DisplayPort 1.4 TX Subsystem v3.0

Product Guide

Vivado Design Suite

PG299 (v3.0) January 13, 2021

XILINX®

Table of Contents

Introduction

The Xilinx® DisplayPort 1.4 TX Subsystem implements the functionality of a video source as defined by the Video Electronics Standards Association (VESA) DisplayPort standard v1.4 and supports driving resolutions of up to Full Ultra HD (FUHD) 8K at 60 fps. The Xilinx DisplayPort subsystem provides highly integrated IP blocks requiring very little customization.

Features

- Support for DisplayPort Source (TX) transmissions
- Supports multi-stream transport (MST) and single stream transport (SST)
- Dynamic lane support (1, 2, or 4 lanes)
- Dynamic link rate support (1.62/2.7/5.4/8.1 Gb/s)
- Dynamic support for 6, 8, 10, 12, or 16 bits per component (BPC)
- Dynamic support for RGB/YCbCr444/YCbCr422/YCbCr420 color formats
- Supports 16-bit Video PHY (GT) interface
- Supports 2 to 8 channel audio with 44/48 kHz sample rates
- Supports HDCP 1.3 and HDCP 2.2/2.3 encryption in SST
- Supports native or AXI4-Stream video input interface
- Pixel mode support in native video interface mode
- Supports Linear PCM 2-channel audio format
- Supports single audio stream in MST mode
- Supports SDP packet for static HDR mode
- Supports eDP v1.4b
- Supports DSC and/or forward error correction (FEC)
- Supports YUV420 colorimetry
- Supports adaptive sync
- Supports in-band 3D stereo

IP Facts

Subsystem IP Facts Table	
Subsystem Specifics	
Supported Device Family[1]	UltraScale+™ Families (GTHE4, GTYE4) UltraScale™ Families (GTHE3) Versal™ ACAP (GTYE5)
Supported User Interfaces	AXI4-Stream, AXI4-Lite, Native video
Resources	Performance and Resource Use web page
Provided with Subsystem	
Design Files	Hierarchical subsystem packaged with DisplayPort TX core and other IP cores
Example Design	Vivado® IP integrator
Test Bench	Not Provided
Constraints File	IP cores delivered with XDC files
Simulation Model	Not Provided
Supported S/W Driver	Standalone, Linux[2]
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	
Release Notes and Known Issues	Master Answer Record: 70295
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. (`<install_directory>/Vitis/<release>/data/embeddedsw/doc/xilinx_drivers.htm`). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of third-party tools, see the Xilinx Design Tools: Release Notes Guide.

XILINX

Chapter 2

Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - Register Space
  - Clocking
  - Resets
  - Programming Sequence
  - Customizing and Generating the Subsystem
  - Chapter 6: Example Design

Core Overview

The DisplayPort 1.4 TX Subsystem is a feature-rich, hierarchically packaged subsystem with a DisplayPort (TX) core ready to use in applications that require a DisplayPort 1.4 compliant source.

The DisplayPort 1.4 TX Subsystem operates in the following video modes:

- Single stream transport (SST)
- Multi-stream transport (MST) up to 4 streams

RECOMMENDED: *Xilinx® recommends a redriver such as the SN65DP141 along with the TX subsystem solution. Using a redriver can make compliance testing simpler; however, using a redriver is not required.*

The following table shows the core support for UltraScale™ and UltraScale+™ families. For more information on the device constraint/dependency, see the *Video PHY Controller LogiCORE IP Product Guide* (PG230) and respective device family datasheets. Speed grade and temperature information can be found in the *UltraScale Architecture and Product Data Sheet: Overview* (DS890) and the *Defense-Grade UltraScale Architecture Data Sheet: Overview* (DS895).

Table 1: **Core Support for UltraScale and UltraScale+ Devices**

Device Family	Device Data Sheet	Speed Grade	Without MST (or) Without HDCP 1.3/2.2/2.3	With MST (or) With HDCP 1.3/2.2/2.3
Kintex UltraScale	*Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892)	-1	5.4 Gb/s	2.7 Gb/s
		-2, -3	8.1 Gb/s	5.4 Gb/s
Virtex UltraScale	*Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS893)	-1	5.4 Gb/s	2.7 Gb/s
		-2, -3	8.1 Gb/s	5.4 Gb/s
Kintex UltraScale+	*Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)	-1LI (V_{CCINT} = 0.72 V)	2.7 Gb/s	
		-1LI (V_{CCINT} = 0.85 V)	5.4 Gb/s	
		-2LE (V_{CCINT} = 0.72 V)	5.4 Gb/s	
		-1, -1E, -1I, -1M, -1Q	5.4 Gb/s	
		-2, -2E, -2I, -3, -3E	8.1 Gb/s	
Zynq UltraScale+ MPSoC	*Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925)	-1LI (V_{CCINT} = 0.72 V)	2.7 Gb/s	
		-1LI (V_{CCINT} = 0.85 V)	5.4 Gb/s	
		-2LE (V_{CCINT} = 0.72 V)	5.4 Gb/s	
		-1, -1E, -1I, -1M, -1Q	5.4 Gb/s	
		-2, -2E, -2I, -3, -3E	8.1 Gb/s	
Virtex UltraScale+	*Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS923)	-1 (V_{CCINT} = 0.85 V)	5.4 Gb/s	
		-2 (V_{CCINT} = 0.72 V)	5.4 Gb/s	
		-2, -3	8.1 Gb/s	
Zynq UltraScale+ RFSoC	*Zynq UltraScale+ RFSoC Data Sheet: DC and AC Switching Characteristics* (DS926)	-1LI (V_{CCINT} = 0.72 V)	2.7 Gb/s	
		-1LI (V_{CCINT} = 0.85 V)	5.4 Gb/s	
		-1, -1E, -1I, -1M	5.4 Gb/s	
		-2LE (V_{CCINT} = 0.72 V)	5.4 Gb/s	
		-1 (V_{CCINT} = 0.72 V)	5.4 Gb/s	
		-2 (V_{CCINT} = 0.72 V)	5.4 Gb/s	
		-1 (V_{CCINT} = 0.85 V)	5.4 Gb/s	
		-2, -2E, -2I	8.1 Gb/s	

The following table shows the core support for Versal™ devices.

Table 2: **Core Support for Versal Devices**

Device	Device Data Sheet	Speed Grade	Without MST (or) Without HDCP 1.3/2.2/2.3	With MST (or) With HDCP 1.3/2.2/2.3
Versal ACAP	*Versal Prime Series Data Sheet: DC and AC Switching Characteristics* (DS956) *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics* (DS957)	All -1 speed grades and -2LP	5.4 Gb/s	2.7 Gb/s
		All -2 speed grades except -2LP	8.1 Gb/s	5.4 Gb/s
		All -3 speed grades	8.1 Gb/s	

Unsupported Features

The following features of the standard are not supported in the subsystem:

- Video AXI4-Stream interface is not scalable with dynamic pixel mode selection
- Dual-pixel splitter is not supported in native video mode
- HDCP is not supported in MST mode
- iDP
- Global Time Code (GTC)
- Non-LPCM audio
- 16/32 channel audio
- Interlaced video in AXI4-Stream interface

Licensing and Ordering

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

Note: 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 subsystem.

For more information about this subsystem, visit the DisplayPort product web page.

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

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 an 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

The subsystem requires video data in either AXI4-Stream or native interface formats. AXI4-Stream is the preferred format. Both use cases are described in the following sections.

AXI4-Stream Video Interface

When configured with the AXI4-Stream interface, the subsystem is packaged with the following sub-cores:

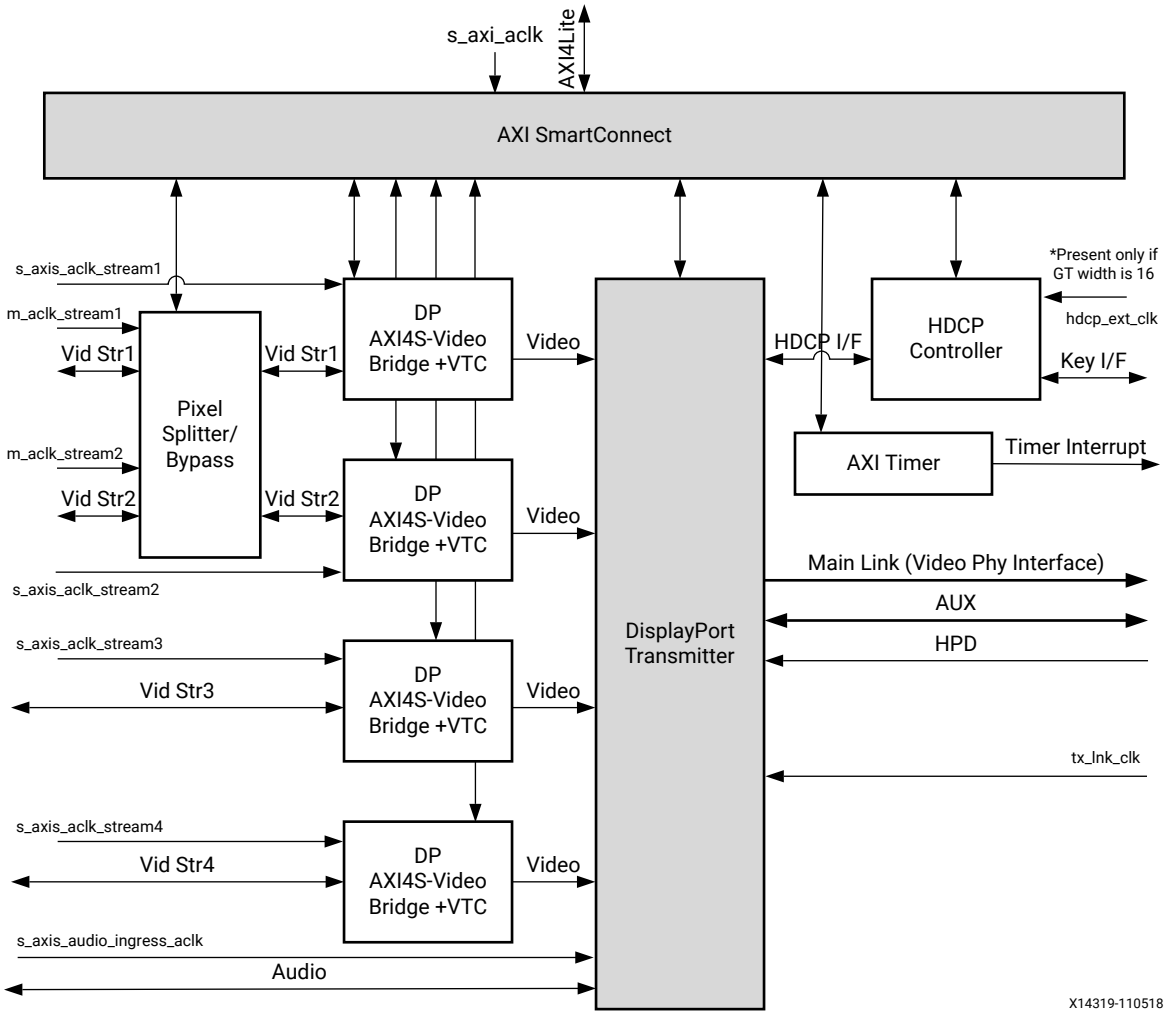
- DisplayPort Transmitter core
- Video Timing Controller (VTC)
- DisplayPort AXI4-Stream to Video Bridge
- HDCP core with AXI Timer when HDCP feature is enabled

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

AXI4-Stream is defined in *AXI4-Stream Video IP and System Design Guide* (UG934). The benefit of using AXI4-Stream is that the video timing is removed from the bus. This allows for simpler clocking, small amounts of burstiness, improved debugging and for the core to verify the incoming data using internal locked signals.

Because the DisplayPort 1.4 TX Subsystem is hierarchically packaged, you select the parameters and the subsystem creates the required hardware. The subsystem includes a multi-pixel AXI4-Stream video protocol interface and outputs the video using the DisplayPort v1.4 protocol. The subsystem works with the Video PHY Controller (*Video PHY Controller LogiCORE IP Product Guide* (PG230)) configured for the DisplayPort protocol. The following figure shows the subsystem architecture of the subsystem assuming MST with four video streams.

Figure 1: **DisplayPort 1.4 TX Subsystem AXI4-Stream Video Interface Block Diagram**



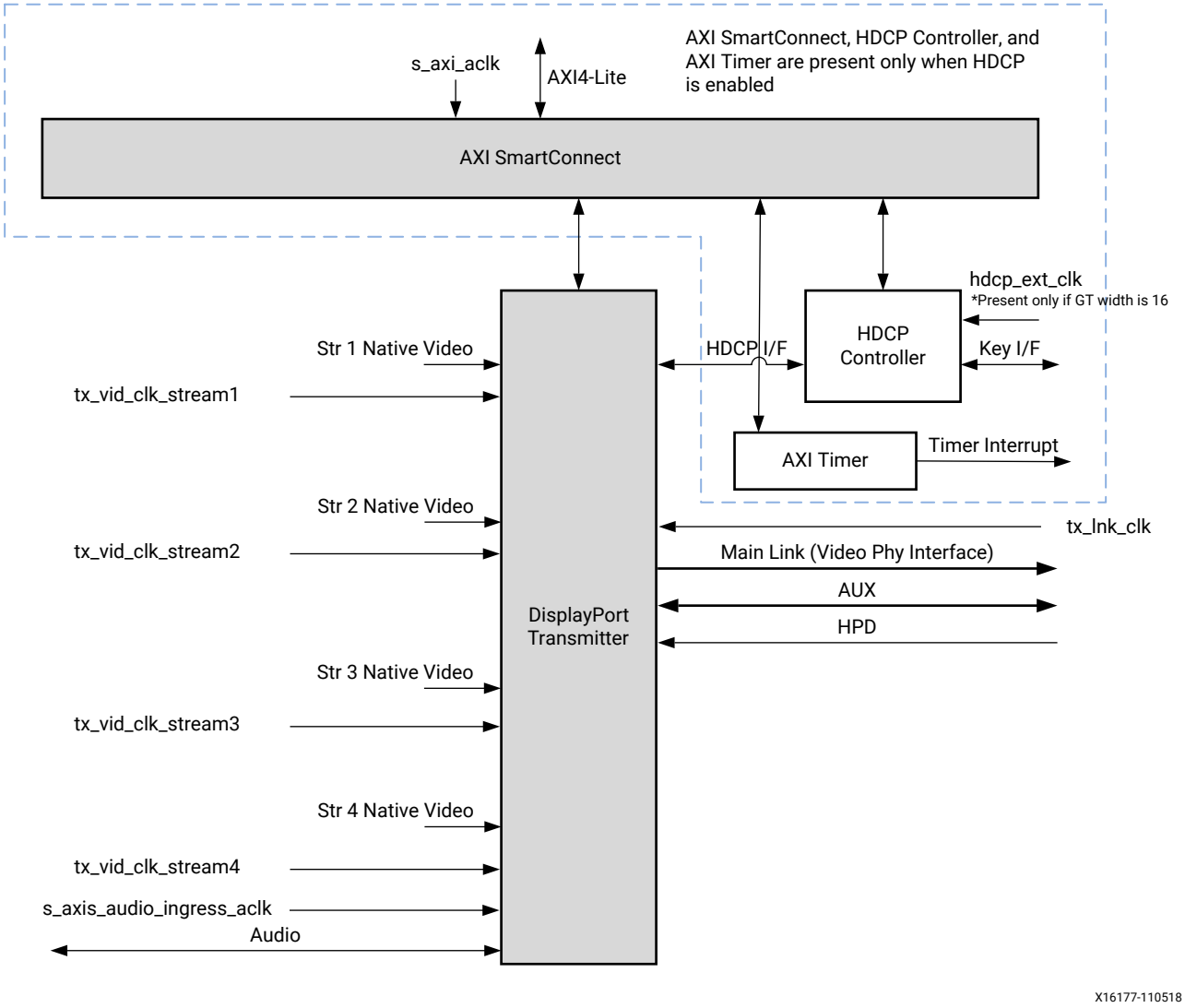
Related Information

Pixel Mapping Examples on AXI4-Stream Interface

Native Video Interface

When the native video interface is enabled, the subsystem is packaged with only one sub-core, the DisplayPort TX core. The following figure 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 1.4 TX Subsystem outputs the video using the DisplayPort v1.4 protocol and works in conjunction with Video PHY Controller configured for the DisplayPort protocol.

Figure 2: **DisplayPort 1.4 TX Subsystem Native Video Block Diagram**



Related Information

Pixel Mapping on Native Video Interface

Subsystem Sub-core Descriptions

The subsystem is comprised of multiple sub-cores, when AXI4-Stream mode is selected. The following sections provide a brief overview of these sub-cores.

DisplayPort AXI4-Stream to Video Bridge IP Core

The DisplayPort AXI4-Stream to video bridge maps the video over the AXI4-Stream interface to the native video format required by the DisplayPort TX IP core. The bridge uses the Xilinx® AXI4-Stream to Video Out IP core to convert the format from AXI4-Stream to DisplayPort native video. See the *AXI4-Stream to Video Out LogiCORE IP Product Guide* (PG044) for information on this core.

For details about video over AXI4-Stream, see the *Vivado Design Suite: AXI Reference Guide* (UG1037) and *AXI4-Stream Video IP and System Design Guide* (UG934).

Video Timing Controller IP Core

The Video Timing Controller IP core is used for generation of video timing. This core is required when the subsystem is configured in the AXI4-Stream interface mode. For details on this core, see the *Video Timing Controller LogiCORE IP Product Guide* (PG016).

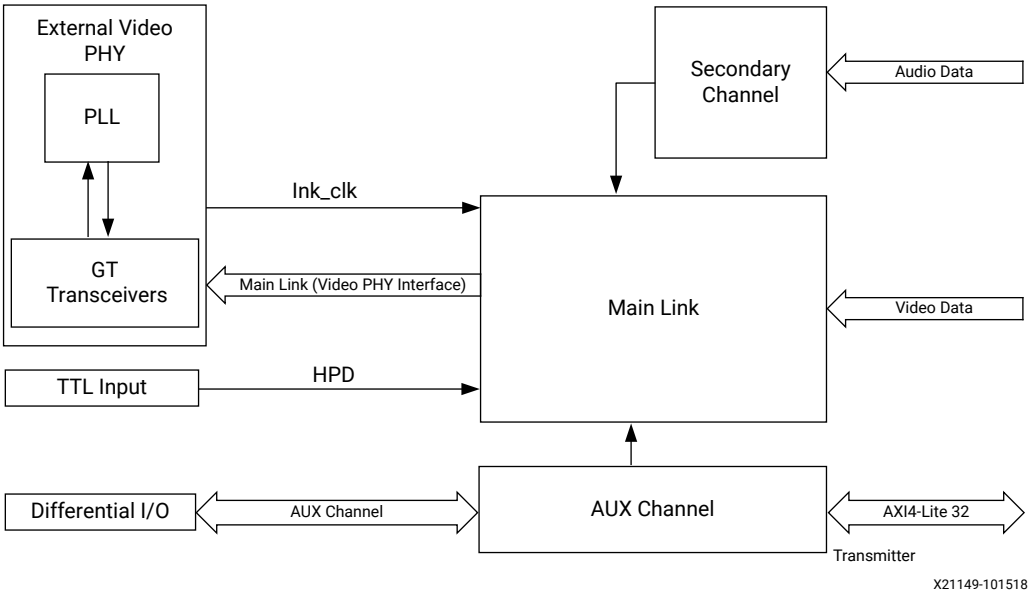
IMPORTANT! *You must program correct front porch and back porch blanking period generation.*

DisplayPort Transmit IP Core

The DisplayPort TX block is delivered as part of the DisplayPort 1.4 TX Subsystem and contains the following components, also shown in the following figure:

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

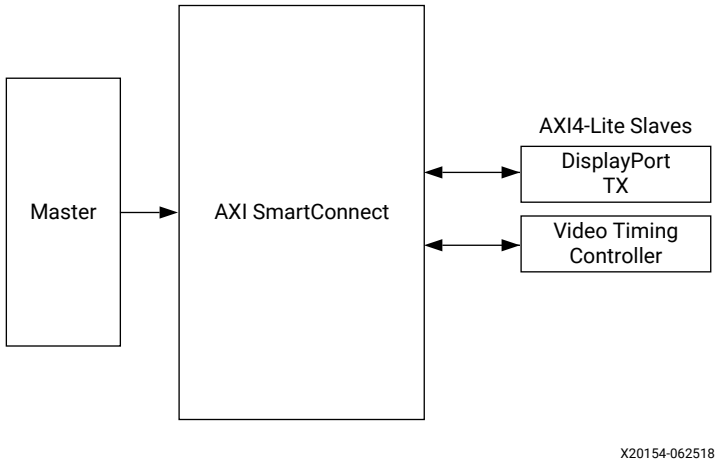
Figure 3: **DisplayPort Transmit Core Block Diagram**



AXI SmartConnect IP Core

The subsystem uses the Xilinx® AXI Smartconnect IP core, as a smartconnect which contains an AXI4-Lite interface. For more details on the AXI Smartconnect functionality, see the *SmartConnect LogiCORE IP Product Guide* (PG247). The following figure shows the AXI slave structure within the DisplayPort 1.4 TX Subsystem.

Figure 4: **AXI4-Lite Interconnect within DisplayPort 1.4 TX Subsystem**



***Note*:**

- The Video Timing Controller IP core and Dual splitter are present only when subsystem is generated in AXI4-Stream interface mode.

- For MST with N streams, there are N Video Timing Controller IP cores. See Address Map Example.

Related Information

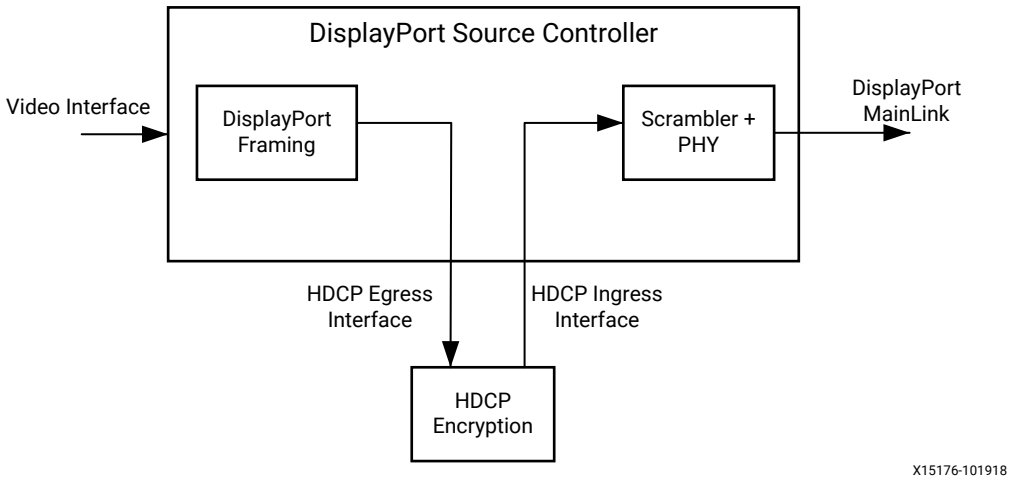
Address Map Example

HDCP Controller IP Core

The HDCP v1.3/v2.2/v2.3 protocol specifies a secure method of transmitting audiovisual content. The audiovisual content can be transmitted over a DisplayPort interface. The HDCP Controller IP core is used for data encryption along with DisplayPort transmit IP core in the DisplayPort 1.4 TX Subsystem.

The following figure shows the DisplayPort 1.4 TX Subsystem with the HDCP controller.

Figure 5: **DisplayPort TX with HDCP Controller**



For more details, on the HDCP 1.x v1.0 IP core, see the *HDCP 1.x Product Guide* (PG224). For more details on the HDCP 2.2 v1.0 IP core, see the *HDCP 2.2 LogiCORE IP Product Guide* (PG249).

AXI Timer IP Core

A 32-bit AXI Timer IP core is used in the DisplayPort 1.4 TX Subsystem. When the HDCP controller is enabled for encryption the AXI Timer can be accessed through the AXI4 master interface for basic timer functionality in the system.

Standards

The DisplayPort 1.4 TX Subsystem is compatible with the DisplayPort v1.4 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 Use

For full details about performance and resource use, visit the Performance and Resource Use web page.

Port Descriptions

The DisplayPort 1.4 TX Subsystem ports are described in the following tables.

AXI4-Lite Interface

Table 3: **AXI4-Lite Interface**

Port Name	I/O	Description
s_axi_aclk	I	AXI Bus clock
s_axi_aresetn	I	AXI reset. Active-Low.
s_axi_awadd[18:0]	I	Write address
s_axi_awpro[2:0]	I	Protection Type
s_axi_awvalid	I	Write address Valid
s_axi_awready	O	Write address Ready
s_axi_wdata[31:0]	I	Write data
s_axi_wstrb[3:0]	I	Write Strobe
s_axi_wvalid	I	Write data valid
s_axi_wready	O	Write data ready
s_axi_bresp[1:0]	O	Write response
s_axi_bvalid	O	Write response valid
s_axi_bready	I	Write response ready
s_axi_araddrs_axi_araddr[18:0]	I	Read address

Table 3: **AXI4-Lite Interface** *(cont'd)*

Port Name	I/O	Description
s_axi_arprot[2:0]	I	Read protection type
s_axi_arvalid	I	Read address valid
s_axi_arready	O	Read address ready
s_axi_rdata[31:0]	O	Read data
s_axi_rresp[1:0]	O	Read data response
s_axi_rvalid	O	Read data valid
s_axi_rready	I	Read data ready

AXI4-Stream Interface

This interface is enabled when the AXI4-Stream interface is selected.

Table 4: **AXI4-Stream Interface**

Port Name	I/O	Description
s_axis_aclk_stream1	I	AXI4-Stream clock
s_axis_aresetn_stream1	I	AXI4-Stream reset. Active-Low.
s_axis_video_stream1_tdata[191:0]	I	Video data input. Maximum width is 192.
s_axis_video_stream1_tlast	I	Video end of line
s_axis_video_stream1_tready	O	AXI4-Stream tready output
s_axis_video_stream1_tuser	I	Video start of frame
s_axis_video_stream1_tvalid	I	Video valid

Native Video Interface

This interface is enabled when native video is selected.

Table 5: **Native Video Interface**

Port Name	I/O	Description
tx_video_stream1_tx_vid_vsync	I	Vertical sync pulse. Active on the rising edge.
tx_video_stream1_tx_vid_hsync	I	Horizontal sync pulse. Active on the rising edge.
tx_video_stream1_tx_vid_enable	I	User data video enable
tx_video_stream1_tx_vid_pixel0[47:0]	I	Video data
tx_video_stream1_tx_vid_pixel1[47:0]	I	Video data
tx_video_stream1_tx_vid_pixel2[47:0]	I	Video data
tx_video_stream1_tx_vid_pixel3[47:0]	I	Video data
tx_video_stream1_tx_vid_oddeven	I	Odd/even field select. Indicates an odd (1) or even (0) field polarity. If not used, this pin should be connected to ground.

MST Interface

Table 6: **MST Stream (<n> = stream number 2 to 4)**

Port Name	I/O	Description
s_axis_aclk_stream<n>	I	MST stream clock.
s_axis_aresetn_stream<n>	I	MST stream reset. Active-Low.
s_axis_video_stream<n>_tdata[191:0]	I	MST stream video data input.
s_axis_video_stream<n>_tlast	I	MST stream video end of line.
s_axis_video_stream<n>_tready	O	MST stream input ready.
s_axis_video_stream<n>_tuser	I	MST stream video start of frame.
s_axis_video_stream<n>_tvalid	I	MST stream video valid.
m_aclk_stream1	I	Video pipe clock for stream1. Used in MST configuration.
m_aresetn_stream1	I	Active-Low video pipe reset for stream 1. Used in MST configuration.
m_aclk_stream2	I	Video pipe clock for stream 2. Used in MST configuration.
m_aresetn_stream2	I	Active-Low video pipe reset for stream 2. Used in MST configuration.
tx_vid_clk_stream<n>	I	User data clock for MST stream n.
tx_vid_rst_stream<n>	I	Active-High user video reset.
tx_video_stream<n>_tx_vid_vsync	I	Vertical sync pulse. Active on the rising edge.
tx_video_stream<n>_tx_vid_hsync	I	Horizontal sync pulse. Active on the rising edge
tx_video_stream<n>_tx_vid_enable	I	User data video enable.
tx_video_stream<n>_tx_vid_pixel0[47:0]	I	Video data
tx_video_stream<n>_tx_vid_pixel1[47:0]	I	Video data
tx_video_stream<n>_tx_vid_pixel2[47:0]	I	Video data
tx_video_stream<n>_tx_vid_pixel3[47:0]	I	Video data
tx_video_stream<n>_tx_vid_oddeven	I	Odd/even field select. Indicates an odd (1) or even (0) field polarity. If not used, this pin should be connected to ground.

User Ports

Table 7: **User Ports**

Port Name	I/O	Description
tx_vid_clk_stream1	I	User video clock
tx_vid_rst_stream1	I	User video reset. Active-High.
tx_hpd	I	Hot-plug detect signal to TX from RX

Audio AXI4-Stream Interface

Table 8: **Audio AXI4-Stream Interface**

Port Name	I/O	Description
s_axis_audio_ingress_aclk	I	AXI4-Stream clock.
s_axis_audio_ingress_aresetn	I	Active-Low reset.
s_axis_audio_ingress_tdata[31:0]	I	AXI4-Stream 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]	I	[3:0] - Audio Channel ID [7:4] - Audio Packet Stream ID
s_axis_audio_ingress_tvalid	I	Valid indicator for audio data from master.
s_axis_audio_ingress_tready	O	Ready indicator from DisplayPort source.

External Video PHY Sideband Status Interface

Table 9: **External Video PHY Sideband Status Interface**

Port Name	I/O	Description
s_axis_phy_tx_sb_status_tdata[7:0]	O	Sideband status to Video PHY
s_axis_phy_tx_sb_status_tready	I	Sideband status ready input from Video PHY
s_axis_phy_tx_sb_status_tvalid	O	Sideband status data valid to Video PHY

External Video PHY Clock Interface

Table 10: **External Video PHY Clock Interface**

Port Name	I/O	Description
tx_lnk_clk	I	Link clock input from external Video PHY
tx_enc_clk	I	8B10B encoder clock from external Video PHY and applicable only for Versal™ device designs. Refer to Versal Device Support.

Related Information

Versal Device Support

External Video PHY Lane n Interface

Table 11: **External Video PHY Lane n Interface**

Port Name[1]	I/O	Description
m_axis_lnk_tx_lanen_tdata[31:0]	O	Lanen Data to External Video PHY
m_axis_lnk_tx_lanen_tvalid	O	Lanen Data Valid to External Video PHY
m_axis_lnk_tx_lanen_tready	I	Lanen Data Ready from External Video PHY
m_axis_lnk_tx_lanen_tuser[11:0]	O	Lanen User data out to External Video PHY

Notes:

1. n = 0 to Lane_Count -1.

External Versal PHY Control Interface

Table 12: **External Versal PHY Control Interface**

Port Name	I/O	Description
tx_gt_ctrl_out[31:0]	O	External Versal device PHY control output and mapped to 0x04C AXI4-Lite register with R/W access. [0] - Reset [3:1] - Line Rate [6:4] - Lane Count [7] - Reserved [12:8] - Vswing [17:13] - Precursor [22:18] - Postcursor [31:23] - Reserved

HDCP Key Interface

Table 13: **Interrupt Interface**

Port Name	I/O	Description
hdcp_ext_clk	I	HDCP external clock
hdcp_key_aclk	I	HDCP key clock
hdcp_key_aresetn	I	Key Interface reset. Active-Low
hdcp_key_tdata[63:0]	I	AXI4-Stream Key Tdata
hdcp_key_last	I	AXI4-Stream Key Tlast
hdcp_key_tready	O	AXI4-Stream Key Tready
hdcp_key_tuser[7:0]	I	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	I	AXI4-Stream Key TValid
reg_key_sel[2:0]	O	To select the one of the eight sets of 40 keys.

Table 13: **Interrupt Interface** *(cont'd)*

Port Name	I/O	Description
start_key_transmit	O	An Active-High pulse that is used to start key transmit.

AUX Signals

Table 14: **AUX Signals**

Port Name	I/O	Description
aux_tx_io_n	O	Negative polarity AUX Manchester-II data.
aux_tx_io_p	O	Positive polarity AUX Manchester-II data.
aux_tx_channel_in_p	I	Positive polarity AUX channel input. Valid when AUX IO Type is unidirectional
aux_tx_channel_in_n	I	Negative polarity AUX channel input. Valid when AUX IO Type is unidirectional
aux_tx_channel_out_p	O	Positive polarity AUX channel Output. Valid when AUX IO Type is unidirectional
aux_tx_channel_out_n	O	Negative Polarity AUX channel output. Valid when AUX IO Type is unidirectional
aux_tx_data_out	O	AUX data out. Valid when AUX IO buffer location is external
aux_tx_data_in	I	AUX data input. Valid when AUX IO buffer location is external
aux_tx_data_en_out_n	O	AUX data output enable. Active-Low. Valid only when AUX IO buffer location is external

Interrupt Interface

Table 15: **Interrupt Interface**

Port Name	I/O	Description
dptxss_dp_irq	O	DisplayPort 1.4 TX IP interrupt out
dptxss_hdcp_irq	O	HDCP IP interrupt out
dptxss_timer_irq	O	AXI Timer IP interrupt output valid only when HDCP is enabled

Register Space

This section details registers available in the DisplayPort 1.4 TX Subsystem. The address map is split into following regions:

- VTC 0 (Up to 3 for 4 streams in MST)
- DisplayPort TX IP

- HDCP Controller

Video Timing Controller Registers

For details about the Video Timing Controller (VTC) registers, see the *Video Timing Controller LogiCORE IP Product Guide* (PG016).

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 the following tables 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 and the base addresses are configured by the AXI Interconnect.

Link Configuration Field

Table 16: **Link Configuration Field**

Offset	Access Type	Description
0x000	R/W	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 • 0x1E = 8.1 Gb/s
0x004	R/W	LANE_COUNT_SET. Sets the number of lanes used by the source in transmitting data. [4:0] - Set to 1, 2, or 4
0x008	R/W	ENHANCED_FRAME_EN [0] - Set to 1 by the source to enable the enhanced framing symbol sequence.

Table 16: **Link Configuration Field** *(cont'd)*

Offset	Access Type	Description
0x00C	R/W	TRAINING_PATTERN_SET. Sets the link training mode. [2:0] - Set the link training pattern according to the 2-bit code. • 000 = Training off • 001 = Training pattern 1, used for clock recovery • 010 = Training pattern 2, used for channel equalization • 011 = Training pattern 3, used for channel equalization • 111 = Training pattern 4, used for channel equalization
0x010	R/W	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	R/W	SCRAMBLING_DISABLE. Set to 1 when the transmitter has disabled the scrambler and transmits all symbols. [0] - Disable scrambling.
0x01C	WO	SOFTWARE_RESET. Reads return zeros. [0] - Soft Video Reset: When set, video logic is reset (stream 1). [7] - AUX Soft Reset. When set, AUX logic is reset.
0x020	R/W	Custom 80-bit quality pattern Bits[31:0]
0x024	R/W	Custom 80-bit quality pattern Bits[63:32]
0x028	R/W	[31:16] - Reserved [15:0] - Customer 80-bit quality pattern Bits[80:64]

Core Enables

Table 17: **Core Enables**

Offset	Access Type	Description
0x080	R/W	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	R/W	MAIN_STREAM_ENABLE. Enable the transmission of main link video information. [0] - When set to 0, the active lanes of the DisplayPort transmitter outputsW 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.
0x0C0	WO	FORCE_SCRAMBLER_RESET. Reads from this register always return 0x0. [0] - 1 forces a scrambler reset.

Table 17: **Core Enables** *(cont'd)*

Offset	Access Type	Description
0x0D0	R/W	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.

Core ID

Table 18: **Core ID**

Offset	Access Type	Description
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 value for the protocol and core is *VESA DisplayPort Standard v1.4* protocol with a Transmit core: 32'h01_04_00_00.

AUX Channel Interface

Table 19: **AUX Channel Interface**

Offset	Access Type	Description
0x100	R/W	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	R/W	AUX_ADDRESS. Specifies the address for the current AUX channel command. [19:0] - 20-bit address for the start of the AUX Channel burst.

Table 19: **AUX Channel Interface** *(cont'd)*

Offset	Access Type	Description
0x10C	R/W	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). [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. [7:0] - Clock divider value. 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.
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. [3:2] • 00 = I2C ACK • 01 = I2C NACK • 10 = I2C DEFER [1:0] • 00 = AUX ACK • 01 = AUX NACK • 10 = AUX DEFER
0x13C	R/W	AUX_REPLY_COUNT. Provides an internal counter of the number of AUX reply transactions received on the AUX Channel. Writing to this register clears the count. [7:0] - Current reply count.

Table 19: **AUX Channel Interface** *(cont'd)*

Offset	Access Type	Description
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. [13] - VBLANK_STREAM4: VBlank interrupt of video stream 4. Set when the video stream 4 is in vertical blanking period. Valid only in MST mode. [12] - VBLANK_STREAM3: VBlank interrupt of video stream 3. Set when video stream 3 is in vertical blanking period. Valid only in MST mode. [11] - VBLANK_STREAM2: VBlank interrupt of video stream 2. Set when video stream 2 is in vertical blanking period. Valid only in MST mode. [10] - VBLANK_STREAM1: VBlank interrupt of video stream 1. Set when video stream 1 is in vertical blanking period. [9] - Audio packet ID mismatch interrupt, sets when incoming audio packet ID over AXI4-Stream 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 ms. [0] - HPD_IRQ: An IRQ framed with the proper timing on the HPD signal has been detected.
0x144	R/W	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. [13] - VBLANK_STREAM4: Mask VBlank interrupt of video stream 4. Valid only in MST mode. [12] - VBLANK_STREAM3: Mask VBlank interrupt of video stream 3. Valid only in MST mode. [11] - VBLANK_STREAM2: Mask VBlank interrupt of video stream 2. Valid only in MST mode. [10] - VBLANK_STREAM1: Mask VBlank interrupt of video stream 1. [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.

Table 19: **AUX Channel Interface** *(cont'd)*

Offset	Access Type	Description
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 µs.
0x154	RO	Free running counter incrementing for every 1 MHz.

Main Stream Attributes

For more details on the DisplayPort Standard, see the *VESA DisplayPort Standard v1.4*.

Table 20: **Main Stream Attributes**

Offset	Access Type	Description
0x180	R/W	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	R/W	MAIN_STREAM_VTOTAL. Provides the total number of lines in the main stream video frame. [15:0] - Total number of lines per video frame.
0x188	R/W	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 *VESA DisplayPort Standard* (VESA website). 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	R/W	MAIN_STREAM_HSWIDTH. Sets the width of the horizontal sync pulse. [14:0] - Horizontal sync width in clock cycles.
0x190	R/W	MAIN_STREAM_VSWIDTH. Sets the width of the vertical sync pulse. [14:0] - Width of the vertical sync in lines.
0x194	R/W	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	R/W	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.

Table 20: **Main Stream Attributes** *(cont'd)*

Offset	Access Type	Description
0x19C	R/W	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	R/W	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	R/W	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. [12] - 0: Default Behavior. 1: Enables mode to sync Ext packet transmission with Vsync event. [11] - Maud control (Advanced Users) [10] - Audio Only Mode. When enabled, controller inserts information/timestamp packets every 512 BS symbols. By default the value is 0. [9] - Sync/Async Mode for Audio [8] - Override Audio Clocking Mode [7:5] - Bit depth per color/component [4] - YCbCr Colorimetry [3] - Dynamic Range [2:1] - Component Format [0] - Synchronous Clock
0x1A8	R/W	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. [5:3] - Reserved [2:1] - Stereo video attribute [0] - Interlaced vertical total even
0x1AC	R/W	M-VID. If synchronous clocking mode is used, this register must be written with the M value as described in section 2.2.3 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	R/W	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 32 or 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	R/W	N-VID. If synchronous clocking mode is used, this register must be written with the N value as described in section 2.2.3 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	R/W	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 equal to the active lane 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 20: **Main Stream Attributes** *(cont'd)*

Offset	Access Type	Description

0x1BC	R/W	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
		
		***Note*:** When MST mode is selected, irrespective of the user selected value of "number of lanes", "USER_DATA_COUNT_PER_LANE" should always be calculated, considering number of lanes as 4. This is because, DP core always works in 4 lane mode in MST.

0x1C0	R/W	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	R/W	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. [6:0] - Set the value to INT((VIDEO_BW/LINK_BW)*TRANSFER_UNIT_SIZE)
0x1C8	R/W	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.
0x1CC	R/W	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) • [6:0] - Set INIT_WAIT to 64 Else if color format is RGB/YCbCr_444/YCbCr_420 • [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) Else if color format is YCbCr_422 • [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 Else if color format is Y_Only • [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/3

PHY Configuration Status

Table 21: **PHY Configuration Status**

Offset	Access Type	Description
0x280	RO	PHY_STATUS. Provides the current status from the PHY. [31:30] - Unused, read as 0. [29:28] - Transmitter buffer status, lane 3. [27:26] - Unused, read as 0. [25:24] - Transmitter buffer status, lane 2. [23:22] - Unused, read as 0. [21:20] - Transmitter buffer status, lane 1. [19:18] - Unused, read as 0. [17:16] - Transmitter buffer status, lane 0. [15:7] - Unused, read as 0. [6] - FPGA fabric clock PLL locked. [5] - PLL for lanes 2 and 3 locked. [4] - PLL for lanes 0 and 1 locked. [3:2] - Reset done for lanes 2 and 3. [1:0] - Reset done for lanes 0 and 1.

MST Mode Registers

Table 22: **DisplayPort Source Core Configuration Space - MST Interface**

Offset	R/W	Definition
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.
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.

Table 22: **DisplayPort Source Core Configuration Space - MST Interface** *(cont'd)*

Offset	R/W	Definition
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 2.2.3 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 2.2.3 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.
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

Table 22: **DisplayPort Source Core Configuration Space - MST Interface** *(cont'd)*

Offset	R/W	Definition

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 1 for the proper transmission of interlaced sources. [0] - Set to 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.
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) [6:0] - Set INIT_WAIT to 64 else if color format is RGB/YCbCr_444 [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) else if color format is YCbCr_422 [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 else if color format is Y_Only [6:0] - 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.

Table 22: **DisplayPort Source Core Configuration Space - MST Interface** *(cont'd)*

Offset	R/W	Definition
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.
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 2.2.3 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 2.2.3 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 22: **DisplayPort Source Core Configuration Space - MST Interface** *(cont'd)*

Offset	R/W	Definition
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

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 1 for the proper transmission of interlaced sources. [0] - Set to 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.
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) [6:0] - Set INIT_WAIT to 64 else if color format is RGB/YCbCr_444 [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) else if color format is YCbCr_422 [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 else if color format is Y_Only [6:0] - 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.

Table 22: **DisplayPort Source Core Configuration Space - MST Interface** *(cont'd)*

Offset	R/W	Definition
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.
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 2.2.3 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).

Table 22: **DisplayPort Source Core Configuration Space - MST Interface** *(cont'd)*

Offset	R/W	Definition
0x5D4	RW	N-VID_STREAM4. If synchronous clocking mode is used, this register must be written with the N value as described in section 2.2.3 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

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 1 for the proper transmission of interlaced sources. [0] - Set to 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 22: **DisplayPort Source Core Configuration Space - MST Interface** *(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): [6:0] - Set INIT_WAIT to 64 else if color format is RGB/YCbCr_444 [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU) else if color format is YCbCr_422 [6:0] - Set INIT_WAIT to (TRANSFER_UNIT_SIZE - MIN_BYTES_PER_TU)/2 else if color format is Y_Only [6:0] - 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 Registers

The DisplayPort Audio registers are listed here.

Table 23: **DisplayPort Audio**

Offset	Access Type	Description
0x300	R/W	TX_AUDIO_CONTROL. Enables audio stream packets in main link and provides buffer control. [19]: Set to 1 to mute the audio over link for MST STREAM 4. NA for SST [18]: Set to 1 to mute the audio over link for MST STREAM 3. NA for SST [17]: Set to 1 to mute the audio over link for MST STREAM 2. NA for SST [16]: Set to 1 to mute the audio over link for SST. in MST, set to 1 to mute the Audio on STREAM 1 [5:4]: Audio Enable for STREAM 4:5 in MST. Default is STREAM1 [3]: Audio Enable for STREAM 4 in MST. NA for SST [2]: Audio Enable for STREAM 3 in MST. NA for SST [1]: Audio Enable for STREAM 2 in MST. NA for SST [0]: Audio Enable for SST. In MST, Audio Enable for STREAM 1
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 23: **DisplayPort Audio** *(cont'd)*

Offset	Access Type	Description
0x308	WO	TX_AUDIO_INFO_DATA. The IP can receive and buffer up to four Info Frames before they are processed and sent over the DP Link. The bits in register offset 0x6A0 indicate a full/overflow status of the internal "Info Frame Buffer", which holds these four Info Frames. [31:0] Word of each Info Frame is formatted as per CEA 861-C Info Frame. Total of eight words per each Info Frame should be written in the following order: 1st word - • [31:24] = HB3 • [23:16] = HB2 • [15:8] = HB1 • [7:0] = HB0 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. The capability to hold multiple Info Frames is useful in scenarios in which "Info Frames" corresponding to different features (such as Audio and HDR) are to be supported.
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. VSC/EXT packets should use the same address space. A total of nine words should be written in following order: First word - • [31:24] = HB3 • [23:16] = HB2 • [15:8] = HB1 • [7:0] = HB0 Second 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.

Table 23: **DisplayPort Audio** *(cont'd)*

Offset	Access Type	Description
0x6A0	RO	TX_AUDIO_INFO_FRAME_BUFFER_STATUS. Indicates full/overflow status of the internal Info Frame buffer which can hold up to 4 Info Frames (each of size 8 bytes as mentioned in the description of 0x308 register) received through offset register 0x308, those are to be processed by the IP. For information about “Info Frame”, see the description of TX_AUDIO_INFO_DATA at offset register 0x308. [0] = Internal Info Frame buffer “Full” status. • 1 - Indicates that the Info Frame buffer is full i.e. it has 4 Info Frames which are yet to be read and processed by the IP. • 0 - Indicates that the internal Info Frame buffer is ready to receive at least one Info frame through the offset register 0x308. [1] = Internal Info Frame buffer “Overflow” status. • 1 - Indicates an error condition resulting from offset register 0x308 being written with new “Info frame”, even though the above bit[0] indicating “Full” is “1”. • 0 - Indicates that there is no overflow.

HDCP 1.3 Registers

For details about the HDCP 1.3 registers, see the *HDCP 1.x Product Guide* (PG224).

HDCP 2.2/2.3 Registers

For details about the HDCP 2.2/2.3 registers, see the *HDCP 2.2 LogiCORE IP Product Guide* (PG249).

Chapter 4

Designing with the Subsystem

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

DisplayPort Overview

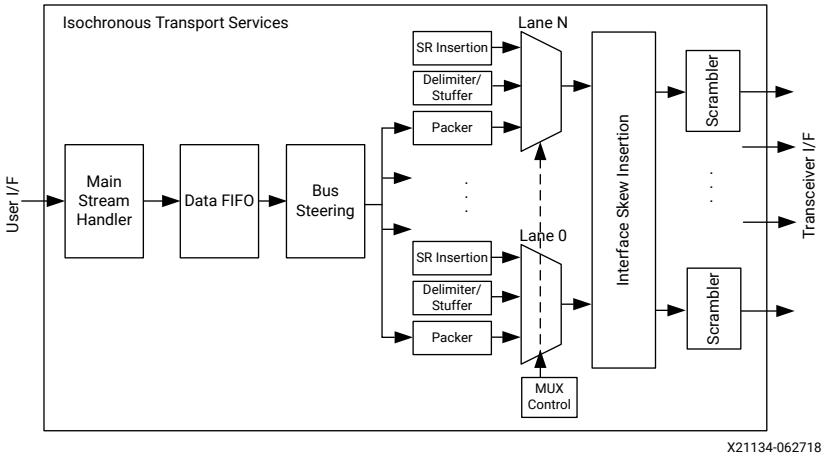
The Source core moves a video stream from a standardized main link through a complete DisplayPort Link Layer and onto High-Speed Serial I/O for transport to a Sink device.

Main Link Setup and Management

This section is intended to elaborate on and act as a companion to the link training procedure in the *VESA DisplayPort Standard v1.4*.

Xilinx® advises all users of the source core to use an Arm® processor, 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, and such controllers are not supported through the service portal.

Figure 6: **Source Main Link Datapath**

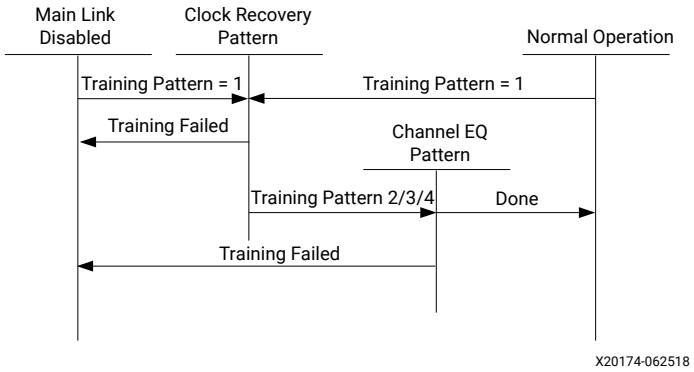


Link Training

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.

The following figure shows the flow diagram for link training. For details, see the *VESA DisplayPort Standard* (VESA website).

Figure 7: **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 the *VESA DisplayPort Standard v1.4*.

IMPORTANT! *During initialization, ensure that TX8B10BEN is not cleared in offset `0x0070` of the corresponding Video PHY Controller . For this release, information on the DisplayPort 1.4 is not available in Video PHY Controller LogiCORE IP Product Guide (PG230).*

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
```

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.

On 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 pre-emphasis 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 2/3/4 Procedure (Symbol Recovery, Interlane Alignment)

1. Turn off scrambling and set training pattern 2 in the source through direct register writes.

   SCRAMBLING_DISABLE = 0x01 (Not applicable for Training Pattern 4)

   Set training pattern to pattern 2, pattern 3, or pattern 4 based on the Sink DPCD capability

2. Set proper state for scrambling and set training pattern 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 pre-emphasis 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
```

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 supported resolutions if the preferred mode is not be available). When 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 scrambler/de-scrambler.*

Accessing the Link Partner

The DisplayPort 1.4 TX Subsystem 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, see the *VESA DisplayPort Standard v1.4* (VESA website).

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.

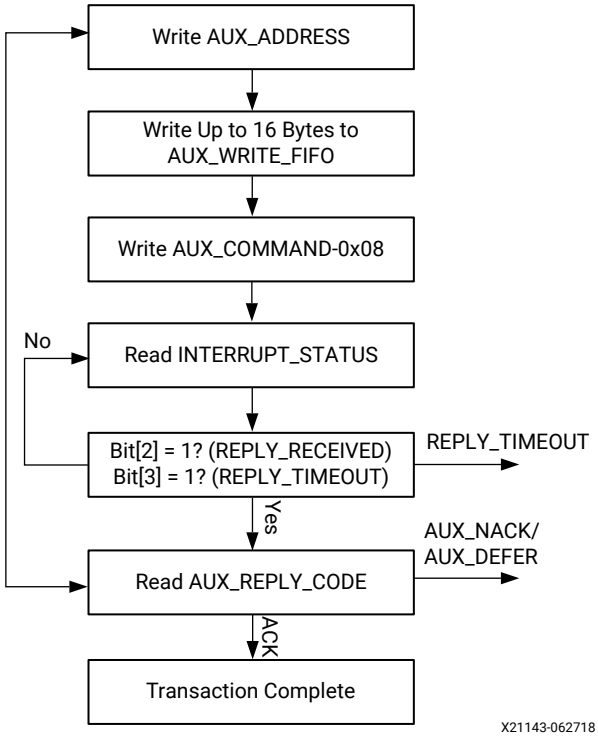
AUX Write Transaction

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.

The following figure shows the flow of an AUX write transaction.

Figure 8: **AUX Write Transaction**



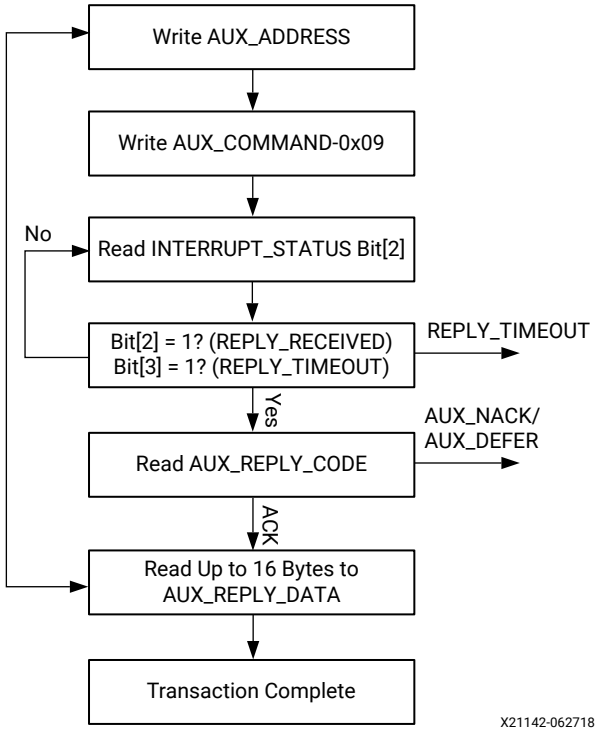
AUX Read Transaction

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.

The following figure shows the flow of an AUX read transaction.

Figure 9: **AUX Read Transaction**



AUX Channel Command for I2C Transactions

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

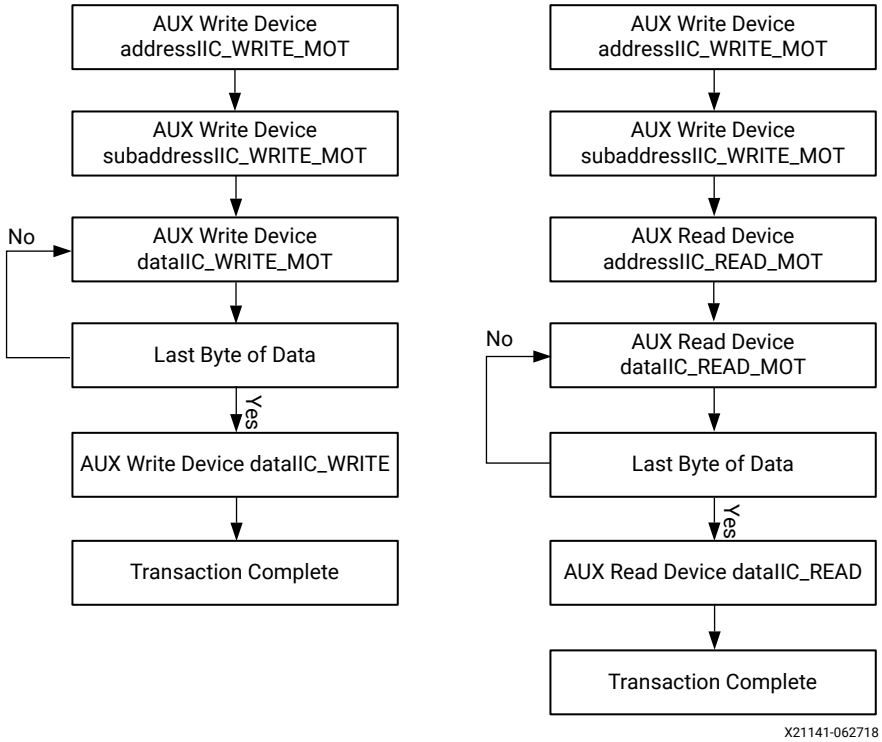
The sequence for performing these transactions is exactly the same as a native AUX channel transaction with a change to the command written to the AUX_COMMAND register. The supported I2C commands are summarized in the following table.

Table 24: **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 can emulate an I2C transaction. The following figure shows the flow of I2C transactions.

Figure 10: **I2C Device Transactions, Write (Left) and Read (Right)**



Because 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

Generation of AUX transactions are described in the following table.

Set up I2C slave for Write to address defined		
Transaction	Write Address only with MOT = 1	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.
AUX Transaction	START -> CMD -> ADDRESS -> STOP	
I2C Transaction	START -> DEVICE_ADDR -> WR -> ACK/NACK	
Set up I2C slave for Read to address defined		
Transaction	Read Address only with MOT = 1	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.
AUX Transaction	START -> CMD -> ADDRESS -> STOP	
I2C Transaction	START -> DEVICE_ADDR -> RD -> ACK/NACK	
To stop the I2C slave, used as Abort or normal stop		
Transaction	Write/Read Address only with MOT = 0	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.
AUX Transaction	START -> ADDRESS -> STOP	
I2C Transaction	STOP	

Set up I2C slave write data		
Transaction	Write with MOT = 1	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.
AUX Transaction	START -> CMD -> ADDRESS -> LENGTH -> D0 to DN -> STOP	
I2C Transaction	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	
Set up I2C slave write data and stop the I2C bus after the current transaction		
Transaction	Write with MOT = 0	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.
AUX Transaction	START -> CMD -> ADDRESS -> LENGTH -> D0 to DN -> STOP	
I2C Transaction	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	
Set up I2C slave read data		
Transaction	Read with MOT = 1	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.
AUX Transaction	START -> CMD -> ADDRESS -> LENGTH -> STOP	
I2C Transaction	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	
Set up I2C slave read data and stop the I2C bus after the current transaction		
Transaction	Read with MOT = 0	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.
AUX Transaction	START -> CMD -> ADDRESS -> LENGTH -> D0 to DN -> STOP	
I2C Transaction	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	
Status of previous write command that was deferred or partially ACKED		
Transaction	Write Status with MOT = 1	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.
AUX Transaction	START -> CMD -> ADDRESS -> STOP	
I2C Transaction	No transaction	

Status of previous write command that was deferred or partially ACKED. MOT = 0 ensures that the bus returns to IDLE at the end of the burst.		
Transaction	Write Status with MOT = 0	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.
AUX Transaction	START -> CMD -> ADDRESS -> STOP	
I2C Transaction	Force a STOP and the end of write burst	

Handling I2C Read Defers/Timeout

The Sink core can 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 can 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 a 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 the GUI. The AUX IO type can be uni-directional/bidirectional when the AUX IO is located inside the IP.

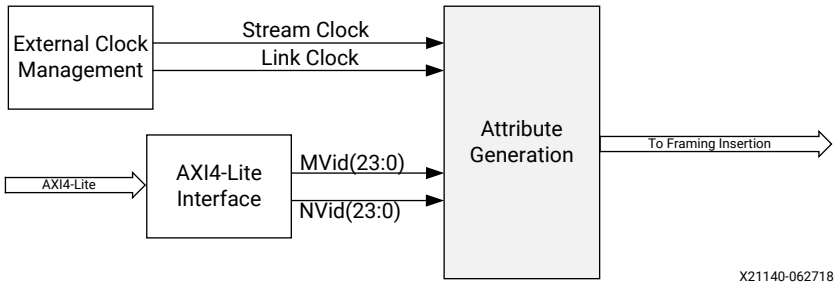
Transmitter Audio/Video Clock Generation

The transmitter clocking architecture supports both the asynchronous and synchronous clocking modes included in the *VESA DisplayPort Standard v1.4*. 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 through 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 transmits a fixed value for NVid and the MVid value provided as a part of the clocking interface.

The following figure shows a block diagram of the transmitter clock generation process.

Figure 11: **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, see the *VESA DisplayPort Standard v1.4*.

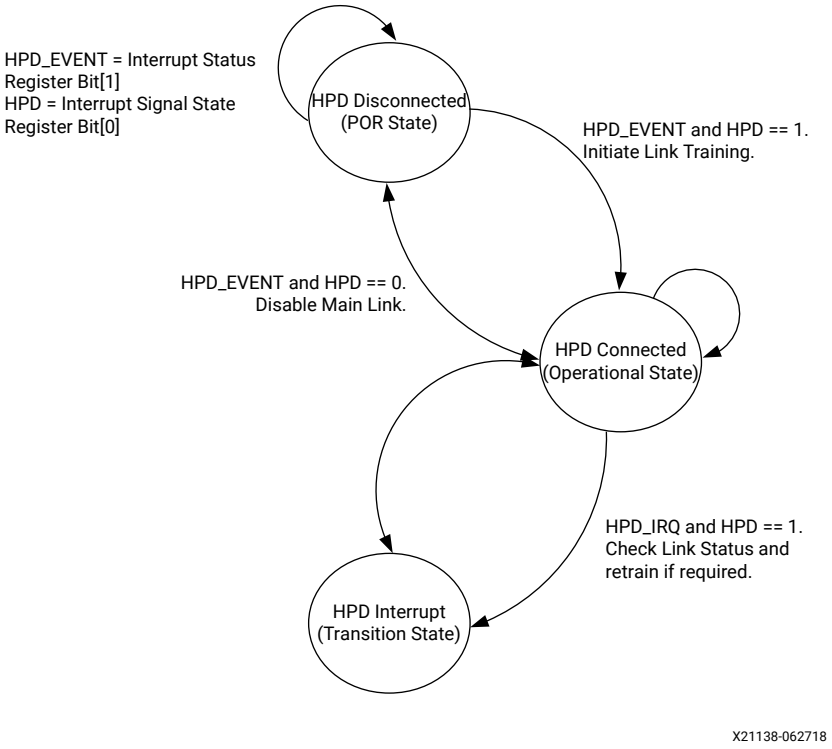
HPD Event Handling

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].

The following figure shows the Source core state and basic actions to be taken based on HPD events.

Figure 12: **HPD Event Handling in Source Core**

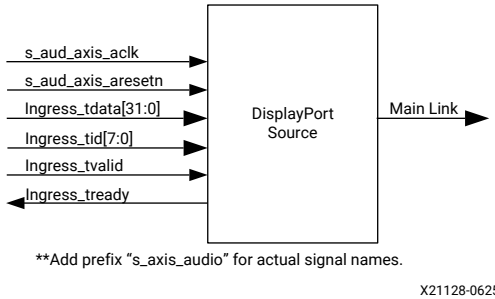


Secondary Channel Operation

The current version of the DisplayPort core supports eight-channel Audio. Secondary Channel features from the DisplayPort Standard v1.4 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 the following figure, the Audio interface to the DisplayPort core is defined using an AXI4-Stream interface to improve system design and IP integration.

Figure 13: **Audio Data Interface of DisplayPort Source System**



32-bit AXI TDATA is formatted as follows:

Control Bits + 24-bit Audio Sample + Preamble

The ingress channel buffer in the DisplayPort core accepts data from the AXI4-Stream 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

The DisplayPort TX requires Info frame configuration to transmit multi-channel audio. The Info frame contains the number of channels and its speaker mapping. The AXI4-Stream `TID` signal 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 AXI4-Stream interface (`TID[7:4]`).

Audio Management

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

Programming DisplayPort Source

1. Disable audio by writing `0x00` to the TX_AUDIO_CONTROL register. The disable bit also flushes the buffers in the 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 the Audio Info Frame (based on your requirement; this might be optional for some systems). The Audio Info Frame consists of eight writes. The order of write transactions are important and follow the steps in the DisplayPort Audio registers, offset `0x308` (see related information).
3. Write Channel Count to the 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.

Related Information

DisplayPort Audio Registers

Re-Programming Source Audio

1. Disable Audio in DisplayPort 1.4 TX core.
2. Wait until Video/Audio clock is recovered and stable.
3. Enable Audio in DisplayPort 1.4 TX core.

Info Packet Management

The core provides an option to program two Info packets and can support 4 Info packets. The packets are transmitted to the Sink once per every video frame or 8192 cycles.

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

1. Disable Audio (because a new Info packet means a new audio configuration). The disable audio also flushes the internal audio buffers.
2. To program the DisplayPort Source, see Programming the DisplayPort Source.

Related Information

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). For VSC Ext packet to be aligned with the Vertical Blanking region, set Bit[12] of `0x1A4` register and then program the packet data.

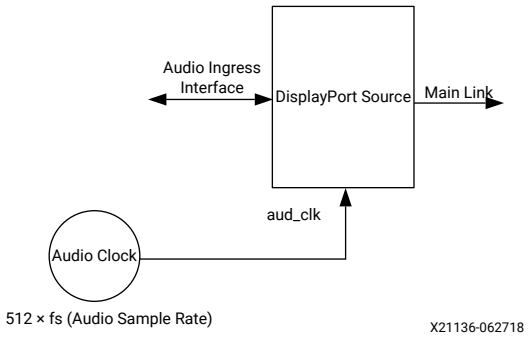
Use the following steps to write an extended packet in the DisplayPort Source controller:

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 the DisplayPort Source device to calculate MAUD and NAUD when running in asynchronous clocking mode.

Figure 14: **Audio Clocking for Source**



***Note*:** This should be > 512 × fs

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 *VESA DisplayPort Standard* (VESA website).
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: Note: Repeat Step 7 for each steam.
9. Enable MST by writing 1 to bit 0 of MST Config register.

Payload Bandwidth Management

The following steps manage payload bandwidth in the source controller.

1. Calculate Target_Average_StreamSymbolTimeSlotsPerMTP based on the *VESA DisplayPort Standard* (VESA website). Program the VC payload size with the calculated Target_Average_StreamSymbolTimeSlotsPerMTP and align it with nearest even boundary.

   For example if the value is 13, program the VC payload size for this particular stream to 14.
2. The VC payload calculation for (1920x2200) stream, RGB color sampling, 8 Bits Per Color at 5.4 Gb/s, 4 lanes is shown here:

   VC Payload Band width = LINK_RATE × Lane_count × 100 (Table 2-136 in the DisplayPort standard) = 5.4 × 4 × 100 = 2160

   Average Stream symbol Time slot per MTP = (Pixel_rate × Bits_per_pixel/8/VC Payload_Band width) × 64 (Section 2.6.4.3.1 in the DisplayPort standard) = (297 MHz × 24 /8/2160) × 64 = 26.4

   VC Payload Size = 2/4 symbol aligned of (Average Stream symbol Time slot per MTP) = 28
3. Program the VC Payload table as defined in DPCD standard.
4. Program the VC Payload table in source controller as defined in registers 12'h0x800 – 12'h0x8FC.

EDID I2C Speed Control

EDID I2C speed can be controlled by programming the I2C Speed Control DPCD register (0x00109) and DisplayPort subsystem supports up to 1 Mb/s I2C speed.

eDP Support

The DisplayPort 1.4 TX Subsystem supports the following eDP features:

- Reduced AUX timing because the AUX sync patterns are reduced from 16 pulses to 8 pulses
- Alternate Scrambler Seed Reset (ASSR)

  Note: Offset `0x08C` bit-0 has been set to enable the ASSR method for content protection in the DP source. This is the R/W register from the AXI4 interface.
- Enhanced Framing

eDP can be enabled by using the following Tcl command:

```
set_property -dict [list CONFIG.EDP_ENABLE {true}] [get_bd_cells
v_dp_txss1_0]
```

When generating the subsystem, if EDP_ENABLE is set to true, the Version register, (Offset `0x0F8`) Bits [7:0] - Internal Revision is updated with `0x01`.

Note: The eDP feature uses the same hardware infrastructure (connector, FMC/PCB, pinouts) of the regular DisplayPort 1.4.

DSC and FEC Support

The DisplayPort 1.4 TX Subsystem supports DSC and FEC features. For DSC/FEC support with the Xilinx subsystem, contact the Xilinx partner, Hardent Incorporated.

Adaptive Sync Support

The DisplayPort 1.4 TX Subsystem supports adaptive sync feature, and the VTC IP is enhanced for adaptive sync support.

Versal Device Support

The DisplayPort 1.4 TX Subsystem supports Versal™ devices and uses a fabric `8B10B` decoder implementation instead of a Xilinx transceiver block `8B10B` decoder. For Versal devices, this results in an additional clock in the subsystem. The following table provides clock frequency values.

Table 26: **Link Clock for Versal Devices**

Clock	Formula	Value
tx_lnk_clk	Link Rate/16	• 506.25 MHz for 8.1 Gb/s • 337.50 MHZ for 5.4 Gb/s • 168.75 MHz for 2.7 Gb/s • 101.25 MHz for 1.62 Gb/s
tx_enc_clk	Link Rate/20	• 405 MHz for 8.1 Gb/s • 270 MHZ for 5.4 Gb/s • 135 MHz for 2.7 Gb/s • 81 MHz for 1.62 Gb/s

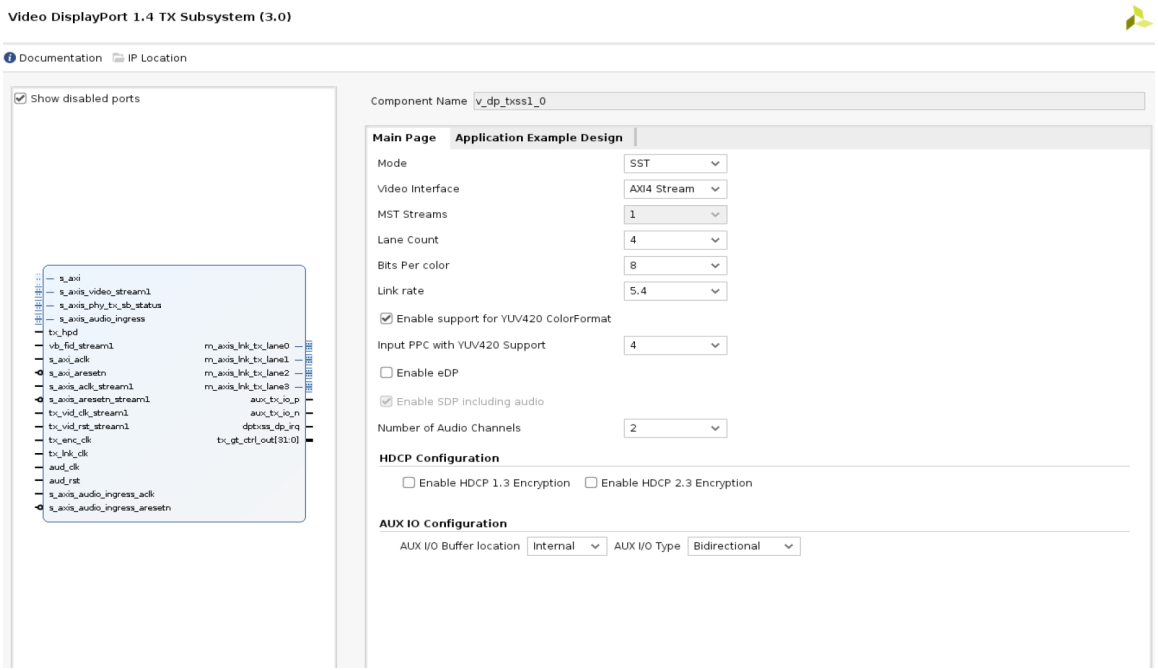
The subsystem supports block automation in IPI for Versal device designs and instantiates the transceiver bridge (PHY) IP and Versal transceiver wizard IP as part of block automation.

YUV420 Colorimetry

In the Vivado® IDE, in the DisplayPort 1.4 TX Subsystem, when the Enable support for YUV420 ColorFormat option is selected (as shown in the following figure), the subsystem can support YUV420 colorimetry as mentioned in *VESA DisplayPort Standard Version 1.4a* (VESA website). When you select the 420 option, you are prompted to select 4PPC or 8PPC for the AXI4-Stream data input interface using the Input PPC with YUV420 Support option. 8PPC can be used for resolutions with YUV420 colorimetry which result in `vid_clk` > 300 MHz (for resolutions such as 8k at 60 fps).

***Note*:** YUV420 colorimetry is not supported in MST mode.

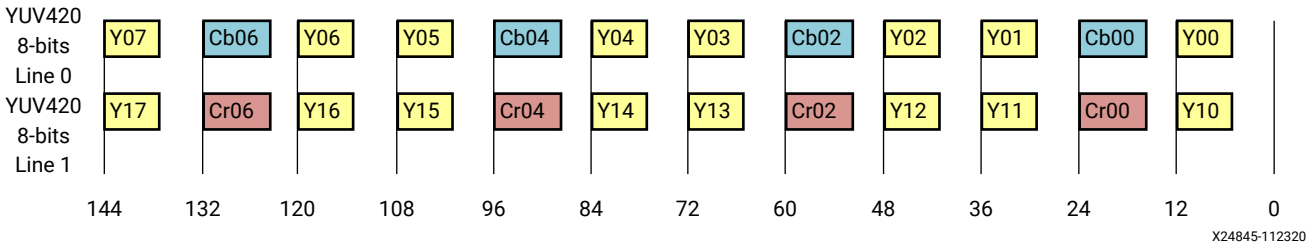
Figure 15: **DisplayPort 1.4 TX Subsystem with Option to Enable YUV420**



The pixel arrangement of the YUV420 color format follows the UG934 standard on the input of the AXI4-Stream interface. This data is then converted into the native mode by the subsystem and is fed to the DisplayPort TX IP. The following figure depicts the native mode data format of 4PPC YUV420 after conversion from the UG934 format.

***Note*:** The figure depicts the case of USER_PIXEL_WIDTH (DisplayPort TX IP offset register `0x1B8`) as 4, MAX_BPC of 12 and Video BPC of 8.

Figure 16: **Native YUV420 Format After Converted From YUV420 Video Format of UG934**



Because the native YUV420 format has two Y-components per pixel, care should be taken to consider “HTOTAL” (total horizontal resolution of the video) as half of the actual value in the software.

Pixel Mapping

Pixel Mapping on AXI4-Stream Interface

The DisplayPort 1.4 TX Subsystem input data interface is always a quad pixel interface (4PPC). 4PPC is internally translated (inside the wrapper of the DisplayPort AXI4-Stream to Video Bridge IP) into 1, 2, or 4PPC, then given as input to DisplayPort TX IP depending on the value programmed in the USER_PIXEL_WIDTH register at offset address `0x1B8` (from now referred to as USER_PIXEL_WIDTH) of the DisplayPort TX IP. With the subsystem input interface always 4PPC, the need to have an external remapper is eliminated, and a significant improvement in resource numbers is achieved. By default, the USER PIXEL WIDTH at offset address `0x1B8` of DisplayPort TX IP is selected based on Pixel Frequency in the subsystem driver. The following shows the different USER_PIXEL_WIDTH (also referred to as PPC) for each pixel frequency:

- For USER_PIXEL_WIDTH of 1, Pixel Frequency < 75 MHz
- For USER_PIXEL_WIDTH of 2, Pixel Frequency ≥ 75 and < 300 MHz
- For USER_PIXEL_WIDTH of 4, Pixel Frequency ≥ 300 MHz

Note: USER_PIXEL_WIDTH of 4 requires four lanes, USER_PIXEL_WIDTH of 2 requires two lanes minimum. For more information on the relationship between USER_PIXEL_WIDTH and lanes, see the description of offset address `0x1B8` in Table 20: Main Stream Attributes.

You can override this dynamically. For example, if the driver selects a value of 2 for USER_PIXEL_WIDTH as default, you can change this to 1.

As the input data interface of the subsystem is always 4PPC, valid pixels are available in pixel 0, pixel 1, pixel 2, and pixel 3 position.

The data width of the AXI4-Stream interface depends on different parameters of the core.

For RGB and YCBCR 4:4:4:4 format `Pixel_Width = MAX_BPC × 3` for 4:2:2 and 4:2:0 format `Pixel_Width = MAX_BPC × 2` and for Y-only `Pixel_Width = MAX_BPC`.

`Interface Width = Pixel Width × 4` (because the subsystem's intput data interface is always 4.)

For example, if the system is generated with MAX_BPC equal to 16, the data width of the AXI4-Stream interface is 16 × 4 × 3 which equals 192.

Pixel Mapping Examples on AXI4-Stream Interface

The following table shows the pixel mapping examples for an AXI4-Stream interface.

Table 27: **Pixel Mapping Examples on AXI4-Stream Interface**

MAX_BPC	Pixel Width	Interface Width	Video BPC	Pixel 3			Pixel 2			Pixel 1			Pixel 0		
16	48	192	16	191:176	175:160	159:144	143:128	127:112	111:96	95:80	79:64	63:48	47:32	31:16	15:0
12	36	144	12	143:132	131:120	119:108	107:96	95:84	83:72	71:60	59:48	47:36	35:24	23:12	11:0
10	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	24	96	8	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
16	48	192	12	191:180	175:164	159:148	143:132	127:116	111:100	95:84	79:68	63:52	47:36	31:20	15:4
12	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	30	120	8	119:112	109:102	99:92	89:82	79:72	69:62	59:52	49:42	39:32	29:22	19:12	9:2
8	24	96	6	95:90	87:82	79:74	71:66	63:58	55:50	47:42	39:34	31:26	23:18	15:10	7:2
16	48	192	10	191:182	175:166	159:150	143:134	127:118	111:102	95:86	79:70	63:54	47:38	31:22	15:6
12	36	144	8	143:136	131:124	119:112	107:100	95:88	83:76	71:64	59:52	47:40	35:28	23:16	11:4
10	30	120	6	119:114	109:104	99:94	89:84	79:74	69:64	59:54	49:44	39:34	29:24	19:14	9:4
16	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
12	36	144	6	143:138	131:126	119:114	107:102	95:90	83:78	71:66	59:54	47:42	35:30	23:18	11:6
16	48	192	6	191:186	175:170	159:154	143:138	127:122	111:106	95:90	79:74	63:58	47:42	31:26	15:10

Notes:

1. The padding bits are zeros.

Pixel Mapping Examples on AXI4-Stream Interface (UG934-Compliant)

The following table shows the pixel mapping examples for an AXI4-Stream interface that is UG934-compliant.

Table 28: **Pixel Mapping Examples on AXI4-Stream Interface (UG934-Compliant Mode)**

MAX_BPC	Pixel Width	Interface Width	Video Sampling Mode	Pixel 3			Pixel 2			Pixel 1			Pixel 0		
8	24	96	444	95:88	87:80	79:72	71:64	63:56	55:48	47:40	39:32	31:24	23:16	15:8	7:0
	16	64	422	63:56		55:48	47:40		39:32	31:24		23:16	15:8		7:0
	16	64	420 (Even line)	63:56		55:48	47:40		39:32	31:24		23:16	15:8		7:0
	16	64	420 (Odd line)	-		55:48	-		39:32	-		23:16	-		7:0
	8	32	Y-Only			31:24			23:16			15:8			7:0
10	30	120	444	119:112	109:102	99:92	89:82	79:72	69:62	59:52	49:42	39:32	29:22	19:12	9:2
	20	80	422	79:72		69:62	59:52		49:42	39:32		29:22	19:12		9:2
	20	80	420 (Even line)	79:72		69:62	59:52		49:42	39:32		29:22	19:12		9:2
	20	80	420 (Odd line)	-		69:62	-		49:42	-		29:22	-		9:2
	10	40	Y-Only			39:32			29:22			19:12			9:2
12	36	144	444	143:136	131:124	119:112	107:100	95:88	83:76	71:64	59:52	47:40	35:28	23:16	11:4
	24	96	422	95:88		83:76	71:64		59:52	47:40		35:28	23:16		11:4
	24	96	420 (Even line)	95:88		83:76	71:64		59:52	47:40		35:28	23:16		11:4
	24	96	420 (Odd line)	-		83:76	-		59:52	-		35:28	-		11:4
	12	48	Y-Only			47:40			35:28			23:16			11:4
16	48	192	444	191:184	175:168	159:152	143:136	127:120	111:104	95:88	79:72	63:56	47:40	31:24	15:8
	32	128	422	127:120		111:104	95:88		79:72	63:56		47:40	31:24		15:8
	32	128	420 (Even line)	127:120		111:104	95:88		79:72	63:56		47:40	31:24		15:8
	32	128	420 (Odd line)	-		111:104	-		79:72	-		47:40	-		15:8
	16	64	Y-Only			63:56			47:40			31:24			15:8[1]

Notes:

1. Video BPC is 8.

Pixel Mapping on Native Video Interface

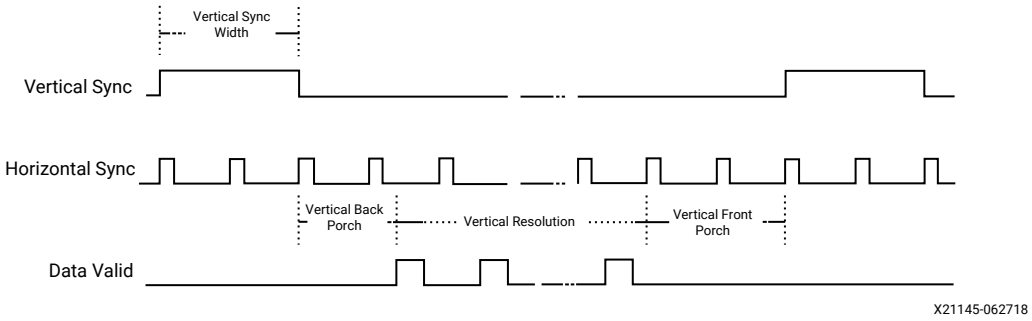
The primary interface for user image data has been modeled on the industry standard for display timing controller signals. The port list consists of video timing information encoded in a vertical and horizontal sync pulse and data valid indicator. These single bit control lines frame the active data and provide flow control for the AXI4-Stream video.

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

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

The following figure shows the typical signaling of a full frame of data.

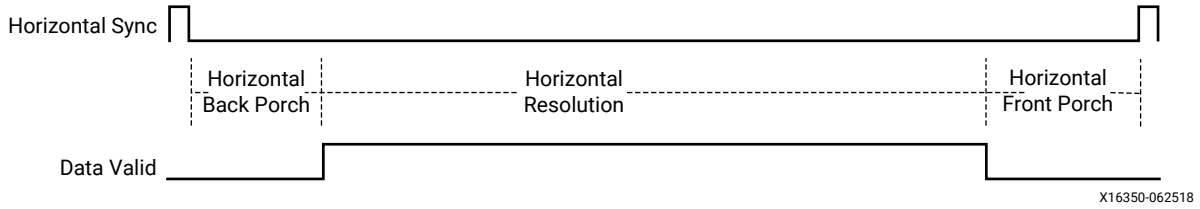
Figure 17: **User Interface Vertical Timing**



Similarly, the horizontal timing information is defined by a front porch, back porch, and pulse width. The porch values are defined as the number of clocks between the horizontal sync pulse and the start or end of active data. Pixel data is only accepted into the image data interface when the data valid flag is active-High. The following figure is an enlarged version of the previous figure, giving more details on a single scan line. The horizontal sync pulse should be used as a line advance signal. Use the rising edge of this signal to increment the line count.

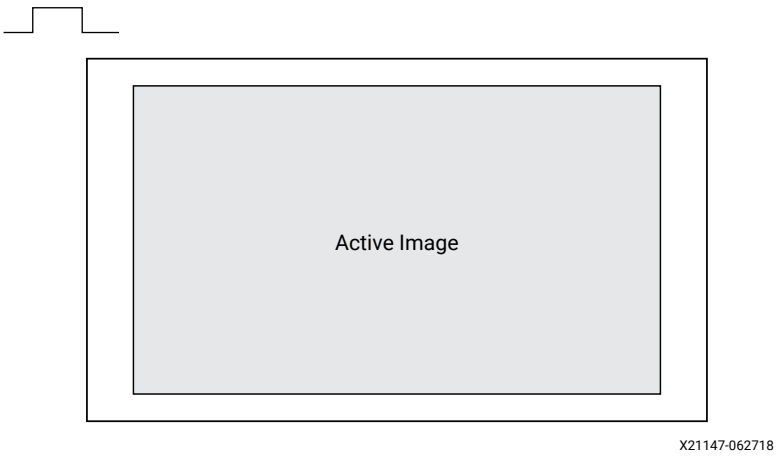
***Note*:** Data Valid might toggle if using a fast clock. Also, Data Valid signal might remain asserted for the duration of a scan line. Dropping the valid signal might result in improper operation.

Figure 18: **User Interface Horizontal Timing**



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

Figure 19: **Active Image Data**



The User Data interface can accept one, two, or four pixels per clock cycle. The `vid_pixel` width is always 48-bits, regardless if all bits are used. For pixel mappings that do not require all 48 bits, the convention used for this core is to occupy the MSB bits first and leave the lower bits either untied or driven to zero.

Pixel Mapping Examples on the User Data Interface

The following table shows the correct mapping for all supported data formats.

Table 29: **Pixel Mapping Examples on the User Data Interface**

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

Table 29: **Pixel Mapping Examples on the User Data Interface** *(cont'd)*

Format	BPC/BPP	R	G	B	Cr	Y	Cb	Cr/Cb	Y
YCrCb444	6/18	–	–	–	[47:42]	[31:26]	[15:10]	–	–
YCrCb444	8/24	–	–	–	[47:40]	[31:24]	[15:8]	–	–
YCrCb444	10/30	–	–	–	[47:38]	[31:22]	[15:6]	–	–
YCrCb444	12/36	–	–	–	[47:36]	[31:20]	[15:4]	–	–
YCrCb444	16/48	–	–	–	[47:32]	[31:16]	[15:0]	–	–
YCrCb422	8/16	–	–	–	–	–	–	[47:40]	[31:24]
YCrCb422	10/20	–	–	–	–	–	–	[47:38]	[31:22]
YCrCb422	12/24	–	–	–	–	–	–	[47:36]	[31:20]
YCrCb422	16/32	–	–	–	–	–	–	[47:32]	[31:16]
YUV420 (even line)	8/24	–	–	–	–	[47:40]	[15:8]	–	[31:24]
YUV420 (odd line)	8/24	–	–	–	[15:8]	[47:40]	–	–	[31:24]
YUV420 (even line)	10/30	–	–	–	–	[47:38]	[15:6]	–	[31:22]
YUV420 (odd line)	10/30	–	–	–	[15:6]	[47:38]	–	–	[31:22]
YUV420 (even line)	12/36	–	–	–	–	[47:36]	[15:4]	–	[31:20]
YUV420 (odd line)	12/36	–	–	–	[15:4]	[47:36]	–	–	[31:20]
YUV420 (even line)	16/48	–	–	–	–	[47:32]	[15:0]	–	[31:16]
YUV420 (odd line)	16/48	–	–	–	[15:0]	[47:32]	–	–	[31:16]
YONLY	8/8	–	–	–	–	–	–	–	[47:40]
YONLY	10/10	–	–	–	–	–	–	–	[47:38]
YONLY	12/12	–	–	–	–	–	–	–	[47:36]
YONLY	16/16	–	–	–	–	–	–	–	[47:32]

Notes:

1. For a YCrCb 4:2:2, the input follows YCr, YCb, YCr, YCb, and so on. This means Cr and Cb are mapped to the same bits on the video input ports of the source core. The source core expects YCb first, followed by YCr.

Selecting the Pixel Interface

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

***Note*:** In a quad pixel interface, if the resolution is not divisible by four, you should add zeros at the end of frame, over the video interface pixel data.

For example, to support an active resolution of 2560 × 1600 @ 60, there are two possible blanking formats: Normal Blanking and Reduced Blanking, as defined by the VESA standard.

2560 × 1600 @ 60 + Blanking = 3504 × 1658 @ 60 requires a pixel clock of 348.58 MHz

2560 × 1600 @ 60 + Reduced Blanking = 2720 × 1646 @ 60 requires a Pixel clock of 268.63 MHz

Assuming a pixel clock of 150 MHz and a dual pixel interface:

2560 × 1600 @ 60 + Blanking = 3504 × 1658 @ 60 = 348.58 MHz

348.58 MHz / 2 = 172.28 MHz

2560 × 1600 @ 60 + Reduced Blanking = 2720 × 1646 @ 60 = 268.63 MHz

268.63 MHz / 2 = 134.31 MHz

With a dual pixel interface, the DisplayPort 1.4 TX Subsystem IP can support 2560 × 1600 only if there is a Reduced Blanking input. If full Blanking support is needed, then a four pixel interface should be used.

The following figures show timing diagrams for the three pixel interface options.

Figure 20: **Single Pixel Timing**

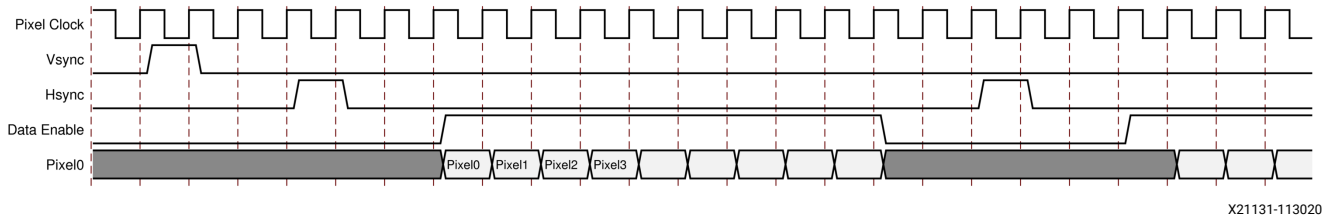


Figure 21: **Dual Pixel Timing**

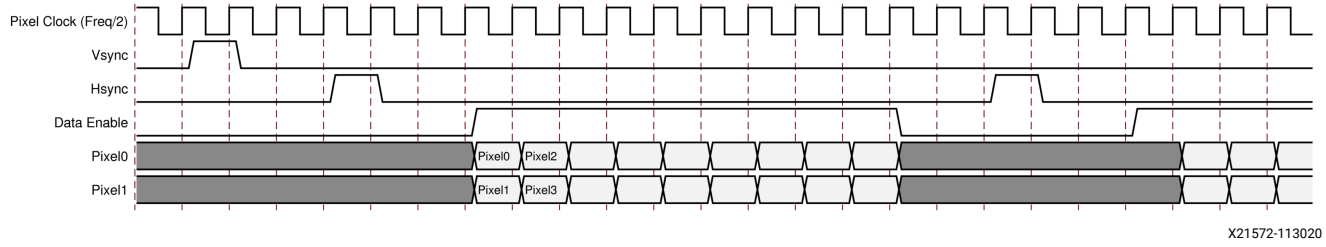
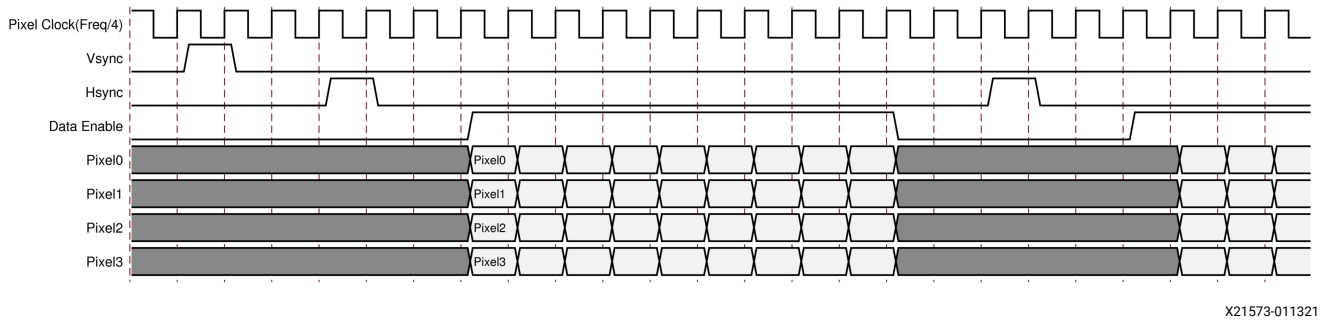


Figure 22: **Quad Pixel Timing**



AXI4-Stream Interface Color Mapping

This table shows the color mapping for the AXI4-Stream interface and this complies with the guidelines in the *AXI4-Stream Video IP and System Design Guide* (UG934).

Table 30: **AXI4-Stream Interface Data Mapping**

	AXI4-Stream Interface											
	Pixel 3			Pixel 2			Pixel 1			Pixel 0		
	Comp3	Comp2	Comp1	Comp3	Comp2	Comp1	Comp3	Comp2	Comp1	Comp3	Comp2	Comp1
RGB	R	B	G	R	B	G	R	B	G	R	B	G
YCbCr444	Cr	Cb	Y	Cr	Cb	Y	Cr	Cb	Y	Cr	Cb	Y
YCbCr422	N/A	Cr/Cb	Y	N/A	Cr/Cb	Y	N/A	Cr/Cb	Y	N/A	Cr/Cb	Y
YUV420 (Even line)	N/A	Cr	Y	N/A	Cb	Y	N/A	Cr	Y	N/A	Cb	Y
YUV420 (Odd line)	N/A	Invalid Data	Y	N/A	Invalid Data	Y	N/A	Invalid Data	Y	N/A	Invalid Data	Y
Y-Only	N/A	N/A	Y	N/A	N/A	Y	N/A	N/A	Y	N/A	N/A	Y

Notes:

1. For component widths, see the Pixel Mapping Examples on AXI4-Stream Interface.
2. Component fields with N/A are not applicable and do not appear on the AXI4-Stream Interface.
3. Component fields with Invalid Data appear on the AXI4-Stream Interface, but contain invalid/zero data.

Related Information

Pixel Mapping Examples on AXI4-Stream Interface

Clocking

This section describes the link clock (`tx_lnk_clk`) and the video clock (`tx_vid_clk_stream1`). 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.

The `tx_lnk_clk` is a link clock input to the DisplayPort 1.4 TX Subsystem generated by the Video PHY (GT). The frequency of `tx_lnk_clk` is `<line_rate>`/20 MHz for 16-bit interface.

The `hdcp_ext_clk` input can be driven from external MMCM or BUFGCDIV where it has a frequency requirement of `hdcp_ext_clk` = `tx_lnk_clk`/2 MHz.

In both, native and AXI4-stream modes TX video clock value is based on the Consumer Electronics Association (CEA)/VESA Display Monitor Timing (DMT) standard for given video resolutions.

The core uses six clock domains:

- **lnk_clk:** The `txoutclk` from the Video PHY is connected to the TX subsystem link clock. Most of the core operates in the link clock domain. This domain is based on the `lnk_clk_p/n` reference clock for the transceivers. The link rate switching is handled by a DRP state machine in the core PHY later. When the lanes are running at 2.7 Gb/s, `lnk_clk` operates at 135 MHz. When the lanes are running at 1.62 Gb/s, `lnk_clk` operates at 81 MHz. When the lanes are running at 5.4 Gb/s, `lnk_clk` operates at 270 MHz.When the lanes are running at 8.1 Gb/s, `lnk_clk` operates at 405 MHz.

  ***Note*:** `lnk_clk` = `link_rate`/20, when the GT data width is 16 bits.

- **vid_clk:** This is the primary user interface clock. It is based on the DisplayPort Standard, the video clock can be derived from the link clock using `mvid` and `nvid`. In addition, `vid_clk` should be at least [(Vtotal x Htotal x frames per second)/pixels per clock]. For YCbCr420 colorimetry, the `vid_clk` frequency will be half of the actual `vid_clk`.

- **s_axi_aclk:** This is the processor domain. It has been tested to run as fast as 135 MHz. The AUX clock domain is derived from this domain, but requires no additional constraints. In UltraScale™ FPGA, `s_axi_aclk` clock is connected to a free-running clock input. `gtwiz_reset_clk_freerun_in` is required by the reset controller helper block to reset the transceiver primitives. A new GUI parameter is added for AXI_Frequency, when the DisplayPort IP is targeted to UltraScale FPGA.

- **aud_clk:** This is the audio interface clock. The frequency will be equal to 512 × audio sample rate.

- **s_aud_axis_aclk:** This clock is used by the source audio streaming interface. This clock should be = 512 × audio sample rate.

- **m_aud_axis_aclk:** This clock is used by the sink audio streaming interface. This clock should be = 512 × audio sample rate.

For more information on clocking, see the *Video PHY Controller LogiCORE IP Product Guide* (PG230).

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

The following table shows an example based on a subsystem base address of `0x44C0_0000` (19 bits). The DisplayPort 1.4 TX Subsystem requires a 19-bit address mapping, starting at an offset address of `0x00000`.

This address map example is applicable when the TX subsystem is configured in the AXI4-Stream interface mode.

Table 31: **Address Map Example**

IP Core/Subsystem	SST	MST
DisplayPort 1.4 TX Core	0x44C0_0000	0x44C0_0000
VTC 0	0x44C0_1000	0x44C0_1000
VTC 1 (N = 2)	N/A	0x44C0_2000
VTC 1 (N = 3)	N/A	0x44C0_3000
VTC 1 (N = 4)	N/A	0x44C0_4000
HDCP Controller	0x44C0_2000	N/A
AXI Timer	0x44C0_3000	N/A

Physical Layout

When designing for the PCB, for pin outputs to the PCB from the PHY, refer to*Video PHY Controller LogiCORE IP Product Guide* (PG230). This section discusses the pin outputs from the subsystem and external IC/PCB considerations.

***Note*:** Schematics and an example FMC card are available for purchase from Tokyo Electron Device Ltd.

Pins

The following pins will need to be driven through a level shifter, such as the SN74AVC4T774, followed by a line driver, such as the SN64MLVD2020A, and then an RC circuit to meet the DisplayPort standard requirements. See the DisplayPort standard for more information on switching requirements.

- `aux_tx_data_out`
- `aux_tx_data_in`
- `aux_tx_data_en_out_n`
- `tx_hpd`

The following pins are detailed in *Video PHY Controller LogiCORE IP Product Guide* (PG230), but Xilinx recommends sending these through a redriver, such as the SN65DP141, to simplify compliance testing.

***Note*:** This is not a requirement to use the core, but doing so removes the need to adjust the transceiver settings during compliance.

- `phy_txn_out[x:0]`
- `phy_txp_out[x:0]`

For more information on schematic availability, refer to AR75465.

For all designs, reference the PCB design user guide and checklist. For UltraScale architectures, refer to *UltraScale Architecture PCB Design User Guide* (UG583) and *UltraScale+ FPGAs and Zynq Ultrascale+ Devices Schematic Review Checklist* (XTP427).

Programming Sequence

For PHY related programming, see the *Video PHY Controller LogiCORE IP Product Guide* (PG230).

The DisplayPort TX Subsystem is delivered with a bare-metal driver and API designed to handle all programming of the subsystem core. The driver is automatically available in the Vitis™ tools when a project is created from an XSA file that includes the DisplayPort TX Subsystem. This bare-metal driver can also be seen on the Xilinx GitHub. See API Documentation for more information. Linux drivers are also available. See the Xilinx Wiki page for more information.

Design Flow Steps

This section describes customizing and generating the subsystem, constraining the subsystem, and the simulation, synthesis, and implementation steps that are specific to this IP 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)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

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) 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 IP for use in your design by specifying values for the various parameters associated with the IP subsystem using the following steps:

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

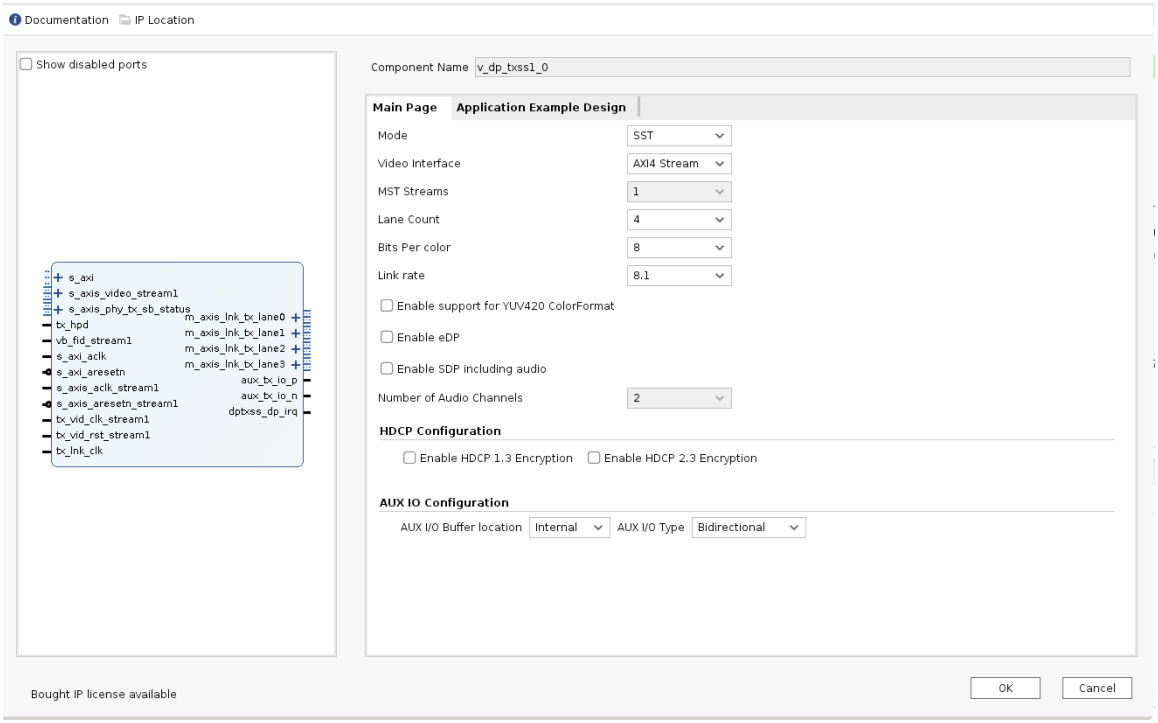
For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

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

Subsystem Configuration Screen

The subsystem configuration screen is shown in the following figure.

Figure 23: **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 v_dp_txss1_0.
- **Mode:** Selects the desired resolution for the video stream out. Options are SST or MST.
- **Video Interface:** Selects the AXI4-Streamor native for the input video interface.
- **MST Streams:** Selects the maximum number of streams in MST mode.
- **Lane Count:** Selects the maximum number of lanes.
- **Bits Per Color:** Selects the desired maximum bit per component (BPC).
- **Link Rate:** Selects the desired link rate in Gb/s.
- **Enable Support for YUV420 ColorFormat:** Selects the YUV420 colorimetry. When YUV420 color format is selected, by default,SDP is enabled.

- **Input PPC with YUV420 Support:** 4 (default), 8. This option is enabled only when YUV420 is enabled and selecting 8 will enable 8K at 60 fps resolution.
- **Enable HDCP 1.3 Encryption:** Enables the HDCP 1.3 encryption.
- **Enable HDCP 2.3 Encryption:** Enables the HDCP 2.2/2.3 encryption.
- **Enable SDP including Audio:** Enables the secondary data packet (SDP) support including audio.
- **Audio Channels:** Selects the number of audio channels.
- **AUX I/O Buffer location:** Selects the buffer location for AUX channel.

User Parameters

The following table shows the relationship between the 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 1.4 TX Subsystem is through software. Maximum pixel mode support is aligned to the lane count.

Table 32: **User Parameters**

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
MST Streams	NUM_STREAMS	1
Lane Count	LANE_COUNT	4
Bits Per Color	BITS_PER_COLOR	8
Enable HDCP 1.3	HDCP_ENABLE	0
Enable SDP	AUDIO_ENABLE	0
Enable HDCP 2.2/2.3	HDCP22_ENABLE	0
YUV420 Support	ENABLE_420	0
PPC for YUV420 Colorimetry	PPC_FOR_420	4
Enable FEC	INCLUDE_FEC_PORTS	0
Enable DSC	ENABLE_DSC	0
Enable 8B10B decoder	ENABLE_8B10B_DEC	0
Number Of Audio Channels	AUDIO_CHANNELS	2
Pixel Mode	PIXEL_MODE	1/2/4 (Valid only in native mode)
AUX I/O Buffer Location	AUX_IO_LOC	Internal
Enable eDP	EDP_ENABLE	0
Link Rate	LINK_RATE	8.1

Output Generation

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

Constraining the Subsystem

Required Constraints

This section is not applicable for this IP subsystem.

Device, Package, and Speed Grade Selections

See IP Facts in the related information below for details about supported devices.

This section is not applicable for this IP subsystem.

Clock Frequencies

See Clocking for more details about clock frequencies. For more information on GT clocking, see the *Video PHY Controller LogiCORE IP Product Guide* (PG230).

Clock Management

This section is not applicable for this IP subsystem.

Clock Placement

This section is not applicable for this IP subsystem.

Banking

For more information on the specific banking constraints, see the *Video PHY Controller LogiCORE IP Product Guide* (PG230).

Transceiver Placement

For more information on the specific transceiver placement constraints, see the *Video PHY Controller LogiCORE IP Product Guide* (PG230).

I/O Standard and Placement

This section is not applicable for this IP subsystem.

AUX Channel

The *VESA DisplayPort Standard* (VESA website) describes the AUX channel as a bidirectional LVDS signal. You should design the board as recommended by the VESA DP Protocol Standard. For reference, see the example design XDC file.

For UltraScale+™ and UltraScale™ families supporting HR IO banks, use the following Source constraints:

```
set_property IOSTANDARD LVDS_25 [get_ports aux_tx_io_p]
set_property IOSTANDARD LVDS_25 [get_ports aux_tx_io_n]
```

For UltraScale+ and UltraScale families supporting HP IO banks, use the following Source constraints:

```
set_property IOSTANDARD LVDS [get_ports aux_tx_io_p]
set_property IOSTANDARD LVDS [get_ports aux_tx_io_n]
```

For Versal™ devices, use the following constraints:

```
set_property IOSTANDARD LVCMOS15 [get_ports aux_tx_io_p]
set_property IOSTANDARD LVCMOS15 [get_ports aux_tx_io_n]
```

HPD

The HPD signal can operate in either a 3.3V, 2.5V, or 1.5V I/O bank. By definition in the standard, it is a 3.3V signal.

For UltraScale+ and UltraScale families supporting HR IO banks, use the following constraints:

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

For UltraScale+ and UltraScale families supporting HP IO banks, use the following constraints:

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

For Versal devices, use the following constraints:

```
set_property IOSTANDARD LVCMOS15 [get_ports hpd]
```

Board design and connectivity should follow DisplayPort standard recommendations with proper level shifting.

Related Information

IP Facts
Clocking

Simulation

There is no simulation support for DisplayPort 1.4 TX Subsystem.

Synthesis and Implementation

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

Example Design

***Note*:** All example designs use the Inrevium TB-FMCH-VFMC-DP FMC card.

This chapter contains step-by-step instructions for generating an Application Example Design from the DisplayPort 1.4 TX Subsystem by using the Vivado® Design Suite flow.

RECOMMENDED: *For ZCU102/VCU118 (Revision 1.0 or later), you should set up a 1.8V setting after connecting the DisplayPort FMC. For details on the FMC voltage, see the Related Information link. For more information, see the ZCU102 System Controller GUI Tutorial (registration required) (XTP433) and the VCU118 System Controller Tutorial (registration required) (XTP447). For VCK190 (Revision 1.0 or later), you should set up a 1.5V setting after connecting the DisplayPort FMC.*

Related Information

Setting the FMC Voltage to 1.8V

Available Example Designs

The following table shows the example designs available for the TX and RX DisplayPort 1.4 subsystems.

Table 33: **Available Example Designs**

GT Type	Topology	Video PHY Config[1]		Hardware	BPC	Processor
		(TXPLL)	(RXPLL)			
GTHE3	Pass-through without HDCP1.3	QPLL	CPLL	KCU105 + Inrevium TB-FMCH-VFMC-DP	8	MicroBlaze™

Table 33: **Available Example Designs** *(cont'd)*

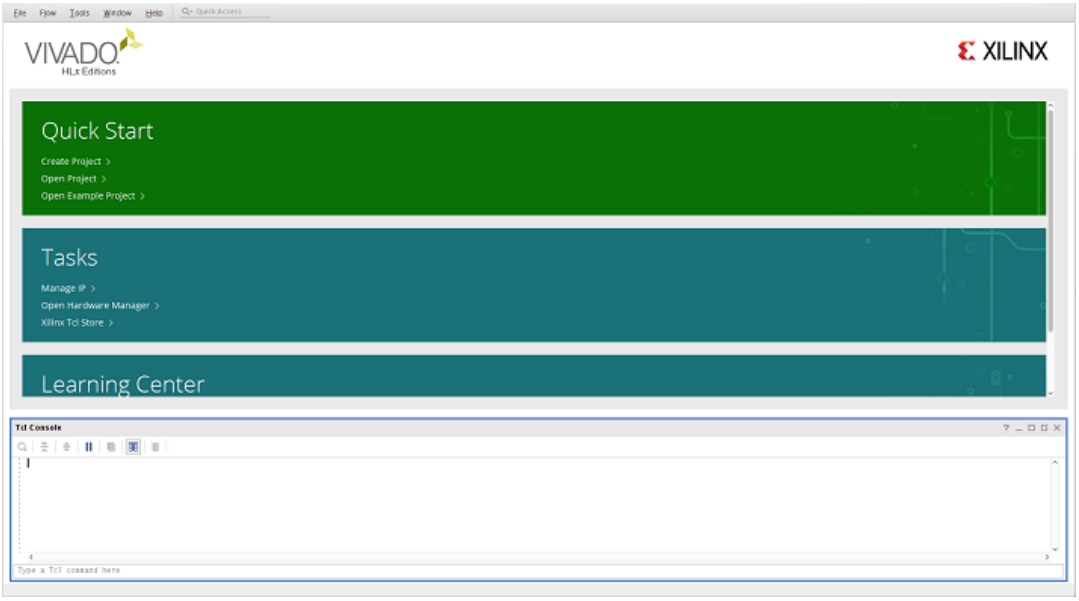
GT Type	Topology	Video PHY Config[1]		Hardware	BPC	Processor
		(TXPLL)	(RXPLL)			
GTHE4	RX only	-	CPLL	ZCU102 + Inrevium TB-FMCH-VFMC-DP	10	A53
	TX only	QPLL	-	ZCU102 + Inrevium TB-FMCH-VFMC-DP	10	A53
	FB Pass-through without HDCP1.3/HDCP2.2/2.3[2]	QPLL	CPLL	ZCU102 + Inrevium TB-FMCH-VFMC-DP	10	A53
	FB Pass-through with HDCP1.3 and HDCP2.2/2.3	QPLL	CPLL	ZCU102 + Inrevium TB-FMCH-VFMC-DP	10	A53
	MST FB Pass-through without HDCP1.3 and TX only	QPLL	CPLL	ZCU102+ Inrevium TB-FMCH-VFMC-DP	10	A53
GTYE4	RX only	-	CPLL	VCU118 + Inrevium TB-FMCH-VFMC-DP	10	MicroBlaze
	TX only	QPLL	-	VCU118 + Inrevium TB-FMCH-VFMC-DP	10	MicroBlaze
GTYE5	TX Only	RPLL	-	VCK190 + Inrevium TBFMCH-VFMC-DP	10	A72
	FB Pass-through without HDCP1.3/HDCP2.2/2.3[2]	RPLL	LCPLL	VCK190 + Inrevium TBFMCH-VFMC-DP	10	A72

Notes:

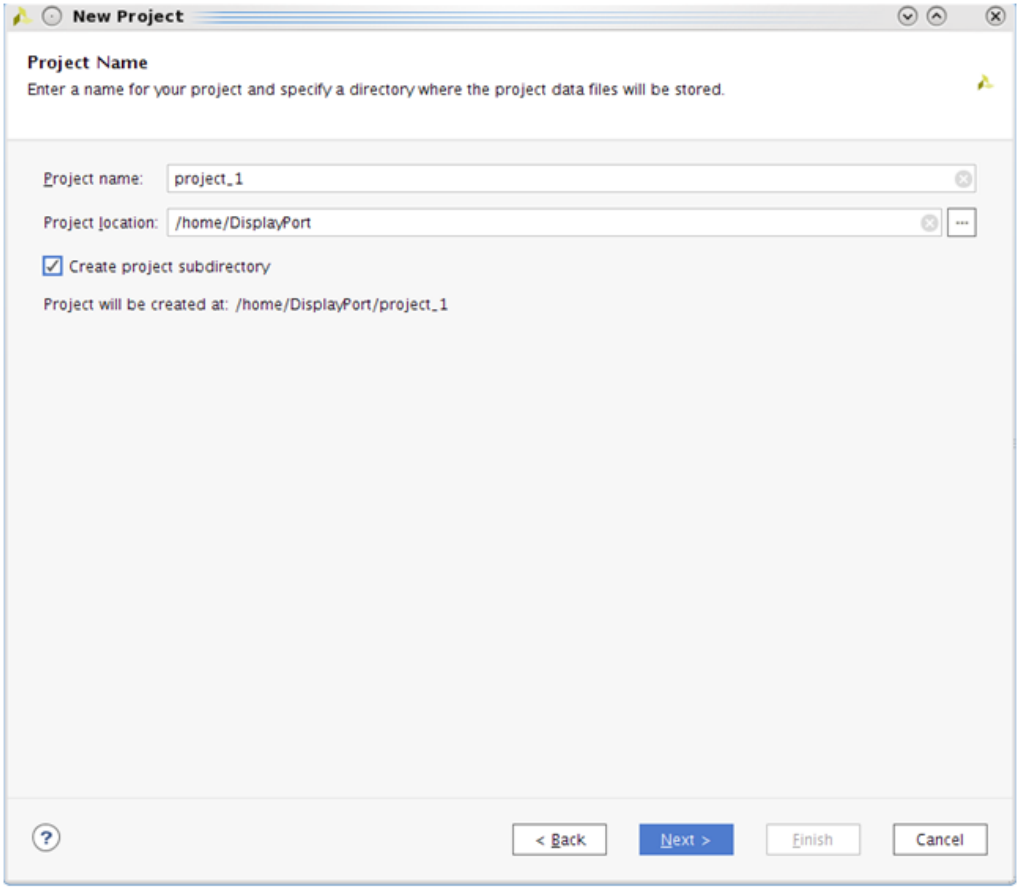
1. GT data width is 2 bytes.
2. Both DP 1.4 TX and DP 1.4 RX Subsystems use the Adaptive-Sync feature.

Building the Example Design

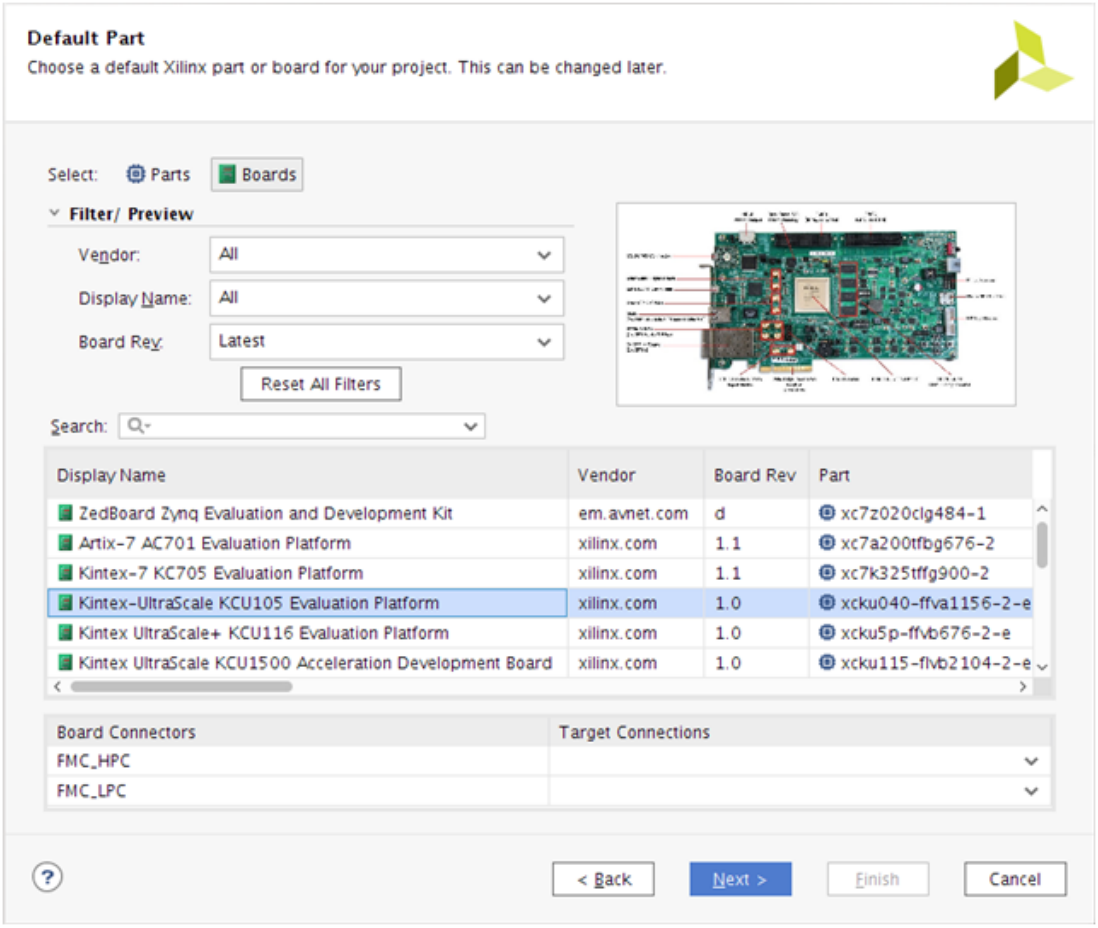
1. Open the Vivado® Design Suite and click **Create Project**.



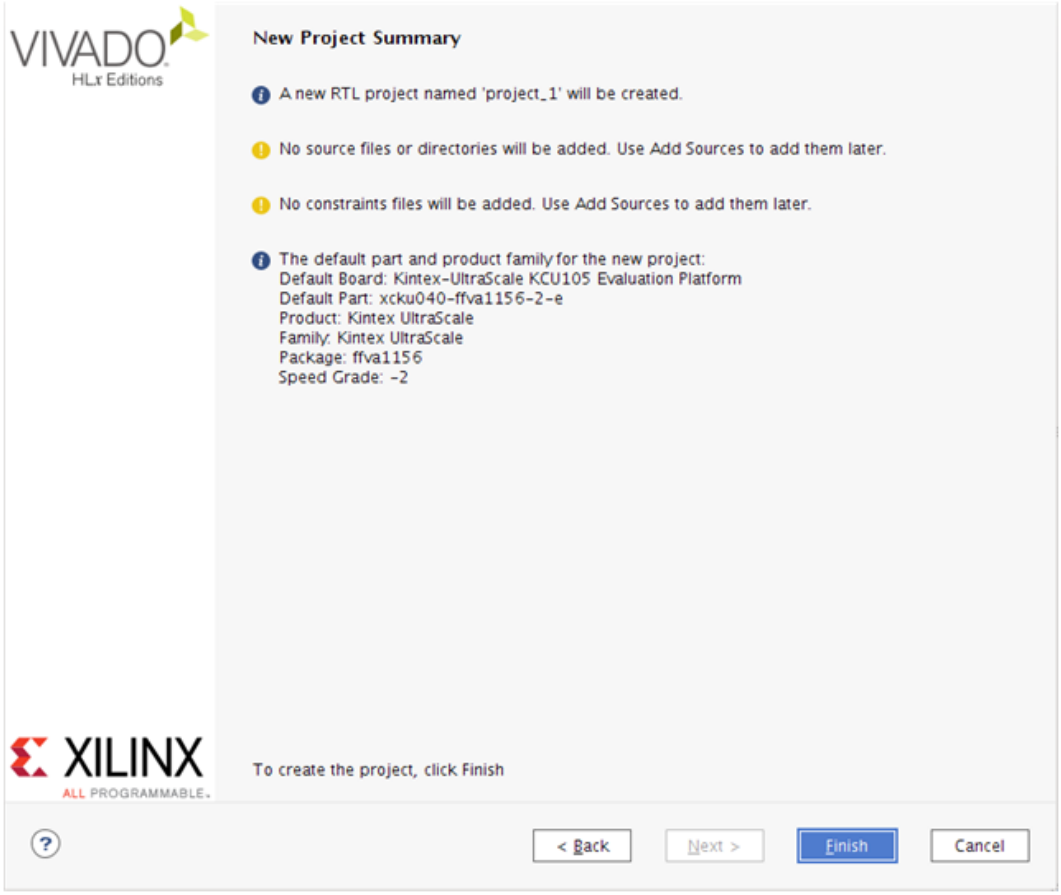
2. In the **New Project** window, enter a **Project name**, **Project location**, and click **Next** up to the Board/Part selection window.



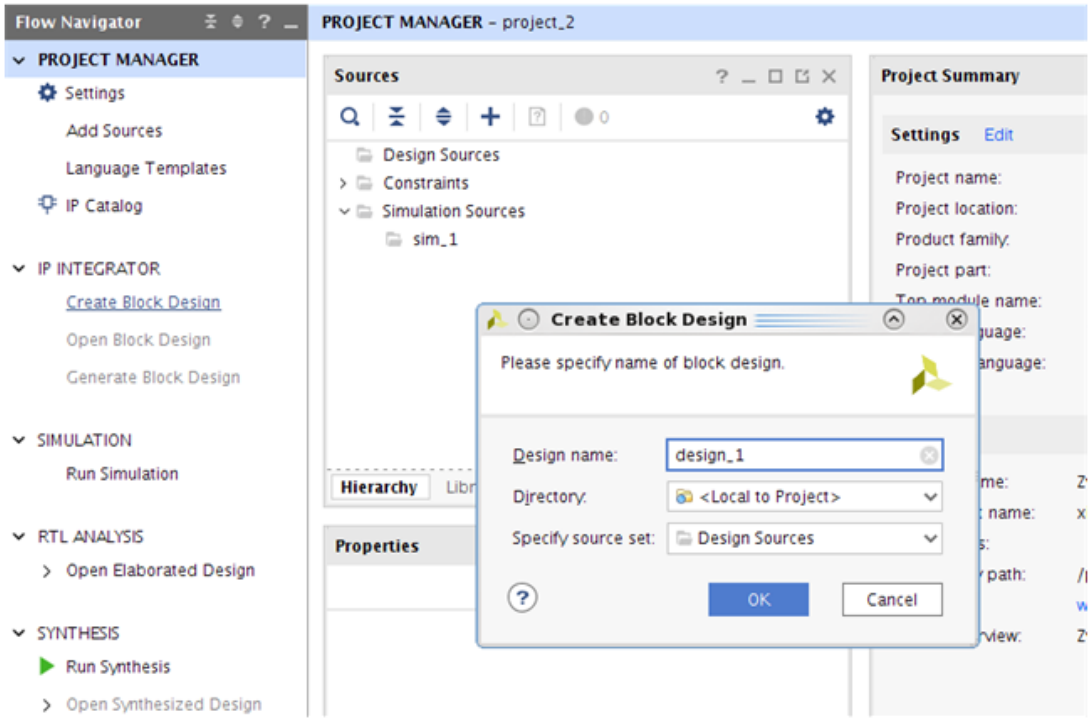
3. In the **Default Part** window, select the Board as per your requirement. Application Example Designs are available for KCU105, ZCU102, VCU118, and VCK190.



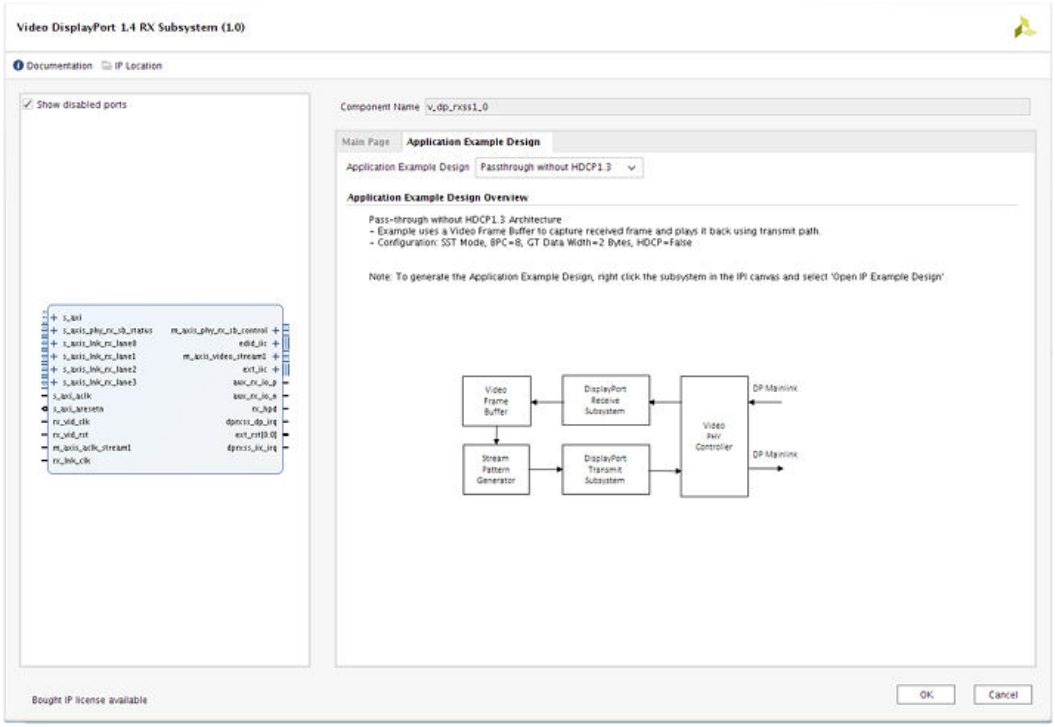
4. Click **Finish**.



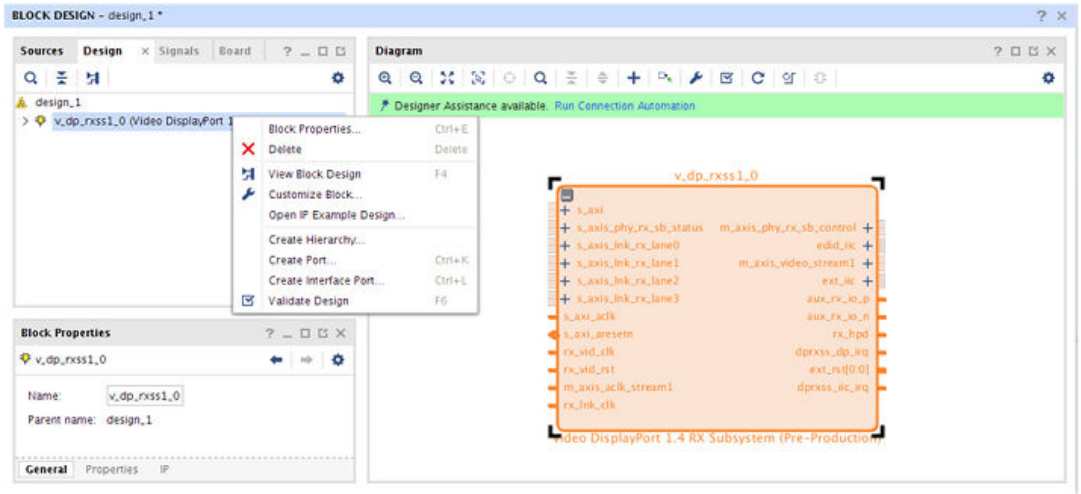
5. In the Flow Navigator window, click **Create Block Design** (BD). Select a name for the Block Design and click **OK**.



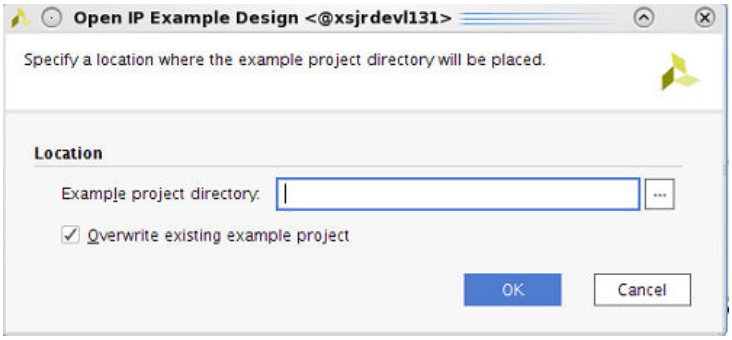
6. Right-click **BD** and click **Add IP**. Search for DisplayPort 1.4 and select either the DisplayPort 1.4 Receiver Subsystem IP (for RX only (ZCU102, VCU118), Pass-through (KCU105, ZCU102, and VCK190) designs) or the DisplayPort 1.4 Transmitter Subsystem IP (for TX only (ZCU102, VCU118, and VCK190), or Pass-through (KCU105) designs).
7. Double-click the **IP** and go to the **Application Example Design** tab in the **Customize IP** window. Select the supported topology in the **Application Example Design** drop-down box. Click **OK** and **Save** the block design.



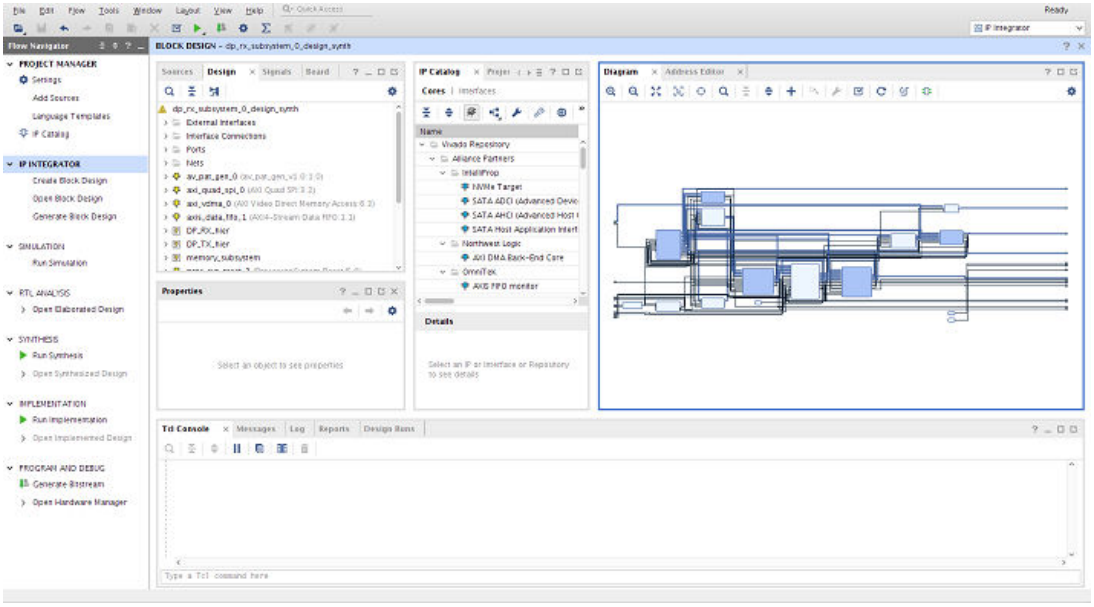
8. Right-click the **DisplayPort Subsystem** IP under Design source in the **Design** tab and click **Open IP Example Design**.



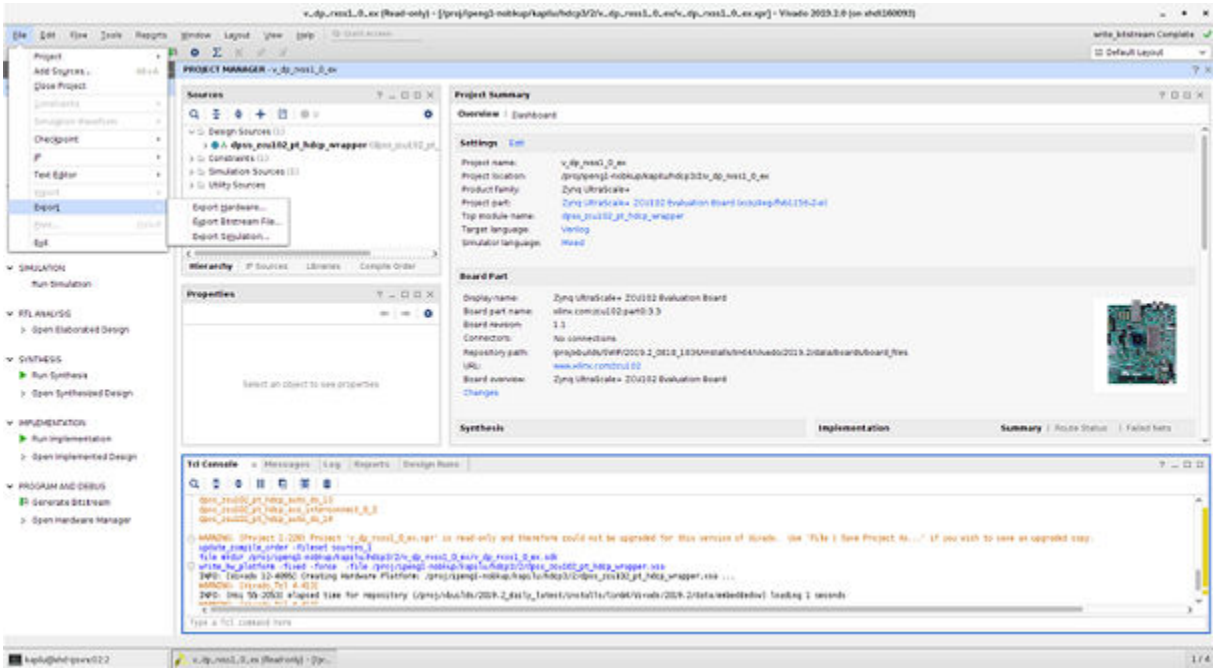
9. Choose **Example project directory** and click **OK**.



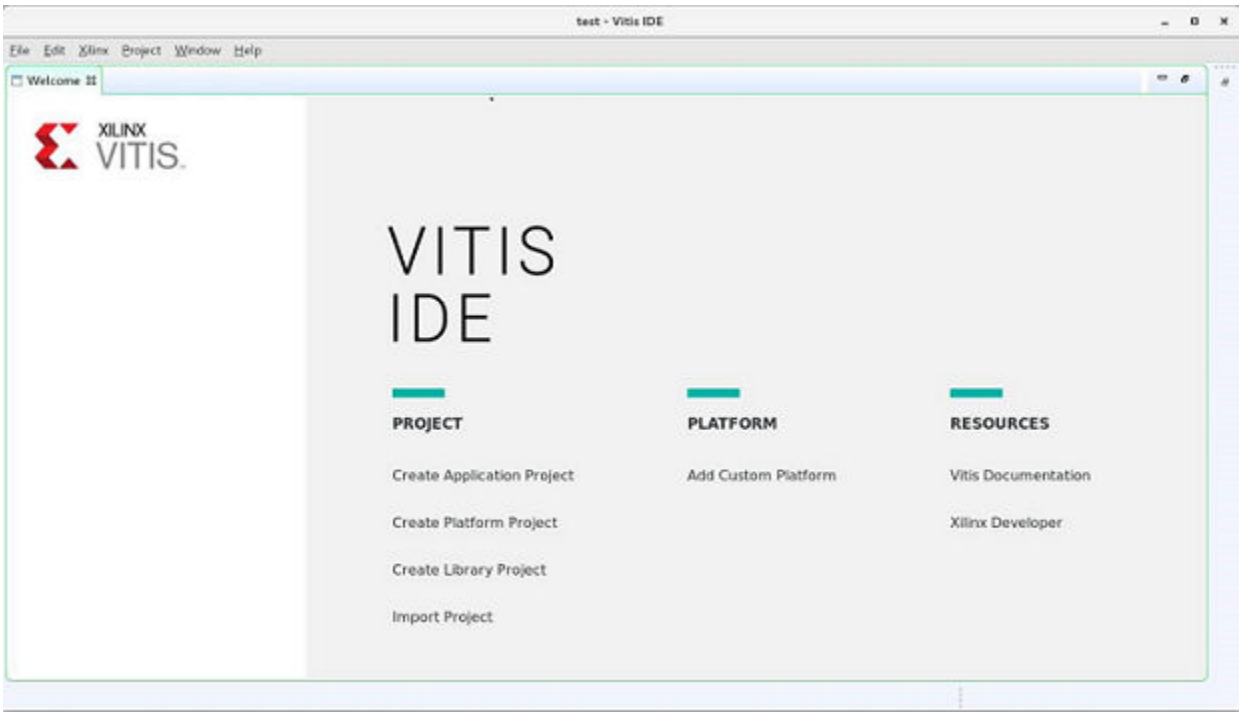
10. The following figure shows the Vivado IP integrator design. Choose the **Generate Bitstream**.



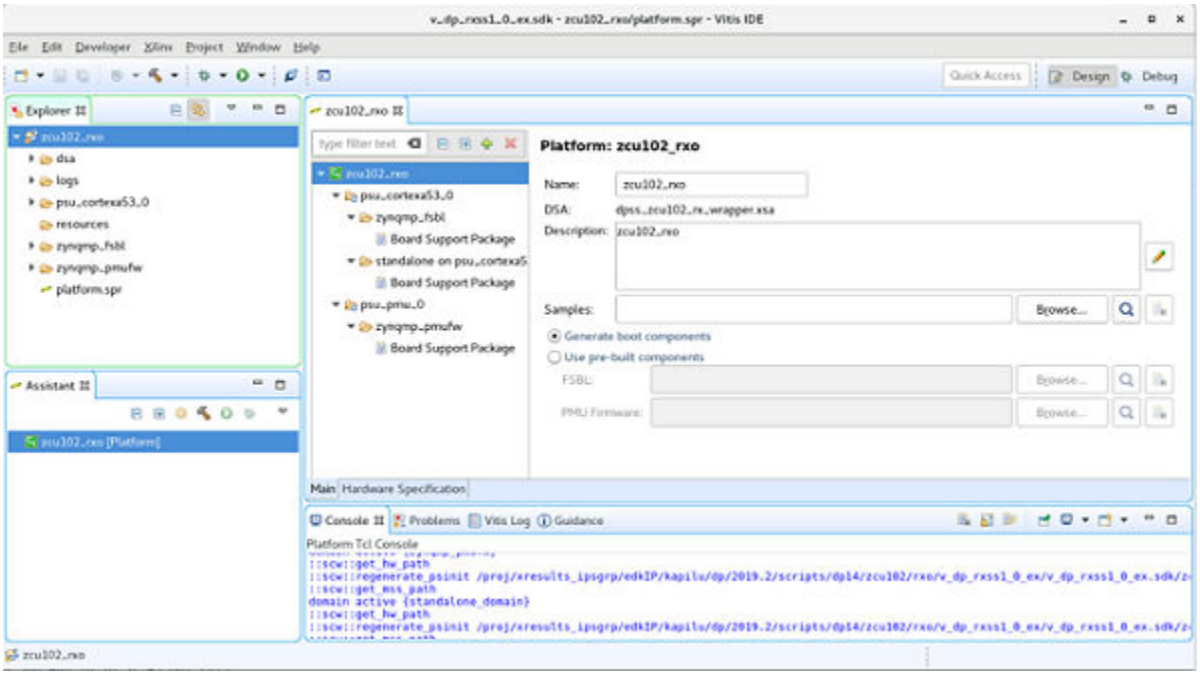
11. Export the hardware (xsa) to Vitis software platform. Click **File → Export → Export Hardware**.



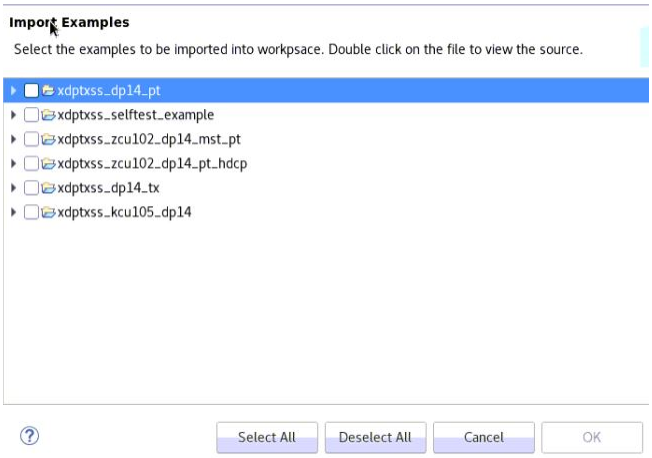
12. Launch the Vitis software platform from command line. Setup a workspace and create a platform project using exported xsa.



13. The following figure shows an example of the launched Vitis platform when built successfully.

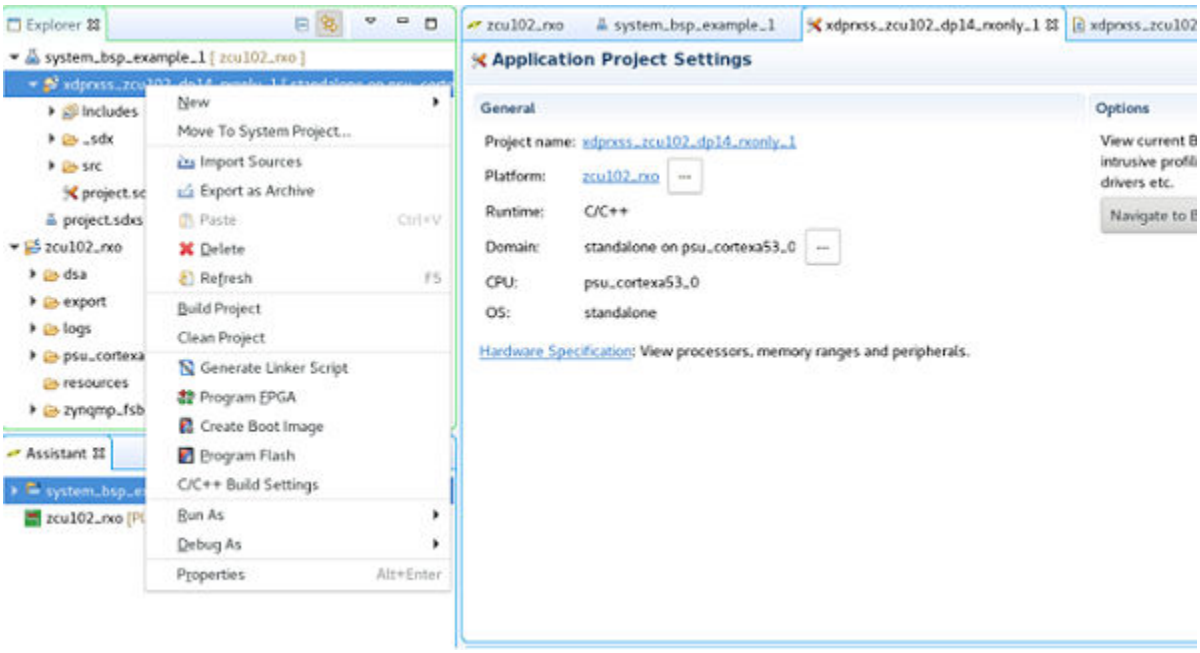


14. Click **Board Support Package** from Vitis. Click **Import Examples**.



15. Select the Example Application corresponding to your hardware:
    - For the Pass-through KCU105 project, select the `*_kcu105_dp14` option.
    - For the TX- only ZCU102/VCU118/VCK190 project, select the `*_dp14_tx` option.
    - For the Pass-Through ZCU102/VCK190 with I2S audio interface project, select the `*_dp14_pt` option.
    - For the Pass-Through ZCU102 with HDCP project, select the `*_dp14_pt_hdcp` option.
    - For the MST FB Pass-Through ZCU102 project, select the `*_zcu102_dp14_mst_pt` option.

16. Build the example in Vitis software platform. Right-click and select **Build Project**.



Hardware Setup and Run

1. Connect the Tokyo Electron Device Limited (TED) TB-FMCH-VFMC-DP module to the HPC FMC connector on the KCU105 board, to the FMCP2 connector on the VCK190 or to the HPC0 connector on the ZCU102, or to the FMCP HSPC connector on the VCU118 depending on your design.
2. Connect a USB cable (Type A to mini B) from the host PC to the USB UART port on the KCU105 for serial communication. For the KCU105, ZCU102, or VCU118, use Type A to micro B type of USB cable. For the VCK190 board, connect the USB cable (Type A to Type C) from the host PC to the USB port (U20).
3. Connect a JTAG USB Platform cable or a USB Type A to Micro B cable from the host PC to the board for programming bit and elf files.
4. For pass-through or TX-only applications, connect a DP cable from the TX port of the TED TB-FMCH-VFMC-DP module to a monitor, as shown in the following figure.
5. For pass-through or RX-only applications, connect a DP cable from the RX port of the TED TB-FMCH-VFMC-DP module to a DP source (GPU), as shown in the following figure.

Figure 24: **KCU105 Board Setup**

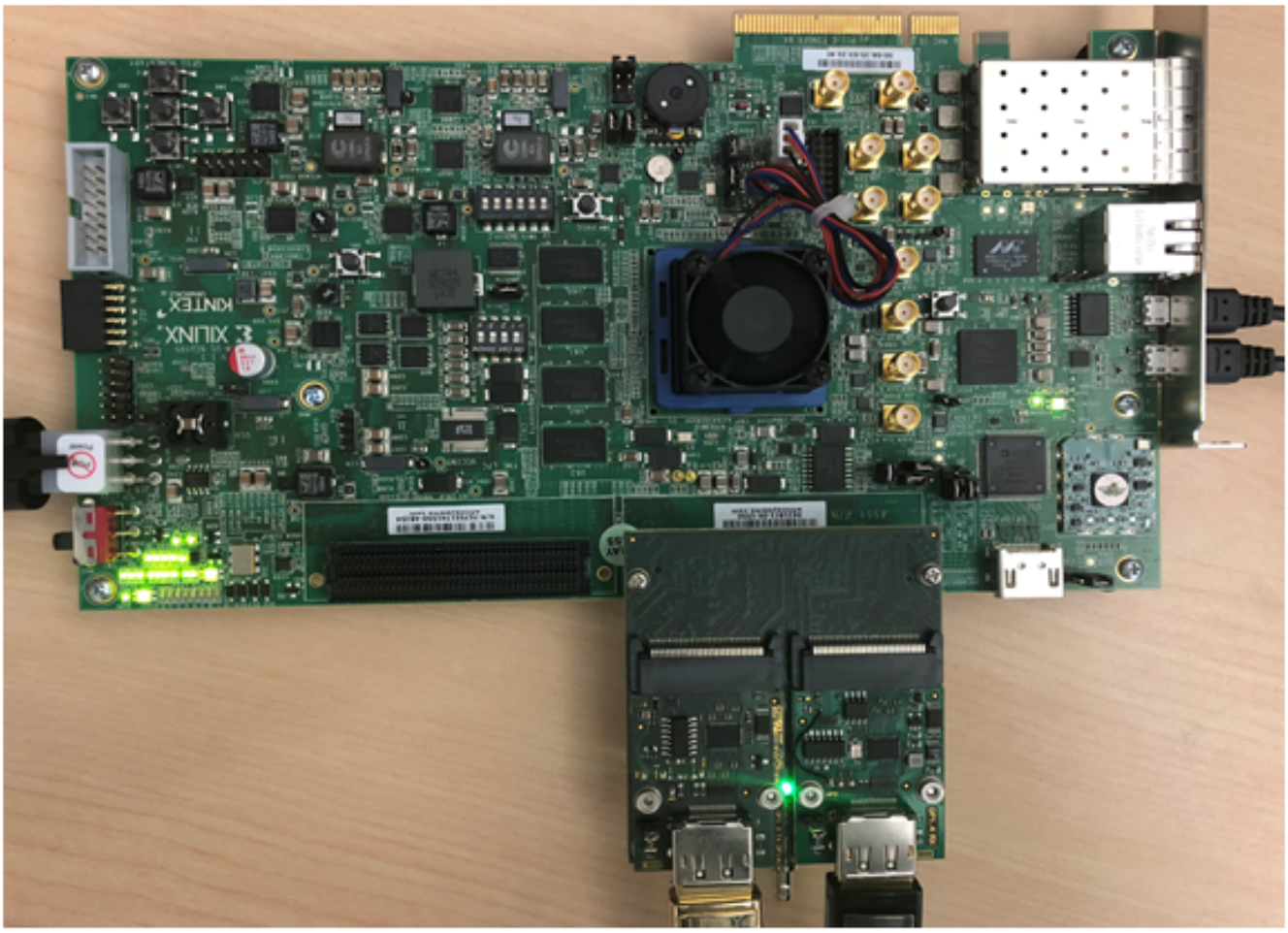


Figure 25: **ZCU102 Board Setup**

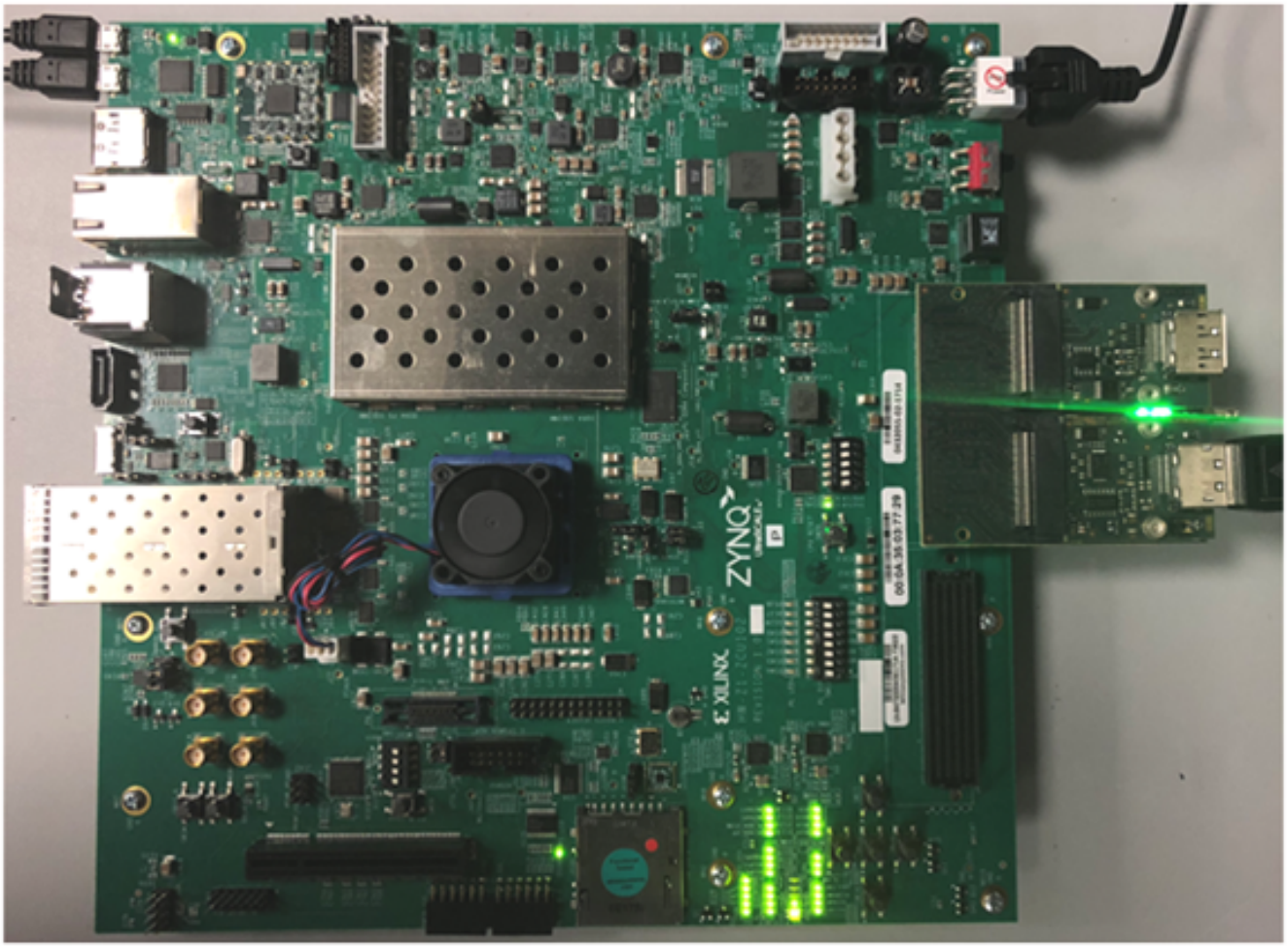
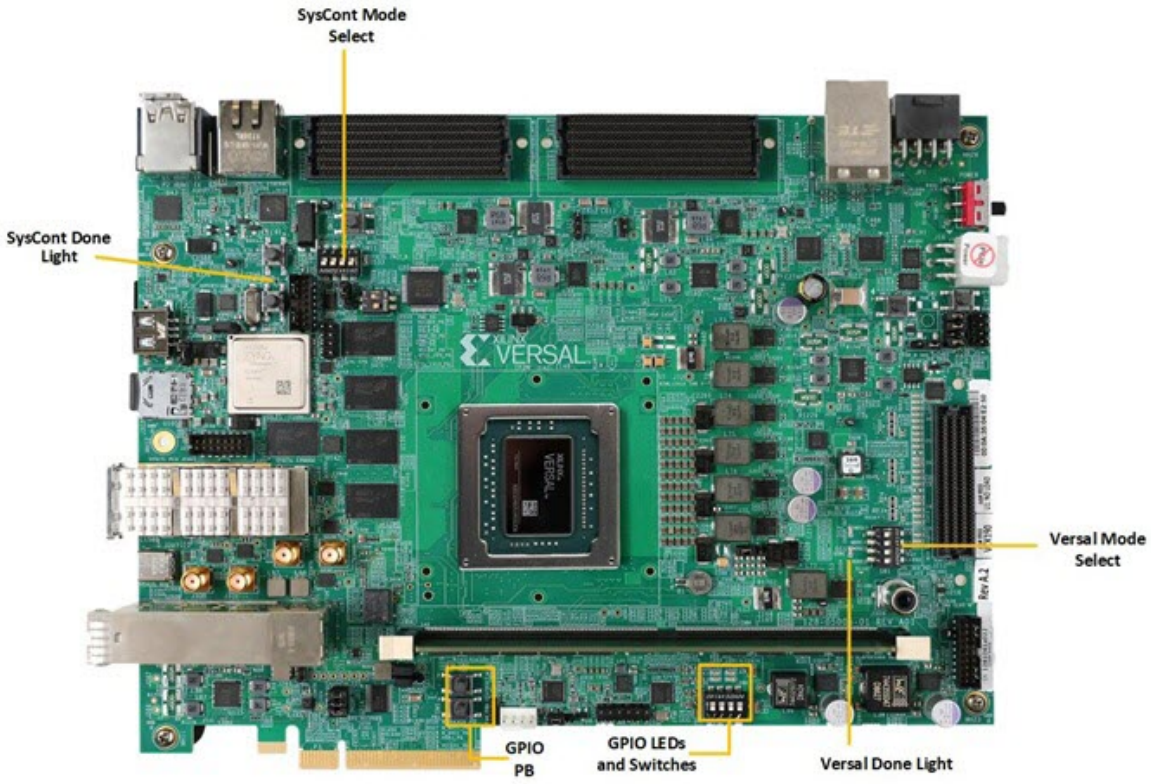
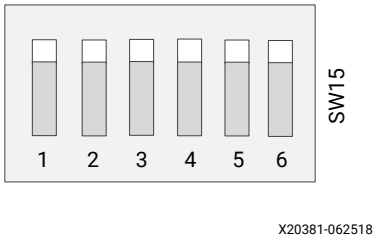


Figure 26: **VCK190 Board Setup**

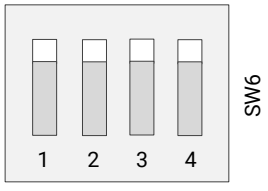


6. On the KCU105 set the mode pin to SW15:



X20381-062518

7. On the ZCU102 set the mode pin to SW6:



X19805-062518

8. Connect the power supply and power on the board.
9. Start an UART terminal program such as Tera Term or Putty with the following settings:
   a. Baud rate = 115200 for KCU105/ZCU102/VCU118/VCK190
   b. Data bits = 8

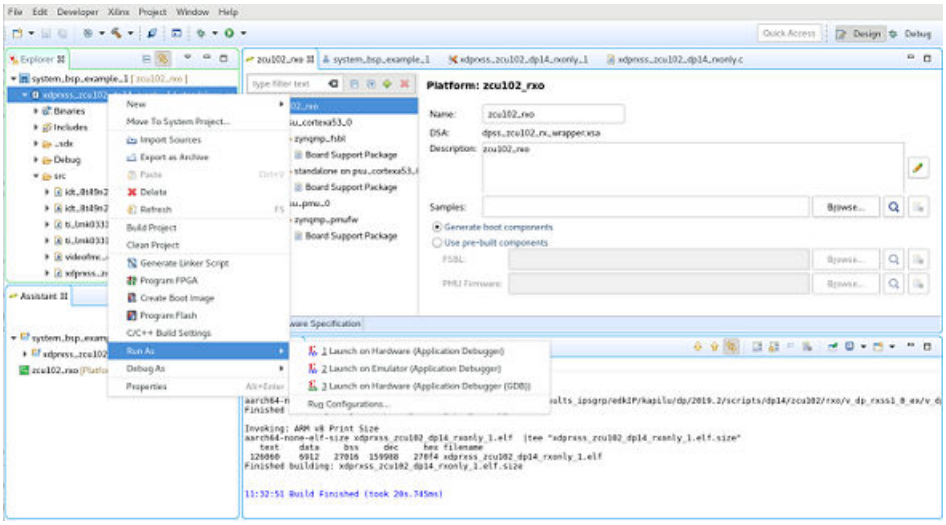
c. Parity = none

d. Stop bits = 1

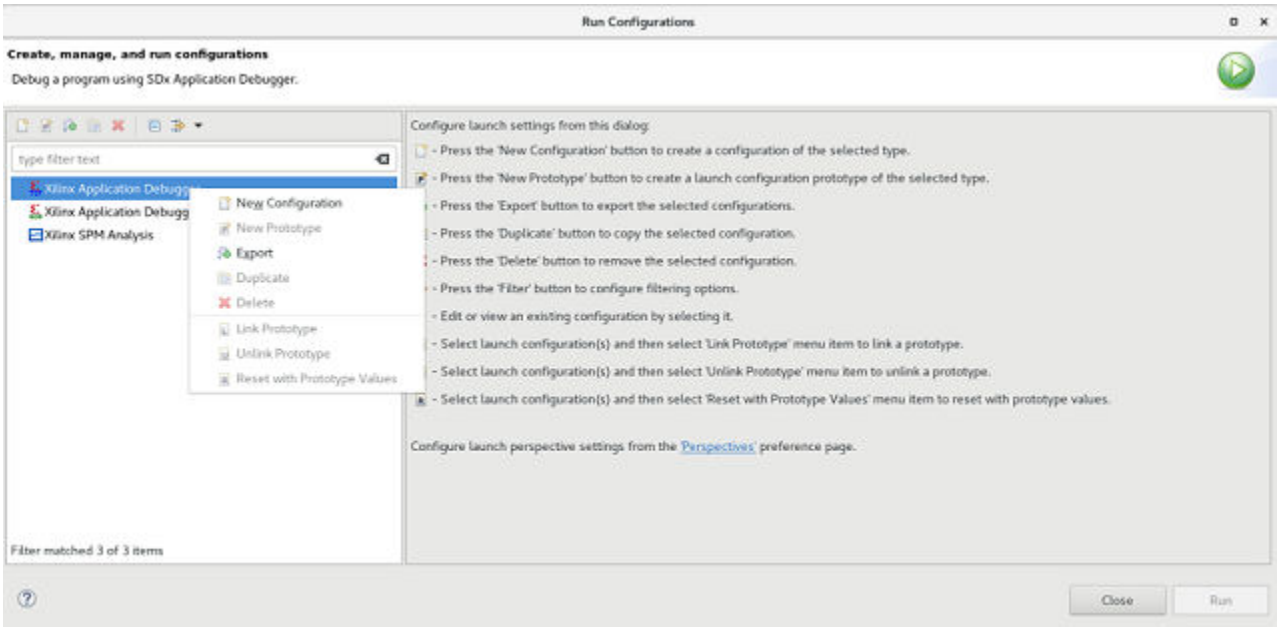
e. Flow Control = none

Note: With the ZCU102 board, there are four COM ports available.

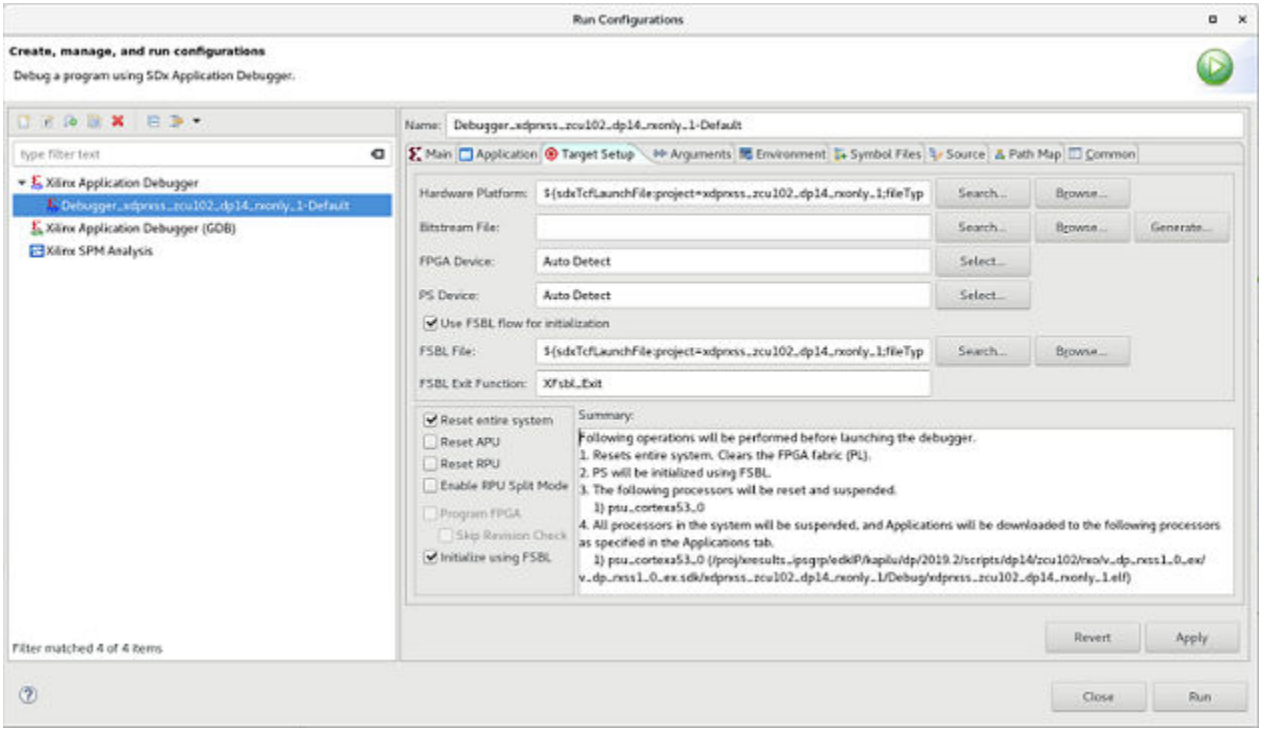
10. In the Vitis IDE, under the **Project Explorer**, right-click the application and click **Run As → Run Configurations**.



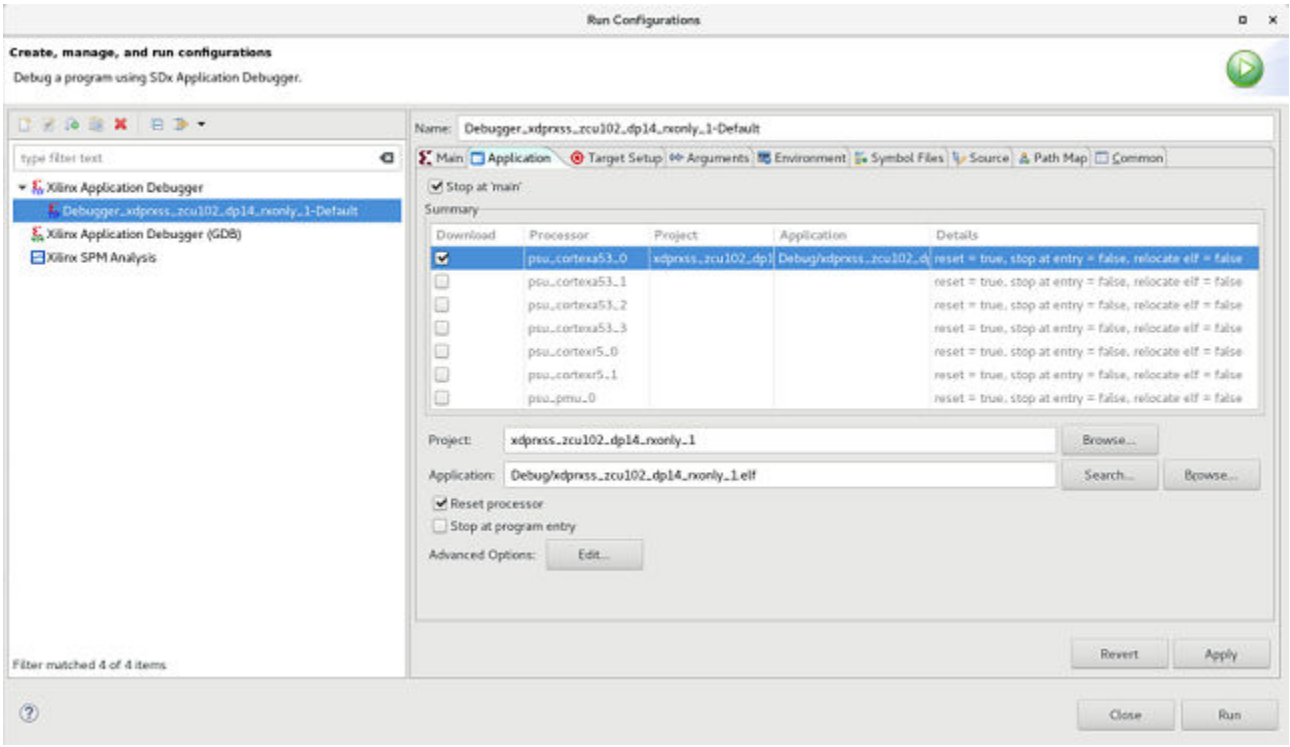
11. In the **Run Configurations** popup menu, right-click **Xilinx Application Debugger** and click **New**.



12. In the **Target Setup** tab, the **Reset entire system** is enabled.



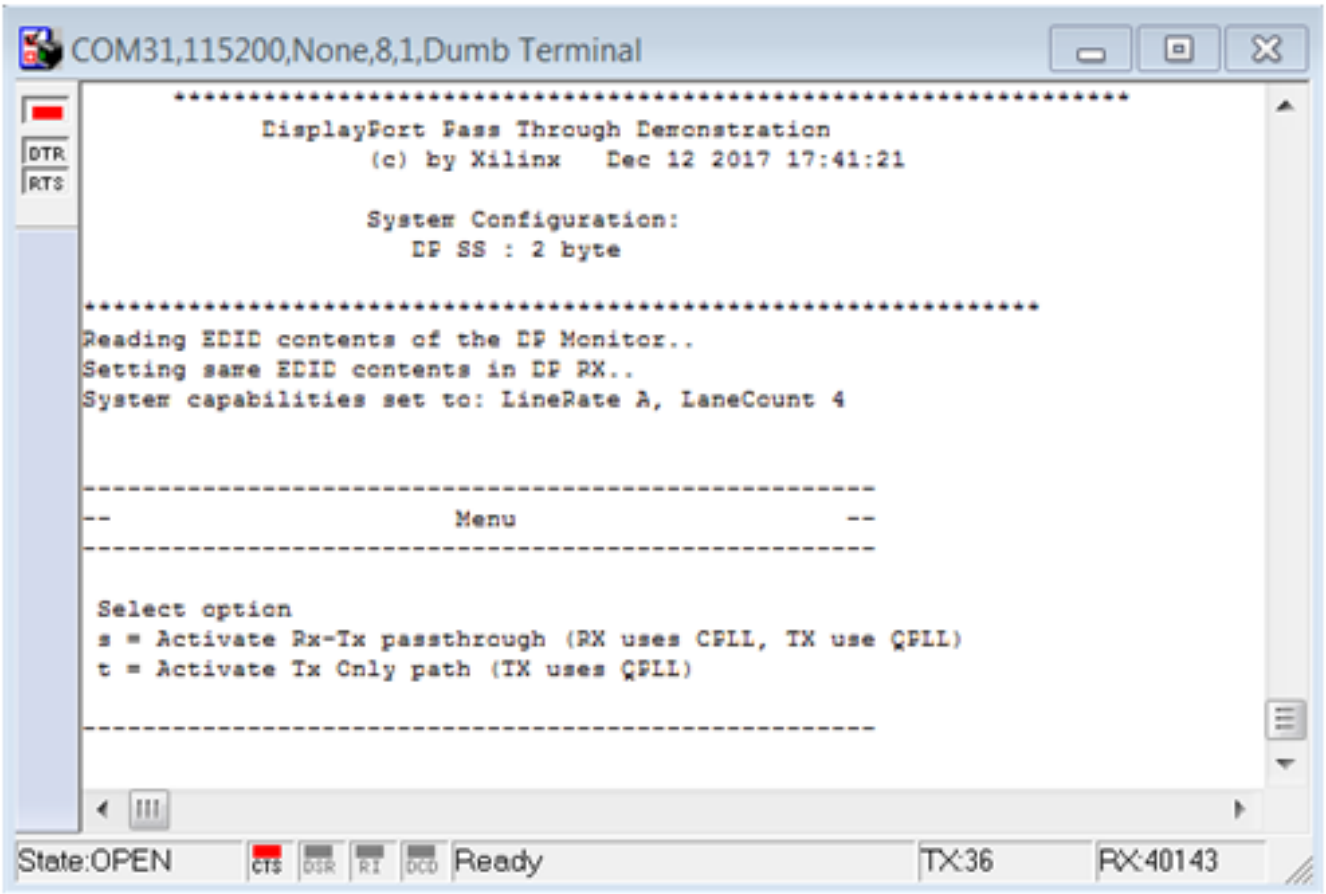
13. In the **Application** tab click **Run**.



Display User Console

In a Pass-Through application (KCU105), as soon as the application is executed, it checks if a Monitor is connected or not. If a monitor is already connected, then it starts up the following options as shown in the following figure to choose from (KCU105).

Figure 27: **DisplayPort User Console**



Selecting either `r` or `s` puts the system in Pass-Through mode, where the video received by the RX is forwarded to the TX. This configures the `vid_phy_controller` and sets up the DisplayPort for RX. If a DisplayPort Source (for example, GPU) is already connected to DisplayPort RX Subsystem, then it starts the training. Otherwise, the training starts when the cable is plugged in. As soon as the training is completed, the application starts the DisplayPort TX Subsystem. The video should be seen on the monitor after the TX is up. The previous figure shows the UART transcript. The transcript might differ based on the training done by the GPU.

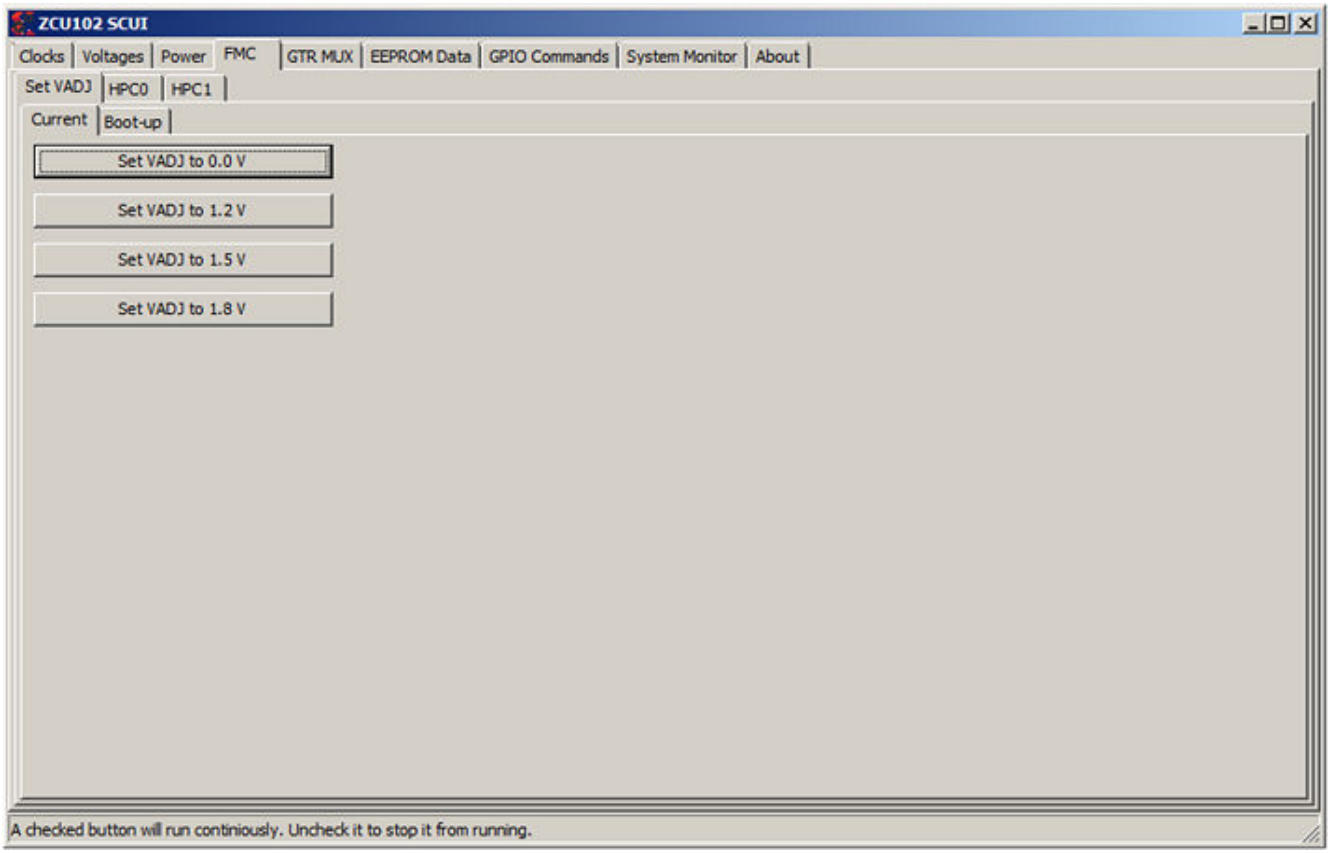
Setting the FMC Voltage to 1.8V

To run the example design on the ZCU102/VCU118 board, ensure that only one ZCU102/VCU118 board is connected to the host PC. This tool does not work with multiple ZCU102/VCU118 connected to the host PC. There is no UART selection in this tool. Also, the FMC voltage is set to 1.8V. If you forget to set the FMC voltage, the following symptoms might occur:

- Random AUX failures
- Training failures

To set the FMC VADJ voltage:

1. Connect the ZCU102/VCU118 board from the host PC to the USB UART port and power up the board.
2. Open the ZCU102/VCU118 SCUI tool and select the **FMC** tab.
3. On the **Set VADJ** tab, select the **Set VADJ to 1.8V**.



Note: The SCUI tool can only be used with one board per one PC. If there are more than two ZCU102 boards connected to a PC, then it does not work.

Configuring HDCP Keys and Key Management

HDCP 1.3 Key Management

The application software does not use the raw HDCP 1.3 keys directly. To use the HDCP 1.3 keys, they have to be first encrypted and then added into the application. This is a manual process. This section provides the scripts and software to help you encrypt the HDCP 1.3 keys.

Using the Encryption Software

For more information on using the encryption software, see AR: 70605. Before you begin, download and extract the ZIP files. To generate the AES encrypted HDCP 1.3 keys, you must have the following keys:

- 32-byte AES key
- Valid HDCP 1.3 keys

***Note*:** Xilinx does not provide any of the above keys. The application delivered with this software is created with invalid keys and therefore does not play HDCP content.

Generate the AES encrypted HDCP key block by following these steps:

1. Unzip the project and navigate to the `hdcp_util/keys` directory. This is where the encryption block is located.
2. Modify the `gDefaultKey` array in the `hdcp_util/keys/key-encryptor/common/src/keyfile.c` file to a user specified 32-byte unique key. This is the 32-byte AES key mentioned in step 1.
3. Navigate to the `hdcp_util/keys/key-encryptor/build/linux` folder and execute the following command from linux terminal.

   ```
   ./build.sh
   ```

   This creates `hdcp-enc.bin` file in the same folder.
4. From the same directory, execute the following command to create the AES encrypted file `keymgmt_data.c`.

   ```
   ./hdcp-enc.bin -c devb1_keys.dat devb2_keys.dat …. devb_keys.dat
   ```

   ***Note*:** The user-provided `devb_keys.dat` file is expected to have only one original HDCP key. An original HDCP key has one 5-byte Key Selection Vector and 40 private keys. If you have multiple HDCP 1.3 keys, each key should be housed exclusively in one `.dat` file.

   The AES encrypted HDCP key block is now created as an array in the `keymgmt_data.c` file.

5. Ensure that the following AES keys in the reference design matches with the keys in step 2.

```
\**zcu102_system_directly**\**sw_directly**\src\keys.c
```

   The HDCP 1.3 keys are now encrypted and ready to be used.

6. Replace the dummy 32-byte AES keys in the `gDefaultKey[32]` array of the example design application's source file `src/keymgmt_keyfile.c` with the keys mentioned in step 2.
7. Replace the dummy AES encrypted HDCP keys in the `KEYMGMT_ENCDATA[]` array of the example design application's source file `src/keys.c` with the encrypted keys generated in step 5.

HDCP 2.x Key Management

1. Populate the 128 bit global constant LC128 keys in the `XHdcp22Lc128[]` array of the `keys.c` file. Replace zeros in the array with the LC128 keys.
2. Populate the HDCP 2.x Rx private keys in the `XHdcp22RxPrivateKey[]` array of the `keys.c` file. Replace the zeros in the array with Rx private keys.
3. Populate the System Renewability Revoke Check Message (SRM) keys in the `Hdcp22Srm[]` array of the `keys.c` file. Replace the zeros in the array with the SRM keys.

Tested Equipment

The following table lists the tested equipment used with the example design.

Table 34: **Sink Equipment**

Sink Type	Brand Name	Model Name
Monitor	Acer	S277HKWMIDPP
Monitor	Dell	S2817
Monitor	Dell	UP3218K
Monitor	Dell	U3014T
Monitor	Dell	U2713
Monitor	LG	27UD68P
Monitor	LG	24UD58
Monitor	Asus	PQ321Q
Monitor	Samsung	LU28D590DS
Tester	Unigraf	UCD-323
Tester	Unigraf	UCD-400
Tester	Unigraf	DPR-100

Appendix A

Upgrading

There is no direct upgrade path from the older Xilinx® DisplayPort1.2 solution due to the new retimer. Xilinx recommends starting with a new design.

Appendix B

Questions and Answers

Q. Can both RX and TX be used on the same GT quad for DisplayPort?

A. Yes. The Video PHY Controller supports the capability of performing both RX and TX on the GT quads. However, they cannot be different protocols.

Q. Does the Video PHY Controller support different protocols for RX and TX?

A. No. The Video PHY Controller must use the same protocol if both RX and TX is being used.

Q. I am having link training issues. What are some things that can be done to improve link training?

A. Perform the following:

1. Verify that all relevant ARs are taken into account.
2. Increase the AUX_DEFER value in register offset `0x004`.
3. Use an AUX analyzer, such DPA400 to check AUX traffic.

Q. Does the Xilinx® subsystem support my resolution and frame rate?

A. Perform the following:

1. DisplayPort should operate at any resolution and frame rate as long as the DisplayPort link is not oversubscribed. Use the following equation to determine if the custom resolution can be supported:

   $(H_{Total} \times V_{Total} \times bits\_per\_component \times frame\ rate) < (0.8 \times link\_lane \times num\_lanes)$
2. Verify with the sink EDID that the resolution and frame rate are supported by the sink device.

XILINX

Appendix C

Driver Documentation

The bare-metal driver documentation can be found at the Xilinx GitHub page. The Linux driver documentation can be found on the Xilinx Wiki.

XILINX

Appendix D

Helper Core

This appendix describes Audio Video (AV) pattern generator (av_pat_gen) , Video Frame CRC (video_frame_crc) and AXI4S Video Re-mapper (v_axi4s_remapper) helper cores. Helper cores are IPs that are not part of Vivado IP catalog but only available in the as part of example design. They are not supported outside the original example design context. Users are expected to design their own IPs in their designs.

Audio Video (AV) pattern generator (av_pat_gen)

`av_pat_gen` module acts as a test pattern generator for both video and audio and in passthrough design this IP is connected before TX CRC module so as to enable user to pass test patterns on to the TX CRC module and then to the DisplayPort TX. `av_pat_gen` also has a switch for the user to select whether to pass received input video/audio data (from DP RX in passthrough system) to its output directly, or to generate test pattern video/audio data on to its output.

Note that this `av_pat_gen` is not part of the Display Port protocol requirements and is used only in the example design to assist users in validation. Hence, Xilinx do not provide separate product guide for `av_pat_gen` and do not provide support related to this IP.

Feature Summary

1. 1, 2 and 4 PPC are supported:

   "Pixel per clock" (PPC) denotes the number of video pixels that `av_pat_gen` can send, per input clock. Data outputted by `av_pat_gen` on `vid_out_axi4s` is such that the first pixel will be aligned to the lower significant part of `vid_out_axis_tdata`. For example, in 4 PPC mode:

   `vid_in_axis_tdata` = {pixel3, pixel2, pixel1, pixel0}

   For audio, `av_pat_gen` always outputs 32 bit audio sample per clock.

2. RGB, YCbCr 422 video formats are supported:

   a. RGB:

`av_pat_gen` outputs RGB video format such that “R” component is aligned to most significant part of the pixel and “G” is aligned to least significant part of the pixel. This is to mimic DP specification in arrangement of RGB components. For example,

pixel0 = {R, B, G}

b. YCbCr 422:

`av_pat_gen` outputs YCbCr 422 video format such that “Cr/Cb” component is aligned to most significant part and “Y” is aligned to least significant part of the pixel. For example,

Pixel0 = {Cb/Cr, Y}

3. 6, 8, 10, 12 and 16 BPCs are supported:

Bits per component (BPC) denotes number of bits per component of each pixel. Bits corresponding to components of the pixel are tightly packed. For example,

If pixel0 (RGB) is being outputted by DisplayPort with 6bpc, then

Pixel0[17:12] = R-component

Pixel0[11:6] = B-component

Pixel0[5:0] = G-component

4. Supported test patterns:

`av_pat_gen` can generate various types of video test patterns as listed in DP compliance specification and can generate up to 8 channels of audio streams. Below are the video patterns it supports

- Color ramp pattern
- Black and white vertical lines pattern
- Color square pattern
- Flat red pattern
- Flat green pattern
- Flat blue pattern
- Flat Yellow pattern

When `test_pattern` (address 0x308, [2:0]) is selected as 0, then video/audio stream input is directly passed to the output of `av_pat_gen`. Otherwise, depending on the other configuration written in to the above-mentioned configuration registers, video and audio test patterns are generated and outputted by `av_pat_gen` IP.

Port Descriptions

Signal Name	Interface	Type	Description
av_axi	s_axi_ctrl	Slave	AXI Slave bus interface to support register interface
vid_in_axi4s	Video AXI4S Interface	Slave	AXI4 streaming video input interface. This data will be transferred on to the video output directly if av_pat_gen is acting just as passthrough for video.
aud_in_axi4s	Audio AXI4S Interface	Slave	AXI4 streaming audio input interface. This data will be transferred on to the audio output directly if av_pat_gen is acting just as passthrough for audio.
vid_out_axi4s	Video AXI4S Interface	Master	AXI4 streaming video output interface
aud_out_axi4s	Audio AXI4S Interface	Master	AXI4 streaming audio output interface
av_axi_aclk	Clock	Input	Clock for s_axi_ctrl
av_axi_aresetn	Reset	Input	Active low reset for s_axi_ctrl
aud_out_axi4s_aclk	Clock	Input	Clock for input and output audio interface
aud_out_axi4s_aresetn	Reset	Input	Active low reset for input and output audio interfaces
aud_clk	Clock	Input	“aud_clk” and “aud_out_axi4s_aclk” both are connected to same source
vid_out_axi4s_clk	Clock	Input	Clock for input and output video interface
vid_out_axi4s_aresetn	Reset	Input	Active low reset for input and output video interfaces

Register Details

Register Offset	Access	Name	Default Value	Register Description
0x0	R/W	Enable	0	[0] – enables the generation of video data, when set to ‘1’.
0x4	R/W	VSYNC_POLARITY	0	[0] - This bit indicates the polarity of VSYNC (Currently not used and the default VSYNC polarity is active high)
0x8	R/W	HSYNC_POLARITY	0	[0] – This bit indicates the polarity of HSYNC (Currently not used and the default HSYNC polarity is active high)

Register Offset	Access	Name	Default Value	Register Description
0xC	R/W	ENABLE_POLARITY	0	[0] – This bit indicates the polarity of ENABLE i.e [0] bit of 0x00 (Currently not used and the default polarity of ENABLE is active high).
0x10	R/W	VSYNC_WIDTH	0	[13:0] – Holds the value of VSYNC width
0x14	R/W	VERT_BACK_PORCH	0	[13:0] – Holds the value of Vertical Back Porch
0x18	R/W	VERT_FRONT_PORCH	0	[13:0] – Holds the value of Vertical Front Porch
0x1C	R/W	VRES	0	[13:0] - Holds the value of Vertical Resolution VRES
0x20	R/W	HSYNC_WIDTH	0	[13:0] - Holds the value of HSYNC width
0x24	R/W	HORIZ_BACK_PORCH	0	[13:0] – Holds the value of Horizontal Back Porch
0x28	R/W	HORIZ_FRONT_PORCH	0	[13:0] – Holds the value of Horizontal Front Porch
0x2C	R/W	HRES	0	[13:0] - Holds the value of Horizontal Resolution HRES
0x34	R/W	FRAMELOCK_DELAY FRAMELOCK_ENABLE	0	[10:0] – Frame lock delay (Currently not used) [31] – Frame lock enable (Currently not used)
0x3C	R/W	FRAMELOCK_LINE_FRAC FRAMELOCK_ALIGN_SYNC	0	[10:0] – Frame lock line fraction (Currently not used) [16] – Frame lock align HSYNC (Currently not used)
0x40	R/W	Hdcolorbar_config	0	[2:0] – HD Color bar config (Currently not used)
0x44	R/W	TC_HSBLNK	0	[13:0] – HSBLANK, holds the start value of HBLANK. Number of active pixels = HSBLANK + 1 For example, if HSBLANK is 4, then the number of active pixels before start of BLANKING region are 5.
0x48	R/W	TC_HSSYNC	0	[13:0] - HSSYNC, holds the start value of HSYNC. For example, if HSSYNC is 8, then the number of total pixel clocks which occur before HSYNC is asserted are 9.

Register Offset	Access	Name	Default Value	Register Description
0x4C	R/W	TC_HESYNC	0	[13:0] - HESYNC, holds the end value of HSYNC. For example, if HESYNC is 11, then the number of total pixel clocks which occur before HSYNC is de-asserted are 10. Note that, the total number of pixel clocks for HSYNC = HESYNC - HSSYNC
0x50	R/W	TC_HEBLNK	0	[13:0] – HEBLANK, holds the end value of HBLANK. For example, if HEBLANK is 20, then the number of total pixel clocks before end of BLANKING region are 19. Note that, the total number of pixel clocks for blanking = HEBLANK - HSBLANK
0x54	R/W	TC_VSBLNK	0	[13:0] – VSBLANK, holds the start value of VBLANK. Number of active lines = VSBLANK + 1 For example, if VSBLANK is 4, then the number of active lines before start of VERTICAL BLANKING region are 5.
0x58	R/W	TC_VSSYNC	0	[13:0] - VSSYNC, holds the start value of VSYNC. For example, if VSSYNC is 8, then the number of total lines which occur before VSYNC is asserted are 9.
0x5C	R/W	TC_VESYNC	0	[13:0] - VESYNC, holds the end value of VSYNC. For example, if VESYNC is 11, then the number of total lines which occur before VSYNC is de-asserted are 10. Note that, the total number of lines for VSYNC = VESYNC - VSSYNC

Register Offset	Access	Name	Default Value	Register Description
0x60	R/W	TC_VEBLNK	0	[13:0] – VEBLANK, holds the end value of VBLANK. For example, if VEBLANK is 20, then the number of total lines before end of BLANKING region are 19. Note that, the total number of lines for blanking = HEBLANK - HSBLANK
0x300	R/W	MISC0	0	[2:1] – Denotes if video format is RGB or YCbCr422 00 – RGB 01 - YCbCr 422 [3] - If asserted, then dynamic range will be enabled in RGB colorimetry. [4] – If 0, then YCbCr_ITU_R_BT601 version of YCbCr will be enabled If 1, then YCbCr_ITU_R_BT709 version of YCbCr will be enabled [7:5] – If 000, then 6BPC If 001, then 8BPC If 010, then 10BPC If 011, then 12BPC If 100, then 16BPC
0x304	-	MISC1	0	Currently not used

Register Offset	Access	Name	Default Value	Register Description
0x308	R/W	Quad_pixel_mode dual_pixel_mode En_sw_pattern Test_pattern	0	[2:0] – Denotes video test pattern 000 – Indicates no test pattern is generated in the IP, instead input video stream is directly passed to output. 001 – Color ramp pattern 010 – Blank and White vertical lines pattern 011 – Color square pattern 100 - Flat Red pattern 101 – Flat Green pattern 110 – Flat Blue pattern 111 – Flat yellow pattern [4] – Currently not used [8] – When asserted, dual pixel mode is enabled [9] – When asserted, quad pixel mode is enabled
0x400	R/W	Aud_reset aud_start Aud_drop (Not used)	1	[0] – aud_reset, When asserted, the audio path gets reset. [1] - aud_start, When asserted, audio test pattern data gets started. When de-asserted, audio path of this IP acts as a passthrough to the incoming audio data. [2] – Currently not used.
0x404	R/W	Aud_sample_rate	0	[3:0] – Holds the value of audio sample rate [11:8] - Holds the value of number of audio channels selected
0x410	R/W	Aud_pattern1 Aud_period1	0	[1:0] – Holds the value of audio pattern 1 00 – Indicates silence 10 – Indicates ping [11:8] – Holds the value of audio period for audio pattern 1
0x420	R/W	Aud_pattern2 Aud_period2	0	[1:0] – Holds the value of audio pattern 2 [11:8] – Holds the value of audio period for audio pattern 2

Register Offset	Access	Name	Default Value	Register Description
0x430	R/W	Aud_pattern3 Aud_period3	0	[1:0] – Holds the value of audio pattern 3 [11:8] – Holds the value of audio period for audio pattern 3
0x440	R/W	Aud_pattern4 Aud_period4	0	[1:0] – Holds the value of audio pattern 4 [11:8] – Holds the value of audio period for audio pattern 4
0x450	R/W	Aud_pattern5 Aud_period5	0	[1:0] – Holds the value of audio pattern 5 [11:8] – Holds the value of audio period for audio pattern 5
0x460	R/W	Aud_pattern6 Aud_period6	0	[1:0] – Holds the value of audio pattern 6 [11:8] – Holds the value of audio period for audio pattern 6
0x470	R/W	Aud_pattern7 Aud_period7	0	[1:0] – Holds the value of audio pattern 7 [11:8] – Holds the value of audio period for audio pattern 7
0x480	R/W	Aud_pattern8 Aud_period8	0	[1:0] – Holds the value of audio pattern 8 [11:8] – Holds the value of audio period for audio pattern 8
0x4A0	R/W	Aud_channel_status	0	[31:0] – Holds the value of channel status that is to be sent along with audio sample
0x4A4	R/W	Aud_channel_status	0	[9:0] – Holds the value of channel status that is to be sent along with audio sample
0x4A8	-	-	-	Currently not used
0x4AC	-	-	-	Currently not used
0x4B0	-	-	-	Currently not used
0x4B4	-	-	-	Currently not used
0x4B8	R/W	Audio_chk_start	0	[0] – When asserted, internal audio checker gets started [7:4] – Holds the user defined stream ID and this is sent over output TID so that other external IPs know the stream ID of the audio data being sent.

Register Offset	Access	Name	Default Value	Register Description
0x4C0	R/W	Audio_stream_id	0	[31:0] - Holds the count value after which the number of audio sample counted by the internal checker will be reset.

- **Clocking:** av_axi_aclk - Clock for AXI lite programming interfaceaud_out_axi4s_aclk - Clock for input and output audio interfaceaud_clk - “aud_clk” and “aud_out_axi4s_aclk” both are connected to same sourcevid_out_axi4s_clk - Clock for input and output video interface
- **Resets:** av_axi_aresetn - Active low reset for AXI lite programming interfaceaud_out_axi4s_aresetn - Active low reset for input and output audio interfacesvid_out_axi4s_aresetn - Active low reset for input and output video interfaces

Programming sequence:

For Video

1. `av_pat_gen` work in two modes for video

   Mode – 0 : When bits [2:0] value is 3’b000 in register 0x308, this IP just acts as a pass through to the incoming video.

   Mode – 1: When bits [2:0] value is not 3’b000 in register 0x308, then this IP generates test patterns for video.
2. When mode-1 is selected, make bit [0] of 0x0 as “0” initially and configure appropriate values in the configuration registers (like HRES, VRES, TEST PATTERN TYPES etc.) corresponding to the video that is intended to be generated by the test pattern generator.
3. Then program “1” to bit [0] of 0x0 to start the generation of required video test pattern.

For Audio

1. `av_pat_gen` work in two modes for audio

   Mode – 0 : When bit [1] value is 1’b0 in register 0x400, this IP just acts as a pass through to the incoming audio.

   Mode – 1: When bit [1] value is 1’b1 in register 0x400, then this IP generates test patterns for audio.
2. To generate audio test patterns, make bit [1] of 0x400 as “0” initially and configure appropriate values in the configuration registers (like audio_test_pattern etc.) corresponding to the audio that is intended to be generated by the test pattern generator.
3. Then program “1” to bit [1] of 0x400 to start the generation of required audio test pattern.

4. User can enable bit [0] of 0x4B8, to start the internal audio checker of the IP which performs counting of number of audio samples being outputted by the IP. However currently there is no support to read this count through register interface. This checker can be used to count the audio samples on the input data when bit [1] of register 0x400 is programmed as 0.

Video Frame CRC (video_frame_crc)

Cyclic Redundancy Check (CRC) is generally used to detect errors in digital data and is commonly employed in video transmission to detect errors in pixel transmission. Using CRC, data integrity can be checked at various levels namely, pixel level, horizontal line level, frame level of a video.

CRC (video_frame_crc) is used in Display Port's example designs to calculate CRC at frame level, on the data received by Display Port RX subsystem and on the data being fed to Display Port TX subsystem (in passthrough system). Each color component's CRC value is calculated separately once per every frame and can be compared with the transmitted video frame's CRC value to check the data integrity.

Note that, CRC is not part of the Display Port core data path requirements but is necessary for validation/compliance requirements as per DP 1.4a specification. Xilinx's CRC IP complies to the requirements mentioned in DP 1.4a specification but do not provide separate product guide for CRC and do not provide support related to this CRC IP.

Feature Summary

- 1,2 and 4 PPCs are supported:

  Data is received by `video_frame_crc` on Vid_In_AXIS such that the first pixel will be aligned to the lower significant part of `vid_in_axis_tdata`. For example, in 4 PPC mode,

  `vid_in_axis_tdata` = {pixel3, pixel2, pixel1, pixel0}

- RGB, YCbCr 422 video formats are supported:
  - RGB:

    DisplayPort outputs YCbCr 422 video format such that Cr/Cb component is aligned to most significant part and "Y" is aligned to least significant part of the pixel. For example,

    If pixel0 (YCbCr) is being outputted by DisplayPort to `video_frame_crc`, then

    Pixel0 = {Cb/Cr, Y}
  - YCbCr 422:

    DisplayPort outputs YCbCr 422 video format such that Cr/Cb component is aligned to most significant part and "Y" is aligned to least significant part of the pixel. For example,

    If pixel0 (YCbCr) is being outputted by DisplayPort to `video_frame_crc`, then

Pixel0 = {Cb/Cr, Y}

- 6, 8, 10, 12 and 16 BPCs are supported:

  Bits corresponding to components of the pixel are tightly packed. For example,

  If pixel0 (RGB) is being outputted by DisplayPort with 6bpc, then

  Pixel0[17:12] = R-component

  Pixel0[11:6] = B-component

  Pixel0[5:0] = G-component

Port Descriptions

Signal name	Interface	Type	Description
S_AXI	s_axi_ctrl	Slave	AXI Slave bus interface to support register interface
Vid_In_AXIS	Video AXI4S Interface	Slave	AXI4 streaming video input interface to receive data on which CRC is to be calculated.
Vid_Out_AXIS	Video AXI4S Interface	Master	AXI4 streaming video output interface for passing the input data to next IPs in the video pipeline
s_axi_aclk	Clock	Input	Clock for s_axi_CTRL
s_axi_aresetn	Reset	Input	Active low reset for s_axi_CTRL
vid_in_axis_aclk	Clock	Input	Clock for input and output streaming video interfaces
vid_in_axis_aresetn	Reset	Input	Active low reset for input and output streaming video interfaces

Register Details

Register offset	Access	Name	Default Value	Register description
0x0	R/W	Pixel Mode	0x1	[2:0] - 3'b001 – 1PPC, 3'b010 – 2PPC, 3'b100 – 4PPC [4] - When set to '1', this clears all the CRC values [31] - When set to '1', this specifies that the incoming data is YCbCr 422 mode.

Register offset	Access	Name	Default Value	Register description
0x4	R	G_CRC R_CRC	0	[15:0] - CRC value of 'R' component in RGB mode and 'Cr' component in YCbCr 422 mode [31:16] - CRC value of 'G' component in RGB mode and 'Cb' component in YCbCr 422 mode
0x8	R	B_CRC	0	[15:0] - CRC value of 'B' component in RGB mode and 'Y' component in YCbCr 422 mode
0xC	R/W	HACTIVE VACTIVE	0	[15:0] - Denotes number of HACTIVE pixels [31:16] – Denotes number of VACTIVE Lines
0x10	R	Red_miss Green_miss Blue_miss	0	[3:0] – Denotes number of missing 'R' components in RGB mode and number of missing 'Cr' components in YCbCr 422 mode [7:4] – Denotes number of missing 'G' components in RGB mode and number of missing 'Cb' components in YCbCr 422 mode [11:8] – Denotes number of missing 'B' components in RGB mode and number of missing 'Y' components in YCbCr 422 mode

- **Clocking::** `s_axi_aclk` - Clock for `s_axi_ctrl` register interface`vid_in_axis_aclk` - Clock for input and output streaming video interface
- **Resets::** `s_axi_aresetn` - This reset when enabled, resets all the registers to default values`vid_in_axis_aresetn` - This reset when enabled, reset the entire data path related to CRC calculations.
- **Programming sequence::** Program HACTIVE, VACTIVE, PPC and color formats in to the registers before inputting corresponding video stream data.On every VSYNC of the frame, 0x4 and 0x8 registers are updated with the calculated CRC values. Hence the values in 0x4 and 0x8 registers denote CRC values calculated for the previous frame.

Functionality

CRC is calculated per component for the entire frame and the final CRC value per component is written in to the register for the software to read (in addresses 0x4 and 0x8), once VSYNC is detected. Hence, if the input is 1PPC RGB, then total of 3 CRC modules are needed for each component. Similarly, if it is 2 PPC, then total of 6 CRC modules are needed.

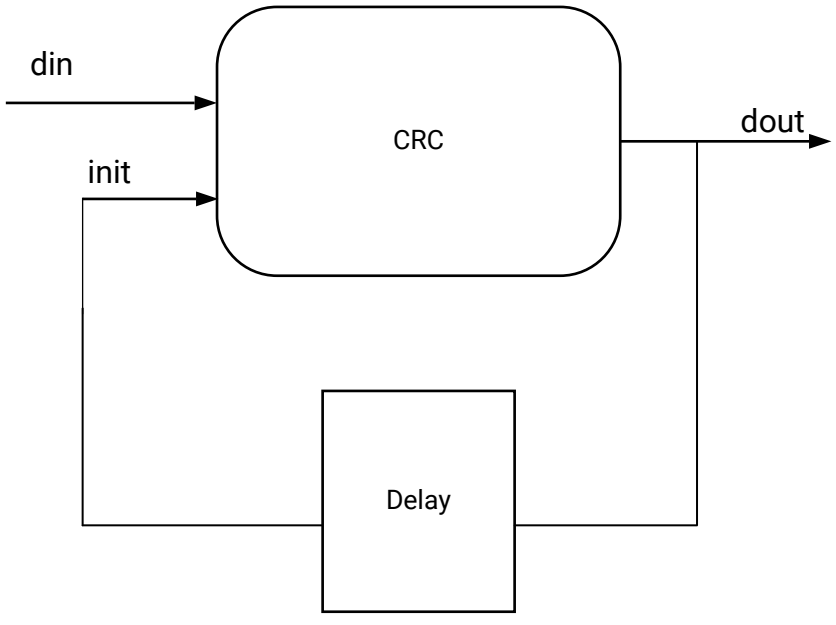
Polynomial used is,

```
f(x) = x16 + x15 + x2 + 1
```

Implementation is done as mentioned in the appendix of DP 1.4a specification except for the fact that there is an extra “init” input which takes the result of the CRC module from the previous clock cycle. In DP 1.4a specification, there is no separate input for previous clock’s CRC output. Instead it is denoted directly with the name of the output (which is ‘d’) of CRC module. The value of the per component’s CRC module’s output (‘dout’ in the example below) when VSYNC is detected is considered as the final calculated CRC value of specific component in that frame.

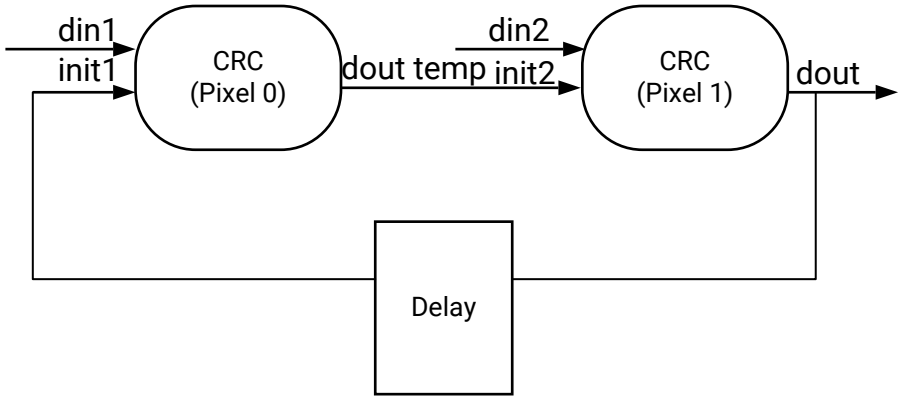
- **Special case for YCbCr 422 mode in 1PPC mode:** For RGB video format, calculation of CRC per component is straight forward, but for YCbCr 422 mode, VESA recommends calculating CRC for each of the component namely Y, Cb and Cr. Hence there will be a special case of handling the data when it is 1PPC mode. This is because on Vid_In_AXIS_tdata, Cb and Cr alternate on the same most significant part of the data on every clock cycle in 1PPC mode and care is taken such that the same part of the data is routed to CRC module handling ‘Cb’ on one clock and to the CRC module handling ‘Cr’ on another clock. In this mode, for each of the ‘Cb’ and ‘Cr’ components, same data (din and init) will be held for 2 clock cycles at their respective CRC module’s inputs since there will be only one “Cb” component and only one “Cr” component per 2 clock cycles in 1 PPC mode.
- **Examples:**
  - **Block diagram of per component CRC calculator in 1PPC mode:** Here, “din” is loaded on every clock cycle, with data corresponding to a component of each pixel.

Figure 28: **CRC calculator in 1PPC mode**



- **Block diagram of per component CRC calculator in 2PPC mode:** Here, “din1” and “din2” are loaded on every clock cycle, with data corresponding to a component of two consecutive pixels received in same clock (since 2 PPC mode).

Figure 29: **CRC calculator in 2PPC mode**



AXI4S Video Re-mapper (v_axi4s_remapper)

`v_axi4s_remapper` is used in DisplayPort example designs to remap video pixels between different “pixel per clock (PPC)” requirements. Multiple color formats are supported for remap namely, YUV444, YUV422 and YUV420. This IP is employed in DP TX data pipeline after frame buffer and in DP RX data pipeline before frame buffer in DP example designs, as the frame buffer always outputs data in 4PPC mode, but we may require other PPC modes for DP TX and RX.

Apart from “PPC” conversion, this IP also supports pixel drop and pixel repeat features which enables dropping and repetition of user specific pixels on the output and supports YUV420_HDMI color format too, which are out of scope of this document.

Note that this `v_axi4s_remapper` is not part of the Display Port protocol requirements and is used only in the example design to assist users in validation. Hence, Xilinx do not provide separate product guide for `v_axi4s_remapper` and do not provide support related to this IP.

Feature Summary

1. 1,2 and 4 PPC support

   `v_axi4s_remap` supports mapping of AXI4S stream video data between 1,2 and 4 PPC modes. For enabling the feature of PPC conversions, “Convert Samples per clock” needs to be selected in the GUI of the IP.

This IP has two register configurable values namely, “inPixClk” and “outPixClk” (as mentioned in Register Details section), which specifies the PPC on input and the PPC required on output respectively. These values are used by the IP for remapping between various PPCs.

In all the PPCs, first pixel is aligned to the lower significant part of video AXI4S stream’s TDATA. For example, in 4PPC mode

`vid_in_axis_tdata / vid_out_axis_tdata` = {pixel3, pixel2, pixel1, pixel0}

2. YUV444/RGB, YUV422, YUV420 video formats support

`v_axi4s_remap` does not manipulate the component order in each pixel as it only alters the number of pixels per clock. Hence, the DisplayPort specific input component ordering on `v_axi4s_remap` is preserved on the output of the IP.

Also, BPCs does not matter for re-mapper module as it only manipulates video data bits in terms of pixels. Hence, all the BPCs are supported by this IP.

Port Description

Signal name	Interface	Type	Description
s_axi_CTRL	s_axi_ctrl	Slave	AXI Slave bus interface to support register interface
s_axis_video	Video AXI4S Interface	Slave	AXI4 streaming video input interface.
m_axis_video	Video AXI4S Interface	Master	AXI4 streaming video output interface.
ap_clk	Clock	Input	Clock for streaming interface and s_axi_CTRL
ap_reset_n	Reset	Input	Active low reset for streaming interface and s_axi_CTRL

Register Details

Register Offset	Access	Name	Default Value	Register Description
0x10	R/W	Picture Height	0	[15:0] Holds the value of picture height
0x18	R/W	Picture Width	0	[15:0] Holds the value of picture width
0x20	R/W	Picture Color format	1	[7:0] Holds the value of color format 8’h1 – For YUV444 8’h3 – For YUV422 / YUV420
0x28	R/W	inPixClk	0	[7:0] – Holds the value of number of pixels per clock on the input video streaming interface

Register Offset	Access	Name	Default Value	Register Description
0x30	R/W	outPixClk	0	[7:0] – Holds the value of number of pixels per clock required on the output video streaming interface
0x38	-	-		This is out of scope as it is meant for HDMI
0x40	-	-		This is out of scope as it is meant for HDMI
0x48	R/W	inPixDrop	0	[0] – When asserted, pixel drop feature on the input video stream gets enabled. This feature is not used in DisplayPort.
0x50	R/W	outPixRepeat	0	[0] – When asserted, pixel repeat feature on the output video stream gets enabled. This feature is not used in DisplaypPort.

- **Clocking:** This IP has only one clock namely `ap_clk` on which all of the three interfaces (`s_axi_ctrl`, `s_axis_video` and `m_axis_video`) work.
- **Resets:** This IP has an active low reset namely, `ap_reset_n` which when enabled, puts all the registers in default state.
- **Programming sequence:**
  - The IP should be generated with “Convert Samples per clock” feature enabled in GUI so as to support re-mapping between various PPCs.
  - Always program the pic height, width, color format, inPixClk and outPixClk values before inputting the corresponding video streaming data.
- **Functionality:**
  - AXIvideo2MultiPixStream: In this first stage of data path, the input AXI video stream data is converted in to multi pixel data basing on the `inPixClk` value which tells how many pixels are being inputted per input clock.
  - pixClkUpConvert, pixClkDownConvert: The output from the first stage comes to this stage, where appropriate up conversion followed by down conversion takes place to output required PPC.
  - MultiPixStream2AXIvideo: In this final stage, the up and down converted multi pixel stream is again converted to AXI video stream output which is finally mapped to output `m_axis_video` interface.

XILINX

Appendix E

Debugging

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

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 an 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.*

Finding Help on Xilinx.com

To help in the design and debug process when using the subsystem, the 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. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

Documentation

This product guide is the main document associated with the 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 subsystem can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use 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 1.4 TX Subsystem

AR 70295

Technical Support

Xilinx provides technical support on the Xilinx Community Forums 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 ask questions, navigate to the Xilinx Community Forums.

Debug Tools

There are many tools available to address TX 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 to interact with the logic debug LogiCORE 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).

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.

TX General Checks

- Check the DisplayPort Source is DisplayPort 1.4 compliant.
- Ensure you are using proper DisplayPort 1.4 certified cable which is tested to run at 8.1 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.

1. Try with a working sink such as the DisplayPort Analyzer sink device.
2. Use a DisplayPort 1.4 certified cable. Change the cable and check again.
3. Put a DisplayPort AUX Analyzer in the Transmit path and check if the various training stages match with those mentioned in Main Link Setup and Management.
4. Probe the `lnk_clk` output and check if the SI of the clock is within the Phase Noise mask of the respective GT. The Noise mask requirement is listed in the respective GT documentation.

5. Check status registers in the Video PHY Controller for Reset done (`0x0020`) and PLL lock Status (`0x0018`).

Related Information

Main Link Setup and Management

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.

1. Perform a software reset on the register (`0x01C`) and check if the video is correct now.
2. Check if the MAIN_STREAM_ENABLE register is set to 1.
3. Ensure that the MSA parameters match the Video being sent by TX.
4. Check the video pixel clock generation. Ensure that the Video Clock is based on the resolution being sent.
5. Dump the DisplayPort source registers and compare against a working log.
6. Check the symbol and disparity errors in the Sink through the DPCD registers. This could be due to cable issue or PHY (GT) alignment issue.
7. Compare the design against the applicable example design.
8. Use the example design with a Xilinx development board to test the physical setup.

Transmit – Audio

This section contains debugging steps for issues with audio communication.

1. Check if MAUD and NAUD registers are correctly programmed and `aud_clk` is calculated as expected to be 512 × fs.
2. Follow steps mentioned in Programming DisplayPort Source.
3. Check if the TX_AUDIO_CHANNELS register value matches with the input audio samples sent.
4. Check if the TX_AUDIO_INFO_DATA is correctly formatted as per CEA 861-C info frame specification.
5. 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.

Related Information

Programming DisplayPort Source

Transmit – Misaligned Data

This section contains debugging steps for issues with data appearing to be misaligned or shifted on the monitor.

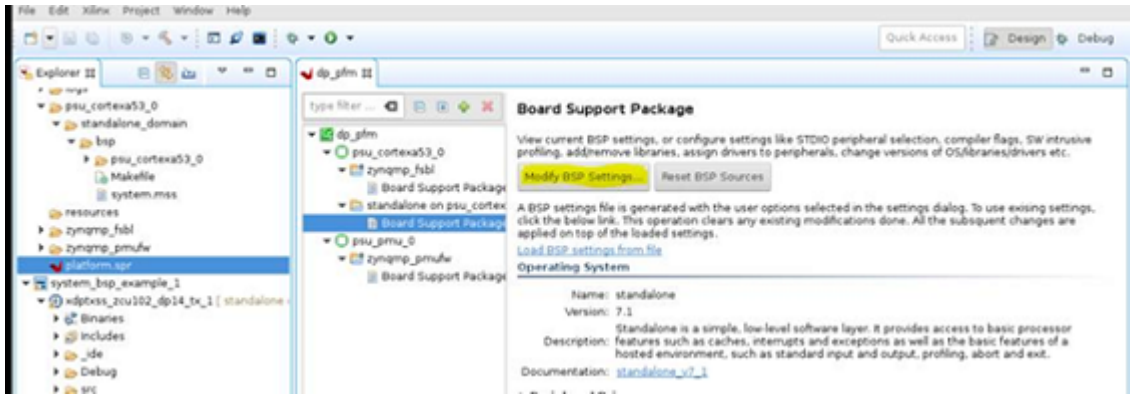
1. Check the EDID timings to verify they are within the CVT standard RB and RB2 reduced blanking resolutions.
2. Using EDID timings outside of the CVT standard can cause timing issues.

To fix this, define VTC_ADJUST_FOR_BS_TIMING in the `xdptxss_vtc.c`. This moves the BS symbol into the front porch to fix a swing in the BS timing caused by a non-standard CVT timing.

Software Debug

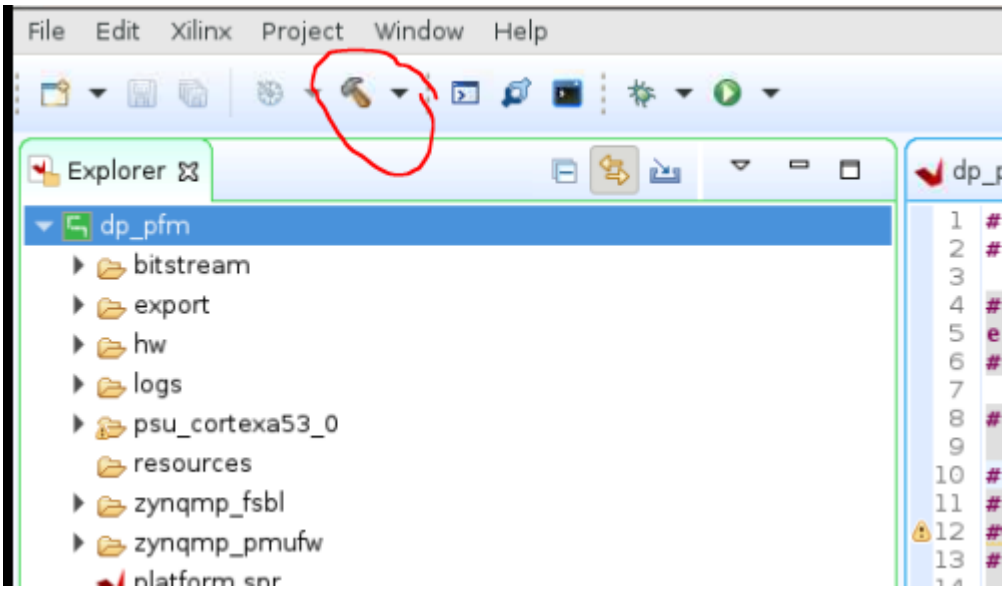
This section shows how to navigate to the DisplayPort debug driver information.

1. Open the platform project file, select Board Support Package under standalone and click on **Modify BSP settings**.



2. Add the option -DDEBUG to the extra compiler flags then close the BSP settings.

3. Select the platform and click **build**.



This will enable the debug symbol.

XILINX

Appendix F

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 (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open 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 DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

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

References

These documents provide supplemental material useful with this guide:

1. *Video PHY Controller LogiCORE IP Product Guide* (PG230)
2. *UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs* (UG949)
3. *VESA DisplayPort Standard* (VESA website)
4. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *AXI4-Stream Video IP and System Design Guide* (UG934)
6. *SmartConnect LogiCORE IP Product Guide* (PG247)
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *Vivado Design Suite User Guide: Getting Started* (UG910)
9. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
10. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
11. *Vivado Design Suite User Guide: Implementation* (UG904)
12. *Vivado Design Suite: AXI Reference Guide* (UG1037)
13. *AXI4-Stream to Video Out LogiCORE IP Product Guide* (PG044)
14. *Video Timing Controller LogiCORE IP Product Guide* (PG016)
15. *HDCP 1.x Product Guide* (PG224)
16. *AXI Timer LogiCORE IP Product Guide* (PG079)
17. *ZCU102 System Controller GUI Tutorial* (registration required) (XTP433)
18. *VCU118 System Controller Tutorial* (registration required) (XTP447)
19. *UltraScale Architecture and Product Data Sheet: Overview* (DS890)
20. *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892)
21. *Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS893)
22. *Defense-Grade UltraScale Architecture Data Sheet: Overview* (DS895)
23. *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)
24. *Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS923)
25. *Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925)
26. *Zynq UltraScale+ RFSoC Data Sheet: DC and AC Switching Characteristics* (DS926)
27. *HDCP 2.2 LogiCORE IP Product Guide* (PG249)
28. *Versal Prime Series Data Sheet: DC and AC Switching Characteristics* (DS956)
29. *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics* (DS957)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
01/13/2021 v3.0	
Core Overview	Added line rate support for Versal devices.
AUX Channel Interface	Updated 0x140 and 0x144 offset description.
DisplayPort Audio Registers	Updated 0x308 offset description and added 0x6A0 offset to table.
HDCP 1.3 Registers	Added new section.
HDCP 2.2/2.3 Registers	Added new section.
Chapter 4: Designing with the Subsystem	• Added YUV420 support • Added DSC and FEC support • Added Adaptive Sync support • Added Versal device support • Added Physical Layout section • Added Programming Sequence section
AXI4-Stream Interface Color Mapping	Added additional footnotes to table.
Chapter 6: Example Design	Added Versal device example design based on VCK190 platform.
Available Example Designs	Added additional footnote to table.
08/31/2020 v2.1	
Core Overview	Updated with device specific line rate information.
Chapter 4: Designing with the Subsystem	Added eDP support.
Chapter 6: Example Design	Added Configuring HDCP Keys and Key Management sections.
12/02/2019 v2.1	
General updates	• HDCP 2.2 support • FB Pass-through with HDCP 1.3 and HDCP 2.2 • Added Appendix D for helper cores • Vitis flow updated in chapter 6
05/22/2019 v2.1	
Chapter 6: Example Design	MST FB Pass-through example design details added
12/05/2018 v2.0	
MST Interface	Added MST interface ports
HDCP Key Interface	HDCP Ports added
Programming the Core in MST Mode	MST Programming added
Pixel Mapping Examples on AXI4-Stream Interface (UG934-Compliant)	UG934-compliant pixel mapping
04/04/2018 v1.0	
Initial release.	N/A

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright