayPort TX Subsystem implements functionality of a video source as defined by the Video Electronics Standards Association (VESA)'s DisplayPort standard v1.2a and supports driving resolutions of up to Ultra HD (UHD) at 60 fps. The Xilinx® DisplayPort subsystems provide highly integrated but straightforward IP blocks requiring very little customization by the user.

Features

- Support for DisplayPort Source (TX) transmissions.
- Supports multi-stream transport (MST) and single stream transport (SST) at UHD at 60 fps
- Dynamic link rate support (1.62/2.7/5.4 Gb/s)
- Dynamic support of 6, 8, 10, 12, or 16 bits per component (BPC).
- Dynamic support of RGB/YCbCr444/ YCbCr422/Y_Only color formats.
- Wide screen support with internal split of up to two streams of the same resolution in streaming video interface mode.
- Support 32 or 16-bit Video PHY (GT) Interface
- Supports 2 to 8 channel Audio.
- Supports HDCP 1.3 encryption.
- Supports native or streaming video input interface.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)(2)	UltraScale+™ Families (GTHE4) Kintex® UltraScale™ (GTHE3) Virtex® UltraScale (GTHE3) 7 series (GTXE2) 7 series (GTHE2) (Discontinued in Vivado 2017.3) Artix®-7 (GTPE2) (Discontinued in Vivado 2017.4)
Supported User Interfaces	AXI4-Stream, AXI4-Lite
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Hierarchical subsystem packaged with DisplayPort TX core and other IP cores
Example Design	N/A
Test Bench	N/A
Constraints File	IP cores delivered with XDC files
Simulation Model	N/A
Supported S/W Driver	Standalone
Tested Design Flows(3)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. For HDCP: UltraScale/UltraScale+ supports up to 5.4 Gb/s, Kintex-7/Virtex-7 (-1 speed grade supports up to 2.7 Gb/s, -2/ -3 supports up to 5.4 Gb/s), and Artix-7 is not supported.
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 features, licensing, and standards. The DisplayPort TX Subsystem is a full feature, hierarchically packaged subsystem with a DisplayPort Transmit (TX) core ready to use in applications in large video systems.

Feature Summary

- UHD up to 60 fps supports multi-stream transport (MST) and single stream transport (SST) modes.
- Dynamic support of different bits per color (6, 8, 10, 12 or 16) and line rates.
- Dynamic support of RGB/YCbCr444/ YCbCr422/Y_Only color formats.
- Support optional HDCP 1.3 Controller.
- Support for native and streaming video input interface.

Unsupported Features

- Audio is not supported in MST mode.
- In-band stereo is not supported.
- HDCP is not supported in MST mode.
- HDCP 2.x is not supported.
- Video Streaming interface is not scalable with dynamic pixel mode selection.
- Dual-pixel splitter is not supported in native video mode.

Licensing and Ordering

License Type

This subsystem requires a license for the DisplayPort Transmit core, which is provided under the terms of the Xilinx Core License Agreement. The subsystem is shipped as part of the Vivado® Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. To generate a full license, visit the product licensing web page. Evaluation licenses and hardware timeout licenses might be available for this core or subsystem. Contact your local Xilinx sales representative for information about pricing and availability.

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

Information about 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.

TIP: To verify that you need a license, check the "License" column of the IP Catalog. "Included" means that a license is included with the Vivado Design Suite; "Purchase" means that you have to purchase a license to use the core or subsystem.

License Checkers

If the IP requires a license key, the key must be verified. The Vivado design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Product Specification

This chapter contains a high-level overview of the core as well as performance and port details.

Overview

The DisplayPort TX Subsystem, in both streaming and native interface, operates in the following video modes:

- Single stream transport (SST)
- Multi-stream transport (MST)

Streaming Video Interface

In the SST mode, the subsystem is packaged with three subcores: DisplayPort Transmit core, Video Timing Controller (VTC) and DP AXI4-Stream to Video Bridge. In the SST mode, the TX subsystem also includes optional HDCP controller for encryption and AXI Timer as a helper core for HDCP functionality.

Because the DisplayPort TX Subsystem is hierarchically packaged, you select the parameters and the subsystem creates the required hardware. Figure 2-1 shows the architecture of the subsystem assuming MST with four streams.

The subsystem includes a multi-pixel AXI4-Stream Video Protocol interface. The DisplayPort TX Subsystem outputs the video using the DisplayPort v1.2 protocol. The DisplayPort TX Subsystem works in conjunction with Video PHY Controller configured for the DP protocol.

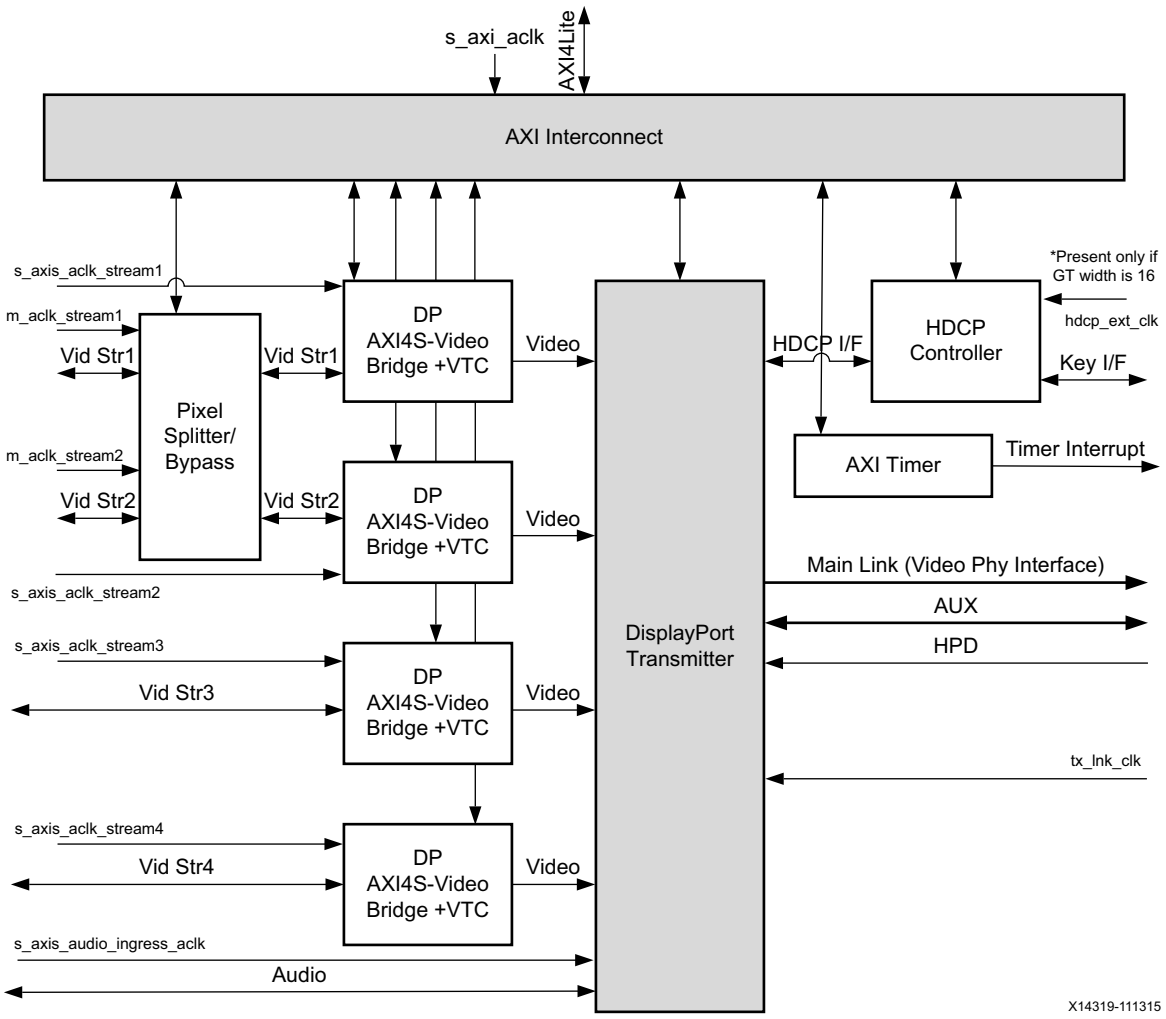

Figure 2-1: **DisplayPort TX Subsystem Streaming Video Interface Block Diagram**

In the MST mode, the subsystem has four subcores: Dual Splitter, DisplayPort AXI4-Stream to Video Bridge, Video Timing Controller and DisplayPort Transmitter core.

Native Video Interface

In the SST mode with the Native interface enabled, the subsystem is by default packaged with only one subcore: DisplayPort Transmit core. In the SST mode, the TX subsystem also includes option to enable the HDCP controller for encryption and AXI Timer as a helper core for HDCP functionality.

Figure 2-2 shows the architecture of the subsystem assuming MST with four native video streams. The subsystem includes a multi-pixel Native Video Protocol interface. The DisplayPort TX subsystem outputs the video using the DisplayPort v1.2 protocol, works in conjunction with Video PHY Controller configured for the DP protocol.

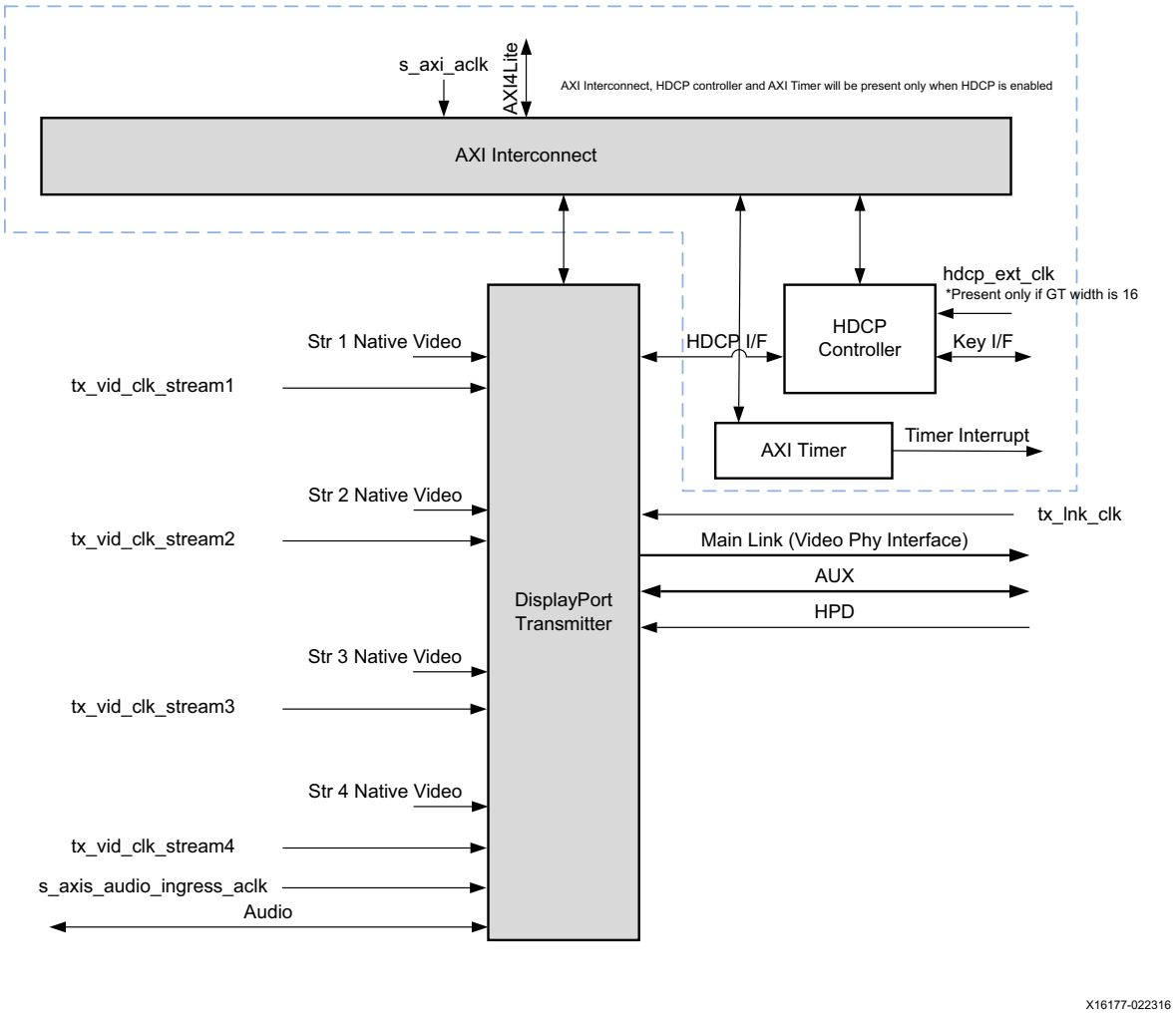

Figure 2-2: **DisplayPort TX Subsystem Native Video Block Diagram**

DisplayPort Dual Splitter

The Dual Splitter is used to vertically split the frame to support MST with two streams, as shown in Figure 2-3. Despite the frames being split, you will see this as one frame. The Dual Splitter has a buffer to hold the data for up to one and a half scan lines.

Note: The Dual Splitter is present only when MST is enabled in streaming interface mode. While using the Dual Splitter, ensure that the unused input video streams are grounded.

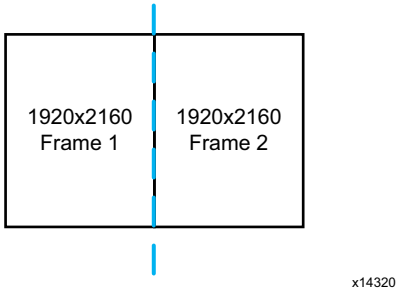

Figure 2-3: **Vertically Split Frame**

Splitter Interface

The splitter input and output are video over AXI4-Stream interface. Figure 2-4 shows the timing of this interface.

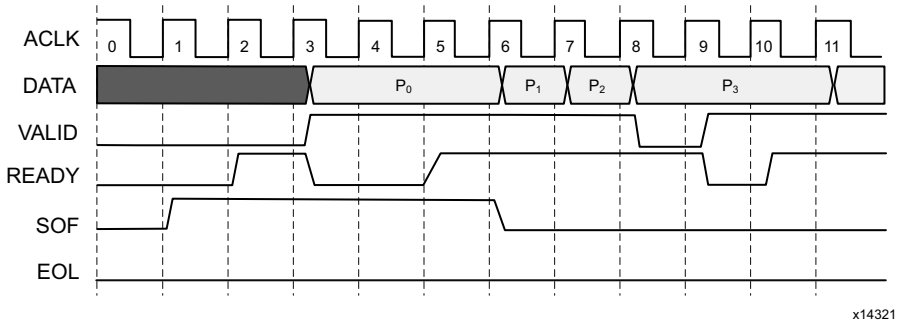

Figure 2-4: **Video over AXI4-Stream Interface Timing**

Based on the mode, the Core Control register (CORE_CONTROL_REG) of the Dual Splitter must be configured for input and output samples per clock. See Dual Splitter Registers for a description of CORE_CONTROL_REG.

DisplayPort AXI4-Stream to Video Bridge

The DisplayPort AXI4-Stream to Video Bridge maps the video over the AXI4-Stream interface to native video format as required by the DisplayPort Transmit IP core. The bridge uses the Xilinx AXI4 to Video Out core to convert the format between AXI4-Stream to DisplayPort native video.

For details about the Video Out Bridge, see the *AXI4-Stream to Video Out Product Guide* (PG044) [Ref 11].

For details about the video over AXI4-Stream, see the *AXI Reference Guide* (UG1037) [Ref 9].

The *DisplayPort Product Guide* (PG064) [Ref 10] contains information about the DisplayPort input video format.

The receive side of the bridge is Video over AXI4-Stream. For more details, see Port Descriptions.

In MST mode, there are N number of bridges in the subsystem, where N = the number of AXI4-Stream inputs to the subsystem.

Pixel Mapping on Streaming Interface

By default, the pixel mode is equal to the lane count during subsystem generation.

```
Pixel_Width = MAX_BPC x 3

Interface Width = Pixel Width x LANE_COUNT
```

For example, if the system is generated using 4 lanes with `MAX_BPC` of 16, the data width will be 16x4x3 which equals to 192.

MAX_BPC	LANES	PIXEL WIDTH	INTERFACE	VIDEO BPC	Pixel 3			Pixel2			Pixel 1			Pixel 0		
					B	G	R	B	G	R	B	G	R	B	G	R
16	4	48	192	8	191:184	175:168	159:152	143:136	127:120	111:104	95:88	79:72	63:56	47:40	31:24	15:8
16	2	48	96	8							95:88	79:72	63:56	47:40	31:24	15:8
12	4	36	144	10	143:134	131:122	119:110	107:98	95:86	83:74	71:62	59:50	47:38	35:26	23:14	11:2
10	4	30	120	10	119:110	109:100	99:90	89:80	79:70	69:60	59:50	49:40	39:30	29:20	19:10	9:0
8	2	24	48	8							47:40	39:32	31:24	23:16	15:8	7:0

Figure 2-5: **Pixel Mapping Examples on Streaming Interface**

Video Timing Controller

The Xilinx Video Timing Controller is used for generation of video timing. Video Timing Controller is required when the subsystem is configured in Streaming interface mode. For details on this core, see the *Video Timing Controller Product Guide* (PG016) [Ref 12].

IMPORTANT: *You must program proper front porch and back porch blanking period generation.*

DisplayPort Transmit

The DisplayPort Transmit core contains the following components as shown in Figure 2-6:

- **Main Link**: Provides 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.

For more details, see the *DisplayPort Product Guide* (PG064) [Ref 10].

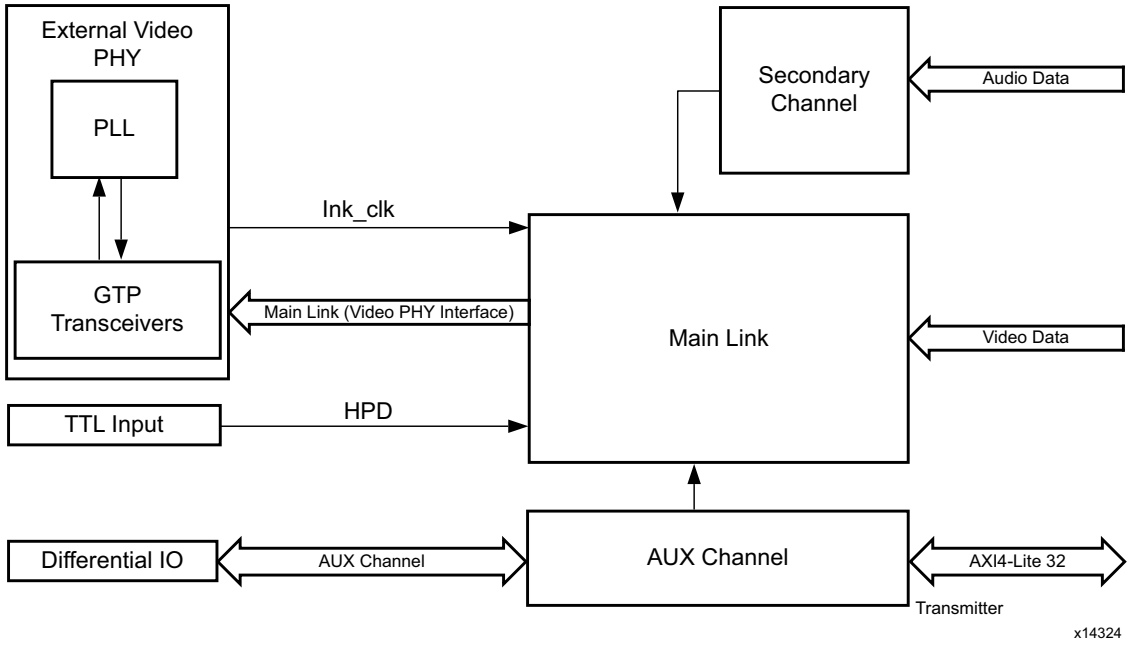

Figure 2-6: **DisplayPort Transmit Core Block Diagram**

AXI Interconnect

The subsystem uses Xilinx AXI Interconnect IP core, as a crossbar which contains an AXI4-Lite interface. Figure 2-7 shows the AXI slave structure within the DisplayPort TX Subsystem.

Note: For MST with *N* streams, there are *N* Video Timing Controllers. See Address Map Example in Chapter 3.

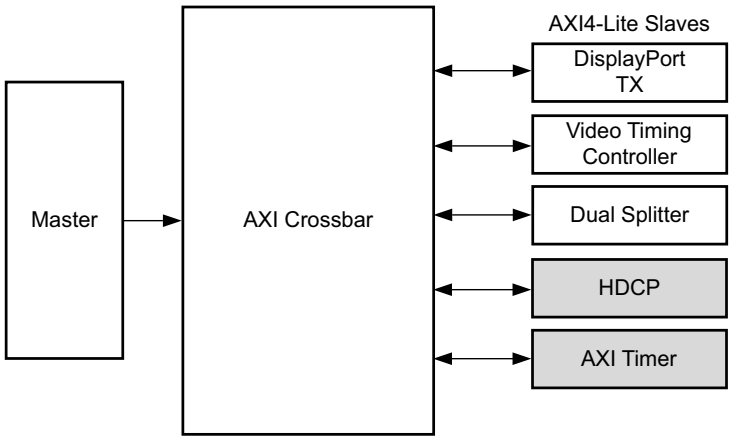

Figure 2-7: **AXI4-Lite Interconnect within DisplayPort TX Subsystem**

Note: Video Timing Controller and Dual splitter are present only when subsystem is generated in streaming interface mode.

HDCP Controller

The HDCP v1.3 protocol specifies a secure method of transmitting audiovisual content. Further, the audiovisual content can be transmitted over a DisplayPort interface. HDCP Controller is used for data encryption along with DisplayPort transmit IP in DisplayPort TX subsystem.

Figure 2-8 shows the DisplayPort TX Subsystem with HDCP controller. For more details on HDCP, see the *HDCP Controller Product Guide* (PG224) [Ref 13].

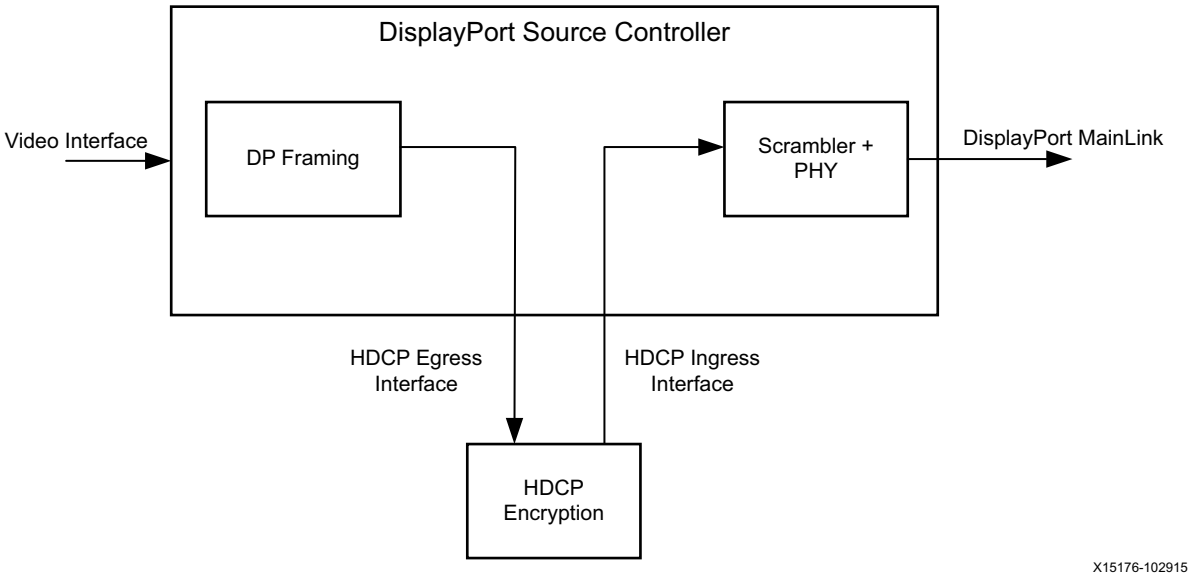

Figure 2-8: **DisplayPort TX with HDCP Controller**

AXI Timer

A 32-bit AXI Timer is used in the DisplayPort TX subsystem when the HDCP controller is enabled for encryption. The AXI Timer can be accessed through an AXI4 master interface for basic timer functionality in the system.

Standards

The DisplayPort TX Subsystem is compatible with the DisplayPort v1.2 Standard, HDCP v1.3 standard, as well as the AXI4-Lite and AXI4-Stream interfaces.

IMPORTANT: *Xilinx DisplayPort subsystems have passed compliance certification. If you are interested in accessing the compliance report or seeking guidance for the compliance certification of your products, contact your local Xilinx sales representative.*

Resource Utilization

For details about Resource Utilization, visit Performance and Resource Utilization.

Port Descriptions

The DisplayPort TX Subsystem ports are described in Table 2-1.

Table 2-1: **DisplayPort TX Subsystem Ports**

Signal Name	Direction from Core	Description
AXI4-Lite Interface		
s_axi_aclk	Input	AXI Bus Clock.
s_axi_aresetn	Input	AXI Reset. Active-Low.
s_axi_awaddr[18:0]	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[18:0]	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.
AXI4-Stream Interface (Enabled when the streaming interface is selected)		
s_axis_aclk_stream1	Input	AXI4-Stream clock.
s_axis_aresetn_stream1	Input	AXI4-Stream reset. Active-Low.

Table 2-1: **DisplayPort TX Subsystem Ports *(Cont'd)***

Signal Name	Direction from Core	Description
s_axis_video_stream1_tdata[191:0]	Input	Video data input.
s_axis_video_stream1_tlast	Input	Video end of line.
s_axis_video_stream1_tready	Output	AXI4-Stream tready output.
s_axis_video_stream1_tuser	Input	Video start of frame.
s_axis_video_stream1_tvalid	Input	Video valid.
Native Video Interface (Enabled when native video is selected)		
tx_video_stream1_tx_vid_vsync	Input	Vertical sync pulse. Active on the rising edge.
tx_video_stream1_tx_vid_hsync	Input	Horizontal sync pulse. Active on the rising edge
tx_video_stream1_tx_vid_enable	Input	User data video enable.
tx_video_stream1_tx_vid_pixel0[47:0]	Input	Video data
tx_video_stream1_tx_vid_pixel1[47:0]	Input	Video data
tx_video_stream1_tx_vid_pixel2[47:0]	Input	Video data
tx_video_stream1_tx_vid_pixel3[47:0]	Input	Video data
tx_video_stream1_tx_vid_oddeven	Input	Odd/even field select. Indicates an odd (1) or even (0) field polarity.
MST Stream (*n* = stream number 2 to 4) ***Note:*** See Clocking in Chapter 3 for the clock values.		
s_axis_aclk_stream*n*	Input	MST stream clock.
s_axis_aresetn_stream*n*	Input	MST stream reset. Active-Low.
s_axis_video_stream*n*_tdata[191:0]	Input	MST stream video data input.
s_axis_video_stream*n*_tlast	Input	MST stream video end of line.
s_axis_video_stream*n*_tready	Output	MST stream input ready.
s_axis_video_stream*n*_tuser	Input	MST stream video start of frame.
s_axis_video_stream*n*_tvalid	Input	MST stream video valid.
m_aclk_stream1	Input	Video pipe clock for stream1. Used in MST configuration.
m_aresetn_stream1	Input	Active-Low video pipe reset for stream 1. Used in MST configuration.
m_aclk_stream2	Input	Video pipe clock for stream 2. Used in MST configuration.
m_aresetn_stream2	Input	Active-Low video pipe reset for stream 2. Used in MST configuration.
tx_vid_clk_stream*n*	Input	User data clock for MST stream *n*.
tx_vid_rst_stream*n*	Input	Active-High user video reset.

Table 2-1: **DisplayPort TX Subsystem Ports *(Cont'd)***

Signal Name	Direction from Core	Description
tx_video_stream*n*_tx_vid_vsync	Input	Vertical sync pulse. Active on the rising edge.
tx_video_stream*n*_tx_vid_hsync	Input	Horizontal sync pulse. Active on the rising edge
tx_video_stream*n*_tx_vid_enable	Input	User data video enable.
tx_video_stream*n*_tx_vid_pixel0[47:0]	Input	Video data
tx_video_stream*n*_tx_vid_pixel1[47:0]	Input	Video data
tx_video_stream*n*_tx_vid_pixel2[47:0]	Input	Video data
tx_video_stream*n*_tx_vid_pixel3[47:0]	Input	Video data
tx_video_stream*n*_tx_vid_oddeven	Input	Odd/even field select. Indicates an odd (1) or even (0) field polarity.
User Ports		
tx_vid_clk_stream1	Input	User video clock.
tx_vid_rst_stream1	Input	User video reset. Active-High.
tx_hpd	Input	Hot-plug detect signal to TX from RX.
Audio Streaming Interface		
s_axis_audio_ingress_aclk	Input	AXI4-Stream clock.
s_axis_audio_ingress_aresetn	Input	Active-Low reset.
s_axis_audio_ingress_tdata[31:0]	Input	Streaming data input. • [3:0] - Preamble Code ◦ 4'b0001: Subframe1/ Start of audio block ◦ 4'b0010: Subframe 1 ◦ 4'b0011: Subframe 2 • [27:4] - Audio Sample Word • [28] - Validity Bit (V) • [29] - User Bit (U) • [30] - Channel Status (C) • [31] - Parity (P)
s_axis_audio_ingress_tid[7:0]	Input	• [3:0] - Audio Channel ID • [7:4] - Audio Packet Stream ID
s_axis_audio_ingress_tvalid	Input	Valid indicator for audio data from master.
s_axis_audio_ingress_tready	Output	Ready indicator from DisplayPort source.
External Video PHY Sideband status Interface		
s_axis_phy_tx_sb_status_tdata[7:0]	Output	Sideband status to Video PHY

Table 2-1: **DisplayPort TX Subsystem Ports *(Cont'd)***

Signal Name	Direction from Core	Description
s_axis_phy_tx_sb_status_tready	Input	Sideband status ready input from Video PHY
s_axis_phy_tx_sb_status_tvalid	Output	Sideband status data valid to Video PHY
External Video PHY clock Interface		
tx_lnk_clk	Input	Link clock input from external Video PHY
External Video PHY Lane n [n = 0 to Lane_Count-1] Interface		
m_axis_lnk_tx_lanen_tdata[31:0]	Output	Lanen Data to External Video PHY
m_axis_lnk_tx_lanen_tvalid	Output	Lanen Data Valid to External Video PHY
m_axis_lnk_tx_lanen_tready	Input	Lanen Data Ready from External Video PHY
m_axis_lnk_tx_lanen_tuser[11:0]	Output	Lanen User data out to External Video PHY
HDCP Key Interface		
hdcp_ext_clk	Input	HDCP external clock (enabled when HDCP is selected with 16-bit GT interface)
hdcp_key_aclk	Input	Key clock
hdcp_key_aresetn	Input	Key Interface reset. Active low
hdcp_key_tdata[63:0]	Input	AXI4-Stream Key Tdata
hdcp_key_last	Input	AXI4-Stream Key Tlast
hdcp_key_tready	Output	AXI4-Stream Key Tready
hdcp_key_tuser[7:0]	Input	AXI4-Stream Key TUSER. KMB should send the Key number from 0 to 41. 0 corresponds to KSV and 1 to 40 are the HDCP Keys count.
hdcp_key_tvalid	Input	AXI4-Stream Key TValid
reg_key_sel[2:0]	Output	To select the one of the eight sets of 40 keys.
start_key_transmit	Output	An Active-High pulse that is used to start key transmit.
AUX Signals		
aux_tx_io_n	Output	Negative polarity AUX Manchester-II data.
aux_tx_io_p	Output	Positive polarity AUX Manchester-II data.

Table 2-1: **DisplayPort TX Subsystem Ports *(Cont'd)***

Signal Name	Direction from Core	Description
aux_tx_channel_in_p	Input	Positive polarity AUX channel input. Valid when AUX IO Type is unidirectional
aux_tx_channel_in_n	Input	Negative polarity AUX channel input. Valid when AUX IO Type is unidirectional
aux_tx_channel_out_p	Output	Positive polarity AUX channel Output. Valid when AUX IO Type is unidirectional
aux_tx_channel_out_n	Output	Negative Polarity AUX channel output. Valid when AUX IO Type is unidirectional
aux_tx_data_out	Output	AUX data out. Valid when AUX IO buffer location is external
aux_tx_data_in	Input	AUX data input. Valid when AUX IO buffer location is external
aux_tx_data_en_out_n	Output	AUX data output enable. Active low. Valid only when AUX IO buffer location is external
Interrupt Interface		
dptxss_dp_irq	Output	DisplayPort TX IP interrupt out
dptxss_hdcp_irq	Output	HDCP IP interrupt out
dptxss_timer_irq	Output	AXI Timer IP interrupt output valid only when HDCP is enabled

Register Space

This section details registers available in the DisplayPort TX Subsystem. The address map is split into following regions:

- Dual Splitter
- VTC 0 (Up to 3 for 4 streams)
- DisplayPort Transmit
- HDCP Controller
- AXI Timer

TIP: *For details about accessing these registers, see Programming Sequence in Chapter 3.*

Dual Splitter Registers

Table 2-2 defines the Dual Splitter registers.

Table 2-2: **Dual Splitter Register Definitions**

Offset	Register	Access	Default Value	Definition
0x0000	GENR_CONTROL_REG	R/W	0x2	• [0] – Enables the splitter. • [1] – Register update. • [31] - Soft reset bit. Other registers can be programed by writing a value 2 to this register. At the end of programing set the register to 3.
0x0008	GENR_ERROR_REG	R/W	0x0	• [0] – Slave EOL early. • [1] – Slave EOL late. • [2] – Slave SOF early. • [3] – Slave SOF late.
0x000C	IRQ_ENABLE	R/W	0	[0] – Interrupt based on the error conditions.
0x0020	TIME_CONTROL REG(1)	R/W	0x0870_0F00	Contains the input image size: • [15:0] - Height. • [15:0] - Width. ***Note:*** For UHD @60 frame split mode, HRES must be programmed to actual HRES/4.
0x0100	CORE_CONTROL_REG	R/W	0x00_01_01_01	For UHD @ 60 frame split mode, this register can be programmed to 0x020404. For all other modes, it can be 0x10404. • [7:0] – Input number of samples per clock. • [15:8] – Output number of samples per clock. • [23:16] – Number of image segments. • [31:24] – Number of samples overlapping the segments. Should be programmed to 0 because the subsystem supports two frames without overlap.

Notes:

1. Height refers to VRES and the WIDTH refers to HRES.

Video Timing Controller Registers

For details about the Video Timing Controller (VTC) registers, see the *Video Timing Controller Product Guide* (PG016) [Ref 12].

DisplayPort Registers

The DisplayPort Configuration Data is implemented as a set of distributed registers which can 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 might change while being read from the configuration space, two scenarios might exist. In the case of single bits, either the new value or the old value is read as valid data. In the case of multiple bit fields, a lock bit might be used to prevent the status values from being updated while the read is occurring. For multi-bit configuration data, a toggle bit is used indicating that the local values in the functional core should be updated.

Any bits not specified in Table 2-3 are considered reserved and returns 0 upon read. The power on reset values of all the registers are 0 unless it is specified in the definition. Only address offsets are listed in Table 2-3. Base addresses are configured by the AXI Interconnect.

Table 2-3: **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 Gb/s ◦ 0x0A = 2.7 Gb/s ◦ 0x14 = 5.4 Gb/s
0x004	RW	LANE_COUNT_SET. Sets the number of lanes 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 Standard v1.2a.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x010	RW	LINK_QUAL_PATTERN_SET. Transmit the link quality pattern. • [2:0] – Enable transmission of the link quality test patterns. ◦ 000 = Link quality test pattern not transmitted ◦ 001 = D10.2 test pattern (unscrambled) transmitted ◦ 010 = Symbol Error Rate measurement pattern ◦ 011 = PRBS7 transmitted ◦ 100 = Custom 80-Bit pattern ◦ 101 = HBR2 compliance pattern
0x014	RW	SCRAMBLING_DISABLE. Set to 1 when the transmitter has disabled the scrambler and transmits all symbols. • [0] – Disable scrambling.
0x01C	WO	SOFTWARE_RESET. Reads will return zeros. • [0] – Soft Video Reset: When set, video logic is reset (stream 1). • [1] – Soft Video Reset: When set, video logic is reset (stream 2). • [2] – Soft Video Reset: When set, video logic is reset (stream 3). • [3] – Soft Video Reset: When set, video logic is reset (stream 4). • [7] – AUX Soft Reset. When set, AUX logic is reset.
0x020	RW	Custom 80-Bit quality pattern Bits[31:0]
0x024	RW	Custom 80-Bit quality pattern Bits[63:32]
0x028	RW	[15:0] - Customer 80-bit quality pattern Bits[80:64] [31:16] - Reserved
Core Enables		
0x080	RW	TRANSMITTER_ENABLE. Enable the basic operations of the transmitter. • [0] – When set to 1, stream transmission is enabled. When set to 0, all lanes of the main link 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 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x090	RW	VIDEO_PACKING_CLOCK_CONTROL: This register is used when GT data width is 32-bit.To meet the bandwidth requirement for the resolutions where vid_clk/vid_pixel_mode < lnk_clk frequency and with BPC 12/16 the video packing has to work at lnk_clk, setting the bit to '1' enables the packing from lnk_clk domain. By default video data packing is done in Vid_clk.All the resolutions with less than or equal to 10-BPC works with packing at vid_clk. • [0] – set to '1' to enable the video data packing to work in lnk_clk for SST video or for Stream 1 in MST. • [1] – set to '1' to enable the video data packing to work in lnk_clk for Stream 2 in MST. • [2] – set to '1' to enable the video data packing to work in lnk_clk for Stream 3 in MST. • [3] – set to '1' to enable the video data packing to work in lnk_clk for Stream 4 in MST.
0x0C0	WO	FORCE_SCRAMBLER_RESET. Reads from this register always return 0x0. • [0] – 1 forces a scrambler reset.
0x0D0	RW	TX_MST_CONFIG: MST Configuration. • [0] – MST Enable: Set to 1 to enable MST functionality. • [1] – VC Payload Updated in sink: This is an WO bit. Set to 1 after reading DPCD register 0x2C0 (bit 0) is set.
0x0F0	RW	TX_LINE_RESET_DISABLE. TX line reset disable. This register bits have to be used to disable the end of line reset to the internal video pipe in case of reduced blanking video support. • [0] - End of line reset disable to the SST video stream/ MST video stream1 • [1] - End of line reset disable to the MST video stream2 • [2] - End of line reset disable to the MST video stream3 • [3] - End of line reset disable to the MST video stream4
Core ID		
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 Standard v1.1a protocol with a Transmit core: 32'h01_01_0a_00 • DisplayPort Standard v1.2a protocol with a Transmit core: 32'h01_02_0a_00

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
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 initiates Address only transfers (STOP is 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.
0x10C	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. • [15:8] – The number of AXI4-Lite clocks (defined by the AXI4-Lite clock name: s_axi_aclk) equivalent to the recommended width of AUX pulse. Allowable values include: 8,16,24,32,40 and 48. From DisplayPort Protocol spec, AUX Pulse Width range = 0.4 to 0.6 µs. For example, for AXI4-Lite clock of 50 MHz (= 20 ns), the filter width, when set to 24, falls in the allowable range as defined by the protocol spec. ((20 × 24 = 480)) Program a value of 24 in this register.
0x110	RC	TX_USER_FIFO_OVERFLOW. Indicates an overflow in the user FIFO. The event can occur if the video rate does not match the TU size programming. • [0] – FIFO_OVERFLOW_FLAG: 1 indicates that the internal FIFO has detected an overflow condition. This bit clears upon read.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x130	RO	INTERRUPT_SIGNAL_STATE. Contains the raw signal values for those conditions which might cause an interrupt. • [3] – REPLY_TIMEOUT: 1 indicates that a reply timeout has occurred. • [2] – REPLY_STATE: 1 indicates that a reply is currently being received. • [1] – REQUEST_STATE: 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 Standard. ***Note:*** The core does 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.
0x140	RC	INTERRUPT_STATUS. Source core interrupt status register. A read from this register clears all values. Write operation is illegal and clears the values. • [9] – Audio packet ID mismatch interrupt, sets when incoming audio packet ID over streaming interface does not match with the info frame packet stream ID. • [5] – EXT_PKT_TXD: Extended packet is transmitted and controller is ready to accept new packet. Extended packet address space can also be used to send the audio copy management packet/ISRC packet/VSC packets. • [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.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
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. • [9] – Mask Audio packet ID mismatch interrupt. • [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.
0x154	RO	Free running counter incrementing for every 1 MHz.
Main Stream Attributes (Refer to the DisplayPort Standard for more details [Ref 3].)		
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.
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.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x188	RW	MAIN_STREAM_POLARITY. Provides the polarity values for the video sync signals. Polarity information is packed and sent in the MSA packet. See the Main Stream Attribute Data Transport section of the *DisplayPort Standard v1.2 Specification* [Ref 4]. • 0 = Active-High • 1 = Active-Low • [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. • [8] – Override Audio Clocking Mode • [9] – Sync/Async Mode for Audio • [10] – Audio Only Mode. When enabled, controller inserts information/timestamp packets every 512 BS symbols. By default the value is 0. • [11] – Maud control (Advanced Users)
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. • [6:3] – Reserved.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x1AC	RW	M-VID. If synchronous clocking mode is used, this register must be written with the M value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream is automatically computed by the source core and written to the main stream. These values are not written into the M-VID register for readback. • [23:0] – Unsigned M value.
0x1B0	RW	TRANSFER_UNIT_SIZE. Sets the size of a transfer unit in the framing logic On reset, transfer size is set to 64. This register must be written as described in section 2.2.1.4.1 of the standard. • [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).
0x1B4	RW	N-VID. If synchronous clocking mode is used, this register must be written with the N value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream is automatically computed by the source core and written to the main stream. These values are not written into the N-VID register for readback. • [23:0] – Unsigned N value.
0x1B8	RW	USER_PIXEL_WIDTH. Selects the width of the user data input port. Use quad pixel mode in MST. In SST, the user pixel width should always be less than or equal to the active line count generated in hardware. • [2:0]: ◦ 1 - Single pixel wide interface ◦ 2 - Dual pixel wide interface. Valid for designs with 2 or 4 lanes. ◦ 4 - Quad pixel wide interface.Valid for designs with 4 lanes only.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition

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 divisible 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 divisible 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 sets 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 Standard. MIN_BYTES_PER_TU should be ≥ 4 when GT Data width is selected as 32-bit. • [6:0] – Set the value to INT((VIDEO_BW/LINK_BW)*TRANSFER_UNIT_SIZE)
0x1C8	RW	FRAC_BYTES_PER_TU. Calculating MIN bytes per TU is often not a whole number. This register is used to hold the fractional component. • [9:0] – The fraction part of ((VIDEO_BW/LINK_BW)*TRANSFER_UNIT_SIZE) scaled by 1024 is programmed in this register.

Table 2-3: **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. The default value of INIT_WAIT is 0x20. If (MIN_BYTES_PER_TU ≤ 4) • [7:0] – Set INIT_WAIT to 64 else if color format is RGB/YCbCr_444 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) else if color format is YCbCr_422 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 else if color format is Y_Only • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/3
0x1D0	RW	STREAM1. Average Stream Symbol Timeslots per MTP Config: • [9:0] – TS_FRAC: Program fraction × 1000 in this field. See the *DisplayPort Standard* section 2.6.3.3 VC Payload Size Determination by a Source Payload Bandwidth Manager. • [23:16] – TS_INT: Program integer value based on the calculations.
0x1D4	RW	STREAM2. Average Stream Symbol Timeslots per MTP Config: • [9:0] – TS_FRAC: Program fraction × 1000 in this field. See the *DisplayPort Standard* section 2.6.3.3 VC Payload Size Determination by a Source Payload Bandwidth Manager. • [23:16] – TS_INT: Program integer value based on the calculations.
0x1D8	RW	STREAM3. Average Stream Symbol Timeslots per MTP Config: • [9:0] – TS_FRAC: Program fraction × 1000 in this field. See the *DisplayPort Standard* section 2.6.3.3 VC Payload Size Determination by a Source Payload Bandwidth Manager. • [23:16] – TS_INT: Program integer value based on the calculations.
0x1DC	RW	STREAM4. Average Stream Symbol Timeslots per MTP Config: • [9:0] – TS_FRAC: Program fraction × 1000 in this field. See the *DisplayPort Standard* section 2.6.3.3 VC Payload Size Determination by a Source Payload Bandwidth Manager. • [23:16] – TS_INT: Program integer value based on the calculations.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
PHY Configuration Status		
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] – Unused, read as 0. • [21:20]- Transmitter buffer status, lane 1. • [23:22] – Unused, read as 0. • [25:24] – Transmitter buffer status, lane 2. • [27:26] – Unused, read as 0. • [29:28] – Transmitter buffer status, lane 3. • [31:30] – Unused, read as 0.
0x4FC	RO	SINK_VID_FRAMING_ERROR_STATUS: Sink Video Framing error status. This is a debug register that is valid when GT data width is 32-bit. • [1:0] – Stream1 error status in framing. • [9:8] – Stream2 error status in framing. • [17:16] – Stream3 error status in framing. • [25:24] – Stream4 error status in framing.
MST Interface		
0x500	RW	MAIN_STREAM_HTOTAL_STREAM2. 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.
0x504	RW	MAIN_STREAM_VTOTAL_STREAM2. Provides the total number of lines in the main stream video frame. • [15:0] – Total number of lines per video frame.
0x508	RW	MAIN_STREAM_POLARITY_STREAM2. 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.
0x50C	RW	MAIN_STREAM_HSWIDTH_STREAM2. Sets the width of the horizontal sync pulse. • [14:0] – Horizontal sync width in clock cycles.
0x510	RW	MAIN_STREAM_VSWIDTH_STREAM2. Sets the width of the vertical sync pulse. • [14:0] – Width of the vertical sync in lines.
0x514	RW	MAIN_STREAM_HRES_STREAM2. Horizontal resolution of the main stream video source. • [15:0] – Number of active pixels per line of the main stream video.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x518	RW	MAIN_STREAM_VRES_STREAM2. Vertical resolution of the main stream video source. • [15:0] – Number of active lines of video in the main stream video source.
0x51C	RW	MAIN_STREAM_HSTART_STREAM2. Number of clocks between the leading edge of the horizontal sync and the start of active data. • [15:0] – Horizontal start clock count.
0x520	RW	MAIN_STREAM_VSTART_STREAM2. Number of lines between the leading edge of the vertical sync and the first line of active data. • [15:0] – Vertical start line count.
0x524	RW	MAIN_STREAM_MISC0_STREAM2. 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.
0x528	RW	MAIN_STREAM_MISC1_STREAM2. 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. • [6:3] – Reserved.
0x52C	RW	M-VID_STREAM2. If synchronous clocking mode is used, this register must be written with the M value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream as automatically computed by the source core and written to the main stream. These values are not written into the M-VID register for readback. • [23:0] – Unsigned M value.
0x530	RW	TRANSFER_UNIT_SIZE_STREAM2. 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).
0x534	RW	N-VID_STREAM2. If synchronous clocking mode is used, this register must be written with the N value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream as automatically computed by the source core and written to the main stream. These values are not written into the N-VID register for readback. • [23:0] – Unsigned N value.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x538	RW	USER_PIXEL_WIDTH_STREAM2. Selects the width of the user data input port. Use quad pixel mode in MST. • [2:0]: ◦ 1 = Single pixel wide interface ◦ 2 = Dual pixel wide interface ◦ 4 = Quad pixel wide interface

0x53C	RW	USER_DATA_COUNT_PER_LANE_STREAM2. This register is used to translate the number of pixels per line to the native internal 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 divisible 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 divisible 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

0x540	RW	MAIN_STREAM_INTERLACED_STREAM2. 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.
0x544	RW	MIN_BYTES_PER_TU_STREAM2: Programs source to use MIN number of bytes per transfer unit. The calculation should be done based on the DisplayPort Standard. • [7:0] – Set the value to INT((LINK_BW/VIDEO_BW)*TRANSFER_UNIT_SIZE)
0x548	RW	FRAC_BYTES_PER_TU_STREAM2: 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 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x54C	RW	INIT_WAIT_STREAM2: 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 if color format is RGB/YCbCr_444 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) else if color format is YCbCr_422 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 else if color format is Y_Only • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/3
0x550	RW	MAIN_STREAM_HTOTAL_STREAM3. 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.
0x554	RW	MAIN_STREAM_VTOTAL_STREAM3. Provides the total number of lines in the main stream video frame. • [15:0] – Total number of lines per video frame.
0x558	RW	MAIN_STREAM_POLARITY_STREAM3. 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.
0x55C	RW	MAIN_STREAM_HSWIDTH_STREAM3. Sets the width of the horizontal sync pulse. • [14:0] – Horizontal sync width in clock cycles.
0x560	RW	MAIN_STREAM_VSWIDTH_STREAM3. Sets the width of the vertical sync pulse. • [14:0] – Width of the vertical sync in lines.
0x564	RW	MAIN_STREAM_HRES_STREAM3. Horizontal resolution of the main stream video source. • [15:0] – Number of active pixels per line of the main stream video.
0x568	RW	MAIN_STREAM_VRES_STREAM3. Vertical resolution of the main stream video source. • [15:0] – Number of active lines of video in the main stream video source.
0x56C	RW	MAIN_STREAM_HSTART_STREAM3. Number of clocks between the leading edge of the horizontal sync and the start of active data. • [15:0] – Horizontal start clock count.
0x570	RW	MAIN_STREAM_VSTART_STREAM3. Number of lines between the leading edge of the vertical sync and the first line of active data. • [15:0] – Vertical start line count.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x574	RW	MAIN_STREAM_MISC0_STREAM3. 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.
0x578	RW	MAIN_STREAM_MISC1_STREAM3. 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. • [6:3] – Reserved.
0x57C	RW	M-VID_STREAM3. If synchronous clocking mode is used, this register must be written with the M value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream as automatically computed by the source core and written to the main stream. These values are not written into the M-VID register for readback. • [23:0] – Unsigned M value
0x580	RW	TRANSFER_UNIT_SIZE_STREAM3. 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).
0x584	RW	N-VID_STREAM3. If synchronous clocking mode is used, this register must be written with the N value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream as automatically computed by the source core and written to the main stream. These values are not written into the N-VID register for readback. • [23:0] – Unsigned N value
0x588	RW	USER_PIXEL_WIDTH_STREAM3. Selects the width of the user data input port. Use quad pixel mode in MST. • [2:0]: ◦ 1 = Single pixel wide interface ◦ 2 = Dual pixel wide interface ◦ 4 = Quad pixel wide interface

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition

0x58C	RW	USER_DATA_COUNT_PER_LANE_STREAM3. 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 divisible 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 divisible 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

0x590	RW	MAIN_STREAM_INTERLACED_STREAM3. 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.
0x594	RW	MIN_BYTES_PER_TU_STREAM3: Programs source to use MIN number of bytes per transfer unit. The calculation should be done based on the DisplayPort Standard. • [7:0] – Set the value to INT((LINK_BW/VIDEO_BW)*TRANSFER_UNIT_SIZE)
0x598	RW	FRAC_BYTES_PER_TU_STREAM3: Calculating MIN bytes per TU is often not 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 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x59C	RW	INIT_WAIT_STREAM3: 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 if color format is RGB/YCbCr_444 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) else if color format is YCbCr_422 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 else if color format is Y_Only • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/3
0x5A0	RW	MAIN_STREAM_HTOTAL_STREAM4. 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.
0x5A4	RW	MAIN_STREAM_VTOTAL_STREAM4. Provides the total number of lines in the main stream video frame. • [15:0] – Total number of lines per video frame.
0x5A8	RW	MAIN_STREAM_POLARITY_STREAM4. 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.
0x5AC	RW	MAIN_STREAM_HSWIDTH_STREAM4. Sets the width of the horizontal sync pulse. • [14:0] – Horizontal sync width in clock cycles.
0x5B0	RW	MAIN_STREAM_VSWIDTH_STREAM4. Sets the width of the vertical sync pulse. • [14:0] – Width of the vertical sync in lines.
0x5B4	RW	MAIN_STREAM_HRES_STREAM4. Horizontal resolution of the main stream video source. • [15:0] – Number of active pixels per line of the main stream video.
0x5B8	RW	MAIN_STREAM_VRES_STREAM4. Vertical resolution of the main stream video source. • [15:0] – Number of active lines of video in the main stream video source.
0x5BC	RW	MAIN_STREAM_HSTART_STREAM4. Number of clocks between the leading edge of the horizontal sync and the start of active data. • [15:0] – Horizontal start clock count.
0x5C0	RW	MAIN_STREAM_VSTART_STREAM4. Number of lines between the leading edge of the vertical sync and the first line of active data. • [15:0] – Vertical start line count.

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x5C4	RW	MAIN_STREAM_MISC0_STREAM4. 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.
0x5C8	RW	MAIN_STREAM_MISC1_STREAM4. 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. • [6:3] – Reserved.
0x5CC	RW	M-VID_STREAM4. If synchronous clocking mode is used, this register must be written with the M value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream as automatically computed by the source core and written to the main stream. These values are not written into the M-VID register for readback. • [23:0] – Unsigned M value.
0x5D0	RW	TRANSFER_UNIT_SIZE_STREAM4. 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).
0x5D4	RW	N-VID_STREAM4. If synchronous clocking mode is used, this register must be written with the N value as described in section 2.2.5.2 of the standard. When in asynchronous clocking mode, the M value for the video stream as automatically computed by the source core and written to the main stream. These values are not written into the N-VID register for readback. • [23:0] – Unsigned N value.
0x5D8	RW	USER_PIXEL_WIDTH_STREAM4. Selects the width of the user data input port. Use quad pixel mode in MST. • [2:0]: ◦ 1 = Single pixel wide interface ◦ 2 = Dual pixel wide interface ◦ 4 = Quad pixel wide interface

Table 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition

0x5DC	RW	USER_DATA_COUNT_PER_LANE_STREAM4. 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 divisible 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 divisible 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

0x5E0	RW	MAIN_STREAM_INTERLACED_STREAM4. Informs the DisplayPort transmitter main link that the source video is interlaced. By setting this bit to a 1, the core sets 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.
0x5E4	RW	MIN_BYTES_PER_TU_STREAM4. Programs source to use MIN number of bytes per transfer unit. The calculation should be done based on the DisplayPort Standard. • [7:0] – Set the value to INT((LINK_BW/VIDEO_BW)*TRANSFER_UNIT_SIZE)
0x5E8	RW	FRAC_BYTES_PER_TU_STREAM4. Calculating MIN bytes per TU is often not 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 2-3: **DisplayPort Source Core Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x5EC	RW	INIT_WAIT_STREAM4. 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 if color format is RGB/YCbCr_444 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) else if color format is YCbCr_422 • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 else if color format is Y_Only • [7:0] – Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/3
0x800 to 0x8FF	WO	PAYLOAD_TABLE. This address space maps to the VC payload table that is maintained in the core. • [7:0] – Payload data

DisplayPort Audio

The DisplayPort Audio registers are listed in Table 2-4.

Table 2-4: **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 • [16]: Set to '1' to mute the audio over link
0x304	R/W	TX_AUDIO_CHANNELS. Used to input active channel count. Transmitter collects audio samples based on this information. • [2:0] Channel Count

Table 2-4: **DisplayPort Audio Registers *(Cont'd)***

Offset	R/W	Definition
0x308	WO	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 to 0x350	WO	TX_AUDIO_EXT_DATA. [31:0] = Word formatted as per Extension packet described in protocol standard. Extended packet is fixed to 32 Bytes length. The controller has buffer space for only one extended packet. Extension packet address space can be used to send the audio Copy management packet/ISRC packet/VSC packets. TX is capable of sending any of these packets. 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 Standard 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.

HDCP Registers

For details about the HDCP registers, see the *HDCP Controller Product Guide* (PG224) [Ref 13]

AXI Timer Registers

For details about the AXI Timer registers, see the *AXI Timer Product Guide* (PG079) [Ref 14]

Designing with the Core

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

DisplayPort Overview

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 Setup and Management

This section is intended to elaborate on and act as a companion to the link training procedure as described in section 3.5.1.3 of the *VESA DisplayPort Standard v1.2a* [Ref 1].

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. Xilinx does not recommend using the RTL based controllers.

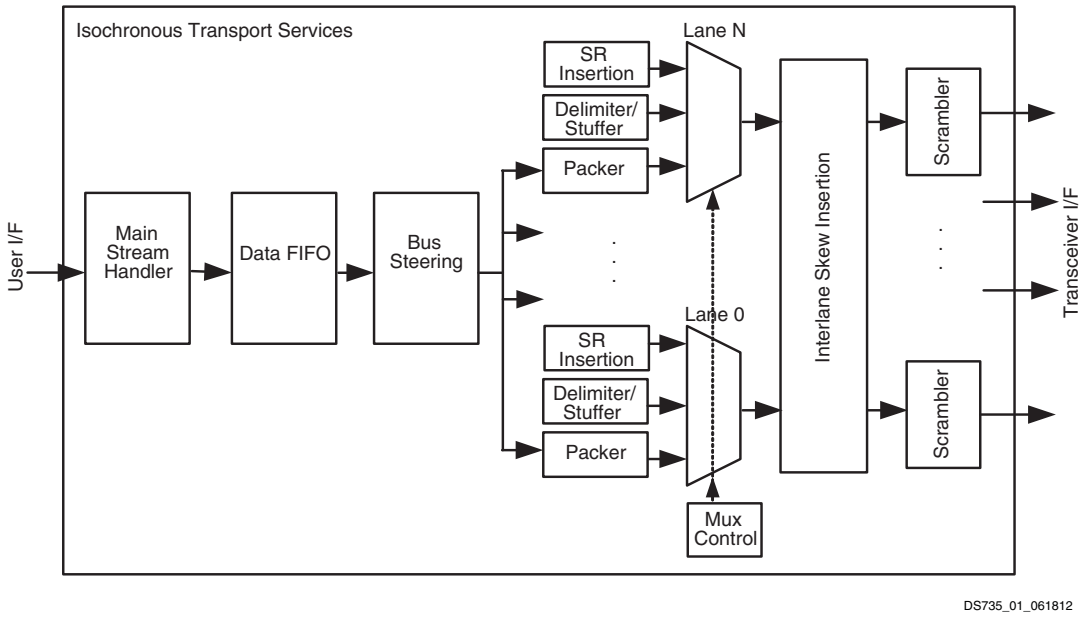

Figure 3-1: **Source Main Link Datapath**

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 datapath 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 might occur.

Figure 3-2 shows the flow diagram for link training. For details, refer to the *VESA DisplayPort Standard v1.2a* [Ref 4].

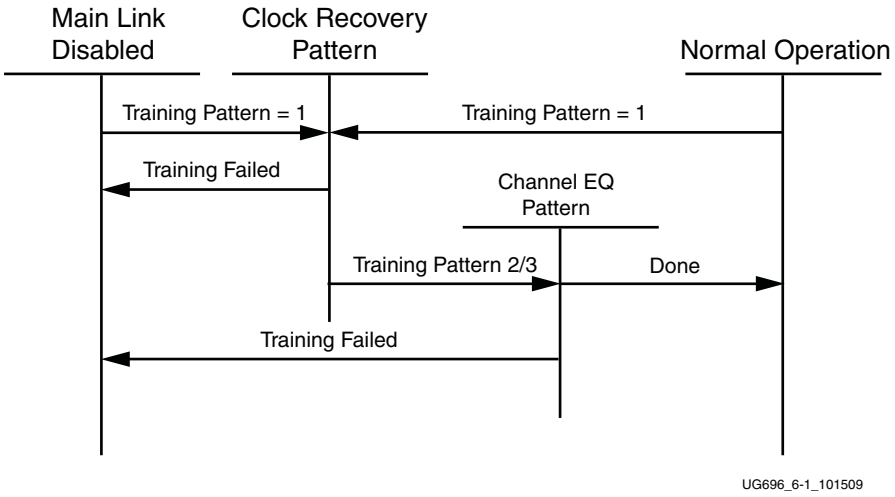

Figure 3-2: **Link Training States**

Source Core Setup and Initialization

The following text contains the procedural tasks required to achieve link communication. For description of the DPCD, see *VESA DisplayPort Standard v1.2a* [Ref 4].

IMPORTANT: *During initialization ensure that TX8B10BEN is not cleared in offset 0x0070 of the corresponding Video PHY Controller Product Guide (PG230) [Ref 15].*

Source Core Setup

1. Place the PHY into reset.
2. Disable the transmitter.
   - TRANSMITTER_ENABLE = 0x00
3. Set the clock divider.
   - AUX_CLOCK_DIVIDER = (see register description for proper value)
4. Select and set up the reference clock for the desired link rate in the Video PHY Controller.
5. Bring the PHY out of reset.
6. Wait for the PHY to be ready.
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 to 0x0000B) though 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 to 0x00101) of the DPCD through 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 through direct register writes.
   - SCRAMBLING_DISABLE = 0x01
   - TRAINING_PATTERN_SET = 0x01
2. Turn off scrambling and set training pattern 1 in the sink DPCD (0x00102 to 0x00106) through the AUX channel.
3. Wait for the aux read interval configured in TRAINING_AUX_RD_INTERVAL DPCD Register (0x0000E) before reading status registers for all active lanes (0x00202 to 0x00203) through the AUX channel.
4. If clock recovery failed, check for voltage swing or preemphasis level increase requests (0x00206 to 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 through direct register writes.
   - SCRAMBLING_DISABLE = 0x01
   - TRAINING_PATTERN_SET = 0x02
2. Turn off scrambling and set training pattern 2 in the sink DPCD (0x00102 to 0x00106) through the AUX channel.
3. Wait for aux read interval configured in TRAINING_AUX_RD_INTERVAL DPCD Register (0x0000E) then read status registers for all active lanes (0x00202 to 0x00203) through the AUX channel.
4. Check the channel equalization, symbol lock, and interlane alignment status bits for all active lanes (0x00204) through the AUX channel.
5. If any of these bits are not set, check for voltage swing or preemphasis level increase requests (0x00206 to 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) through the AUX channel.
8. On the source side, re-enable scrambling and turn off 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 Gb/s link rate devices. See the DisplayPort Standard v1.2a 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 wants 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 to 0x1B0). Enable the main stream (0x084) only when a reliable video source is available.

IMPORTANT: *When the main link video is enabled, the scrambler/de-scrambler must be reset every 512th BS Symbol as described in section 2.2.1.1 of the DisplayPort standard. For simulation purposes, you should force a scrambler reset by writing a '1' to 0x0c0 before the main link is enabled to reduce the amount of time after startup needed to align the scramber/de-scrambler.*

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 v1.2a*.)

Before any AUX channel operation can be completed, you must first set the proper clock divider 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 3-3 shows a flow of an AUX write transaction.

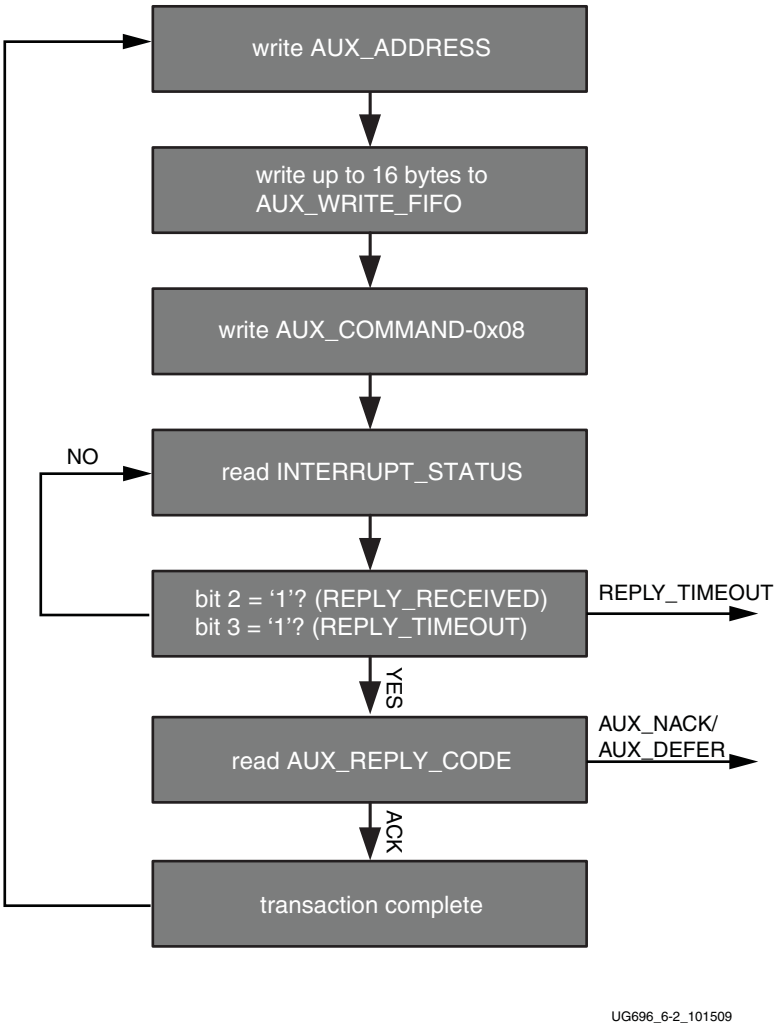

Figure 3-3: **AUX Write Transaction**

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 might 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 3-4 shows a flow of an AUX read transaction.

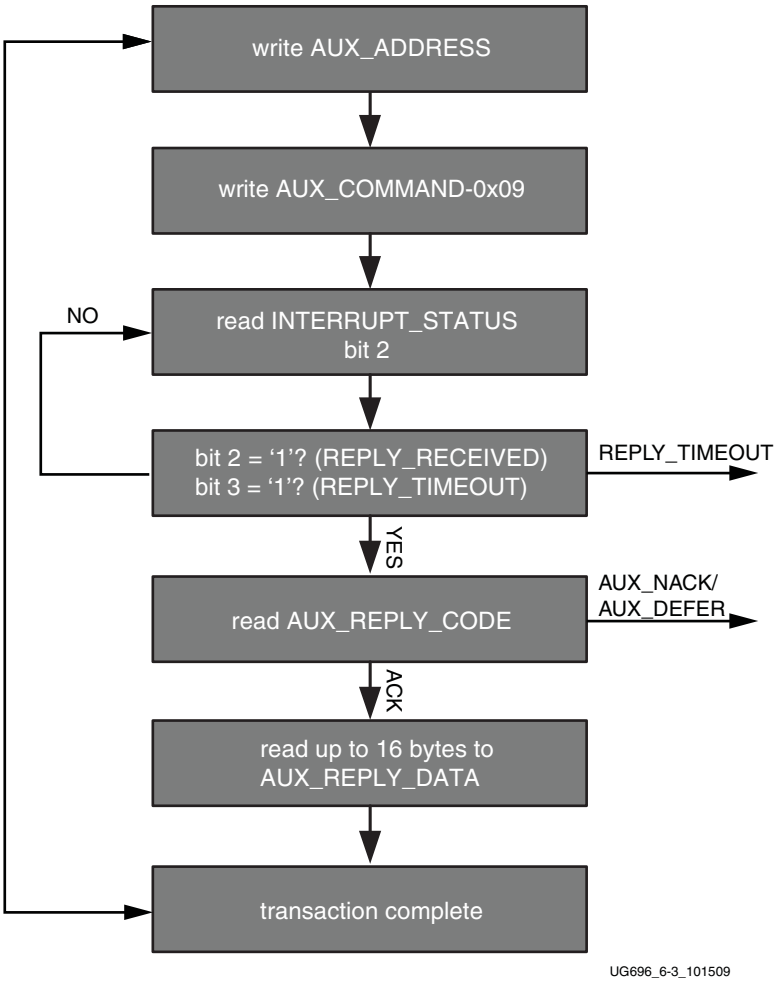

Figure 3-4: **AUX Read Transaction**

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
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 might emulate an I2C transaction.

Figure 3-5 shows the flow of commanded I2C transactions.

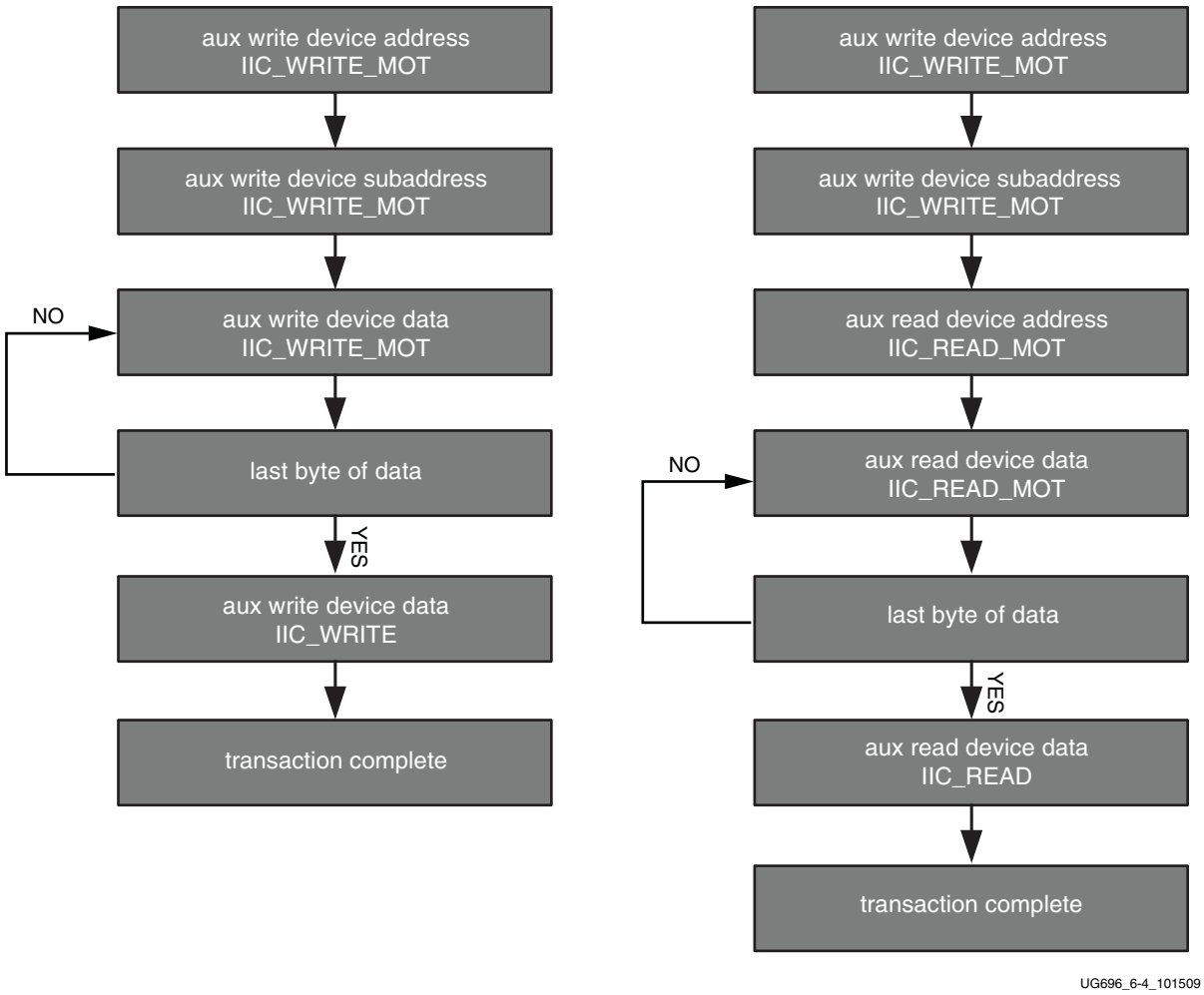

Figure 3-5: **Commanded I2C Device Transactions, Write (Left) and Read (Right)**

Since I2C transactions might 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.
- IIC Timeout: The sink controller monitors the IIC bus after a transaction starts and looks for an IIC stop occurrence within 1 second. If an IIC stop is not received, it is considered as an IIC timeout and the sink controller issues a stop condition to release the bus. This timeout avoids a lock-up scenario.

Generation of AUX transactions are described in Table 3-2.

Table 3-2: **Generation of 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.

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

Transaction	AUX Transaction	I2C Transaction	Usage	Sequence
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 of 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 of 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/Timeout:

- 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 a NACK response to these requests while intermediate ones will get a partial ACK.
  - Issue the same command for a response (previously issued with the same device address, length and data) and wait for a response. On completion of the write to I2C (after multiple partial ACKs), 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/Timeout:

- 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 requests while intermediate ones will get partial ACKs.
  - 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.

AUX IO Location

DisplayPort source can have AUX IO located inside the IP or external to the IP based on the AUX IO location selection through GUI. The AUX IO type can be unidirectional/bidirectional when the AUX IO is located inside the IP.

Transmitter Audio/Video Clock Generation

The transmitter clocking architecture supports both the asynchronous and synchronous clocking modes included in the *DisplayPort Standard v1.2a*. 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 3-6 shows a block diagram of the transmitter clock generation process.

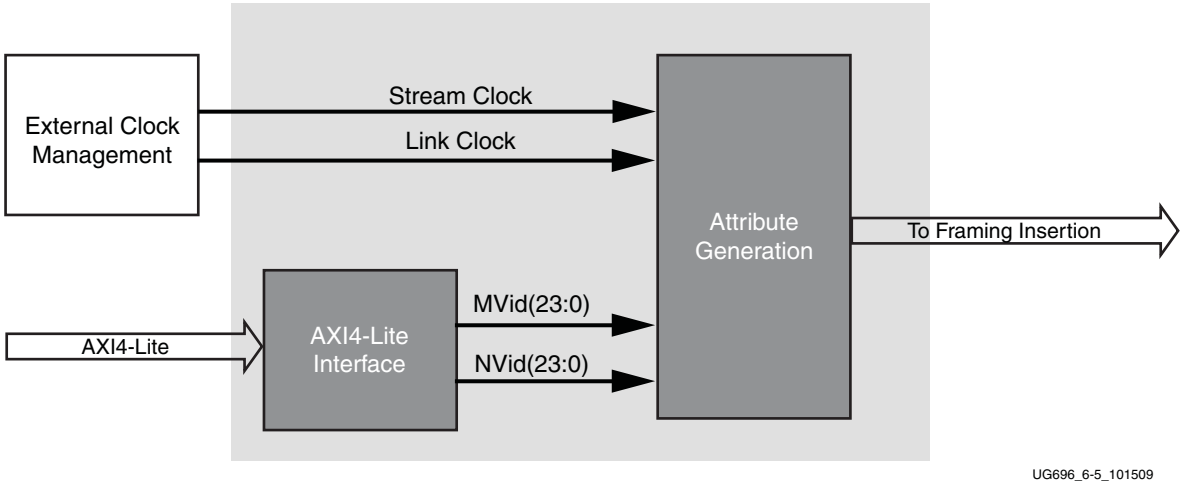

Figure 3-6: **Transmitter Audio/Video Clock Generation**

Hot Plug Detection

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

If HPD signal remains Low for > 2 ms, then 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. For details, refer to the *VESA DisplayPort Standard v1.2a* [Ref 4].

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 3-7 shows the source core state and basic actions to be taken based on HPD events.

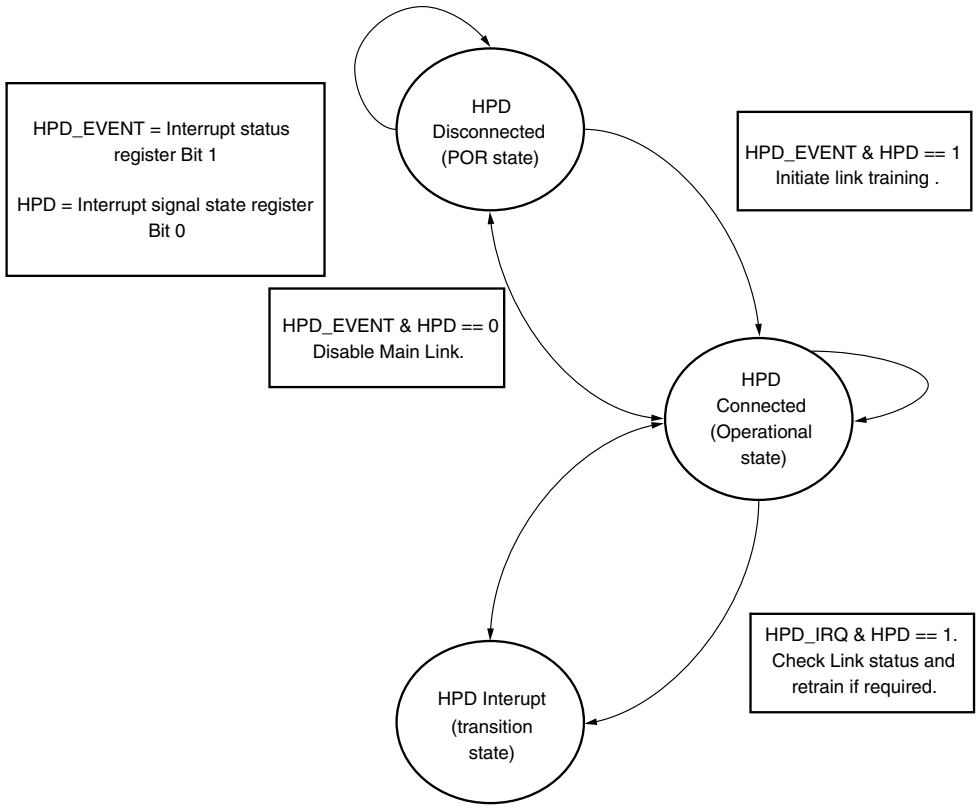

Figure 3-7: **HPD Event Handling in Source Core**

Secondary Channel Operation

The current version of the DisplayPort IP supports 8-channel Audio. Secondary Channel features from the DisplayPort Standard v1.2a are supported.

The DisplayPort Audio IP core is offered as modules to provide flexibility and freedom to modify the system as needed. As shown in Figure 3-8, the Audio interface to the

DisplayPort core is defined using an AXI4-Stream interface to improve system design and IP integration.

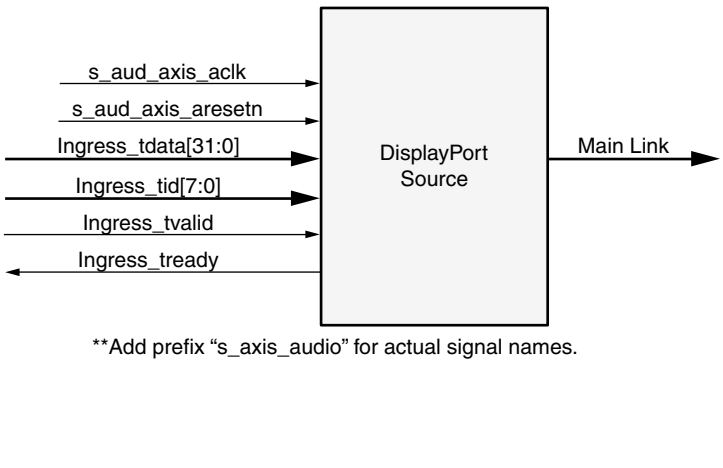

Figure 3-8: **Audio Data Interface of DisplayPort Source System**

32-bit AXI TDATA is formatted according as follows:

Control Bits + 24-bit Audio Sample + Preamble

The ingress channel buffer in the DisplayPort core accepts data from the streaming interface 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.

Multi Channel Audio

DisplayPort transmitter requires Info frame configuration to transmit multi-channel audio. The Info frame contains the number of channels and its speaker mapping. Streaming TID should contain the Audio channel ID along with audio data, based on the number of channels configured.

For multi-stream audio, secondary data packet ID in the Info frame packet should match with the stream ID over the audio streaming interface (`TID[7:4]`).

Programming DisplayPort Source

1. Disable Audio by writing 0x00 to TX_AUDIO_CONTROL register. The disable bit also flushes the buffers in DisplayPort Source and sets the MUTE bit in VB-ID. Xilinx recommends following this step when there is a change in video/audio parameters.

2. Write Audio Info Frame (Based on your requirement. This might 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 DisplayPort Audio Registers, offset 0x308 (Table 2-4).
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 to TX_AUDIO_MAUD and TX_AUDIO_NAUD registers, respectively.
5. Enable Audio by writing 0x01 to TX_AUDIO_CONTROL register.

Re-Programming Source Audio

1. Disable Audio in DisplayPort TX core.
2. Wait until Video/Audio clock is recovered and stable.
3. Enable Audio in DisplayPort TX core.
4. Wait for some time (in µs).

Info Packet Management

The core provides an option to program a single Info packet. The packet is 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 also flushes 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 × fs (Audio Sample Rate) clock frequency. The same clock (aud_clk) is used by DisplayPort Source device to calculate MAUD and NAUD when running in asynchronous clocking mode.

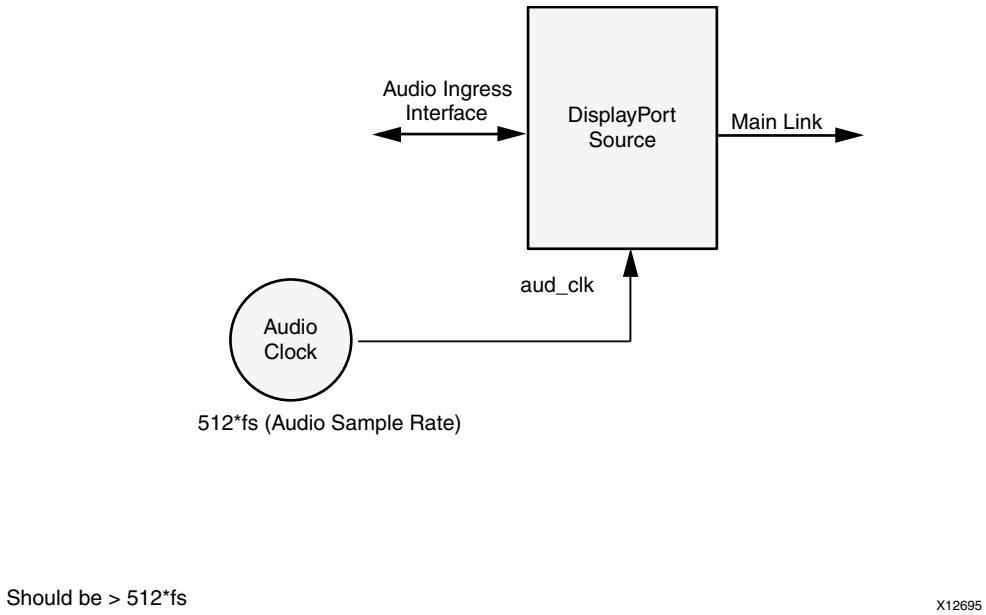

Figure 3-9: **Source: Audio Clocking**

Programming the Core in MST Mode

The section details the steps to program the core in MST mode.

Enabling MST

The following steps are recommended to enable MST functionality:

1. Bring up the main link by following training procedure.
2. Send side band messages using the AUX channel to discover the link (how many downstream nodes are connected and their capabilities).
3. Enable MST by writing '1' to bit 0 of the MST Config register.
4. Discover MST downstream devices as recommended in section 1.2.1 in the *DisplayPort Standard*.
5. Allocate timeslots based on configuration and the Sink Payload Bandwidth Number (PBN). Typical sideband messages used before VC Payload allocation are Link Address Request, Clear Payload Table, and Enumerate Path Resources.
   a. Program VC Payload Buffer 12'h0x800 onwards as per allocation requirement.

b. Program the Sink core with the same allocation timeslots using AUX channel as described in section 2.6.4 in the *DisplayPort Standard*.

c. Wait until Sink accepts allocation programming (check DPCD reads to monitor status).

d. After Sink sets VC Payload Allocated (DPCD Address = 0x02C0), set VC Payload Allocated bit in MST Config register (12'h0x0D0). This enables the source controller to send an ACT trigger.

6. Wait until ACT Handled bit is set in DPCD Address (0x02C0).
7. Program Video attributes for required streams. Program user pixel width to 4 for all the streams.
8. Program Rate Governing registers 0x1D0, 0x1D4, 0x1D8, and 0x1DC based on the stream requirement.
   - Program TRANSFER UNIT Size = # of timeslots allocated for that stream. (VC payload size source)
   - Program FRAC_BYTES_PER_TU = TS_FRAC
   - Program MIN_BYTES_PER_TU = TS_INT
   - Program INIT_WAIT = 0

   Note: Repeat step 7 for each steam.
9. Enable MST by writing 1 to bit 0 of MST Config register.

After these steps are done, the source controller starts sending MST traffic as per VC Payload programming in the main link.

Payload Bandwidth Management

The following steps manage payload bandwidth in the source controller.

1. Calculate Target_Average_StreamSymbolTimeSlotsPerMTP based on the *DisplayPort Standard v1.2* or later. Program VC payload size with calculated Target_Average_StreamSymbolTimeSlotsPerMTP and align it with nearest even boundary.

   For example if the value is 13, program VC payload size for this particular stream to 14.
2. In MST mode when GT data width is 4 bytes the VC Payload should be multiple of 4.
3. The VC payload calculation for UHD (1920X2200) stream, RGB color sampling, 8 Bits Per Color at 5.4 Gb/s, 4 lanes is given here.

   VC Payload Band width = LINK_RATE × Lane_count × 100 (see Table 2-61 in the *VESA DisplayPort Standard v1.2a* [Ref 1])

= 5.4 × 4 × 100

= 2160

Average Stream symbol Time slot per MTP = (Pixel_rate × Bits_per_pixel/8/VC Payload_Band width) × 64 (see Section 2.6.3.3 in the *VESA DisplayPort Standard v1.2a* [Ref 1])

= (297 MHz × 24 /8/2160) × 64

= 26.4

VC Payload Size = 2/4 symbol aligned of (Average Stream symbol Time slot per MTP)

= 28

4. Program VC Payload table as defined in DPCD standard.
5. Program VC Payload table in source controller as defined in registers 12'h0x800 – 12'h0x8FC.

Reduced Blanking

DisplayPort IP supports CVT standard RB and RB2 reduced blanking resolutions. As per the CVT specifications RB/RB2 resolution has HBLANK ≤ 20% HTOTAL, HBLANK = 80/160 and HRES%8 = 0.

For the CVT standard, RB/RB2 resolutions end of the line reset need to be disabled by setting the corresponding bit in the Line reset disable register (offset address 0x0F0 for transmitter). For the Non-CVT reduced blanking resolutions, where HRES is non multiple of 8, end of line reset is required to clear extra pixels in the video path for each line.

DisplayPort transmitter knows the resolution ahead of time hence reset disable can be done during initialization. In DisplayPort receiver when video mode change interrupt occurs the MSA registers can be read to know whether the resolution is reduced blanking or standard resolution and the corresponding bit can be set.

Clocking

This section describes the link clock (`tx_lnk_clk`) and the video clock (`tx_vid_clk_stream1`). For information on other clocks, see the *DisplayPort Product Guide* (PG064) [Ref 10].

The AXI4-Stream to Video bridge can handle asynchronous clocking. The value is based on the Consumer Electronics Association (CEA)/VESA Display Monitor Timing (DMT) standard

for given video resolutions. Similarly for MST mode, tx_vid_clk_streamn and s_axis_aclk_streamn can be same or the s_axis_aclk_streamn can be at higher frequency than tx_vid_clk_streamn.

The tx_lnk_clk is a link clock input to the DisplayPort TX Subsystem generated by the Video PHY (GT). The frequency of tx_lnk_clk is <line_rate>/40 MHz for the 32-bit video PHY(GT) data interface and <line_rate>/20 MHz for 16-bit interface. See Table 3-3 for the recommended values.

In 16-bit GT interface hdcp_ext_clk input has to be driven from external MMCM where it has a frequency requirement of hdcp_ext_clk = tx_lnk_clk/2 MHz.

In native mode, the TX video clock has to be as per the value based on the Consumer Electronics Association (CEA)/VESA Display Monitor Timing (DMT) standard for given video resolutions.

Table 3-3: **Clocking**

Resolution	AXI4-Stream (s_axis_aclk_stream1)	Video Pipe (m_aclk_stream1)	User Video Clock (tx_vid_clk_stream1)
UHD at the 60 fps (frame split mode)	148.5(1)	74.25(1)	74.25(1)
Other Modes	Video Clock(2)	Video Clock(2)	Video Clock(2)

Notes:
1. For MST stream 1 and stream 2 only.
2. For all four streams when MST mode is enabled. See DMT/CEA spec for video clock range for each DMT resolution.

Resets

The subsystem has one reset input for each of the AXI4-Lite, AXI4-Stream and Video interfaces:

- s_axi_aresetn: Active-Low AXI4-Lite reset. This resets all the programming registers.
- tx_vid_reset_stream1: Active-High video pipe reset. For MST with four streams, there are four video resets.
- s_axis_aresetn_stream1: Active-Low AXI4-Stream interface reset. For MST with four streams, there are four resets corresponding to each stream.
- m_aresetn_stream1: Active-Low reset for streams one and two.

Address Map Example

Table 3-4 shows an example based on a subsystem base address of 0x44C0_0000 (19 bits). The DisplayPort TX Subsystem requires a 19-bit address mapping, starting at an offset address of 0x00000.

This address map example is applicable when TX subsystem is configured in Streaming Interface mode. In Native Interface mode, the Dual Splitter and Video Timing Controller are not present. ‘

Table 3-4: **Address Map Example**

	SST	MST
DisplayPort TX Core	0x44C0_0000	0x44C0_0000
Dual Splitter	N/A	0x44C1_0000
VTC 0	0x44C2_0000	0x44C2_0000
VTC 1 (N = 2)	N/A	0x44C3_0000
VTC 2 (N = 3)	N/A	0x44C4_0000
VTC 3 (N = 4)	N/A	0x44C5_0000
HDCP Controller	0x44C3_0000	N/A
AXI Timer	0x44C4_0000	N/A

Programming Sequence

This section contains the programming sequence for the subsystem using UHD@60 in MST mode with two streams. Program and enable the components of the DisplayPort TX Subsystem in the following order. HDCP Controller and AXI Timer address map exist when HDCP is enabled in SST mode.

1. DisplayPort TX Core
2. Dual Splitter
3. Video Timing Controller

Dual Splitter Programming

Use the following steps to program the Dual Splitter.

1. Write 0x02 in GENR_CONTROL_REG. This begins the programming sequence and the Dual Splitter register update bit is set.
2. Write vertical resolution and horizontal resolution in TIME_CONTROL_REG.
3. The Dual Splitter is used in a configuration where the input frame must be split into two vertical halves. Write the overlap, number of segments, output samples per clock and input samples per clock in CORE_CONTROL_REG.

   For 4k frame split mode, write 0x02_04_04 to register 0x100 (number of segments = 2; number of samples per clock at output = 4; number of samples per clock at input = 4).

For other modes, write 0x010404 (number of segments =1 (bypass); number of output samples = 4; number of input samples = 4).

4. Write 0x03 in GENR_CONTROL_REG to enable the Dual Splitter for programmed resolutions and splitting functionality.

When programming the Dual Splitter, note the following:

- There should be no overlap of the two segments in a frame.
- Segment 0 of the Dual Splitter is the left frame and Segment 1 is the right frame.
- The timing of two segments of the splitter is independent, but by the start of a new line, both the segments complete the previous line.
- For UHD @ 60 in frame split mode, the width of the frame (HRES) must be equal to actual HRES/4.

Design Flow Steps

This chapter describes customizing and generating the subsystem. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 2]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5]

Customizing and Generating the Subsystem

This section includes information about using Xilinx tools to customize and generate the subsystem in the Vivado Design Suite.

If you are customizing and generating the subsystem in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 2] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the subsystem by specifying values for the various parameters associated with the subsystem IP cores using the following steps:

1. Select the subsystem from the IP catalog.
2. Double-click the selected subsystem or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4].

Note: Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Customizing the IP

The configuration screen is shown in Figure 4-1.

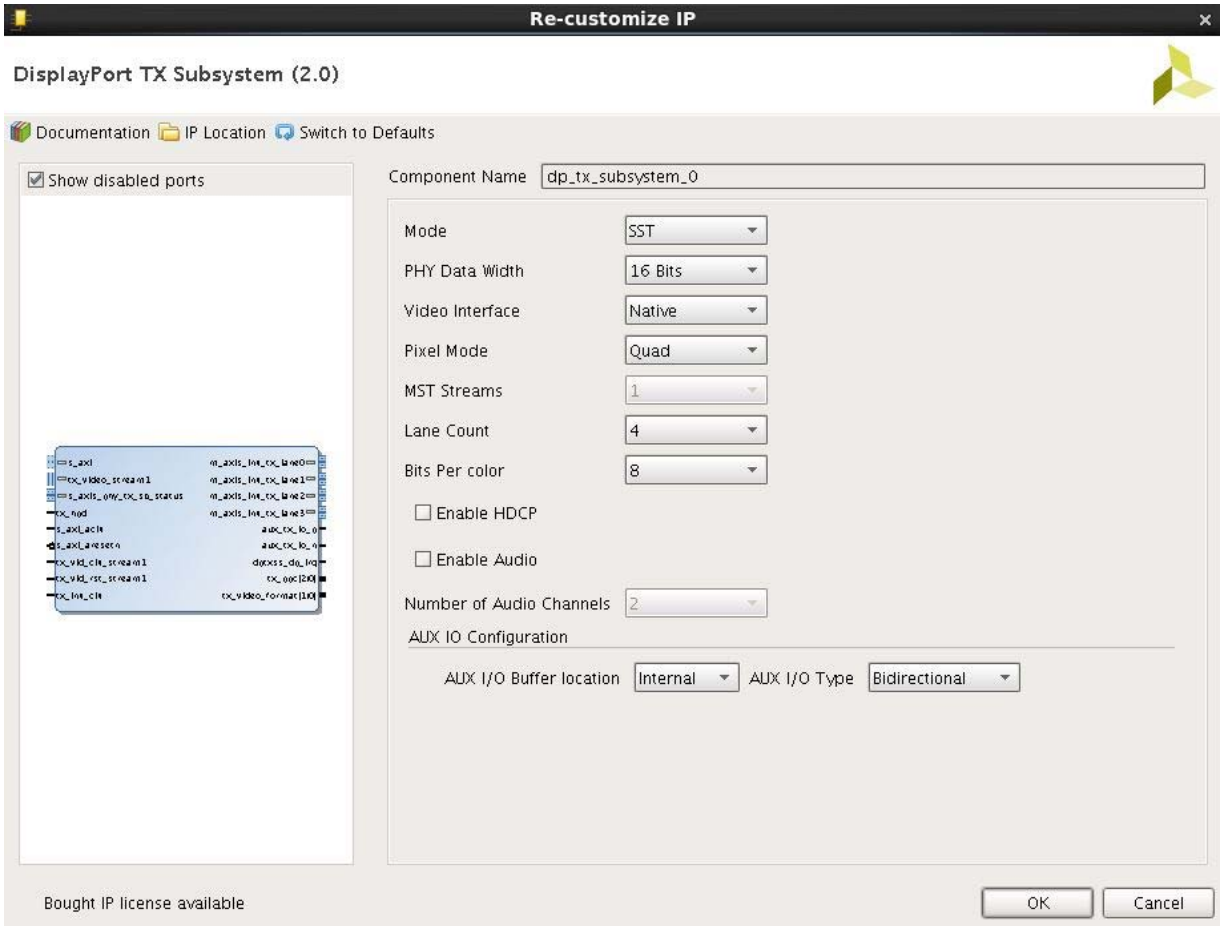

Figure 4-1: **Configuration Screen**

- **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_0 is used as internal module name and should not be used for the component name. The default is dp_tx_subsystem_0.
- **Mode**: Select the desired resolution for the video stream out. The default value is SST.
- **PHY Data Width**: Select 16-bit or 32-bit GT data width.
- **Video Interface**: Select streaming or native input video interface.
- **Pixel Mode**: Enabled when native interface is selected. Select single, dual or quad pixel mode.
- **MST Streams**: Select the number of streams in MST mode.

- **Lane Count**: Select the number of lanes. Maximum pixel mode supported is aligned with lane count. Pixel mode can be changed dynamically through software but this does not affect the video streaming width.
- **Bits Per Color**: Select the desired bit per component (BPC).
- **Enable Audio**: Enables audio support.
- **Enable HDCP**: Enables HDCP encryption.
- **Audio Channels**: Select the number of audio channels.
- **AUX I/O Buffer location**: Select buffer location for AUX channel
- **AUX I/O Type**: Selection of Bi-Directional or Uni directional buffer type.

User Parameters

Table 4-1 shows the relationship between the GUI fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console). The line rate and pixel mode support in the DisplayPort TX Subsystem is through software. Maximum pixel mode support is aligned to the lane count.

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
Mode	MODE	SST
PHY Data Width	PHY_DATA_WIDTH	16
Video Interface	VIDEO_INTERFACE	AXI4 Stream
Pixel Mode	PIXEL_MODE	Quad
MST Streams	NUM_STREAMS	1
Lane Count	LANE_COUNT	4
Bits Per Color	BITS_PER_COLOR	8
Enable HDCP	HDCP_ENABLE	0
Enable Audio	AUDIO_ENABLE	0
Number Of Audio Channels	AUDIO_CHANNELS	2
AUX IO Buffer Location	AUX_IO_LOC	Internal
AUX IO Type	AUX_IO_TYPE	Bidirectional

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

There are no required constraints for this core.

Device, Package, and Speed Grade Selections

See IP Facts for details about supported devices.

Clock Frequencies

See Clocking in Chapter 3 for more details about clock frequencies.

Clock Management

There are no specific clock management constraints.

Clock Placement

There are no specific clock placement constraints.

Banking

There are no specific banking constraints.

Transceiver Placement

There are no specific transceiver placement constraints.

I/O Standard and Placement

For details on the specific I/O constraints, see the *DisplayPort Product Guide* (PG064) [Ref 10].

Simulation

There is no example design simulation support for DisplayPort TX Subsystem.

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the DisplayPort Subsystem, the Xilinx Support web page (Xilinx Support web page) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the DisplayPort Subsystem. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name

- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the DisplayPort Subsystem

AR: 59384

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address DisplayPort Subsystem design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. Xilinx recommends having an external auxiliary channel analyzer to understand the transactions between the Source and Sink cores.

General Checks

- Check the DisplayPort Source is DisplayPort1.2a compliant.
- Make sure you are using proper DisplayPort1.2a certified cable which is tested to run at 5.4 Gb/s.
- Ensure that the Signal Integrity of the lines is as per the DisplayPort standards for the AUX, TX, and Clock Input lines.

Transmit – Training Issue

This section contains debugging steps for issues with the clock recovery or channel equalization at sink and if the Training Done is Low.

- Try with a working sink such as the DisplayPort Analyzer sink device.
- Use a DisplayPort v1.2a certified cable. Change the cable and check again.
- Put a DisplayPort AUX Analyzer in the Transmit path and check if the various training stages match with the one's mentioned in Main Link Setup and Management in Chapter 3.
- Probe the `lnk_clk` output and check if the SI of the clock is within the Phase Noise mask of the respective GT.
- Check status registers in the Video PHY Controller for Reset done (0x0020) and PLL lock Status (0x0018)

Transmit – Main Link Problem After Training

This section contains debugging steps if the monitor is not displaying video even after a successful training, or if the monitor display is noisy and has many errors.

- Perform a software reset on the register 0x01C and check if the video is proper now.
- Check if the MAIN_STREAM_ENABLE register is set to 1.
- Ensure that the MSA parameters match the Video being sent by the TX.
- Check the video pixel clock generation. Ensure that the Video Clock is based on the resolution being sent.

- Dump the DisplayPort source registers and compare against a working log.
- Check the symbol and disparity errors in the Sink through DPCD registers. This could be due to cable issue or PHY (GT) alignment issue.

Transmit – Audio

This section contains debugging steps for issues with audio communication.

- Check if MAUD and NAUD registers are correctly programmed and `aud_clk` is calculated as expected to be 512 × fs.
- Follow steps mentioned in Programming DisplayPort Source in Chapter 3.
- Check if the TX_AUDIO_CHANNELS register value matches with the input audio samples sent
- Check if the TX_AUDIO_INFO_DATA is correctly formatted as per CEA 861-C info frame specification.
- Ensure all the inputs data bits of `s_axis_audio_ingress_tdata` and `s_axis_audio_ingress_tid` are correctly sent as per the format specified.

Application Software Development

The software is capable of detecting an MST/SST RX connected to the subsystem based on if a MST or SST software flow is executed. Figure B-1 shows the DisplayPort TX Subsystem application software flow.

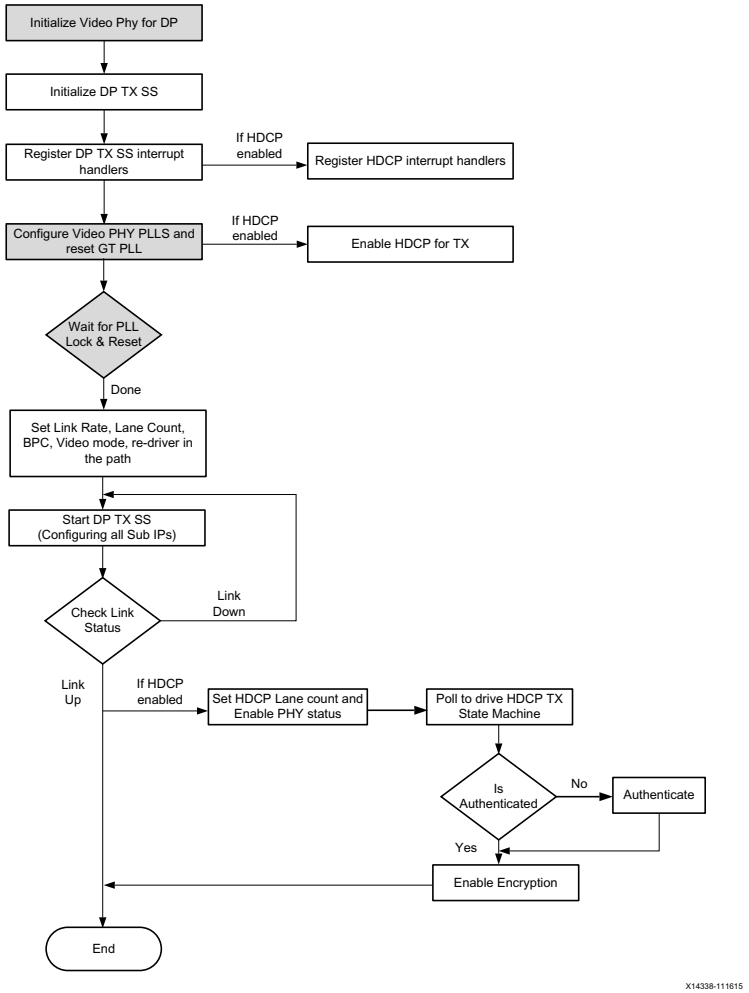

Figure B-1: **Software Flow**

Note: Video PHY is external to DisplayPort TX Subsystem and must be configured for the subsystem to work as expected. For more details on Video PHY configuration, see the *Video PHY Product Guide* (PG230) [Ref 15].

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. VESA *DisplayPort Standard v1.2a*, December 22, 2009
2. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
3. *Vivado Design Suite User Guide: Designing with IP* (UG896)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *ISE to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
8. *Vivado Design Suite User Guide: Implementation* (UG904)
9. *AXI Reference Guide* (UG1037)
10. *DisplayPort Product Guide* (PG064)
11. *AXI4-Stream to Video Out LogiCORE IP Product Guide* (PG044)
12. *Video Timing Controller LogiCORE IP Product Guide* (PG016)
13. *HDCP Controller Product Guide* (PG224)
14. AXI Timer Product Guide (PG079)
15. Video PHY Controller Product Guide (PG230)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/14/2017	2.0	Updated 7 series (GTHE2) and Artix-7 support in IP Facts.
06/07/2017	2.0	Vivado Design Suite release for DisplayPort TX v2.0.
04/05/2017	2.0	• Updated Supported Device Family in IP Facts table. • Added In-band bullet in Unsupported Features section. • Added DisplayPort Registers Source Core in Product Specification chapter. • Added Source Overview in Designing with the Core chapter. • Added Reduced Blanking in Designing with the Core chapter. • Updated Hardware Debug section in Debug Appendix.
12/20/2016	2.0	Added HDCP note for Supported Device Family in IP Facts table.

Date	Version	Revision
11/30/2016	2.0	Added Important note in Standards section.
10/05/2016	2.0	Updated HDCP features.
04/06/2016	2.0	Added support for 16 bit GT interface and native with pixel mode.
11/18/2015	1.0	Initial Xilinx release.

Please Read: Important Legal Notices