DisplayPort RX Subsystem v2.1

Product Guide

Vivado Design Suite

PG233 June 3, 2020

XILINX

Table of Contents

Introduction

The DisplayPort RX Subsystem is a plug-in solution for serial digital video data reception in large video systems with video resolutions up to Ultra HD (UHD) at 60 fps as defined by the Video Electronics Standards Association (VESA) DisplayPort standard v1.2a. The subsystem provides easy to use mode selection and automated customization.

Features

- Support for DisplayPort Sink (RX) capabilities
- Supports single stream transport (SST) and multi-stream transport (MST)
- Dynamic lane supports (1, 2, or 4 lanes)
- Dynamic support for 1.62/2.7/5.4 Gb/s line rates
- Dynamic support of 6, 8, 10, 12, or 16 bits per component (BPC).
- Dynamic support of RGB and YCbCr444/ YCbCr422/Y-Only color formats.
- Supports Audio
- Supports HDCP 1.3
- AXI IIC controller for DP159 retimer programming
- Supports native or AXI4-Stream video input interface
- Supports both 16-bit and 32-bit video PHY (GT) interface

LogiCORE IP Facts Table	
Subsystem Specifics	
Supported Device Family(1)(2)	UltraScale+™ Families (GTHE4) UltraScale™ Families (GTHE3) Zynq®-7000 SoC (GTXE2) Virtex®-7 (GTXE2) and Kintex®-7 (GTXE2)
Supported User Interfaces	AXI4-Stream, AXI4-Lite, Native video
Resources	Performance and Resource Utilization web page
Provided with Subsystem	
Design Files	Hierarchical subsystem packaged with DisplayPort RX core and other IP cores
Example Design	Vivado IP Integrator
Test Bench	N/A
Constraints File	IP cores delivered with XDC files
Simulation Model	N/A
Supported S/W Driver	Standalone
Tested Design Flows(3)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 65447
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. For HDCP: UltraScale/UltraScale+ supports up to 5.4 Gb/s, Kintex-7/Virtex-7 (-1 speed grade supports up to 2.7 Gb/s, -2/ -3 supports up to 5.4 Gb/s), and Artix-7 is not supported.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The DisplayPort RX subsystem is a full feature, hierarchically packaged subsystem with a DisplayPort sink (RX) core ready to use in applications in large video systems. The DisplayPort RX subsystem requires use of a DP159 Retimer.

Feature Summary

- UHD up to 60 fps supports multi-stream transport (MST) and single stream transport (SST) modes
- Dynamic lane supports (1, 2, or 4 lanes)
- Dynamic support of different BPC and color formats
- Pixel mode support in native video interface mode
- Support for 2 to 8 channel audio with 44/48 kHz sample rates
- Support Linear PCM 2-channel audio format
- Support optional HDCP 1.3 Controller
- Support for 16-bit or 32-bit GT width
- Support for native and AXI4-Stream video input interface

Unsupported Features

- Audio in MST mode
- In-band stereo
- HDCP 1.3 in MST mode
- HDCP 2.x
- Video AXI4-Stream interface is not scalable with dynamic pixel mode selection
- Dual-pixel splitter in native video mode
- MCCS over DDC/CI

- No Interlaced video support in AXI4-Streaming mode
- eDP and iDP
- GTC
- Non-LPCM audio

Licensing and Ordering

License Type

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

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

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

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

License Checkers

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

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

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

Product Specification

The DisplayPort RX Subsystem operates in the following video modes:

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

The DisplayPort RX Subsystem works in conjunction with the Video PHY Controller, configured for DisplayPort protocol. The subsystem outputs multi-pixel video over an AXI4-Stream interface. For more information on the Video PHY Controller, see the *Video PHY Controller Product Guide* (PG230) [Ref 1].

AXI4-Stream Video Interface

In SST mode and when the AXI4-Stream video interface is selected the subsystem is packaged with the following cores:

- DisplayPort Receive core
- DisplayPort Video to AXI4-Stream Bridge
- AXI IIC controller for connecting to the TI DP159

The HDCP core and an AXI Timer core also form part of the DisplayPort RX Subsystem when the HDCP feature is enabled.

In MST mode, in addition to the subcores listed for SST, the Video to AXI4-Stream Bridge instances increase to the number of video streams.

Because the DisplayPort RX Subsystem is hierarchically packaged, you select the parameters and the subsystem creates the required hardware. Figure 2-1 shows the architecture of the subsystem assuming MST with four streams. The DisplayPort RX Subsystem receives the video using the DisplayPort v1.2a protocol over a 32-bit or 16-bit video PHY interface.

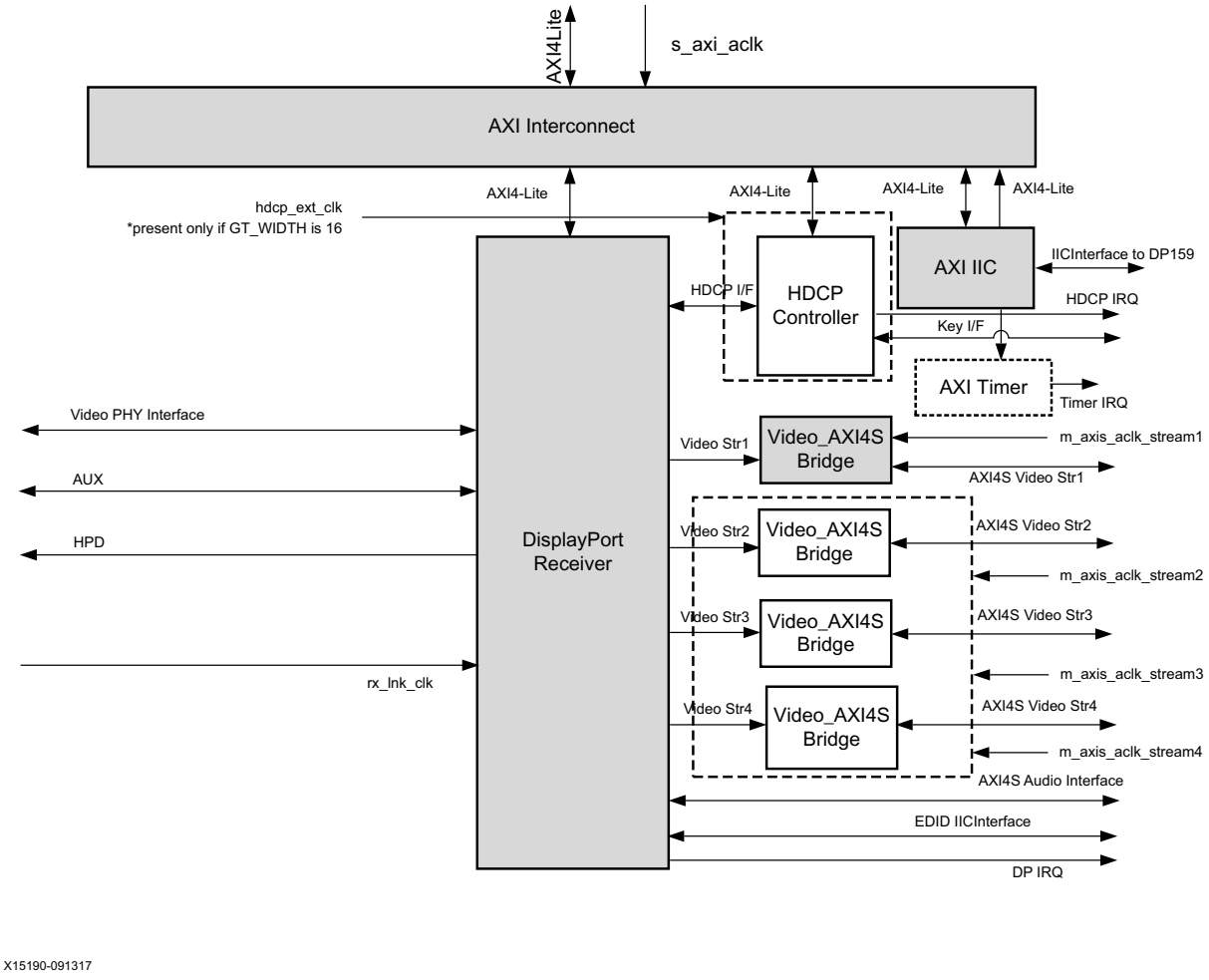


Figure 2-1: **DisplayPort RX Subsystem Block Diagram**

Note: MST HDCP is not supported.

Pixel Mapping on AXI4-Stream Interface

By default, the pixel mode is equal to the lane count during subsystem generation. You can override pixel width dynamically. For example, if the driver selects a 2 pixel mode as default, you can change the pixel mode to 1.

- For pixel mode of 1, valid pixels are available only in pixel 0 position.
- For pixel mode of 2, valid pixels are available only in pixel 0 and pixel 1 position.
- For pixel mode of 4, valid pixels are available only 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.

```
Pixel_Width = MAX_BPC x 3
Interface Width = Pixel Width x LANE_COUNT
```

For example, if the system is generated using 4 lanes with **MAX_BPC** of 16, the data width of the AXI4-Stream interface is 16 × 4 × 3 = 192. For more detailed information on the mapping of the color components on the AXI4-Stream interface, see the Data Format section in the *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 18].

Table 2-1 shows pixel mapping on AXI4-Stream for all the video sampling modes (4:4:4, 4:2:2, Y-only). Implementation of 4:2:2 and Y-only sampling modes pixel mapping is different from the guidelines defined in the *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 18].

Table 2-1: **Pixel Mapping Examples on AXI4-Stream Interface**

MAX_BPC	LANES	Pixel Width	Interface Width	Video BPC	Pixel 3			Pixel 2			Pixel 1			Pixel 0		
16	4	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
	2		96		–	–	–	–	–	–	95:80	79:64	63:48	47:32	31:16	15:0
	1		48		–	–	–	–	–	–	–	–	–	47:32	31:16	15:0
12	4	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
	2		72		–	–	–	–	–	–	71:60	59:48	47:36	35:24	23:12	11:0
	1		36		–	–	–	–	–	–	–	–	–	35:24	23:12	11:0
10	4	30	120	10	119:110	109:100	99:90	89:80	79:70	69:60	59:50	49:40	39:30	29:20	19:10	9:0
	2		60		–	–	–	–	–	–	59:50	49:40	39:30	29:20	19:10	9:0
	1		30		–	–	–	–	–	–	–	–	–	29:20	19:10	9:0
8	4	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
	2		48		–	–	–	–	–	–	47:40	39:32	31:24	23:16	15:8	7:0
	1		24		–	–	–	–	–	–	–	–	–	23:16	15:8	7:0
16	4	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
	2		96		–	–	–	–	–	–	95:84	79:68	63:52	47:36	31:20	15:4
	1		48		–	–	–	–	–	–	–	–	–	47:36	31:20	15:4
12	4	36	144	10	143:134	131:122	119:110	107:98	95:86	83:74	71:62	59:50	47:38	35:26	23:14	11:2
	2		72		–	–	–	–	–	–	71:62	59:50	47:38	35:26	23:14	11:2
	1		36		–	–	–	–	–	–	–	–	–	35:26	23:14	11:2
10	4	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
	2		60		–	–	–	–	–	–	59:52	49:42	39:32	29:22	19:12	9:2
	1		30		–	–	–	–	–	–	–	–	–	29:22	19:12	9:2
8	4	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
	2		48		–	–	–	–	–	–	47:42	39:34	31:26	23:18	15:10	7:2
	1		24		–	–	–	–	–	–	–	–	–	23:18	15:10	7:2
16	4	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
	2		96		–	–	–	–	–	–	95:86	79:70	63:54	47:38	31:22	15:6
	1		48		–	–	–	–	–	–	–	–	–	47:38	31:22	15:6

Table 2-1: **Pixel Mapping Examples on AXI4-Stream Interface *(Cont'd)***

MAX_BPC	LANES	Pixel Width	Interface Width	Video BPC	Pixel 3			Pixel 2			Pixel 1			Pixel 0		
12	4	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
	2		72		–	–	–	–	–	–	71:64	59:52	47:40	35:28	23:16	11:4
	1		36		–	–	–	–	–	–	–	–	–	35:28	23:16	11:4
10	4	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
	2		60		–	–	–	–	–	–	59:54	49:44	39:34	29:24	19:14	9:4
	1		30		–	–	–	–	–	–	–	–	–	29:24	19:14	9:4
16	4	48	192	8	191:184	175:168	159:152	143:136	127:120	111:104	95:88	79:72	63:56	47:40	31:24	15:8
	2		96		–	–	–	–	–	–	95:88	79:72	63:56	47:40	31:24	15:8
	1		48		–	–	–	–	–	–	–	–	–	47:40	31:24	15:8
12	4	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
	2		72		–	–	–	–	–	–	71:66	59:54	47:42	35:30	23:18	11:6
	1		36		–	–	–	–	–	–	–	–	–	35:30	23:18	11:6
16	4	48	192		191:186	175:170	159:154	143:138	127:122	111:106	95:90	79:74	63:58	47:42	31:26	15:10
	2		96		–	–	–	–	–	–	95:90	79:74	63:58	47:42	31:26	15:10
	1		48		–	–	–	–	–	–	–	–	–	47:42	31:26	15:10

Notes:

1. The padding bits are zeros.

Table 2-2 shows the color component mapping for different video sampling modes for the AXI4-Stream pixel mapping shown in Table 2-1. The 422 and Y-only sampling modes are shown to highlight the difference between video sampling mode mappings. In Table 2-2 MAX_BPC = 16, LANES = 4, and Video BPC = 16.

Table 2-2: **Color Component Mapping for Different Video Sampling Rates**

Pixel Width	Interface Width	Video Sampling Mode	Pixel 3			Pixel 2			Pixel 1			Pixel 0		
48	192	444	191:176	175:160	159:144	143:128	127:112	111:96	95:80	79:64	63:48	47:32	31:16	15:0]
32	128	422	V2 [191:176]	Y3 [175:160]	–	U2 [143:128]	Y2 [127:112]	–	V0 [95:80]	Y1 [79:64]	–	U0 [47:32]	Y0 [31:16]	–
16	64	Y-only	Y2 [191:176]	–	–	Y2 [143:128]	–	–	Y1 [95:80]	–	–	Y0 [47:32]	–	–

Table 2-3 shows the pixel mapping examples on the AXI4-Stream interface that are compliant with the guidelines defined in the *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 18].

Table 2-3: **Pixel Mapping Examples on AXI4-Stream Interface (UG934-Compliant)**

MAX_BPC	LANES	Pixel Width	Interface Width	Video BPC	Video Sampling Mode	Pixel 3			Pixel 2			Pixel 1			Pixel 0		
8	4	24	96	8	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
		8	32		Y-Only			31:24			23:16			15:8			7:0
	2	24	48		444	–	–	–	–	–	–	47:40	39:32	31:24	23:16	15:8	7:0
		16	32		422	–	–		–		–	31:24		23:16	15:8		7:0
		8	16		Y-Only			–			–			15:8			7:0
	1	24	24		444	–	–	–	–	–	–	–	–	–	23:16	15:8	7:0
		16	16		422	–		–	–		–			–	15:8		7:0
		8	8		Y-Only			–			–			–			7:0
10	4	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
		10	40		Y-Only			39:32			29:22			19:12			9:2
	2	30	60		444	–	–	–	–	–	–	59:52	49:42	39:32	29:22	19:12	9:2
		20	40		422	–		–	–		–	39:32		29:22	19:12		9:2
		10	20		Y-Only			–						19:12			9:2
	1	30	30		444	–	–	–	–	–	–	–	–	–	29:22	19:12	9:2
		20	20		422	–		–	–		–			–	19:12		9:2
		10	10		Y-Only												9:2
12	4	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
		12	48		Y-Only			47:40			35:28			23:16			11:4
	2	36	72		444	–	–	–	–	–	–	71:64	59:52	47:40	35:28	23:16	11:4
		24	48		422	–		–	–		–	47:40		35:28	23:16		11:4
		12	24		Y-Only			–						23:16			11:4
	1	36	36		444	–	–	–	–	–	–	–	–	–	35:28	23:16	11:4
		24	24		422	–		–	–		–			–	23:16		11:4
		12	12		Y-Only			–			–			–			11:4
16	4	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
		16	64		Y-Only			63:56			47:40			31:24			15:8
	2	48	96		444	–	–	–	–	–	–	95:88	79:72	63:56	47:40	31:24	15:8
		32	64		422	–		–	–		–	63:56		47:40	31:24		15:8
		16	32		Y-Only			–						31:24			15:8
	1	48	48		444	–	–	–	–	–	–	–	–	–	47:40	31:24	15:8
		32	32		422	–		–	–		–			–	31:24		15:8
		16	16		Y-Only			–			–			–			15:8

Native Video Interface

In SST or MST mode, by default, the subsystem is packaged with two subcores:

- DisplayPort Receive core
- AXI IIC controller

An HDCP core and an AXI Timer core also form part of the DisplayPort RX Subsystem when the HDCP feature is enabled.

Because the DisplayPort RX Subsystem is hierarchically packaged, you select the parameters and the subsystem creates the required hardware. Figure 2-2 shows the architecture of the subsystem assuming MST with four streams. The DisplayPort RX Subsystem receives the video using the DisplayPort v1.2a protocol over 32-bit or 16-bit video PHY interface.

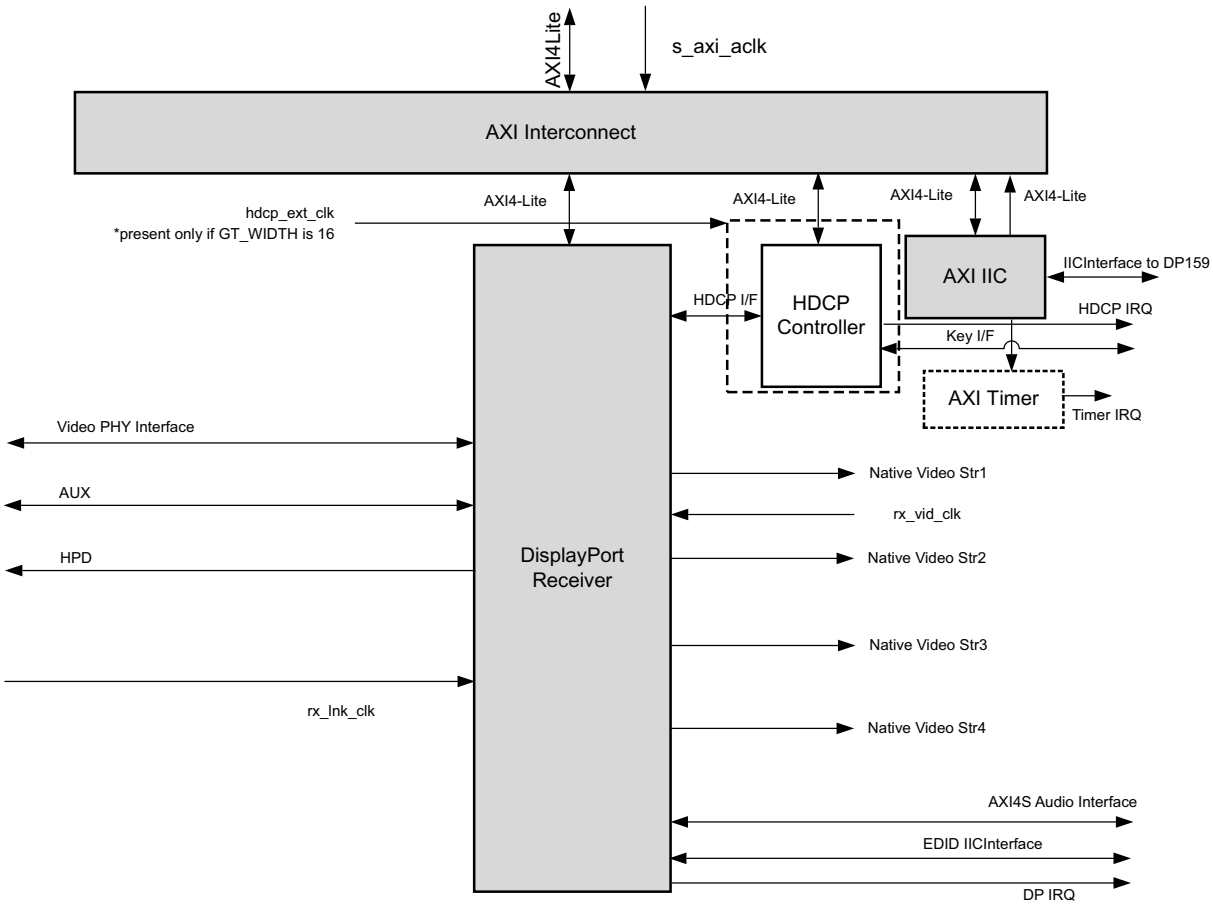


Figure 2-2: **DisplayPort RX Subsystem Block Diagram with Native Video Interface**

Pixel Mapping on Native 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 the 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 user has the option to use the resolved M and N values from the stream to generate a clock, or to use a sufficiently-fast clock and pipe the data into a line buffer. Xilinx recommends using a fast clock and ignoring the M and N values unless you can be certain of the source of these values. Unlike the Source core, when using a fast clock, the data valid signal might toggle within a scan line. Figure 2-3 shows the typical signaling of a full frame of data.

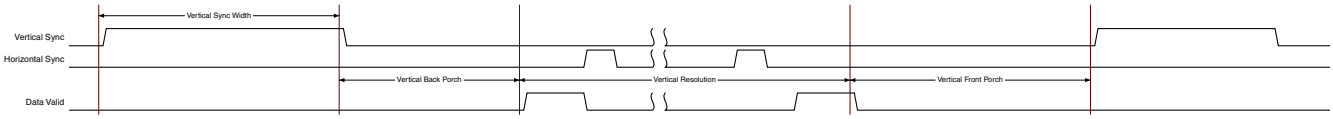


Figure 2-3: **User Interface Vertical Timing**

The horizontal timing information is defined by a front porch, back porch, and pulse width. The porch values are defined as the number of clocks between the horizontal sync pulse and the start or end of active data. Pixel data is only accepted into the image data interface when the data valid flag is active-High. Figure 2-4 is an enlarged version of Figure 2-3, giving more detail on a single scan line.

The horizontal sync pulse should be used as a line advance signal. Use the rising edge of this signal to increment the line count. Note that Data Valid might toggle if using a fast clock.

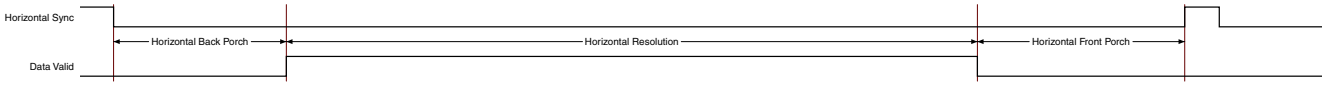


Figure 2-4: **User Interface Horizontal Timing**

In the two-dimensional image plane, these control signals frame a rectangular region of active pixel data within the total frame size. This relationship of the total frame size to the active frame size is shown in Figure 2-5.

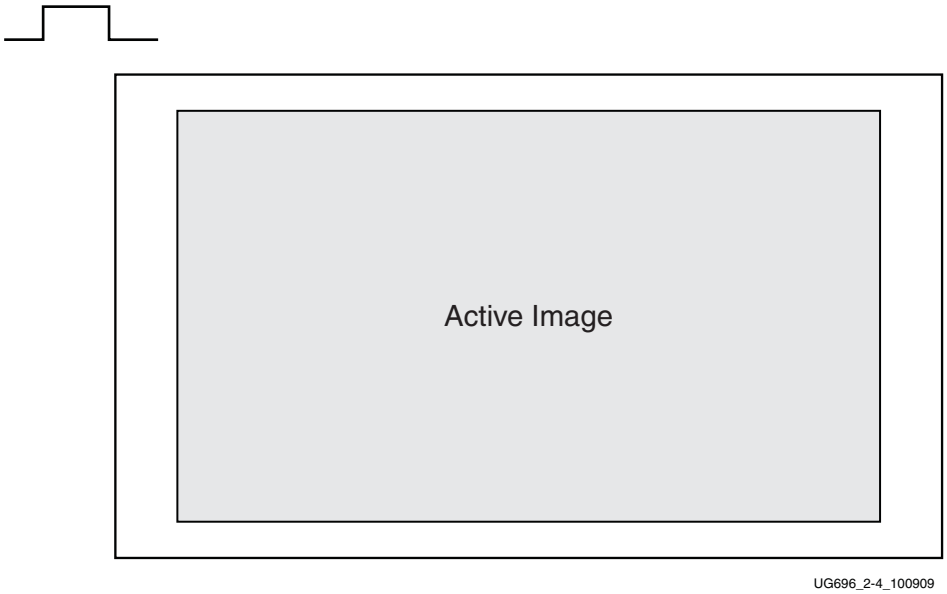


Figure 2-5: **Active Image Data**

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

The **vid_pixel** width is always 48-bits, regardless of if all bits are used. For pixel mappings that do not require all 48-bits, the convention used for this core is to occupy the MSB bits first and leave the lower bits either untied or driven to zero. Table 2-4 provides the mapping for all supported data formats.

Table 2-4: **Pixel Mapping for the User Data Interface**

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

Notes:

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

The design allows use of a faster pixel clock, for example, 150 MHz or higher video clock frequency for all standard video resolutions. The DisplayPort RX supports DMA mode without any internal line buffers for video display. You need to reproduce the exact video timing from the M_vid and N_vid values reported in the Main Stream Attribute (MSA) data. The interface timing in this case is shown in Figure 2-6.

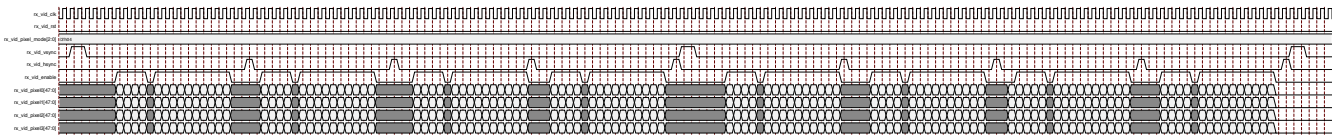


Figure 2-6: **RX Pixel Timing**

Note: The width of **rx_vid_vsync**, **rx_vid_hsync**, **rx_vid_enable** and the number of hsync pulses shown in Figure 2-6 are scaled down to have better visibility. The number of hsync pulses are equal to the number of active lines in a frame. The default widths of **rx_vid_hsync** pulse is 16 and **rx_vid_vsync** pulse is 63. The widths hsync and vsync can be controlled through software as per the MSA.

DisplayPort Receive (RX)

The DisplayPort Receive (RX) core contains the following four major blocks as shown in Figure 2-7:

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

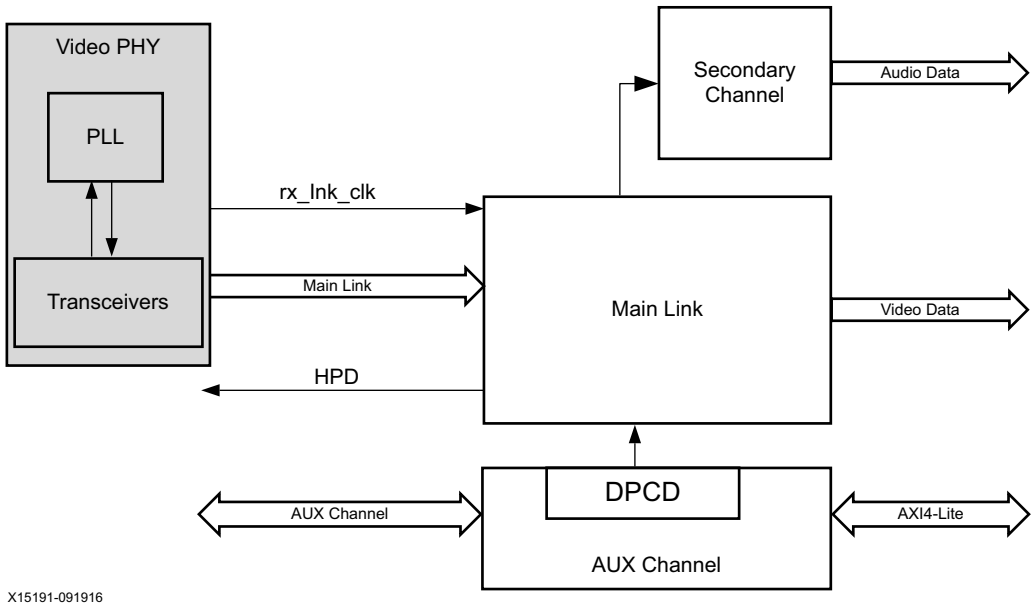


Figure 2-7: **DisplayPort Receive Core Block Diagram**

Video to AXI4-Stream Bridge

Video to AXI4 stream Bridge is used in DisplayPort RX Subsystem to convert the video output of DisplayPort receive IP to the AXI4 stream standard.

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

Table 2-5 shows the color mapping for the AXI4-Stream interface.

Table 2-5: **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	Cr/Cb	Y	–	Cr/Cb	Y	–	Cr/Cb	Y	–	Cr/Cb	Y	–
Y-Only	Y	–	–	Y	–	–	Y	–	–	Y	–	–

Notes:

1. For component widths, see Table 2-1.

AXI Interconnect

The subsystem uses the Xilinx AXI Interconnect IP core as a crossbar, which contains one AXI4-Lite slave interface and two AXI4-Lite master interfaces when HDCP is not enabled in the system. With HDCP enabled, the subsystem has four AXI4-Lite master interfaces.

Figure 2-8 shows the AXI slave structure within the DisplayPort RX Subsystem. For more details on the AXI crossbar functionality, see the *AXI Interconnect Product Guide* (PG059) [Ref 3].

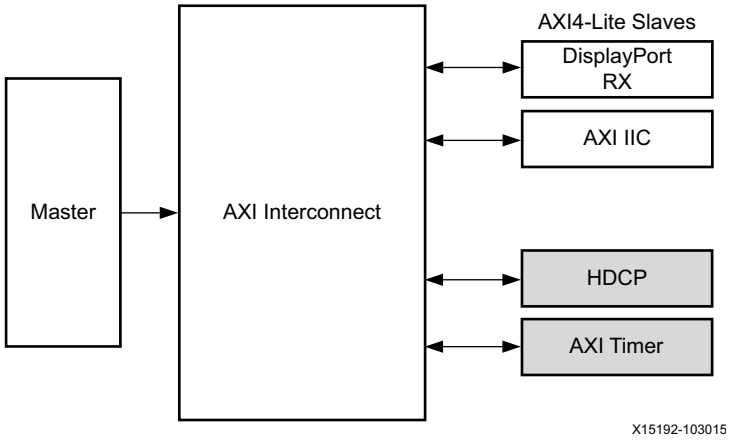


Figure 2-8: **AXI Interconnect in the DisplayPort RX Subsystem**

AXI IIC

The AXI IIC controller is used in the DisplayPort RX Subsystem to configure the DP159 retimer through the IIC interface. The AXI IIC controller in the DisplayPort RX Subsystem operates at 400 kHz. The DisplayPort RX Subsystem driver handles the DP159 programming.

For more details on IIC programming for DP159, see the DP159 Retimer in Chapter 3.

HDCP Controller

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

Figure 2-9 shows the DisplayPort RX Subsystem with HDCP controller. For more details on HDCP, see the *HDCP Product Guide* (PG224) [Ref 4].

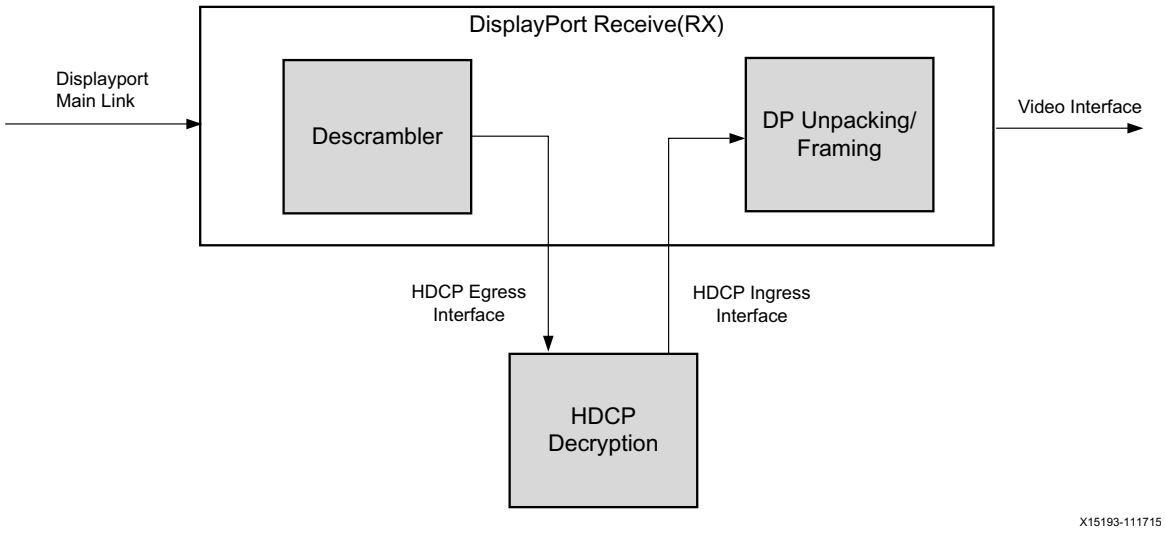


Figure 2-9: **DisplayPort RX with HDCP Controller**

Note: MST HDCP is not supported.

AXI Timer

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

Standards

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

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

Resource Utilization

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

Port Descriptions

The DisplayPort RX Subsystem ports are described in the following tables.

AXI4-Lite Interface Ports

Table 2-6: **AXI4-Lite Interface Ports**

Signal Name	I/O	Description
s_axi_aclk	I	AXI bus clock
s_axi_aresetn	I	AXI reset. Active-Low.
s_axi_awaddr[13:0]	I	Write address
s_axi_awprot[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_araddr[13:0]	I	Read address
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

Video PHY Sideband Status Ports

Table 2-7: **Video PHY Sideband Status Ports**

Signal Name	I/O	Description
s_axis_phy_rx_sb_status_tdata[15:0]	I	Video PHY status input
s_axis_phy_rx_sb_status_tready	O	Ready to video PHY for status
s_axis_phy_rx_sb_status_tvalid	I	Video PHY status valid

Video PHY Sideband Control Ports

Table 2-8: **Video PHY Sideband Control Ports**

Signal Name	I/O	Description
m_axis_phy_rx_sb_control_tdata[7:0]	O	Control output to video PHY
m_axis_phy_rx_sb_control_tvalid	O	Control output valid to video PHY
m_axis_phy_rx_sb_control_tready	I	Control data ready input

Link Clock Interface Port

Table 2-9: **Link Clock Interface Port**

Signal Name	I/O	Description
rx_lnk_clk	I	Link clock

Video PHY Main Link (Lane0 to Lane3)

Table 2-10: **Video PHY Main Link Ports**

Signal Name	I/O	Description
s_axis_lnk_rx_lane0_tdata[31:0]	I	Main link data for lane0
s_axis_lnk_rx_lane0_tvalid	I	Main link data valid for lane0
s_axis_lnk_rx_lane0_tready	O	Main link data ready for lane0
s_axis_lnk_rx_lane0_tuser[11:0]	I	Main link user data for lane0
s_axis_lnk_rx_lane1_tdata[31:0]	I	Main link data for lane1
s_axis_lnk_rx_lane1_tvalid	I	Main link data valid for lane1
s_axis_lnk_rx_lane1_tready	O	Main link data ready for lane1
s_axis_lnk_rx_lane1_tuser[11:0]	I	Main link user data for lane1
s_axis_lnk_rx_lane2_tdata[31:0]	I	Main link data for lane2
s_axis_lnk_rx_lane2_tvalid	I	Main link data valid for lane2
s_axis_lnk_rx_lane2_tready	O	Main link data ready for lane2
s_axis_lnk_rx_lane2_tuser[11:0]	I	Main link user data for lane2
s_axis_lnk_rx_lane3_tdata[31:0]	I	Main link data for lane3

Table 2-10: **Video PHY Main Link Ports *(Cont'd)***

Signal Name	I/O	Description
s_axis_lnk_rx_lane3_tvalid	I	Main link data valid for lane3
s_axis_lnk_rx_lane3_tready	O	Main link data ready for lane3
s_axis_lnk_rx_lane3_tuser[11:0]	I	Main link user data for lane3

Receive Video Interface Ports

Table 2-11: **Receive Video Interface Ports**

Signal Name	I/O	Description
rx_vid_clk	I	DisplayPort RX video clock
rx_vid_rst	I	DisplayPort RX Video reset

SST AXI4-Stream Interface Ports

Enabled when the AXI4-Stream interface is selected (n = 1).

Table 2-12: **SST AXI4-Stream Interface**

Signal 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. For more information on the interface width, see Table 2-1.
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.

MST AXI4-Stream Interface Ports

Enabled when the AXI4-Stream interface is used.

Table 2-13: **MST AXI4-Stream Interface Ports**

Signal Name	I/O	Description
m_axis_aclk_stream<n>	I	
m_axis_ video_stream<n>_tdata[191:0]	O	
m_axis_video_stream<n>_tlast	O	
m_axis_video_stream<n>_tready	I	
m_axis_video_stream<n>_tuser	O	

Table 2-13: **MST AXI4-Stream Interface Ports *(Cont'd)***

Signal Name	I/O	Description
m_axis_video_stream<n>_tvalid	O	

Notes:
1. <n> = 2 to 4.

Native Video Stream Interface Ports

Enabled when the AXI4-Stream interface is selected. <n> = 1.

Table 2-14: **Native Video Stream Interface Ports**

Signal Name	I/O	Description
rx_vid_stream1_tx_vid_enable	O	User data video enable
rx_vid_stream1_tx_vid_hsync	O	Horizontal sync pulse. Active on the rising edge
rx_vid_stream1_tx_vid_oddeven	O	Indicates an odd (1) or even (0) field polarity
rx_vid_stream1_tx_vid_pixel0[47:0]	O	Video data
rx_vid_stream1_tx_vid_pixel1[47:0]	O	Video data
rx_vid_stream1_tx_vid_pixel2[47:0]	O	Video data
rx_vid_stream1_tx_vid_pixel3[47:0]	O	Video data
rx_vid_stream1_tx_vid_vsync	O	Vertical sync pulse. Active on the rising edge.
rx_bpc[2:0]	O	Bits per component and derived from MISC0 and MISC1 fields
rx_vid_pixel_mode[2:0]		User pixel width/mode
rx_cformat[2:0]	O	Colorimetry indicator field and derived from MISC0 and MISC1 fields

Native Video Stream Interface MST Stream

Table 2-15: **Native Video Stream Interface MST Stream**

Signal Name	I/O	Description
rx_vid_stream<n>_tx_vid_enable	O	User data video enable
rx_vid_stream<n>_tx_vid_hsync	O	Horizontal sync pulse. Active on the rising edge
rx_vid_stream<n>_tx_vid_oddeven	O	Odd/even field select. Indicates an odd (1) or even (0) field polarity.
rx_vid_stream<n>_tx_vid_pixel0[47:0]	O	Video data
rx_vid_stream<n>_tx_vid_pixel1[47:0]	O	Video data
rx_vid_stream<n>_tx_vid_pixel2[47:0]	O	Video data
rx_vid_stream<n>_tx_vid_pixel3[47:0]	O	Video data

Table 2-15: **Native Video Stream Interface MST Stream *(Cont'd)***

Signal Name	I/O	Description
rx_vid_stream<n>_tx_vid_vsync	O	Vertical sync pulse. Active on the rising edge.

Notes:
1. <n> = stream number 2 to 4.

AUX IO Interface – Internal Bidirectional IOB

Table 2-16: **AUX IO Interface – Internal Bidirectional IOB**

Signal Name	I/O	Description
aux_rx_io_p	I/O	Bidirectional AUX IO - P
aux_rx_io_n	I/O	Bidirectional AUX IO - n

Audio Streaming Signals

The DisplayPort RX Sink audio streaming signals are shown in Table 2-17.

Table 2-17: **DisplayPort Sink Audio Interface**

Name	I/O	Description
m_axis_audio_egress_tdata[31:0]	O	Streaming data output. [3:0] – PR (Preamble Code) ◦ 4'b0001 -> Subframe1 / start of audio block ◦ 4'b0010 -> Subframe 1 ◦ 4'b0011 -> Subframe 2 [27:4] – Audio Sample Word [28] – V (Validity Bit) [29] – U (User Bit) [30] – C (Channel Status) [31] – P (Parity)
m_axis_audio_egress_tid [7:0]	O	[3:0] – Audio Channel ID [7:4] – Audio Packet Stream ID
m_axis_audio_egress_tvalid	O	Valid indicator for audio data from master.
m_axis_audio_egress_tready	I	Ready indicator from external streaming module.

AUX IO Interface – Internal Unidirectional IOB

Table 2-18: **AUX IO Interface – Internal Unidirectional IOB Ports**

Signal Name	I/O	Description
aux_rx_channel_in_p	I	Unidirectional AUX channel in - P
aux_rx_channel_in_n	I	Unidirectional AUX channel in - N

Table 2-18: **AUX IO Interface – Internal Unidirectional IOB Ports *(Cont'd)***

Signal Name	I/O	Description
aux_rx_channel_out_p	O	Unidirectional AUX channel out - P
aux_rx_channel_out_n	O	Unidirectional AUX channel out - N

AUX IP Interface – External IOB

Table 2-19: **AUX IP Interface – External IOB**

Signal Name	I/O	Description
aux_rx_data_in	I	External AUX data input
aux_rx_data_out	O	External AUX data output
aux_rx_data_en_out_n	O	External AUX data enable out. Active-Low.

HPD Interface Ports

Table 2-20: **HPD Interface Ports**

Signal Name	I/O	Description
rx_hpd	O	HPD from DisplayPort RX

EDID IIC Ports

Table 2-21: **EDID IIC Interface Ports**

Signal Name	I/O	Description
edid_iic_sci_i	I	EDID IIC SCL input
edid_iic_sci_o	O	EDID IIC SCL output
edid_iic_sci_t	O	EDID IIC SCL enable. IIC SCL enable is Active-Low.
edid_iic_sda_i	I	EDID IIC SDA input
edid_iic_sda_o	O	EDID IIC SDA output
edid_iic_sda_t	O	EDID IIC SDA enable. IIC SDA enable is Active-Low.

DP159 Interface Ports

Table 2-22: **DP159 Interface Ports**

Signal Name	I/O	Description
dp159_iic_sci_i	I	DP159 IIC SCL input
dp159_iic_sci_o	O	DP159 IIC SCL output
dp159_iic_sci_t	O	DP159 IIC SCL enable
dp159_iic_sda_i	I	DP159 IIC SDA input
dp159_iic_sda_o	O	DP159 IIC SDA output

Table 2-22: **DP159 Interface Ports *(Cont'd)***

Signal Name	I/O	Description
dp159_iic_sda_t	O	DP159 IIC SDA enable
dp159_rst	O	DP159 IIC reset through AXI IIC controller GPIO port0

HDCP External Clock

Enabled when HDCP is selected with 16-bit GT interface.

Table 2-23: **HDCP External Clock**

Signal Name	I/O	Description
hdcp_ext_clk	I	HDCP external clock (enabled when HDCP is selected with 16-bit GT interface)

HDCP Key Interface

Table 2-24: **HDCP Key Interface**

Signal Name	I/O	Description
hdcp_key_aclk	I	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_tuse[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	Selects one of the eight sets of 40 keys.
Start_key_transmit	O	Active-High pulse starts key transmit.

Interrupts

Table 2-25: **Interrupts**

Signal Name	I/O	Description
dprxss_dp_irq	O	DisplayPort RX IP interrupt out
dprxss_iic_irq	O	AXI IIC IP interrupt out
dprx_hdcp_irq	O	HDCP IP interrupt out
dprx_timer_irq	O	AXI Timer interrupt out

Register Space

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

- DisplayPort Receive (RX) IP
- AXI IIC
- HDCP
- AXI Timer

The subsystem address propagation in the Vivado® IP integrator assigns the maximum addresses based on full featured configuration. Ensure the following steps are taken:

1. Confirm that the DisplayPort RX subsystem IP is mapped to a base address where the 14th bit in the address is 0. For example, **0x44A00000** is correct and **0x44A02000** causes errors.
2. Ensure that all 14 bits of the address range are reserved for the DisplayPort RX subsystem IP.

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 maintained to prevent the status values from being updated while the read is occurring. For multi-bit configuration data, a toggle bit is used indicating that the local values in the functional core should be updated.

Any bits not specified in Table 2-26 to Table 2-35 are to be considered reserved and returns 0 upon read. Only address offsets are listed in these register tables; base addresses are configured by the AXI Interconnect.

DisplayPort Receive IP Core Registers

Receiver Core Configuration

Table 2-26: **Receiver Core Configuration**

Offset	R/W	Definition
0x000	RW	LINK_ENABLE. Enable the receiver 1 – Enables the receiver core. Asserts the HPD signal when set.
0x004	RW	AUX_CLOCK_DIVIDER. Contains the clock divider value for generating the internal 1 MHz clock from the AXI4-Lite host interface clock. The clock divider register provides integer division only and does not support fractional AXI4-Lite clock rates (for example, set to 75 for a 75 MHz AXI4-Lite clock). [27:24] – Valid values are 0-8. Non-zero value in this field will issue defers as per programmed value to DPCD read of LANE0_1_STATUS register. This functionality is needed to extend the clock recovery period from default. [[15:8] – This is the filter value, used to sample the AUX input with the AXI clock. It conveys the minimum number of clocks that AUX data is stable in terms of the AXI clock. But it does not represent an actual AUX pulse width. Allowed filter values are 0, 8, 16, 24, 32, 40, and 48. To calculate, use the largest filter value that satisfies the equation: filter value * AXI clock period ≤ 0.4 µs The allowed values are defined as: 0, 8 - AUX data should be constant for at least 7 clocks 16 – AUX data should be constant for at least 15 clocks 24 – AUX data should be constant for at least 23 clocks 32 – AUX data should be constant for at least 31 clocks 40 – AUX data should be constant for at least 40 clocks 48 – AUX data should be constant for at least 47 clocks Example: 50 MHz - Clock divider value: 50; the filter value should be 16 because the filter should be ≤ 20 (0.4 µs/(1/50 MHz)) and the nearest allowed value is 16. 100MHz - Clock divider value: 100; the filter value should be 40 because the filter value should be ≤ 40 (0.4 µs/(1/100 MHz)) and the nearest allowed value is 40. 135MHz – Clock divider value: 135; the filter value should be 48 because the filter value should be ≤ 54 (0.4 µs/(1/135 MHz)) and the nearest allowed value is 48 [7:0] Clock divider value. This value should satisfy the following conditions with 0.025 µs tolerance. (Clock divider value/8) * 3 * AXI clock period > 0.4 µs (as per RTL allowable value is 0.375 µs with tolerance) (Clock divider value/8) * 5 * AXI clock period < 0.6 µs (as per RTL allowable value is 0.625 µs with tolerance) (Clock divider value/2) * AXI clock period > 0.4 µs and < 0.6 µs
0x008	RW	RX_LINE_RESET_DISABLE. RX line reset disable. This register bit can be used to disable the end of line reset to the internal video pipe for reduced blanking video support. [3] – End of line reset disable to the MST video stream4 [2] – End of line reset disable to the MST video stream3 [1] – End of line reset disable to the MST video stream2 [0] – End of line reset disable to the SST video stream/ MST video stream1

Table 2-26: **Receiver Core Configuration *(Cont'd)***

Offset	R/W	Definition
0x00C	RW	DTG_ENABLE. Enables the display timing generator in the user interface. [0] – DTG_ENABLE: Set to 1 to enable the timing generator. The DTG should be disabled when the core detects the no-video pattern on the link.
0x010	RW	USER_PIXEL_WIDTH. Configures the number of pixels output through the user data interface. The sink controller programs the pixel width to the active lane count (default). User can override this by writing a new value to this register. Use quad pixel mode in MST. [2:0] 1 = Single pixel wide interface. 2 = Dual pixel output mode. Valid for designs with 2 or 4 lanes. 4 = Quad pixel output mode. Valid for designs with 4 lanes only.
0x014	RW	INTERRUPT_MASK. Masks the specified interrupt sources from asserting the axi_init signal. When set to a 1, the specified interrupt source is masked. This register resets to all 1s at power up. [31] – Mask for Cable disconnect/unplug interrupt [30] – CRC test start interrupt [29] – Mask MST Act sequence received interrupt [28] – Mask interrupt generated when DPCD registers 0x1C0, 0x1C1 and 0x1C2 are written for allocation/de-allocation/partial deletion [27] – Audio packet FIFO overflow interrupt [18] – Training pattern 3 start interrupt [17] – Training pattern 2 start interrupt [16] – Training pattern 1 start interrupt [15] – Bandwidth change interrupt [14] – TRAINING_DONE [13] – DOWN_REQUEST_BUFFER_READY [12] – DOWN_REPLY_BUFFER_READ [11] – VC Payload Deallocated [10] – VC Payload Allocated [9] – EXT_PKT_RXD: Set to 1 when extension packet is received [8] – INFO_PKT_RXD: Set to 1 when info packet is received [6] – VIDEO: Set to 1 when valid video frame is detected on main link. Video interrupt is set after a delay of eight video frames following a valid scrambler reset character [4] – TRAINING_LOST: Training has been lost on any one of the active lanes [3] – VERTICAL_BLANKING: Start of the vertical blanking interval [2] – NO_VIDEO: The no-video condition has been detected after active video received [1] – POWER_STATE: Power state change, DPCD register value 0x00600 [0] – VIDEO_MODE_CHANGE: Resolution change, as detected from the MSA fields.

Table 2-26: **Receiver Core Configuration *(Cont'd)***

Offset	R/W	Definition
0x018	RW	MISC_CONTROL. Allows the host to instruct the receiver to pass the MSA values through unfiltered. [2] – When set to 1, I2C DEFERs will be sent as AUX DEFERs to the source device. [1] – When set to 1, the long I2C write data transfers are responded to using DEFER instead of Partial ACKs. [0] – USE_FILTERED_MSA: When set to 0, this bit disables the filter on the MSA values received by the core. When set to 1, two matching values must be detected for each field of the MSA values before the associated register is updated internally.
0x01C	WO	SOFTWARE_RESET_REGISTER. [8] – Soft reset control to external HDCP FIFOs. [7] – AUX Soft Reset. When set, AUX logic will be reset. [0] – Soft Video Reset: When set, video logic will be reset. Reads will return zeros.

Aux Channel Status

Table 2-27: **Aux Channel Status**

Offset	R/W	Definition
0x020	RO	AUX_REQUEST_IN_PROGRESS. Indicates the receipt of an AUX Channel request [0] – 1 indicates a request is in progress.
0x024	RO	REQUEST_ERROR_COUNT. Provides a running total of errors detected on inbound AUX Channel requests. [7:0] – Error count, a write to register address 0x28 clears this counter.
0x028	RW	REQUEST_COUNT. Provides a running total of the number of AUX requests received. [7:0] – Total AUX request count, a write to register 0x28 clears this counter.
0x02C	WO	HPD_INTERRUPT. Instructs the receiver core to assert an interrupt to the transmitter using the HPD signal. A read from this register always returns 0x0. [31:16] – HPD_INTERRUPT_LENGTH: Default value is 0. This field defines the length of the HPD pulse. The value should be given in microsecond units. For example for 750 µs, program 750 in the register. [0] – Set to 1 to send the interrupt through the HPD signal. The HPD signal is brought low for 750 µs to indicate to the source that an interrupt has been requested.
0x030	RO	REQUEST_CLOCK_WIDTH. Holds the half period of recovered AUX clock. [9:0] – Indicates the number of AXI_CLK cycles between sequential edges during the SYNC period of the most recent AUX request.
0x034	RO	REQUEST_COMMAND. Provides the most recent AUX command received. [3:0] – Provides the command field of the most recently received AUX request.
0x038	RO	REQUEST_ADDRESS. Contains the address field of the most recent AUX request. [19:0] – The twenty-bit address field from the most recent AUX request transaction is placed in this register. For I2C over AUX transactions, the address range will be limited to the seven LSBs.

Table 2-27: **Aux Channel Status *(Cont'd)***

Offset	R/W	Definition
0x03C	RO	REQUEST_LENGTH. The length of the most recent AUX request is written to this register. The length of the AUX request is the value of this register plus one. [3:0] – Contains the length of the AUX request. Transaction lengths from 1 to 16 bytes are supported. For address only transactions, the value of this register will be 0.
0x040	RC	INTERRUPT_CAUSE. Indicates the cause of a pending host interrupt. A read from this register clears all values. Write operation is illegal and clears the values. [31] – Cable disconnect/unplug interrupt [30] – CRC test start interrupt [29] – MST Act sequence received interrupt [28] – interrupt generated when DPCD registers 0x1C0, 0x1C1, and 0x1C2 are written for allocation/de-allocation/partial deletion [27] – Audio packet FIFO overflow interrupt. [18] – Training pattern 3 start interrupt. [17] – Training pattern 2 start interrupt. [16] – Training pattern 1 start interrupt. [15] – Bandwidth change interrupt. [14] – TRAINING_DONE: Set to 1 when training is done. [13] – DOWN_REQUEST_BUFFER_READY: set to 1 indicating availability of Down request. [12] – DOWN_REPLY_BUFFER_READ: Set to 1 for a read event from Down Reply Buffer by upstream source. [11] – VC Payload Deallocated: Set to 0 when de-allocation event occurs in controller. [10] – VC Payload Allocated: Set to 1 when allocation event occurs in controller. [9] – EXT_PKT_RXD: Set to 1 when extension packet is received. [8] – INFO_PKT_RXD: Set to 1 when info packet is received. [7] – Reserved. [6] – VIDEO: Set to 1 when a valid video frame is detected on main link. [5] – Reserved [4] – TRAINING_LOST: This interrupt is set when the receiver has been trained and subsequently loses clock recovery, symbol lock or inter-lane alignment, in any of the lanes. [3] – VERTICAL_BLANKING: This interrupt is set at the start of the vertical blanking interval as indicated by the VerticalBlanking_Flag in the VB-ID field of the received stream. [2] – NO_VIDEO: the receiver has detected the no-video flags in the VBID field after active video has been received. [1] – POWER_STATE: The transmitter has requested a change in the current power state of the receiver core. [0] – VIDEO_MODE_CHANGE: A change has been detected in the current video mode transmitted on the DisplayPort link as indicated by the MSA fields. The horizontal and vertical resolution parameters are monitored for changes.

Table 2-27: **Aux Channel Status *(Cont'd)***

Offset	R/W	Definition
0x044	RW	INTERRUPT_MASK_1: 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. [17] – Video Interrupt – Stream 4 [16] – Vertical Blanking Interrupt – Stream4 [15] – No Video Interrupt – Stream 4 [14] – Mode Change Interrupt – Stream 4 [13] – Info Packet Received – Stream 4 [12] – Ext Packet Received – Stream 4 [11] – Video Interrupt – Stream 3 [10] – Vertical Blanking Interrupt – Stream 3 [9] – No Video Interrupt – Stream 3 [8] – Mode Change Interrupt – Stream 3 [7] – Info Packet Received – Stream 3 [6] – Ext Packet Received – Stream 3 [5] – Video Interrupt – Stream 2 [4] – Vertical Blanking Interrupt – Stream 2 [3] – No Video Interrupt – Stream 2 [2] – Mode Change Interrupt – Stream 2 [1] – Info Packet Received – Stream 2 [0] – Ext Packet Received – Stream 2
0x048	RC	INTERRUPT_CAUSE_1: Indicates the cause of a pending host interrupt. A read from this register clears all values. A write operation would be illegal and would clear all values as well. These bits have the same function as those described in the Interrupt Case register of stream 1. Reserved bits return 0. See offset 0x040 for more description on each interrupt. [17] – Video Interrupt – Stream 4 [16] – Vertical Blanking Interrupt – Stream4 [15] – No Video Interrupt – Stream 4 [14] – Mode Change Interrupt – Stream 4 [13] – Info Packet Received – Stream 4 [12] – Ext Packet Received – Stream 4 [11] – Video Interrupt – Stream 3 [10] – Vertical Blanking Interrupt – Stream 3 [9] – No Video Interrupt – Stream 3 [8] – Mode Change Interrupt – Stream 3 [7] – Info Packet Received – Stream 3 [6] – Ext Packet Received – Stream 3 [5] – Video Interrupt – Stream 2 [4] – Vertical Blanking Interrupt – Stream 2 [3] – No Video Interrupt – Stream 2 [2] – Mode Change Interrupt – Stream 2 [1] – Info Packet Received – Stream 2 [0] – Ext Packet Received – Stream 2

Table 2-27: **Aux Channel Status *(Cont'd)***

Offset	R/W	Definition
0x050	RW	HSYNC_WIDTH. The display timing generator control logic outputs a fixed length, active-High pulse for the horizontal sync. The timing of this pulse might be controlled by setting this register appropriately. The default value of this register is 0x0F0F. [15:8] – HSYNC_FRONT_PORCH: Defines the number of video clock cycles to place between the last pixel of active data and the start of the horizontal sync pulse. [7:0] – HSYNC_PULSE_WIDTH: Specifies the number of clock cycles the horizontal sync pulse is asserted. The vid_hsync signal will be high for the specified number of clock cycles.
0x058	RW	VSYNC_WIDTH. The display timing generator control logic outputs a fixed length of 63 RX video clocks, active-High pulse for the vertical sync pulse. The timing of this pulse might be controlled by setting this register appropriately. The default value of this register is 0x003F. [15:0] – VSYNC_WIDTH: Defines the number of RX video clock cycles the vertical sync pulse is asserted. The minimum value to be programmed in this register is 0x003F. User can configure the register based on actual VSYNC duration based on MSA.
0x060	RW	[7:0] – FAST_I2C_DIVIDER. Fast I2C mode clock divider value. Set this value to (AXI4-Lite clock frequency/10) - 1. Valid only for DPCD 1.2.
0x064	RW	[31] – Set to override Training Pattern 1 (TP1) score. [14:0] – Training pattern1 (TP1) score to override.
0x068	RW	[31] – Set to override the Training Pattern2/3 scores. [12:0] – Training pattern2/3 (TP2/TP3) score to override.
0x06C	RO	Provides the contents of DPCD registers 0x1C0, 0x1C1, and 0x1C2 [21:16] – Time slot count [13:8] – Starting Time Slot [5:0] – VC Payload ID
0x074	RW	[5] – Enable the CRC support. The CRC has to be calculated outside the DisplayPort IP and the values have to be provided in 0x078, 0x07C, and 0x080. [3:0] – CRC change count to be configured by SW
0x078	RW	CRC for Red color
0x07C	RW	CRC for Green color
0x080	RW	CRC for Blue color

DPCD Fields

Table 2-28: **DPCD Fields**

Offset	R/W	Definition
0x084	RW	LOCAL_EDID_VIDEO. Indicates the presence of EDID information for the video stream. [0] – Set to 1 to indicate to the transmitter through the DPCD registers that the receiver supports local EDID information.
0x088	RW	LOCAL_EDID_AUDIO. Indicates the presence of EDID information for the audio stream. [0] – Set to 1 to indicate to the transmitter through the DPCD registers that the receiver supports local EDID information
0x08C	RW	REMOTE_COMMAND. General byte for passing remote information to the transmitter. [7:0] – Remote data byte.
0x090	RW	DEVICE_SERVICE_IRQ. Indicates DPCD DEVICE_SERVICE_IRQ_VECTOR state. [4] – Set to 1 to indicate a new DOWN Reply Buffer Message is ready. [1] – Reflects SINK_SPECIFIC_IRQ state of DPCD 0x201 register. This bit is RO. [0] – Set to 1 to indicate a new command. Indicates a new command present in the REMOTE_COMMAND register. A Write of 0x1 to this register sets the DPCD register DEVICE_SERVICE_IRQ_VECTOR (0x201), REMOTE_CONTROL_PENDING bit. A write of 0x0 to this register has no effect. Refer to DPCD register section of the standard for more details. Reads from this register reflect the state of DPCD register.
0x094	RW	VIDEO_UNSUPPORTED. DPCD register bit to inform the transmitter that video data is not supported. [0] – Set to 1 when video data is not supported.
0x098	RW	AUDIO_UNSUPPORTED. DPCD register bit to inform the transmitter that audio data is not supported [0] – Set to 1 when audio data is not supported.
0x09C	RW	Override LINK_BW_SET. This register can be used to override LINK_BW_SET in the DPCD register set. Register 0x0B8 (direct_dpcd_access) must be set to 1 to override DPCD values. LINK_BW_SET corresponds to the 00100h and 0x0001h DPCD address space. [4:0] – Link rate override value for DisplayPort Standard v1.2a designs [3:0] – Link rate override value for DisplayPort Standard v1.1a designs 0x6 - 1.62 Gb/s 0xA - 2.7 Gb/s 0x14 - 5.4 Gb/s
0x0A0	RW	Override LANE_COUNT_SET. This register can be used to override LANE_COUNT_SET in the DPCD register set. Register 0x0B8 (direct_dpcd_access) must be set to 1 to override DPCD values. LANE_COUNT_SET corresponds to the 00101h and 0x00002h DPCD address space. [7] – ENHANCED_FRAME_CAP: Capability override [6] – TPS3_SUPPORTED: Capability override for DisplayPort Standard v1.2a protocol designs only. Reserved for v1.1a protocol. [4:0] – Lane count override value (1, 2, or 4 lanes).

Table 2-28: **DPCD Fields *(Cont'd)***

Offset	R/W	Definition
0x0A4	RW	Override TRAINING_PATTERN_SET. This register can be used to override TRAINING_PATTERN_SET in the DPCD register set. Register 0x0B8 (direct_dpcd_access) must be set to 1 to override DPCD values. TRAINING_PATTERN_SET corresponds to the 00102h DPCD address space. [15:8] – TRAINING_AUX_RD_INTERVAL (the values are based on DisplayPort Standard v1.2a protocol). [7:6] – SYMBOL ERROR COUNT SEL Override [5] – SCRAMBLING_DISABLE Override [4] – RECOVERED_CLOCK_OUT_EN Override [3:2] – LINK_QUAL_PATTERN_SET Override for DisplayPort Standard v1.1a only. [1:0] – TRAINING_PATTERN_SELECT Override
0x0A8	RW	Override TRAINING_LANE0_SET. This register can be used to override TRAINING_LANE0_SET in the DPCD register set. Register 0x0B8 (direct_dpcd_access) must be set to 1 to override DPCD values. TRAINING_LANE0_SET corresponds to the 00103h DPCD address space. [7:6] – Reserved [5] – MAX_PRE-EMPHASIS_REACHED override [4:3] – PRE-EMPHASIS_SET override [2] – MAX_SWING_REACHED override [1:0] – VOLTAGE SWING SET override
0x0AC	RW	Override TRAINING_LANE1_SET. This register can be used to override TRAINING_LANE1_SET in the DPCD register set. Register 0x0B8 (direct_dpcd_access) must be set to 1 to override DPCD values. Same as Override TRAINING_LANE0_SET. TRAINING_LANE1_SET corresponds to the 00104h DPCD address space.
0x0B0	RW	Override TRAINING_LANE2_SET. This register can be used to override TRAINING_LANE2_SET in the DPCD register set. Register 0x0B8 (direct_dpcd_access) must be set to 1 to override DPCD values. Same as Override TRAINING_LANE0_SET. TRAINING_LANE2_SET corresponds to the 00105h DPCD address space.
0x0B4	RW	Override TRAINING_LANE3_SET. This register can be used to override TRAINING_LANE3_SET in the DPCD register set. Register 0x0B8 (direct_dpcd_access) must be set to 1 to override DPCD values. Same as Override TRAINING_LANE0_SET. TRAINING_LANE3_SET corresponds to the 00106h DPCD address space.
0x0B8 *	RW	Override DPCD Control Register. Setting this register to 0x1 enables AXI/APB write access to DPCD capability structure.
0x0BC	RW	Override DPCD DOWNSPREAD control field. Register 0x0B8 must be set to 1 to override DPCD values. [0] – MAX_DOWNSPREAD Override
0x0C0	RW	Override DPCD LINK_QUAL_LANE0_SET field for DPCD1.2 version only. Register 0x0B8 must be set to 1 to override DPCD values. [2:0] – LINK_QUAL_LANE0_SET override
0x0C4	RW	Override DPCD LINK_QUAL_LANE1_SET field for DPCD1.2 version only. Register 0x0B8 must be set to 1 to override DPCD values. [2:0] – LINK_QUAL_LANE1_SET override

Table 2-28: **DPCD Fields *(Cont'd)***

Offset	R/W	Definition
0x0C8	RW	Override DPCD LINK_QUAL_LANE2_SET field for DPCD1.2 version only. Register 0x0B8 must be set to 1 to override DPCD values. [2:0] – LINK_QUAL_LANE2_SET override
0x0CC	RW	Override DPCD LINK_QUAL_LANE3_SET field for DPCD1.2 version only. Register 0x0B8 must be set to 1 to override DPCD values. [2:0] – LINK_QUAL_LANE3_SET override
0x0D0	RW	[8] – Clears the VCPayload Table contents. This bit is auto cleared. [7:5] – Unused [4] – VCPayload Table update bit. SW writes this bit with which the core updates the DPCD register 0x2C0 bit to 1. This bit is used when VC Payload table is controlled through SW. [2] – Set to 1 to override act trigger. Usually, the VCPayload active buffer updates on receiving ACT trigger sequence. This bit can be set when the VC payload table is in SW control. This bit is for advanced users only. [1] – Set to 1 to enable SW control over VCpayload table. This provides SW write access to offset 0x800-0x8ff. This bit is for advanced users only. [0] – MST CAPABILITY: Enable or Disable MST capability. Set to 1 to enable MST capability.This bit should be set during configuration programming stage only.
0x0D4	RW	[6:0] – Sink device count: Recommended to be programmed during initialization of the sink device. In SST mode, the value should be 1.
0x0E0	RW	GUID word 0. Allows you to set up GUID if required from host interface. Valid for DPCD1.2 version only. [31:0] – Lower 4 bytes of GUID DPCD field
0x0E4	RW	GUID word 1. Allows you to set up GUID if required from host interface. Valid for DPCD1.2 version only. [31:0] – Bytes 4 to 7 of GUID DPCD field
0x0E8	RW	GUID word 2. Allows you to set up GUID if required from host interface. Valid for DPCD1.2 version only. [31:0] – Bytes 8 to 11 of GUID DPCD field
0x0EC	RW	GUID word 3. Allows you to set up GUID if required from host interface. Valid for DPCD1.2 version only. [31:0] – Bytes 12 to 15 of GUID DPCD field
0x0F0	RW	GUID Override. [0]: When set to 0x1, the GUID field of the DPCD reflects the data written in GUID Words 0 to 3. Valid for DPCD1.2 version only. When this register is set to 0x1, GUID field of DPCD becomes read only and source-aux writes are NACK-ed.

Core ID

Table 2-29: **Core ID**

Offset	R/W	Definition
0x0F8	RO	VERSION Register. For example, for displayport_v7.0, the VERSION REGISTER is 32'h07_00_0_0_00. [31:24] – Core major version [23:16] – Core minor version [15:12] – Core version revision [11:8] – Core Patch details [7:0] – Internal revision
0x0FC	RO	CORE_ID. Returns the unique identification code of the core and the current revision level. [31:24] – DisplayPort protocol major version [23:16] – DisplayPort protocol minor version [15:8] – DisplayPort protocol revision [7:0] – Core mode of operation 0x00: Transmit 0x01: Receive Depending on the protocol and core used, the CORE_ID values are as follows: DisplayPort Standard v1.1a, Receive core: 32'h01_01_0a_01 DisplayPort Standard v1.2a, Receive core: 32'h01_02_0a_01
0x110	RO	USER_FIFO_OVERFLOW. This status bit indicates an overflow of the user data FIFO of pixel data. This event might occur if the input pixel clock is not fast enough to support the current DisplayPort link width and link speed. [11] – Video Timing FIFO_OVERFLOW_FLAG (Stream 4): 1 indicates that the Video Timing FIFO has overflowed for stream4. [10] – Video Timing FIFO_OVERFLOW_FLAG (Stream 3): 1 indicates that the Video Timing FIFO has overflowed for stream3. [9] – Video Timing FIFO_OVERFLOW_FLAG (Stream 2): 1 indicates that the Video Timing FIFO has overflowed for stream2. [8] – Video Timing FIFO_OVERFLOW_FLAG (Stream 1): 1 indicates that the Video Timing FIFO has overflowed. [7] – Video Unpack FIFO_OVERFLOW_FLAG (Stream 4): 1 indicates that the Video unpack FIFO is overflowed for stream4. [6] – Video Unpack FIFO_OVERFLOW_FLAG (Stream 3): 1 indicates that the Video unpack FIFO has overflowed for stream3. [5] – Video Unpack FIFO_OVERFLOW_FLAG (Stream 2): 1 indicates that the Video unpack FIFO has overflowed for stream2. [4] – Video Unpack FIFO_OVERFLOW_FLAG (Stream 1): 1 indicates that the Video unpack FIFO has overflowed. [3] – FIFO_OVERFLOW_FLAG (Stream 4): 1 indicates that the internal FIFO has detected an overflow condition for Stream 4. This bit clears upon read. [2] – FIFO_OVERFLOW_FLAG (Stream 3): 1 indicates that the internal FIFO has detected an overflow condition for Stream 3. This bit clears upon read. [1] – FIFO_OVERFLOW_FLAG (Stream 2): 1 indicates that the internal FIFO has detected an overflow condition for Stream 2. This bit clears upon read. [0] – FIFO_OVERFLOW_FLAG (Stream 1): 1 indicates that the internal FIFO has detected an overflow condition for Stream 1. This bit clears upon read.
0x114	RO	USER_VSYNC_STATE. Provides a mechanism for the host processor to monitor the state of the video datapath. This bit is set when vsync is asserted. [3] – State of the vertical sync pulse for Stream 4. [2] – State of the vertical sync pulse for Stream 3. [1] – State of the vertical sync pulse for Stream 2. [0] – State of the vertical sync pulse for Stream 1.

PHY Configuration and Status

Table 2-30: **PHY Configuration and Status**

Offset	R/W	Definition
0x208	RO	PHY_STATUS. Provides status for the receiver core PHY. [31:30] – RX buffer status, lane 3. [29:28] – RX buffer status, lane 2. [27:26] – RX buffer status, lane 1. [25:24] – RX buffer status, lane 0. [23] – RXBYTEISALIGNED status of PHY, lane 3. [22] – RXBYTEISALIGNED status of PHY, lane 2. [21] – RXBYTEISALIGNED status of PHY, lane 1. [20] – RXBYTEISALIGNED status of PHY, lane 0. [15:14] – RX voltage low, lanes 2 and 3. [13:12] – RX voltage low, lanes 0 and 1. [11:10] – PRBS error, lanes 2 and 3. [9:8] – PRBS error, lanes 0 and 1. [7] – Receiver Clock locked. [6] – FPGA fabric clock PLL locked. [5] – PLL for lanes 2 and 3 locked (Tile 1). [4] – PLL for lanes 0 and 1 locked (Tile 0). [3:2] – Reset done for lanes 2 and 3 (Tile 1). [1:0] – Reset done for lanes 0 and 1 (Tile 0).
0x214	RW	MIN_VOLTAGE_SWING. Some DisplayPort implementations require the transmitter to set a minimum voltage swing during training before the link can be reliably established. This register is used to set a minimum value which must be met in the TRAINING_LANEX_SET DPCD registers. The internal training logic will force training to fail until this value is met. ***Note:*** It is not recommended to change this register value. [23:14] – PREEMP_TABLE (only for Advanced users). 15:14: Iteration 1 pre-emp request level 17:16: Iteration 2 pre-emp request level 19:18: Iteration 3 pre-emp request level 21:20: Iteration 4 pre-emp request level 23:22: Iteration 5 pre-emp request level [13:12] – SET_PREEMP (only for Advanced users). [11:10] – Channel Equalization options (only for Advanced users). 00: Default (pre-emphasis adjust request will get incremented one by per iteration until maximum pre-emphasis limit (SET_PREEMP) is reached) 01: Hold pre-emphasis adjust request to SET_PREEMP for all iterations 10: Not applicable 11: Pick values from PREEMP_TABLE [9:8] – SET_VSWING (only for Advanced users). Default value is 0x0000. [6:4] – VSWING_SWEEP_CNT (only for Advanced users). [3:2] – Clock Recovery options (only for Advanced users). 00: Default (Voltage swing adjust request will get incremented one by every iteration) 01: Increment adjust request every 4 or VSWING_SWEEP_CNT iterations 10: Hold adjust request to SET_VSWING value for all iterations 11: Not applicable [1:0] – The minimum voltage swing setting matches the values defined in the DisplayPort Standard for the TRAINING_LANEX_SET register.

Table 2-30: **PHY Configuration and Status *(Cont'd)***

Offset	R/W	Definition
0x21C	RW	CDR_CONTROL_CONFIG. [30] – Disable Training Timeout. [19:0] – Controls the CDR tDLOCK timeout value. The counter is run using the AXI4-Lite clock in the PHY Module. Default value is 20'h1388.
0x220	RW	BS_IDLE_TIME. Blanking start symbol idle time. Default is 0x1312D00 (200 ms) considering 100 MHz AXI4-Lite clock frequency. The value is based on AXI4-Lite clock frequency and you are expected to update as needed. [31:0] – The value written in this register is used in the DisplayPort Sink to detect cable disconnect or unplug event. The DisplayPort sink checks the Blanking Start symbol over the link for the specified period and generates cable unplug interrupt. The timeout counter is loaded with this register value which is working with AXI clock. The cable unplug counter is a free running counter and it reloads with BS_IDLE_TIME as and when it receives the BS character over the link or when it times out by reaching maximum count.

DisplayPort Audio

Table 2-31: **DisplayPort Audio**

Offset	R/W	Definition
12'h300	RW	RX_AUDIO_CONTROL. This register enables audio stream packets in main link. [0] – Audio Enable
12'h304	RO	RX_AUDIO_INFO_DATA [31:0] Word formatted as per CEA 861-C Info Frame. Total of eight words should be read. 1st word: [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.Info frame data is copied into these registers (read only).
12'h324	RO	RX_AUDIO_MAUD. M value of audio stream as decoded from audio time stamp packet by the sink (read only). [31:24] – Reserved [23:0] – MAUD
12'h328	RO	RX_AUDIO_NAUD. N value of audio stream as decoded from audio time stamp packet by the sink (read only). [31:24] – Reserved [23:0] – NAUD

Table 2-31: **DisplayPort Audio *(Cont'd)***

Offset	R/W	Definition
12'h32C	RO	RX_AUDIO_STATUS. [9] – Extension Packet Received. Resets automatically after all words (9) are read. Blocks new packet until host reads the data. [8:3] – Reserved. [2:1] – RS Decoder Error Counter. Used for debugging purpose. [0] – Info Packet Received. Resets automatically after all info words (eight) are read. Blocks new packet until host reads the data.
12'h330 to 12'h350	RO	RX_AUDIO_EXT_DATA [31:0] – Word formatted as per extension packet described in protocol standard. Packet length is fixed to 32 bytes in sink controller. You should convey this information to the source using the vendor fields and ensure correct packet size transmission is done by the source controller. A total of nine words should be read. The extension packet address space can be used to hold the audio copy management packet/ISRC packet/VSC packets. 1st word - [31:24] = HB3 [23:16] = HB2 [15:8] = HB1 [7:0] = HB0 2nd word - DB3,DB2,DB1,DB0 ... 9th word -DB31,DB30,DB29,DB28 Extension packet data is copied into these registers (read only). This is a key-hole memory. So, nine reads from this address space is required.

DPCD Configuration Space

Refer to the DisplayPort 1.1a standard for detailed descriptions of these registers.

Table 2-32: **DPCD Configuration Space**

Offset	R/W	Definition
0x400	RO	DPCD_LINK_BW_SET. Link bandwidth setting. [7:0] – Set to 0x0A when the link is configured for 2.7 Gb/s or 0x06 when configured for 1.62 Gb/s or 0x14 when link is configured for 5.4 Gb/s.
0x404	RO	DPCD_LANE_COUNT_SET. Number of lanes enabled by the transmitter. [4:0] – Contains the number of lanes that are currently enabled by the attached transmitter. Valid values fall in the range of 1-4.
0x408	RO	DPCD_ENHANCED_FRAME_EN. Indicates that the transmitter has enabled the enhanced framing symbol mode. [0] – Set to 1 when enhanced framing mode is enabled.

Table 2-32: **DPCD Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x40C	RO	DPCD_TRAINING_PATTERN_SET. Current value of the training pattern registers. [1:0] – TRAINING_PATTERN_SET: Set the link training pattern according to the two bit code: 00 = Training not in progress 01 = Training pattern 1 10 = Training pattern 2 11 = RESERVED
0x410	RO	DPCD_LINK_QUALITY_PATTERN_SET. Current value of the link quality pattern field of the DPCD training pattern register. [1:0] – transmitter is sending the link quality pattern: 00 = Link quality test pattern not transmitted 01 = D10.2 test pattern (unscrambled) transmitted 10 = Symbol Error Rate measurement pattern 11 = PRBS7 transmitted
0x414	RO	DPCD_RECOVERED_CLOCK_OUT_EN. Value of the output clock enable field of the DPCD training pattern register. [0] – Set to 1 to output the recovered receiver clock on the test port.
0x418	RO	DPCD_SCRAMBLING_DISABLE. Value of the scrambling disable field of the DPCD training pattern register. By default, scrambling is disabled. [0] – Set to 1 when the transmitter has disabled the scrambler and transmits all symbols.
0x41C	RO	DPCD_SYMBOL_ERROR_COUNT_SELECT. Current value of the symbol error count select field of the DPCD training pattern register. [1:0] – SYMBOL_ERROR_COUNT_SEL: 00 = Disparity error and illegal symbol error 01 = Disparity error 10 = Illegal symbol error 11 = Reserved
0x420	RO	DPCD_TRAINING_LANE_0_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. [5] – MAX_PRE-EMPHASIS_REACHED: Set to 1 when the maximum pre-emphasis setting is reached. [4:3] – PRE-EMPHASIS_SET 00 = Training Pattern 2 without pre-emphasis 01 = Training Pattern 2 with pre-emphasis level 1 10 = Training Pattern 2 with pre-emphasis level 2 11 = Training Pattern 2 with pre-emphasis level 3 [2] – MAX_SWING_REACHED: Set to '1' when the maximum driven current setting is reached. [1:0] – VOLTAGE_SWING_SET 00 = Training Pattern 1 with voltage swing level 0 01 = Training Pattern 1 with voltage swing level 1 10 = Training Pattern 1 with voltage swing level 2 11 = Training Pattern 1 with voltage swing level 3

Table 2-32: **DPCD Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x424	RO	DPCD_TRAINING_LANE_1_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. The fields of this register are identical to DPCD_TRAINING_LANE_0_SET.
0x428	RO	DPCD_TRAINING_LANE_2_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. The fields of this register are identical to DPCD_TRAINING_LANE_0_SET.
0x42C	RO	DPCD_TRAINING_LANE_3_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0. The fields of this register are identical to DPCD_TRAINING_LANE_0_SET.
0x430	RO	DPCD_DOWNSPREAD_CONTROL. The transmitter uses this bit to inform the receiver core that down-spreading has been enabled. [0] – SPREAD_AMP: Set to 1 for 0.5% spreading or 0 for none.
0x434	RO	DPCD_MAIN_LINK_CHANNEL_CODING_SET. 8B/10B encoding can be disabled by the transmitter through this register bit. [0] – Set to 0 to disable 8B/10B channel coding. The default is 1.
0x438	RO	DPCD_SET_POWER_STATE. Power state requested by the source core. On reset, power state is set to power down mode. [1:0] – requested power state 00 = Reserved 01 = state D0, normal operation 10 = state D3, power down mode 11 = Reserved
0x43C	RO	DPCD_LANE01_STATUS. Value of the lane 0 and lane 1 training status registers. [6] – LANE_1_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_1. [5] – LANE_1_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_1. [4] – LANE_1_CLOCK_RECOVERY_DONE. Set to 1, when D10.2 symbol is locked on LANE_1. [2] – LANE_0_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_0. [1] – LANE_0_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_0. [0] – LANE_0_CLOCK_RECOVERY_DONE. Set to 1, when D10.2 symbol is locked on LANE_0.

Table 2-32: **DPCD Configuration Space *(Cont'd)***

Offset	R/W	Definition
0x440	RO	DPCD_LANE23_STATUS. Value of the lane 2 and lane 3 training status registers. [6] – LANE_3_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_3. [5] – LANE_3_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_3. [4] – LANE_3_CLOCK_RECOVERY_DONE. Set to 1, when D10.2 symbol is locked on LANE_3. [2] – LANE_2_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_2. [1] – LANE_2_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_2. [0] – LANE_2_CLOCK_RECOVERY_DONE. Set to 1, when D10.2 symbol is locked on LANE_2.
0x444	RO	SOURCE_OUI_VALUE. Value of the Organizationally Unique Identifier (OUI) as written by the transmitter via the DPCD register AUX transaction. [23:0] – Contains the value of the OUI set by the transmitter. This value might be used by the host policy maker to enable special functions across the link.
0x448	RC/RO	SYM_ERR_CNT01. Reports symbol error counter of lanes 0 and 1. The lane 0 and lane 1 error counts are cleared when this register is read. [31] = Lane 1 error count valid. [30:16] = Lane 1 error count. [15] = Lane 0 error count valid. [14:0] = Lane 0 error count.
0x44C	RC	SYM_ERR_CNT23. Reports symbol error counter of lanes 2 and 3. The lane 2 and lane 3 error counts are cleared when this register is read. [31] = Lane 3 error count valid. [30:16] = Lane 3 error count. [15] = Lane 2 error count valid. [14:0] = Lane 2 error count.

MSA Values

Table 2-33: **MSA Values**

Offset	R/W	Definition
0x500	RO	MSA_HRES. The horizontal resolution detected in the Main Stream Attributes. [15:0] – Represents the number of pixels in a line of video.
0x504	RO	MSA_HSPOL. Horizontal sync polarity. [0] – Indicates the polarity of the horizontal sync as requested by the transmitter.
0x508	RO	MSA_HSWIDTH. Specifies the width of the horizontal sync pulse. [14:0] – Specifies the width of the horizontal sync in terms of the recovered video clock.
0x50C	RO	MSA_HSTART. This main stream attribute is the number of clock cycles between the leading edge of the horizontal sync and the first cycle of active data. [15:0] – Number of blanking cycles before active data.

Table 2-33: **MSA Values *(Cont'd)***

Offset	R/W	Definition
0x510	RO	MSA_HTOTAL. Tells the receiver core how many video clock cycles will occur between leading edges of the horizontal sync pulse. [15:0] – Total number of video clocks in a line of data.
0x514	RO	MSA_VHEIGHT. Total number of active video lines in a frame of video. [15:0] – The vertical resolution of the received video.
0x518	RO	MSA_VSPOL. Specifies the vertical sync polarity requested by the transmitter. [0] – A value of 1 in this register indicates an active-High vertical sync, and a '0' indicates an active-Low vertical sync.
0x51C	RO	MSA_VSWIDTH. The transmitter uses this value to specify the width of the vertical sync pulse in lines. [14:0] – Specifies the number of lines between the leading and trailing edges of the vertical sync pulse.
0x520	RO	MSA_VSTART. This main stream attribute specifies the number of lines between the leading edge of the vertical sync pulse and the first line of active data. [15:0] – Number of blanking lines before the start of active data.
0x524	RO	MSA_VTOTAL. Total number of lines between sequential leading edges of the vertical sync pulse. [15:0] – The total number of lines per video frame is contained in this value.
0x528	RO	MSA_MISC0. Contains the value of the MISC0 attribute data. [7:5] – COLOR_DEPTH: Number of bits per color/component. [4] – YCbCR_COLOR: Set to 1 (ITU-R BT709-5) or 0 (ITU-R BT601-5). [3] – DYNAMIC_RANGE: Set to 1 (CEA range) or 0 (VESA range). [2:1] – COMPONENT_FORMAT: 00 = RGB 01 = YCbCr 4:2:2 10 = YCbCr 4:4:4 11 = Reserved [0] – CLOCK_MODE: 0= Asynchronous clock mode 1= Synchronous clock mode
0x52C	RO	MSA_MISC1. Contains the value of the MISC1 attribute data. [7] – Implements the attribute information contained in the DisplayPort MISC1 register described in section 2.2.4 of the standard. [6:3] – RESERVED: These bits are always set to 0. [2:1] – STEREO_VIDEO: Used only when stereo video sources are being transmitted. See the *DisplayPort Standard v1.1a* section 2.24 for more information. [0] – INTERLACED_EVEN: 1 indicates that the number of lines per frame is an even number.
0x530	RO	MSA_MVID. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_NVID registers. [23:0] – MVID: Value of the clock recovery M value.

Table 2-33: **MSA Values *(Cont'd)***

Offset	R/W	Definition
0x534	RO	MSA_NVID. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_MVID registers. [23:0] – NVID: Value of the clock recovery N value.
0x538	RO	MSA_VBID. The most recently received VB-ID value is contained in this register. [7:0] – VBID: See Table 2-3 (p44) in the *DisplayPort Standard* for more information. The default value is 0x19.

MST MSA Values

Table 2-34: **MST MSA Values**

Offset	R/W	Definition
0x540	RO	MSA_HRES_STREAM2. The horizontal resolution detected in the Main Stream Attributes. [15:0] – Represents the number of pixels in a line of video.
0x544	RO	MSA_HSPOL_STREAM2. Horizontal sync polarity. [0] – Indicates the polarity of the horizontal sync as requested by the transmitter.
0x548	RO	MSA_HSWIDTH_STREAM2. Specifies the width of the horizontal sync pulse. [14:0] – Specifies the width of the horizontal sync in terms of the recovered video clock.
0x54C	RO	MSA_HSTART_STREAM2. This main stream attribute is the number of clock cycles between the leading edge of the horizontal sync and the first cycle of active data. [15:0] – Number of blanking cycles before active data.
0x550	RO	MSA_HTOTAL_STREAM2. Tells the receiver core how many video clock cycles will occur between leading edges of the horizontal sync pulse. [15:0] – Total number of video clocks in a line of data.
0x554	RO	MSA_VHEIGHT_STREAM2. Total number of active video lines in a frame of video. [15:0] – The vertical resolution of the received video.
0x558	RO	MSA_VSPOL_STREAM2. Specifies the vertical sync polarity requested by the transmitter. [0] – A value of 1 in this register indicates an active-High vertical sync, and a 0 indicates an active-Low vertical sync.
0x55C	RO	MSA_VSWIDTH_STREAM2. The transmitter uses this value to specify the width of the vertical sync pulse in lines. [14:0] – Specifies the number of lines between the leading and trailing edges of the vertical sync pulse.
0x560	RO	MSA_VSTART_STREAM2. This main stream attribute specifies the number of lines between the leading edge of the vertical sync pulse and the first line of active data. [15:0] – Number of blanking lines before the start of active data.

Table 2-34: **MST MSA Values *(Cont'd)***

Offset	R/W	Definition
0x564	RO	MSA_VTOTAL_STREAM2. Total number of lines between sequential leading edges of the vertical sync pulse. [15:0] – The total number of lines per video frame is contained in this value.
0x568	RO	MSA_MISC0_STREAM2. Contains the value of the MISC0 attribute data. [7:5] – COLOR_DEPTH: Number of bits per color/component. [4] – YCbCR_COLOR: Set to 1 (ITU-R BT709-5) or 0 (ITU-R BT601-5). [3] – DYNAMIC_RANGE: Set to 1 (CEA range) or 0 (VESA range). [2:1] – COMPONENT_FORMAT: 00 = RGB 01 = YCbCr 4:2:2 10 = YCbCr 4:4:4 11 = Reserved [0] – CLOCK_MODE: 0 = Synchronous clock mode 1 = Asynchronous clock mode
0x56C	RO	MSA_MISC1_STREAM2. Contains the value of the MISC1 attribute data. [7] – Implements the attribute information contained in the DisplayPort MISC1 register described in section 2.2.4 of the standard. [6:3] – RESERVED: These bits are always set to 0. [2:1] – STEREO_VIDEO: Used only when stereo video sources are being transmitted. See the *DisplayPort Standard v1.1a* section 2.24 for more information. [0] – INTERLACED_EVEN: 1 indicates that the number of lines per frame is an even number.
0x570	RO	MSA_MVID_STREAM2. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_NVID registers. [23:0] – MVID: Value of the clock recovery M value.
0x574	RO	MSA_NVID_STREAM2. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_MVID registers. [23:0] – NVID: Value of the clock recovery N value.
0x578	RO	MSA_VBID_STREAM2. The most recently received VB-ID value is contained in this register. [7:0] – VBID: See Table 2-3 (p44) in the *DisplayPort Standard* for more information. The default value is 0x19.
0x580	RO	MSA_HRES_STREAM3. The horizontal resolution detected in the Main Stream Attributes. [15:0] – Represents the number of pixels in a line of video.
0x584	RO	MSA_HSPOL_STREAM3. Horizontal sync polarity. [0] – Indicates the polarity of the horizontal sync as requested by the transmitter.
0x588	RO	MSA_HSWIDTH_STREAM3. Specifies the width of the horizontal sync pulse. [14:0] – Specifies the width of the horizontal sync in terms of the recovered video clock.

Table 2-34: **MST MSA Values *(Cont'd)***

Offset	R/W	Definition
0x59C	RO	MSA_HSTART_STREAM3. This main stream attribute is the number of clock cycles between the leading edge of the horizontal sync and the first cycle of active data. [15:0] – Number of blanking cycles before active data.
0x590	RO	MSA_HTOTAL_STREAM3. Tells the receiver core how many video clock cycles will occur between leading edges of the horizontal sync pulse. [15:0] – Total number of video clocks in a line of data.
0x594	RO	MSA_VHEIGHT_STREAM3. Total number of active video lines in a frame of video. [15:0] – The vertical resolution of the received video.
0x598	RO	MSA_VSPOL_STREAM3. Specifies the vertical sync polarity requested by the transmitter. [0] – A value of 1 in this register indicates an active-High vertical sync, and a '0' indicates an active-Low vertical sync.
0x59C	RO	MSA_VSWIDTH_STREAM3. The transmitter uses this value to specify the width of the vertical sync pulse in lines. [14:0] – Specifies the number of lines between the leading and trailing edges of the vertical sync pulse.
0x5A0	RO	MSA_VSTART_STREAM3. This main stream attribute specifies the number of lines between the leading edge of the vertical sync pulse and the first line of active data. [15:0] – Number of blanking lines before the start of active data.
0x5A4	RO	MSA_VTOTAL_STREAM3. Total number of lines between sequential leading edges of the vertical sync pulse. [15:0] – The total number of lines per video frame is contained in this value.
0x5A8	RO	MSA_MISC0_STREAM3. Contains the value of the MISC0 attribute data. [7:5] – COLOR_DEPTH: Number of bits per color/component. [4] – YCbCR_COLOR: Set to 1 (ITU-R BT709-5) or 0 (ITU-R BT601-5). [3] – DYNAMIC_RANGE: Set to 1 (CEA range) or 0 (VESA range). [2:1] – COMPONENT_FORMAT: 00 = RGB 01 = YCbCr 4:2:2 10 = YCbCr 4:4:4 11 = Reserved [0] – CLOCK_MODE: 0 = Synchronous clock mode 1 = Asynchronous clock mode
0x5AC	RO	MSA_MISC1_STREAM3. Contains the value of the MISC1 attribute data. [7] – Implements the attribute information contained in the DisplayPort MISC1 register described in section 2.2.4 of the standard. [6:3] – RESERVED: These bits are always set to 0. [2:1] – STEREO_VIDEO: Used only when stereo video sources are being transmitted. See the *DisplayPort Standard v1.1a* section 2.24 for more information. [0] – INTERLACED_EVEN: 1 indicates that the number of lines per frame is an even number.

Table 2-34: **MST MSA Values *(Cont'd)***

Offset	R/W	Definition
0x5B0	RO	MSA_MVID_STREAM3. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_NVID registers. [23:0] – MVID: Value of the clock recovery M value.
0x5B4	RO	MSA_NVID_STREAM3. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_MVID registers. [23:0] – NVID: Value of the clock recovery N value.
0x5B8	RO	MSA_VBID_STREAM3. The most recently received VB-ID value is contained in this register. [7:0] – VBID: See Table 2-3 (p44) in the *DisplayPort Standard* for more information. The default value is 0x19.
0x5C0	RO	MSA_HRES_STREAM4. The horizontal resolution detected in the Main Stream Attributes. [15:0] – Represents the number of pixels in a line of video.
0x5C4	RO	MSA_HSPOL_STREAM4. Horizontal sync polarity. [0] – Indicates the polarity of the horizontal sync as requested by the transmitter.
0x5C8	RO	MSA_HSWIDTH_STREAM4. Specifies the width of the horizontal sync pulse. [14:0] – Specifies the width of the horizontal sync in terms of the recovered video clock.
0x5CC	RO	MSA_HSTART_STREAM4. This main stream attribute is the number of clock cycles between the leading edge of the horizontal sync and the first cycle of active data. [15:0] – Number of blanking cycles before active data.
0x5D0	RO	MSA_HTOTAL_STREAM4. Tells the receiver core how many video clock cycles will occur between leading edges of the horizontal sync pulse. [15:0] – Total number of video clocks in a line of data.
0x5D4	RO	MSA_VHEIGHT_STREAM4. Total number of active video lines in a frame of video. [15:0] – The vertical resolution of the received video.
0x5D8	RO	MSA_VSPOL_STREAM4. Specifies the vertical sync polarity requested by the transmitter. [0] – A value of 1 in this register indicates an active-High vertical sync, and a '0' indicates an active-Low vertical sync.
0x5DC	RO	MSA_VSWIDTH_STREAM4. The transmitter uses this value to specify the width of the vertical sync pulse in lines. [14:0] – Specifies the number of lines between the leading and trailing edges of the vertical sync pulse.
0x5E0	RO	MSA_VSTART_STREAM4. This main stream attribute specifies the number of lines between the leading edge of the vertical sync pulse and the first line of active data. [15:0] – Number of blanking lines before the start of active data.
0x5E4	RO	MSA_VTOTAL_STREAM4. Total number of lines between sequential leading edges of the vertical sync pulse. [15:0] – The total number of lines per video frame is contained in this value.

Table 2-34: **MST MSA Values *(Cont'd)***

Offset	R/W	Definition
0x5E8	RO	MSA_MISC0_STREAM4. Contains the value of the MISC0 attribute data. [7:5] – COLOR_DEPTH: Number of bits per color/component. [4] – YCbCR_COLOR: Set to 1 (ITU-R BT709-5) or 0 (ITU-R BT601-5). [3] – DYNAMIC_RANGE: Set to 1 (CEA range) or 0 (VESA range). [2:1] – COMPONENT_FORMAT: 00 = RGB 01 = YCbCr 4:2:2 10 = YCbCr 4:4:4 11 = Reserved [0] – CLOCK_MODE: 0 = Synchronous clock mode 1 = Asynchronous clock mode
0x5EC	RO	MSA_MISC1_STREAM4. Contains the value of the MISC1 attribute data. [7] – Implements the attribute information contained in the DisplayPort MISC1 register described in section 2.2.4 of the standard. [6:3] – RESERVED: These bits are always set to 0. [2:1] – STEREO_VIDEO: Used only when stereo video sources are being transmitted. See the *DisplayPort Standard v1.1a* section 2.24 for more information. [0] – INTERLACED_EVEN: 1 indicates that the number of lines per frame is an even number.
0x5F0	RO	MSA_MVID_STREAM4. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_NVID registers. [23:0] – MVID: Value of the clock recovery M value.
0x5F4	RO	MSA_NVID_STREAM4. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_MVID registers. [23:0] – NVID: Value of the clock recovery N value.
0x5F8	RO	MSA_VBID_STREAM4. The most recently received VB-ID value is contained in this register. [7:0] – VBID: See Table 2-3 (p44) in the *DisplayPort Standard* for more information. The default value is 0x19.
0xA00 to 0xAFF	RO	DOWN_REQUEST_BUFFER. Down Request Buffer address space. User has to read sideband message request from the address 0xA00 – 0xA30. The rest of the address space is reserved.
0xB00 to 0xBFF	WO	DOWN_REPLY_BUFFER. Down Reply Buffer address space. User has to write sideband message reply in the address starting from 0xB00 for every new reply. Reply Buffer can handle up to 32 bytes. The rest of the address space is reserved.
0xC00 to 0xCFF	RO	UPSTREAM_REQUEST_BUFFER. Reserved.

Table 2-34: **MST MSA Values *(Cont'd)***

Offset	R/W	Definition
0xD00 to 0xDFF	WO	UPSTREAM_REPLY_BUFFER. Reserved.
0x800 to 0x8FF	RW	PAYLOAD_TABLE. This address space maps to the VC Payload table that is maintained in the core. Write access is provided when VCPayload table is in software control.

Vendor-Specific DPCD

Table 2-35: **Vendor-Specific DPCD**

Offset	R/W	Definition
0xE00 to 0xEFC	RW	SOURCE_DEVICE_SPECIFIC_FIELD. User access to source specific field of DPCD address space. AXI accesses are all word-based (32 bits). 0xE00 to 0xE02: Read Only (IEEE OUI Value programmed by source) 0xE03 to 0xEFF: Write/Read
0xF00 to 0xFFC	RW	SINK_DEVICE_SPECIFIC_FIELD. User access to sink specific field of DPCD address space. AXI accesses are all word-based (32 bits). 0xF00 to 0xF02: Read Only (IEEE OUI Value from GUI) 0xF03 to 0xFFF: Write/Read

AXI IIC Registers

For details about the AXI IIC registers, see the *AXI IIC Product Guide* (PG090) [Ref 5].

HDCP Registers

For details about the HDCP registers, see the *HDCP Product Guide* (PG224) [Ref 4].

AXI Timer Registers

For details about the AXI Timer registers, see the *AXI Timer Product Guide* (PG079) [Ref 6].

Designing with the Subsystem

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

DisplayPort Overview

The Sink core requires a series of initialization steps before it begins receiving video. These steps include bringing up the Physical Interface (PHY) and setting the internal registers for the correct management of the AUX channel interface.

The Sink policy maker in the example design provides the basic steps for initialization. The following Sink registers are recommended to program after power up:

- Override LINK_BW_SET
- Override LANE_COUNT_SET
- Override DPCD DOWNSPREAD
- Sink Device Count

These values indicate key DPCD capabilities of sink.

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

While the Display Timing Generator might be enabled at any time, Xilinx recommends keeping the DTG disabled until the receiver core policy maker detects the start of active video. This condition can be detected initially through the assertion of the MODE_INTERRUPT which detects the change in the vertical and horizontal resolution values.

Upon receipt of the interrupt, the receiver policy maker should verify the values of the Main Stream Attributes (offset 0x500–0x530) to ensure that the requested video mode is within the range supported by the sink device. If these values are within range, the Display Timing Generator should be enabled to begin passing valid video frames through the user data interface.

DP159 Retimer

Xilinx expects the use of the TI DP159 Retimer along with the DisplayPort RX solution. The DP159 as a retimer provides better SI features. As a retimer, the DP159 removes the random and ISI jitter from video source. The DP159 configuration is controlled through an I2C interface. The DisplayPort RX design needs an external I2C controller to configure the DP159 Retimer. For more details, refer to the *SNx5DP159_Product_Preview* [Ref 7].

Note: The IIC controller needs to work at 400 kHz or higher speed.

For details on reference clock requirements and its connectivity, see Clocking.

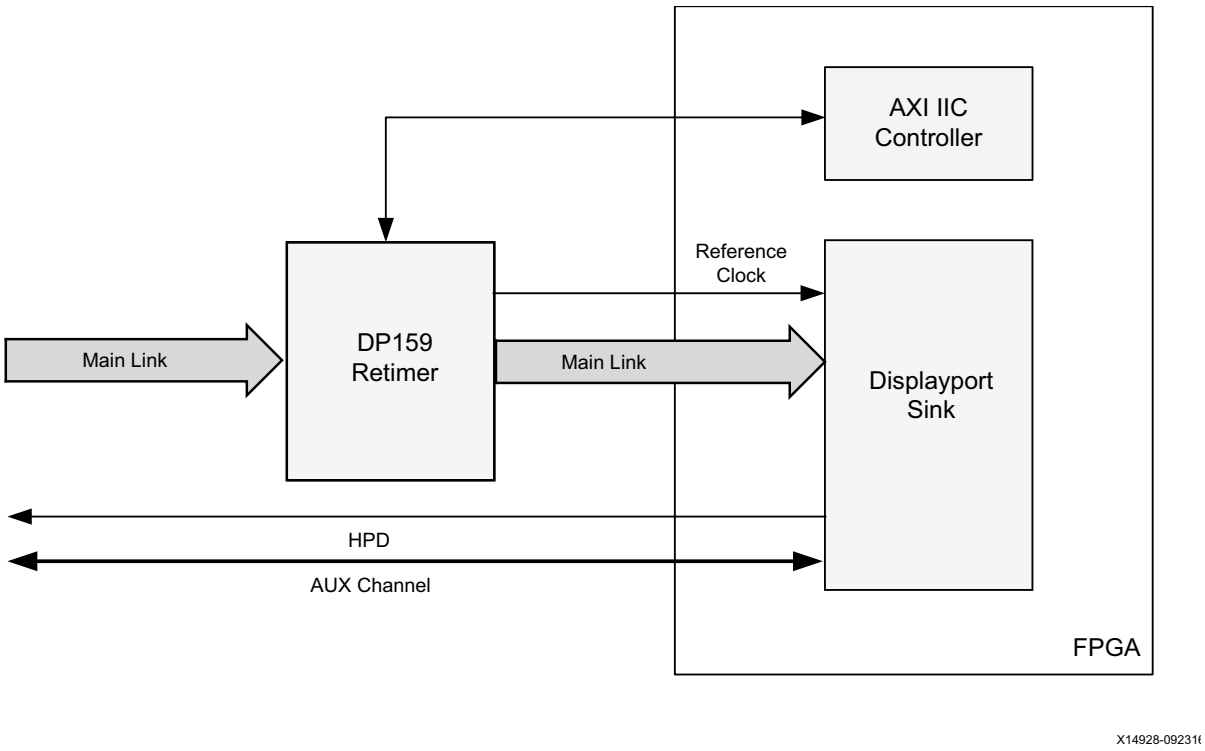


Figure 3-1: **DP159 Retimer**

DP159 Retimer Register Programming

The DP159 does not monitor the AUX transactions between the DisplayPort source and DisplayPort sink during link training, as a typical DisplayPort retimer would. Because of this the DisplayPort software driver handles the required DP159 configuration.

For more details on the programming sequence or the DP159 register details, see *DP159 as DisplayPort Retimer* [Ref 8].

Note: You do not have to control the DP159 retimer because the DisplayPort software driver handles the programming.

The sequences of the DP159 retimer, handled in the DisplayPort driver, are summarized in the following sections:

- DP159 Initialization
- TP1 Interrupt Handler (TP1_Handler)
- TP23 Interrupt Handler (TP23_Handler)
- DP159 Re-Initialization

DP159 Initialization

The following registers must be programmed immediately after power-up to configure the DP159 PLL, TX and RX blocks.

Table 3-1: **DP159 Initialization**

IIC Address	Write/Read	Data	Description
0xFF	Write	0x00	Select Page 0
0x09	Write	0x36	Enable X-mode
0x0A	Write	0x7B	Disable HPD_SNK pass-through to HPD_SRC.
0x0D	Write	0x80	Enable Clock on AUX. Select 1/20 mode.
0x0C	Write	0x6D	Set TX Swing to Maximum
0x10	Write	0x00	Turn off pattern verifier
		0x11	Len = PRBS23, Sel = PRBS mode to turn off char-alignment
0x16	Write	0xF1	Disable char-alignment on all lanes
0xFF	Write	0x01	Select Page 1
Configure PLL			
0x00	Write	0x02	Enable Band Gap
0x04	Write	0x80	PLL_FBDIV[7:0]
0x05	Write	0x00	PLL_FBDIV[10:8]
0x08	Write	0x00	PLL_PREDIV[7:0]
0x0D	Write	0x02	Selects Lane0 for clock
0x0E	Write	0x03	CDR_CONFIG [4:0]. FIXED, LN0
0x01	Write	0x01	CP_EN is PLL mode
0x02	Write	0x3F	CP_Current is High
0x0B	Write	0x33	Loop filter to 8K
0xA1	Write	0x02	Override PLL settings
0xA4	Write	0x02	Override PLL settings

Table 3-1: **DP159 Initialization *(Cont'd)***

IIC Address	Write/Read	Data	Description
Configure TX Block			
0x10	Write	0xF0	Disable for all four TX lanes
0x11	Write	0x30	TX_RATE is Full Rate, TX_TERM = 75 to 150, TX_INVPAIR = None
0x14	Write	0x00	HDMI_TWPST1 is 0dB de-emphasis
0x12	Write	0x03	SLEW_CTRL is Normal, SWING is 600 mV.
0x13	Write	0xFF	FIR_UPD. Load TX settings
0x13	Write	0x00	
Configure RX Block			
0x30	Write	0XE0	Disable Receivers except lane0
0x32	Write	0x00	PD_RXINT
0x31	Write	0x00	RX_Rate is full
0x4D	Write	0x08	EQFTC = 0 and EQLEV = 8
0x4C	Write	0x01	Enable Fixed EQ
0x34	Write	0x01	Enable Offset Correction
0x32	Write	0xF0	Load RX settings
0x32	Write	0X00	
0x33	Write	0xF0	Load EQ settings.
0xFF	Write	0x00	Select Page0
0x0A	Write	0X3B	Enable HPD_SNK pass thru to HPD_SRC. Retimer
0xFF	Write	0x01	Select Page1

TP1 Interrupt Handler (TP1_Handler)

The GPU must inform the sink of the link bandwidth (LINK_BW_SET) and the number of lanes (LANE_COUNT_SET) before beginning the link training. At this stage, the software must program the PLL enable and number of RX lanes of DP159. When PLL lock has been achieved, the software must immediately transition the PLL mode of operation from PLL_MODE to PD_MODE. It is also important to enable the TX lanes, at this stage, so that the DisplayPort sink can start performing the clock recovery.

Table 3-2: **TP1 Interrupt Handler**

Address	Read/Write	Data	Description
Bandwidth and Number of Lanes			
0x00	Write	0x02	Enable Bandgap, DISABLE PLL, clear A_LOCK_OVR
0x01	Write	0x01	CP_EN = PLL (reference) mode
0x0B	Write	0x33	Set PLL control

Table 3-2: **TP1 Interrupt Handler *(Cont'd)***

Address	Read/Write	Data	Description
0x02	Write	0x3F	Set CP_CURRENT
0x30	Write	0xE1 0xC3 0x0F	Set RX Lane count Lane count 1 Lane count2 Lanecount4
0x00	Write	0x03	Enable Bandgap, Enable PLL, clear A_LOCK_OVR
0x4C	Write	0x01	Enable fixed EQ
0x4D	Write	0x08 0x18 0x28	Set EQFTC and EQLEV (fixed EQ) HBR2 HBR RBR
0x10	Write	0xE1 0xC3 0x0F	Enable TX lanes Lane count 1 Lane count2 Lanecount4
0x00	Write	0x23	Enable PLL and Bandgap, set A_LOCK_OVR
Determine the PLL lock of DP159. If achieved, change PLL mode based on the lane rate. Continue programming, after the DP159 PLL lock.			
0x02	Write	0x5F 0x27 0x1F	CP_CURRENT HBR2 HBR RBR
0x0B	Write	0x30	PLL loop filter 1K
0x01	Write	0x02	CP_EN is PD mode
0xFF	Write	0x00	Select Page0
0x16	Write	0x11 0x31 0xF1 0xF1	Set DP_TST_EN per #lanes, latch FIFO errors Lane Count1 Lane count 2 Lane count4 Set DP_TST_EN on all lanes to disable char-alignment
0x10	Write	0x00	Disable PV
0xFF	Write	0x01	Select Page1

TP23 Interrupt Handler (TP23_Handler)

Upon completing the clock recovery phase of link training, the source transitions to the channel equalization phase. Once in the channel equalization phase, the software should enable adaptive equalization in DP159.

Table 3-3: **TP23 Interrupt Handler**

Address	Read/Write	Data	Description
0x4C	Write	0x03	Enable Adaptive Equalization
0xFF	Write	0x00	Select Page0
0x15	Write	0x18	Clear BERT counters and TST_INTQ latches
0x18	Read		BERT counters [7:0] read and error counters increment
0x19	Read		BERT counters[11:8] read and error counters increment
0xFF	Write	0x01	Select Page1

DP159 Re-Initialization

The PLL, RX, and TX settings of the DP159 must be re initialized while the DisplayPort Source is no longer detected. For example, on receiving the cable unplug interrupt event.

Table 3-4: **DP159 Re-Initialization**

Address	Write/Read	Data	Description
0x00	Write	0x02	Disable PLL, clear A_LOCK_OVR
0x34	Write	0x01	Enable Offset Correction
0x02	Write	0x3F	Set CP_CURRENT is high BW
0x01	Write	0x01	CP_EN is PLL mode
0x0B	Write	0x33	PLL Loop filter 8K
0x4D	Write	0x08	EQFTC = 0 and EQLEV = 8
0x4C	Write	0x01	Set to Fixed EQ
0x33	Write	0xF0	Load Equalization settings
0x10	Write	0xF0	Disable all TX lanes
0x30	Write	0xE0	Enable RX Lane 0 only

DisplayPort RX Programming Sequence with DP159

The sequence for programming DisplayPort RX with DP159 is summarized in the following steps:

1. Reset the DP159.
2. Initialize interrupt handler in the system.
3. Initialize IIC controller. The IIC controller is used to program the DP159.
4. Remove reset over the DP159.
5. Initialize the DP159. For details, see the DP159 Initialization.

6. Initialize DisplayPort RX. The initialization sequence is handled by the DisplayPort RX driver.
    a. Disable the main link.
    b. Apply GT (CPLL and PHY) reset.
    c. Set the AUX clock divider (number of AXI clocks for the 1 MHz AUX clock generation).
    d. Set the RX Voltage Swing and pre-emphasis training setting (0x1445 in offset 0x214). Training requests Vswing start at level1 and fixed pre-emphasis at level1.
    e. Configure the tDLOCK period to 10 µs (Offset 0x21C).
    f. Remove the DisplayPort RX out of GT reset.
    g. Wait for PHY ready (CPLL lock and rest done status).
    h. Enable the DisplayPort receiver link.
    i. Enable the DTG.
    j. Apply and remove the soft reset.
7. DisplayPort RX receives the training pattern 1 (TP1) as the source initiates the training sequence after reading the RX capabilities. DisplayPort RX generates TP1 start interrupt.
8. TP1 interrupt handler. For details, see the TP1 Interrupt Handler (TP1_Handler).
    a. After the completion of DP159 TP1 programming, power down the unused lanes of DisplayPort based on the lane count.
    b. Apply GT (CPLL and PHY) reset.
    c. Follow the DisplayPort reset sequencing and remove the reset.
9. DisplayPort RX receives training pattern 2 (TP2)/ training pattern 3 (TP3), as the source initiates the training sequence based on RX DPCD capabilities once the clock recovery is complete. DisplayPort RX generates TP2/TP3 start interrupt.
10. TP23 interrupt handler. For details, see the TP23 Interrupt Handler (TP23_Handler).
11. Monitor the training done or the training lost interrupts for the training status.
12. In case of a cable unplug, re-initialize the DP159 by following the re-initialize sequence details provided in DP159 Re-Initialization.

A graphical representation of the DisplayPort RX programming sequence with DP159 is shown in Figure 3-2.

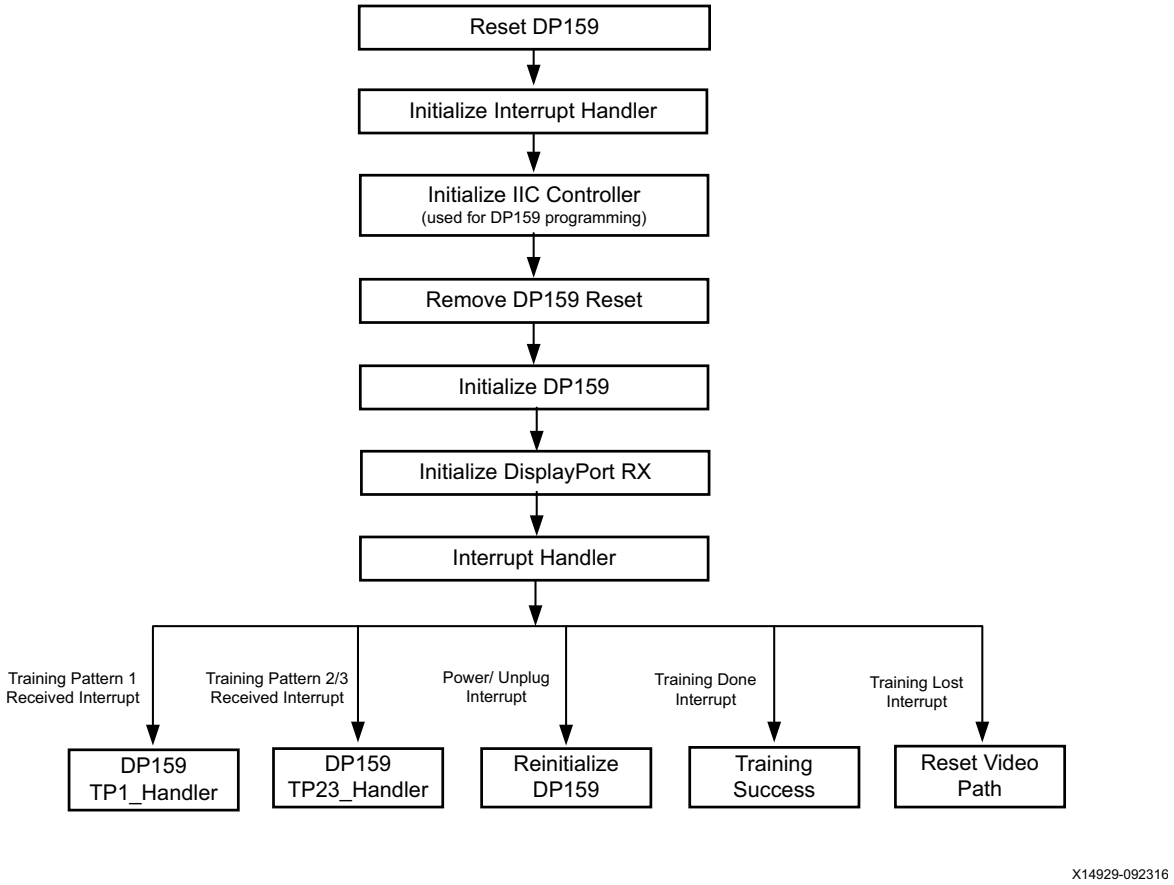


Figure 3-2: **DisplayPort RX Programming Sequence with DP159**

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 monitors the PHY and detects the specified pattern. Care must be taken to place the Sink core into the correct link training mode before the source begins sending the training pattern. Otherwise, unpredictable results might occur.

The link training process is specified in section 3.5.1.3 of the *DisplayPort Standard v1.2a* [Ref 9].

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

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

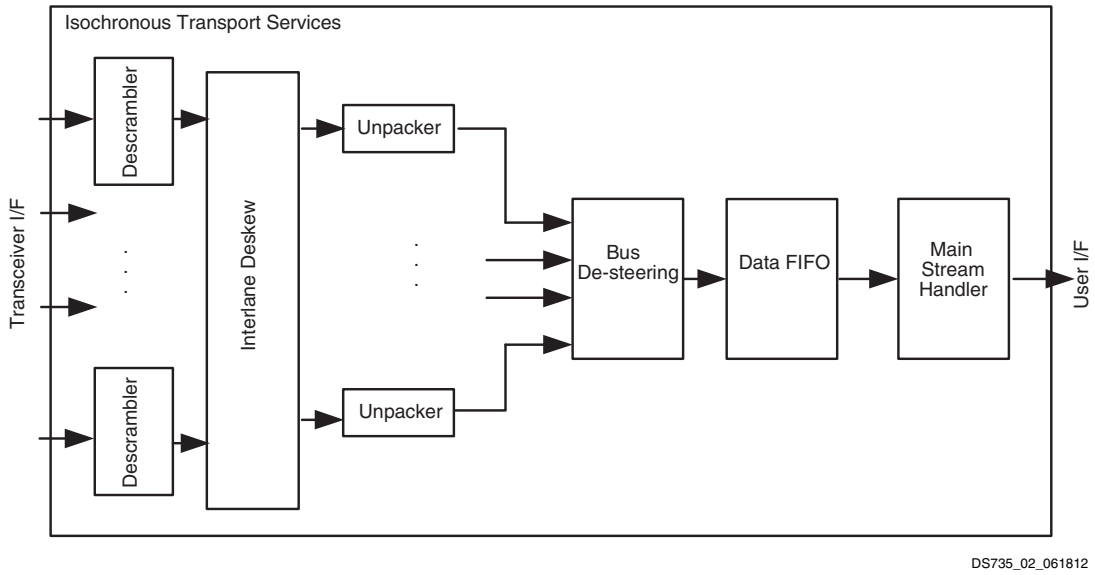


Figure 3-3: **Sink Main Link Datapath**

Figure 3-4 shows the flow diagram for link training.

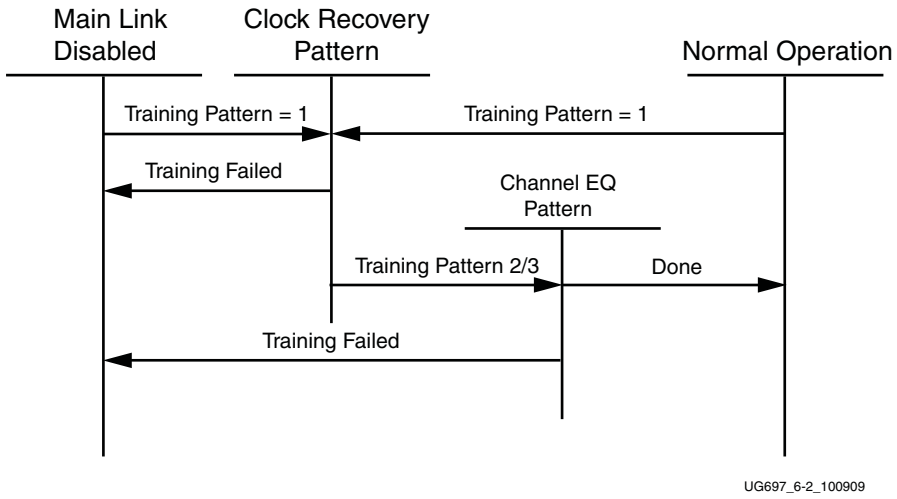


Figure 3-4: **Link Training States**

Receiver Clock Generation

This section describes the frame buffer and non-frame buffer designs.

Frame Buffer

With a frame buffer, you can generate a clock that is equal to or faster than the video clock to clock the user interface into a frame buffer.

This is the Xilinx implemented solution as it does not require any external clock management.

Non-Frame Buffer

For non-frame buffer designs, the DisplayPort receiver core requires the generation of a video stream using the M and N values within the Main Stream Attributes to reconstruct an accurate stream clock. The DisplayPort Receiver core places this information on dedicated signals and provides an update flag to signal a change in these values. Figure 3-5 shows how to use the M and N values from the core to generate a clock. See section 2.2.3 of the *DisplayPort Standard v1.2a* [Ref 9] for more details.

RECOMMENDED: *The Xilinx MMCM is not accurate enough to be used to regenerate the necessary clock for non-frame buffer design. You need to use an external PLL that meets the requirements of the DisplayPort Standard. See section 2.2.3 of the DisplayPort Standard v1.2a [Ref 9] for more details.*

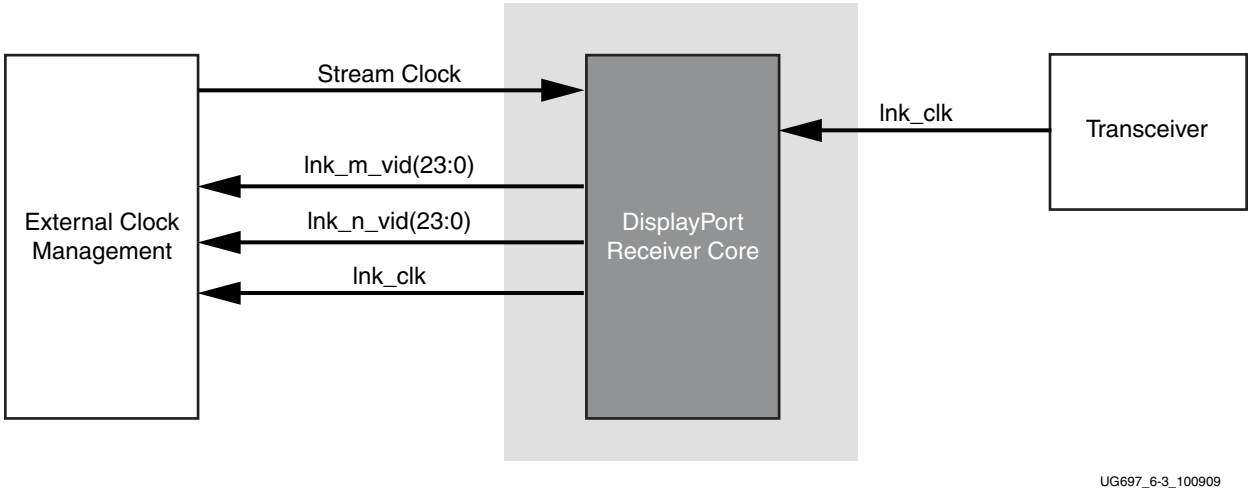


Figure 3-5: **Receiver Clock Generation**

Common Event Detection

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

Transition from Video to No Video

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

Transition from No Video to Video

The transmission of video after a NO_VIDEO interrupt can be detected by the VERTICAL_BLANKING interrupt. Upon the reception of a VERTICAL_BLANKING interrupt, if disabled, you might then re-enable the display timing generator.

Mode Change

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

Cable is Unplugged, Lost Training

When a cable becomes unplugged or training is lost for any reason, the TRAINING_LOST interrupt will occur. At this point, video data and MSA values are unreliable.

When the cable is plugged in again, no action is required from you; the core properly resets itself and applies HPD. In a scenario, where the cable is plugged in but the training is lost, the software is expected to assert a HPD upon the occurrence of a TRAINING_LOST interrupt, so that the source can retrain the link.

Link is Trained

You can determine that the core is properly training by reading from the LANE_STATUS register and observing lane alignment and symbol lock on all active lanes. Additionally, it is advisable to ensure that the PLL is locked (per the Video PHY Controller) and reset is complete, which is also part of the PHY_STATUS register.

Secondary Channel

The current version of the DisplayPort core supports eight-channel audio. The DisplayPort Audio IP core is offered as modules to provide flexibility to modify the system as needed.

As shown in Figure 3-6, the audio interface to the DisplayPort core is defined using the AXI4-Stream interface.

Audio data and secondary packets are received from the main link and stored in internal buffers of the DisplayPort Sink core. The AXI4-Stream interface of the DisplayPort core transfers audio samples along with control bits. The DisplayPort Sink should never be back pressured.

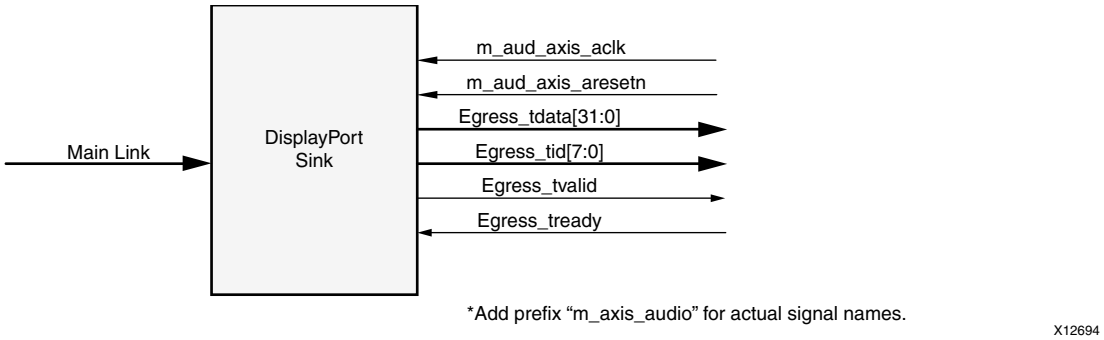


Figure 3-6: **Audio Data Interface of DisplayPort Sink System**

Multi Channel Audio

The DisplayPort receiver captures the audio data received over the link and sends it over AXI4-Stream interface, along with the channel ID (`TID[3:0]`) based on the number of channels and the speaker allocation. Stream ID received over the Info frame is also sent over the `TID[7:4]`. Samples for unallocated channels will be dropped in DisplayPort receiver.

Audio Management

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

Programming DisplayPort Sink

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

Re-Programming Sink Audio

1. Look for MUTE status by polling VB-ID.
2. When MUTE bit is set, disable audio in DisplayPort Receiver.
3. Wait for some time (in µs) or wait until MUTE bit is removed.
4. Enable audio in DisplayPort receiver.

Reading Info/Ext Packet

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

Poll Mode

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

Interrupt Mode

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

Audio Clocking (Recommended)

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

The system should have a clock generator (preferably programmable) to generate 512 × fs (audio sample rate) clock frequency based on MAUD and NAUD values. External clock source is preferred for better precision.

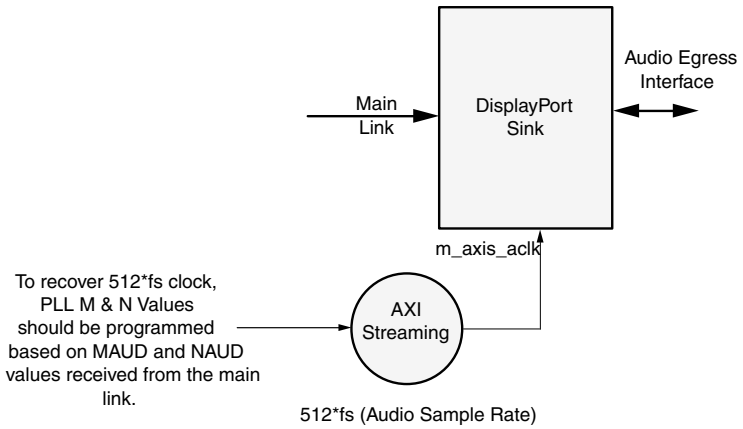


Figure 3-7: **Sink: Audio Clocking**

Sampling Frequencies

The DisplayPort RX Subsystem with a GT data width of 16-bit mode supports up to 8-channels of audio with maximum supported sampling frequency of 192 kHz for all link rates.

The DisplayPort RX Subsystem with a GT data width of 32-bit has limitations in the maximum supported sampling frequency rate depending on link rates. This limitation is due to the **`lnk_clk`** frequency reduction in 32-bit GT data width mode. Table 3-5 shows the maximum sampling rates with support up to eight channels.

Table 3-5: **Maximum Sampling Frequencies**

Link Rate (Gb/s)	Sampling Frequency (kHz)
5.4	192.0
2.7	176.4
1.62	96.0

Programming the Core in MST Mode

This section includes details about programming the Sink core in MST mode.

Enabling MST

To enable MST functionality, perform link bring-up and enable MST capability in MST Capability register (0x0D0). The Source device enables the MST after payload allocation and ACT event process is done.

MST AUX Messaging

Perform the following steps to program MST AUX Messaging:

1. Wait for DOWN_REQUEST_BUFFER_READY status in interrupt, and read from DOWN_REQUEST_BUFFER. Continue to collect side-band messages as per *DisplayPort Standard v1.2a Section 2.1.11.9*. After a complete sideband message is received, the software processes the message and writes the reply to DOWN_REPLY_BUFFER.
2. After the response is written, set DOWN Reply Buffer Message to 1 in the Remote Command New register.
3. Wait for DOWN_REPLY_BUFFER_READ status in interrupts and continue writing responses.

During the MST AUX messaging phase, the required PBN (available BW) is calculated and sent to the source. The source then sends allocation requests based on available bandwidth. Internally, Sink HW updates the VC Payload Table by monitoring the AUX transactions. Alternatively, if you are an advanced user, you can use a software control to the VC Payload

Table (Setting the bit 1 in 0x0D0 enables it), with which the software maintains a VC Payload manager (by monitoring the interrupt bit 28 of 0x014 register and 0x06C register) and writes the resulting stream allocations to 0x800-0x8FF. Once the software finishes writing to the VC payload table, software has to set bit 4 in 0x0D0.

Interrupt

For an interrupt event, read both Interrupt Cause and Interrupt Cause 1 registers.

IMPORTANT: *The software is required to form appropriate LINK_ADDRESS sideband reply as per the Message Transaction protocol given in Section 2.11.2 of DisplayPort Standard v1.2a specification* [Ref 9]. *The LINK_ADDRESS reply helps the source to identify the topology of the sink.*
For example, if the sink core is configured for 4 MST streams, it receives the multi-stream input from the DisplayPort TX and outputs four individual streams in native video format. In this case, the LINK_ADDRESS_REPLY can be modeled to contain 1 input and 4 output logical ports, with their DisplayPort device plug status set as 1 and Peer Device Type set as 3.

Reduced Blanking

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

For the CVT standard, RB/RB2 resolutions end of the line reset need to be disabled by setting the corresponding bit in the Line Reset Disable register (0x008 for the receiver). For the Non-CVT reduced blanking resolutions, where HRES is non multiple of 8, end of line reset is required to clear extra pixels in the video path for each line.

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

EDID I2C Speed Control

The DisplayPort source can control the EDID I2C speed by programming the I2C Speed Control DPCD register (0x00109) through AUX commands. The DisplayPort Subsystem supports up to 1 Mb/s I2C speed.

Clocking

This section describes the link clock (**rx_lnk_clk**), video clock (**rx_vid_clk**) and video bridge AXI4-Stream master interface clock. The **rx_vid_clk** should be 150 MHz or higher and the **m_axis_aclk_stream<n>** can be equal or greater than the **rx_vid_clk**.

The **rx_lnk_clk** is a link clock input to the DisplayPort RX Subsystem generated by the Video PHY (GT). The frequency of **rx_lnk_clk** is **<line_rate>**/40 MHz for the 32-bit video PHY(GT) data interface.

In 16-bit GT interface **hdcp_ext_clk** has to be provided by the user from external MMCM. The frequency requirement of **hdcp_ext_clk** is **rx_lnk_clk/2**.

Table 3-6 shows the clock ranges.

Table 3-6: **Clock Ranges**

Clock Domain	Min (MHz)	Max (MHz)	Description
rx_lnk_clk	40	270	Link clock
rx_vid_clk	150	200(1)	Video clock
s_axi_aclk	25	135	Host processor clock

Notes:

1. 200 MHz is the maximum frequency tested and should cover most of the cases. However "rx_vid_clk" can exceed 200 MHz provided the design meets timing.

The core uses six clock domains:

- **lnk_clk**: The **rxoutclk** from the Video PHY is connected to the RX subsystem link clock. Most of the core operates in 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.

  In the DisplayPort Sink core, **lnk_clk** is derived from the recovered clock from the transceiver. When the cable is disconnected this clock becomes unstable.

  Note: **lnk_clk** = **link_rate**/20, when GT-Data width is 16-bit. **lnk_clk** = **link_rate**/40, when GT-Data width is 32-bit.

- **vid_clk**: In MST mode, a single **rx_vid_clk** connects to all the stream video interfaces. This is the primary user interface clock. It has been tested to run up to 200 MHz, which accommodates to a screen resolution of 2560x1600 when using two-wide pixels and larger when using the four-wide pixels. Based on the *DisplayPort*

Standard, the video clock can be derived from the link clock using **mvid** and **nvid**. Also please make sure that the vid_clk frequency meets below requirement.

```
vid_clk >= (Vtotal * Htotal * fps) / Pixels per clock
```

- **s_axi_aclk**: This is the processor domain. It has been tested to run up to 135 MHz. The AUX clock domain is derived from this domain, but requires no additional constraints. In UltraScale™ devices, **s_axi_aclk** clock is connected to a free-running clock input. The **gtwiz_reset_clk_freerun_in** signal is required by the reset controller helper block to reset the transceiver primitives. A configuration parameter is added for AXI_Frequency, when the DisplayPort IP is targeted to UltraScale devices.
  The requirement is **s_axi_aclk** ≤ **lnk_clk**.
- **aud_clk**: This is the audio interface clock. The frequency is 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 Product Guide* (PG230) [Ref 1].

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.
- **rx_vid_rst**: Active-High video pipe reset. For MST with four streams, there are four video resets.
- **dp159_rst**: Active-High soft reset to the DP159 retimer generated through AXI IIC GPIO port. This reset is asserted through AXI IIC register programming for GPIO ports. For more details, see the *AXI IIC Controller Product Guide* (PG090) [Ref 5].

Address Map Example

Table 3-7 shows an example based on a subsystem base address of 0x44C0_0000 (14 bits). There are no registers in Video to AXI4-Stream bridge.

Table 3-7: **Address Map Example**

Name	SST	MST
DisplayPort RX	0x44C0_0000	0x44C0_0000
AXI IIC Controller	0x44C0_1000	0x44C0_1000
HDCP Controller	0x44C0_2000	0x44C0_2000
AXI Timer	0x44C0_3000	0x44C0_3000

Programming Sequence

For PHY related programming, see the *Video PHY Controller Product Guide* (PG230) [Ref 1].

For HDCP related programming sequence, see the *HDCP Controller Product Guide* (PG224) [Ref 4].

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 10]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 12]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 13]

Customizing and Generating the Subsystem

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

If you are customizing and generating the subsystem in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 10] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the **`validate_bd_design`** command in the Tcl console.

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

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

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 12].

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

Customizing the IP

The configuration screen is shown in Figure 4-1.

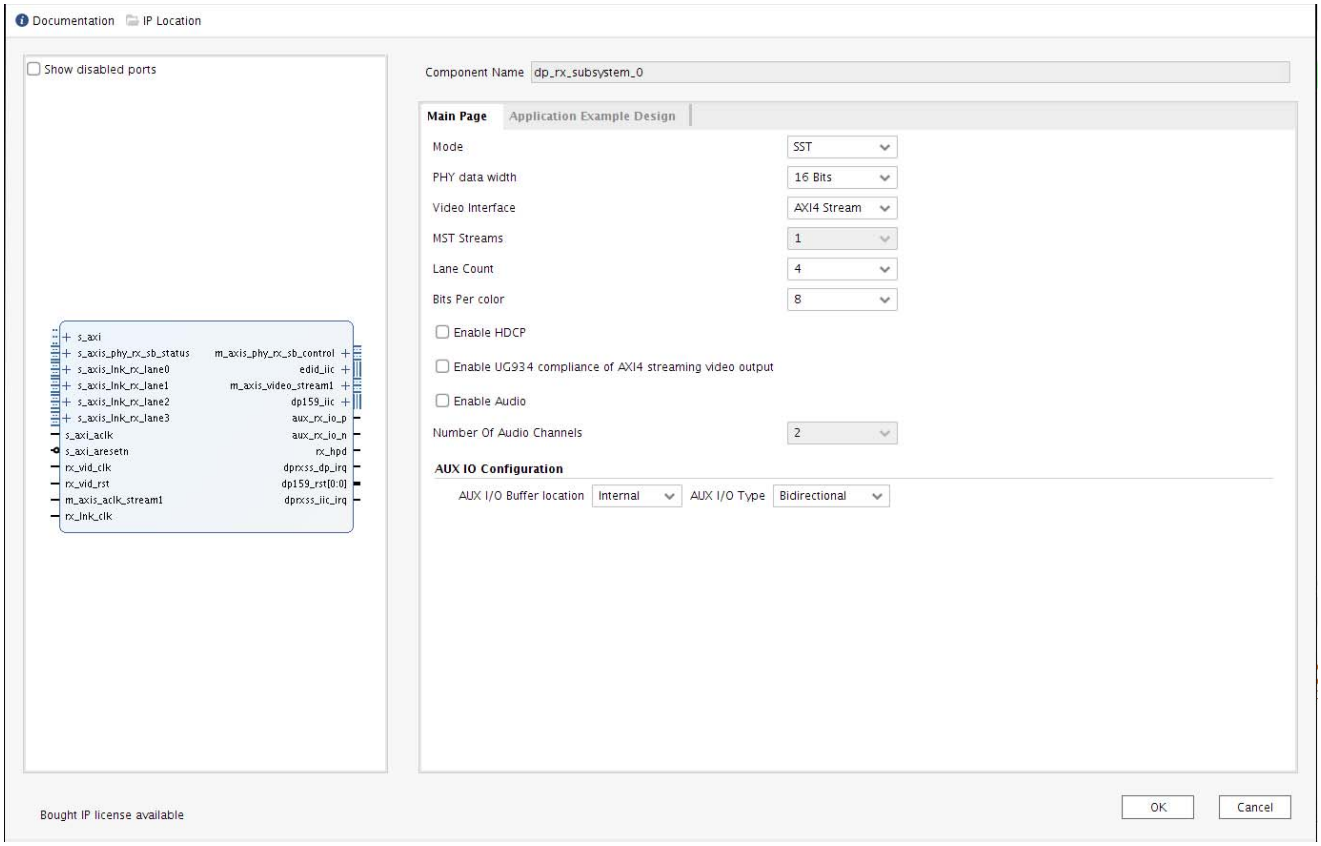


Figure 4-1: **Configuration Screen**

- **Component Name**: The Component Name is used as the name of the top-level wrapper file for the core. The underlying netlist still retains its original name. Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9, and "_". The name displayport_0 is used as internal module name and should not be used for the component name. The default is dp_rx_subsystem_n (n depends on the number of DisplayPort instances in the design. n = 0 for the first instance added to the design.).
- **Mode**: Selects the desired mode for the video stream out. Options are SST or MST.
- **PHY Data Width**: Selects 16-bit or 32-bit GT data width.
- **Video Interface**: Selects AXI4-Stream or native for the video interface.
- **Pixel Mode**: Enables when native interface is selected. Select single, dual or quad pixel mode. Maximum pixel mode supported is aligned with lane count. Pixel mode can be changed dynamically through software but this does not affect the video AXI4-Stream width.

- **MST Streams**: Enables when MST mode is selected. Selects the maximum number of streams in MST mode. This is configured through software up to the maximum value. Valid options are 2, 3, or 4.
- **Lane Count**: Selects the maximum number of lanes. Lane count can be changed dynamically through software. Valid options are 1, 2, or 4.
- **Bits Per Color**: Selects the desired maximum bit per component (BPC). Valid options are 8, 10, 12, or 16. This can be changed in software up to the maximum value.
- **Enable Audio**: Enables audio support. Available only when SST mode is selected.
- **Enable HDCP**: Enables HDCP encryption. Available only when SST mode is selected.

  Note: HDCP requires a separate license.
- **Enable UG934 Compliance of AXI4-Stream Video Output**: Select this option to have UG934-compliant streaming video output (see the *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 18]).
- **Number of Audio Channels**: Enables only when SST mode is selected and audio support enabled. Selects the number of audio channels. Valid options are 2 to 8.
- **AUX I/O Buffer location**: Selects buffer location for AUX channel. Valid options are Internal or External.
- **AUX I/O Type**: Selection of Bidirectional or Unidirectional buffer type.

User Parameters

Table 4-1 shows the relationship between the GUI fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console).

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

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
Mode	MODE	SST
PHY Data Width	PHY_DATA_WIDTH	16
Video Interface	VIDEO_INTERFACE	AXI4-Stream
Pixel Mode	PIXEL_MODE	Quad
MST Streams	NUM_STREAMS	1
Lane Count	LANE_COUNT	4
Bits Per Color	BITS_PER_COLOR	8
Enable HDCP	HDCP_ENABLE	0
Enable Audio	AUDIO_ENABLE	0
Enable UG934 Compliance of AXI4-Stream Video Output	UG934_COMPLIANCE	0
Number Of Audio Channels	AUDIO_CHANNELS	2

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
AUX IO Buffer Location	AUX_IO_LOC	Internal
AUX IO Type	AUX_IO_TYPE	Bidirectional

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11].

Constraining the Core

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

Required Constraints

There are no required constraints for this core. Being a subsystem, all sub-cores generate their own constraints and the same is applied in the subsystem.

Device, Package, and Speed Grade Selections

See Features for details about supported devices.

Clock Frequencies

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

Clock Management

There are no specific clock management constraints.

Clock Placement

There are no specific clock placement constraints.

Banking

For more information on the specific banking constraints, see the *Video PHY Controller Product Guide* (PG230) [Ref 1].

Transceiver Placement

Transceiver is external to DisplayPort RX Subsystem hence there are no specific transceiver placement constraints. For more information on the specific transceiver placement constraints, see the *Video PHY Controller Product Guide* (PG230) [Ref 1].

I/O Standard and Placement

This section contains details about I/O constraints.

AUX Channel

The *VESA DisplayPort Standard* [Ref 9] describes the AUX channel as a bidirectional LVDS signal. For 7 series designs, the core uses IOBUFDS (bidirectional buffer) as the default with the LVDS standard. You should design the board as recommended by the VESA DP Protocol Standard. For reference, see the example design XDC file.

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

For Source:

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

For Sink:

```
set_property IOSTANDARD LVDS_25   [get_ports aux_rx_io_p]
set_property IOSTANDARD LVDS_25   [get_ports aux_rx_io_n]
```

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

For Source:

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

For Sink:

```
set_property IOSTANDARD LVDS   [get_ports aux_rx_io_p]
set_property IOSTANDARD LVDS   [get_ports aux_rx_io_n]
```

HPD

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

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

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

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

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

Board design and connectivity should follow *DisplayPort Standard* recommendations with correct level shifting.

Simulation

There is no example design simulation support for the DisplayPort RX Subsystem.

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11].

Example Design

Note: All example designs use the TED DP1.2 FMC Card from inrevium (TB-FMCH-DP3).

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

RECOMMENDED: *For ZCU102 (Revision 1.0 or later), you should set up a 1.8V setting after connecting the DisplayPort FMC. See the Setting the FMC Voltage to 1.8V section. For more information, see the ZCU102 System Controller GUI Tutorial (XTP433) [Ref 19].*

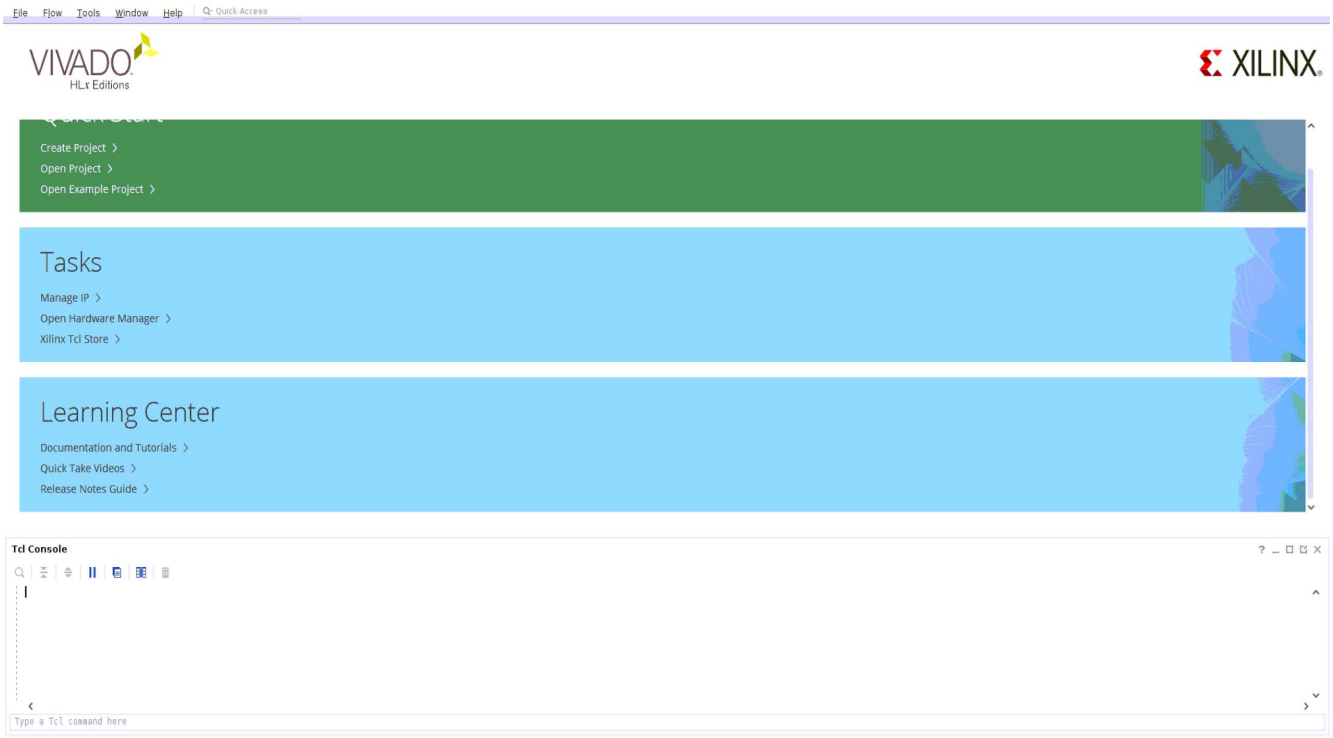
Table 5-1 shows the available example designs for the DisplayPort RX.

Table 5-1: **Available Example Designs**

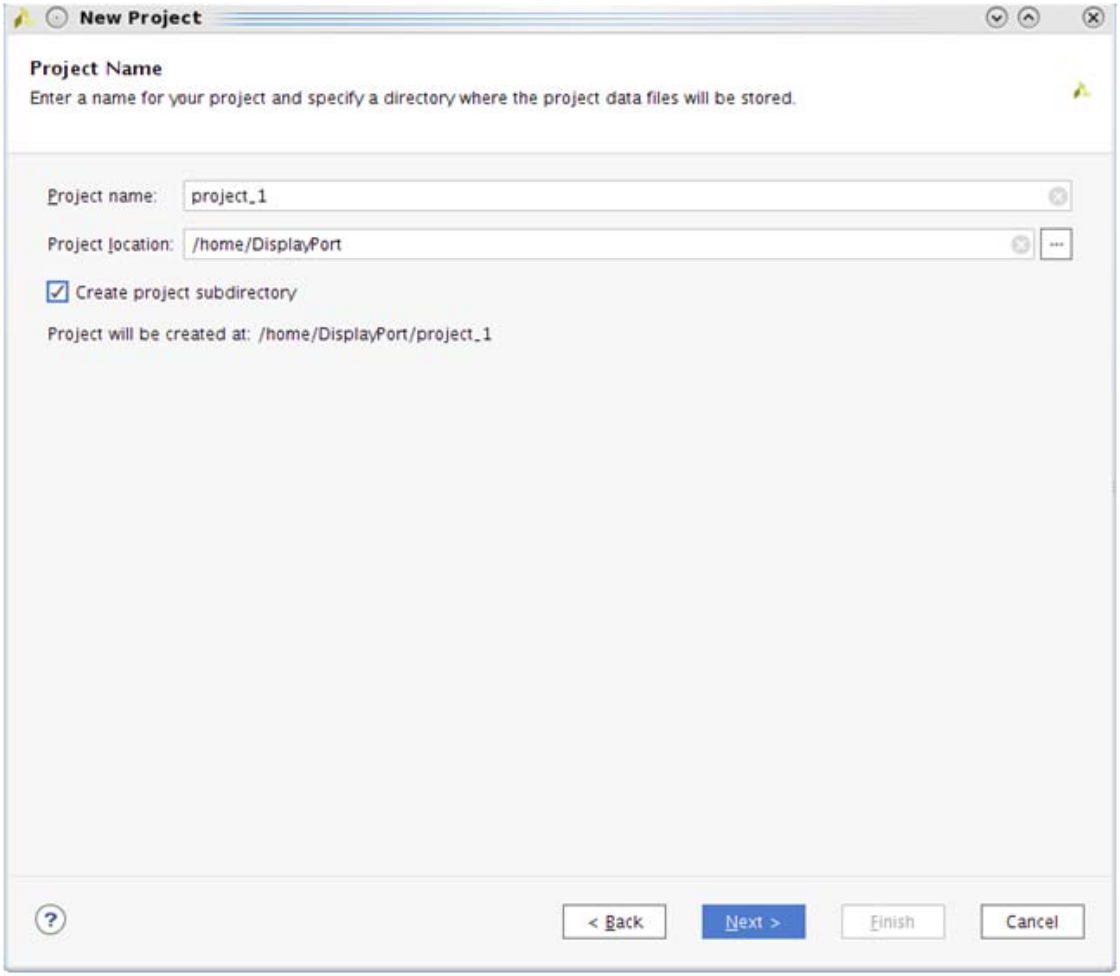
GT Type	Topology	Video PHY Config		Hardware	GT Data Width	BPC	Processor
		(TXPLL)	(RXPLL)				
GTXE2	Pass-through with HDCP1.3	CPLL	CPLL	KC705 + TED DP1.2 FMC	2-byte	10	MicroBlaze
	Pass-through without HDCP1.3	CPLL	CPLL	KC705 + TED DP1.2 FMC	4-byte	10	MicroBlaze
GTHE3	Pass-through without HDCP1.3	QPLL	CPLL	KCU105 + TED DP1.2 FMC	2-byte	10	MicroBlaze
GTHE4	RX only	–	CPLL	ZCU102 + TED DP1.2 FMC	2-byte	10	A53
	TX only	QPLL	–	ZCU102 + TED DP1.2 FMC	2-byte	10	A53

Building the Example Design

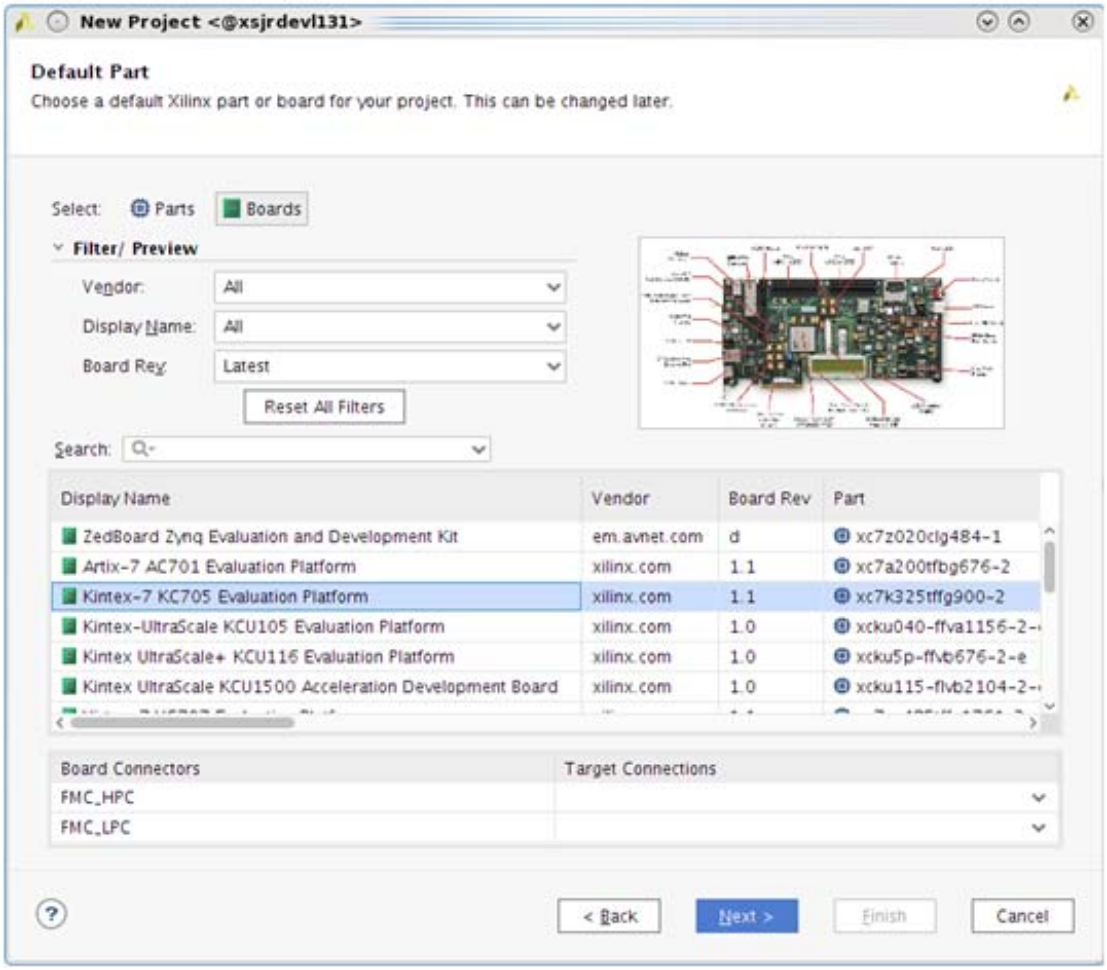
1. Open the Vivado Design Suite and click **Create Project**.



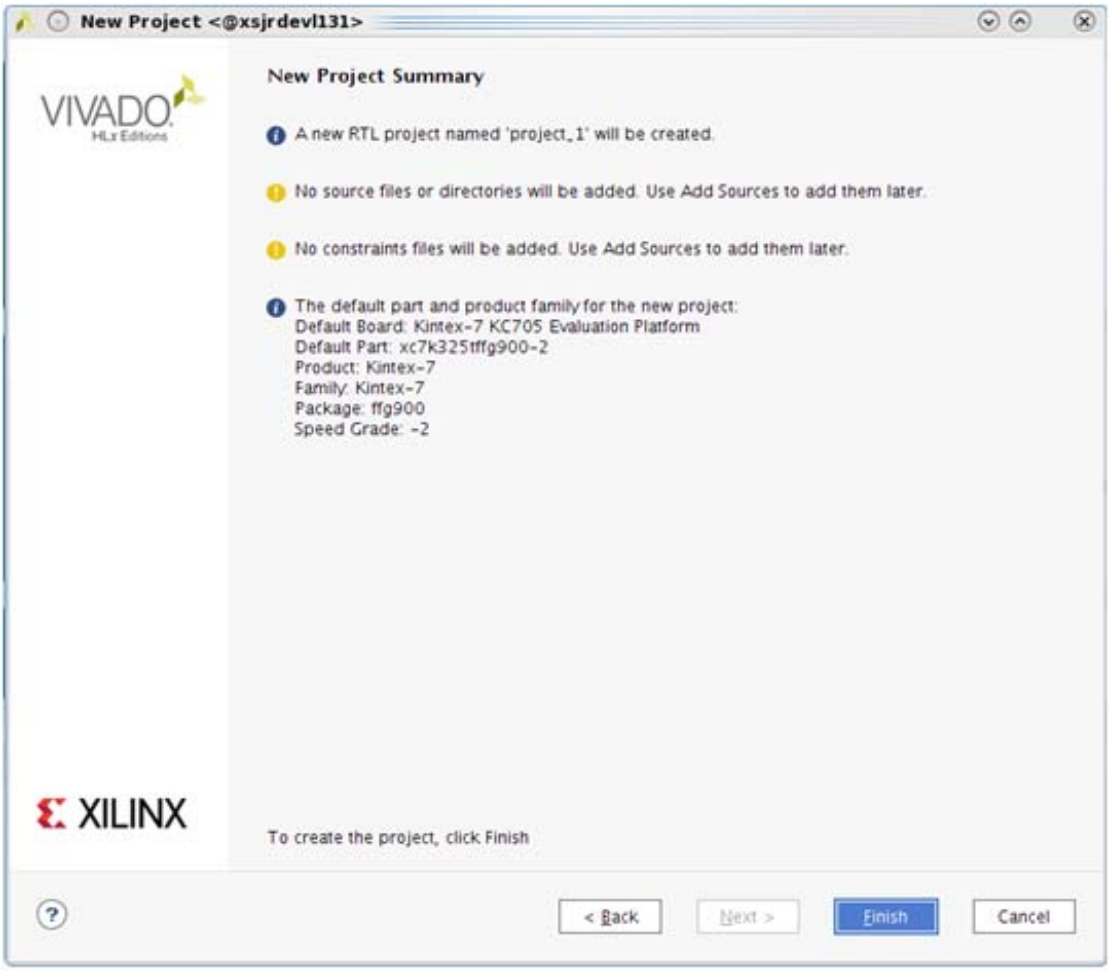
2. In the **New Project** window, enter a **Project name**, **Project location**, and click **Next** up to the Board/Part selection window.



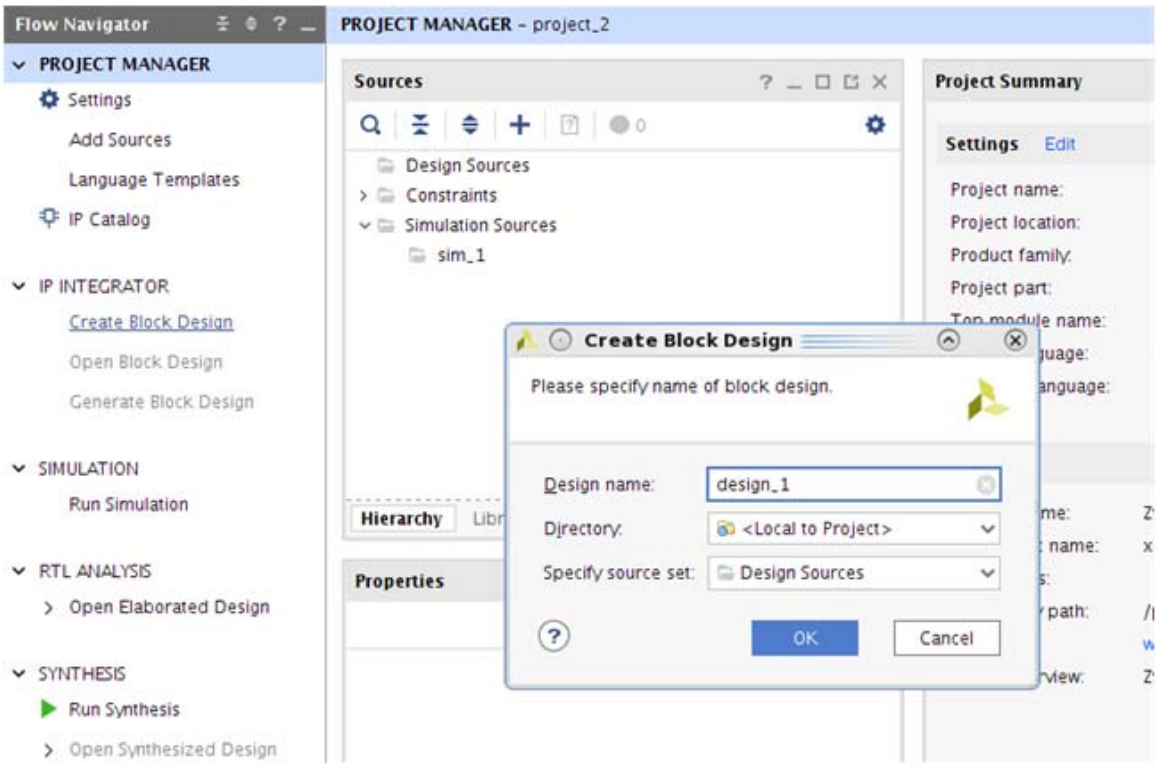
3. In the **Default Part** window, select the Board as per your requirement. Application Example Designs are available for KC705, KCU105, and ZCU102. As an example, the Kintex® -7 KC705 board is selected.



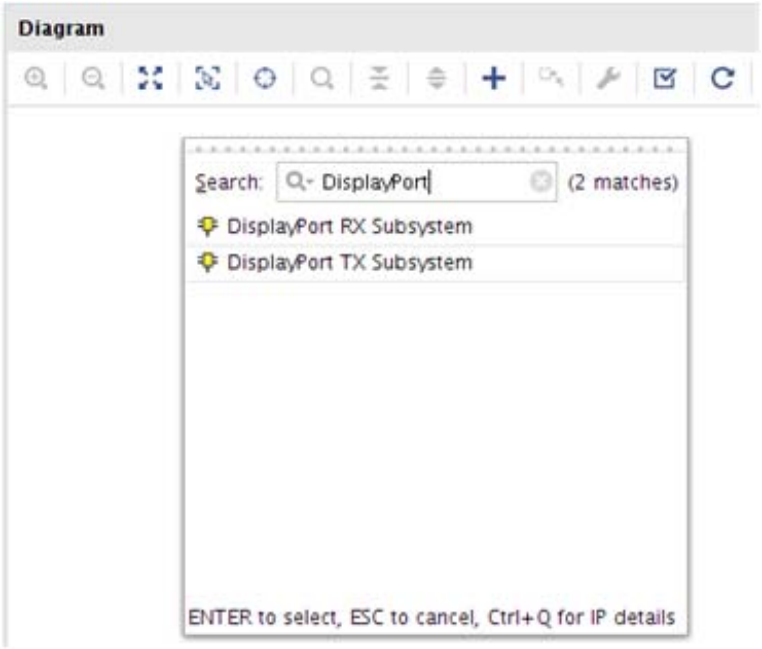
4. Click **Finish**.



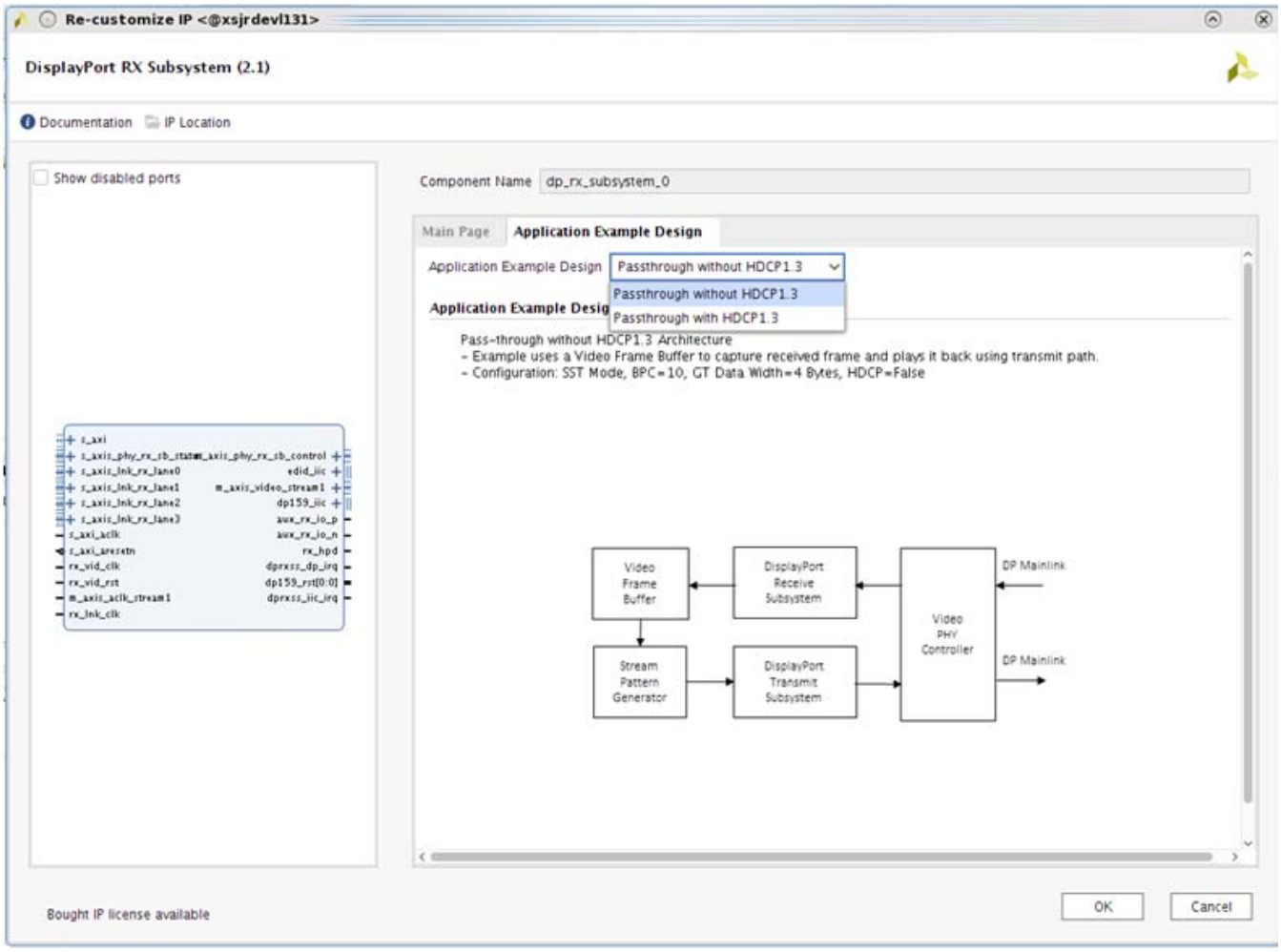
5. In the Flow Navigator, click **Create Block Design** (BD). Select a name for BD and click **OK**.



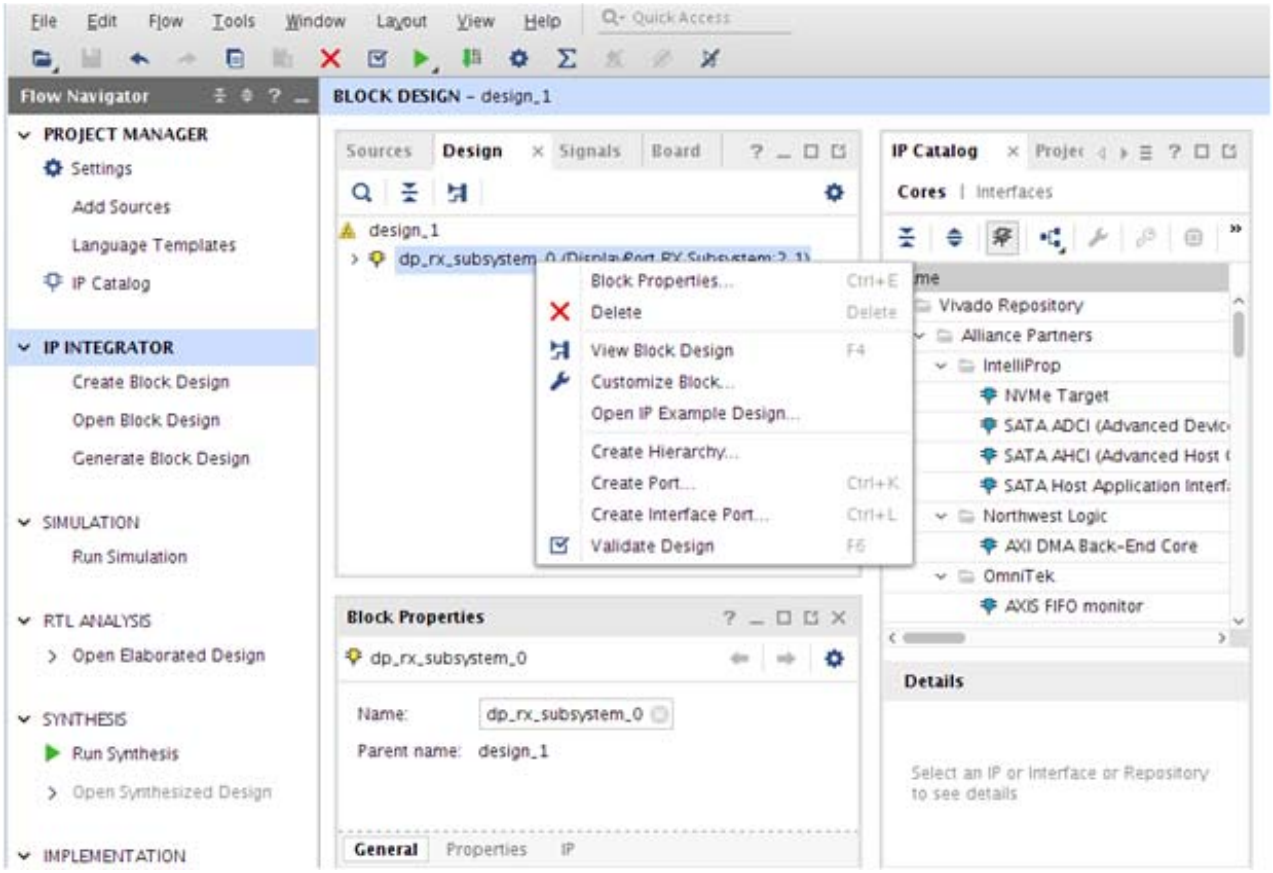
6. Right-click BD and click **Add IP**. Search for DisplayPort and select either the DisplayPort RX Subsystem IP (for RX only (ZCU102) or Pass-through (KC705, KCU105) designs) or the DisplayPort TX Subsystem IP (for TX only (ZCU102) or Pass-through (KC705, KCU105) designs).



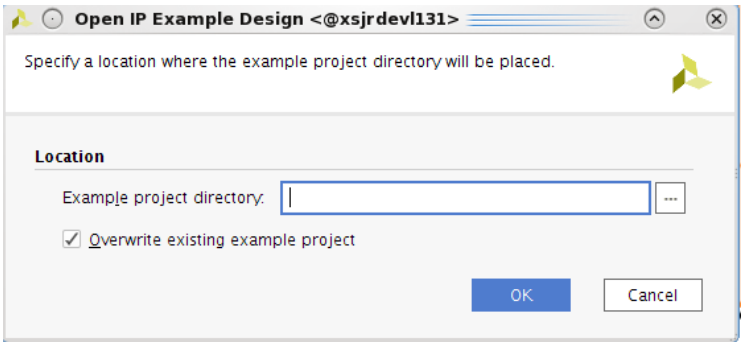
7. Double-click the IP and go to the **Application Example Design** tab in the **Customize IP** window. Select the supported topology in the **Application Example Design** drop-down box. Click **OK** and **Save** the block design.



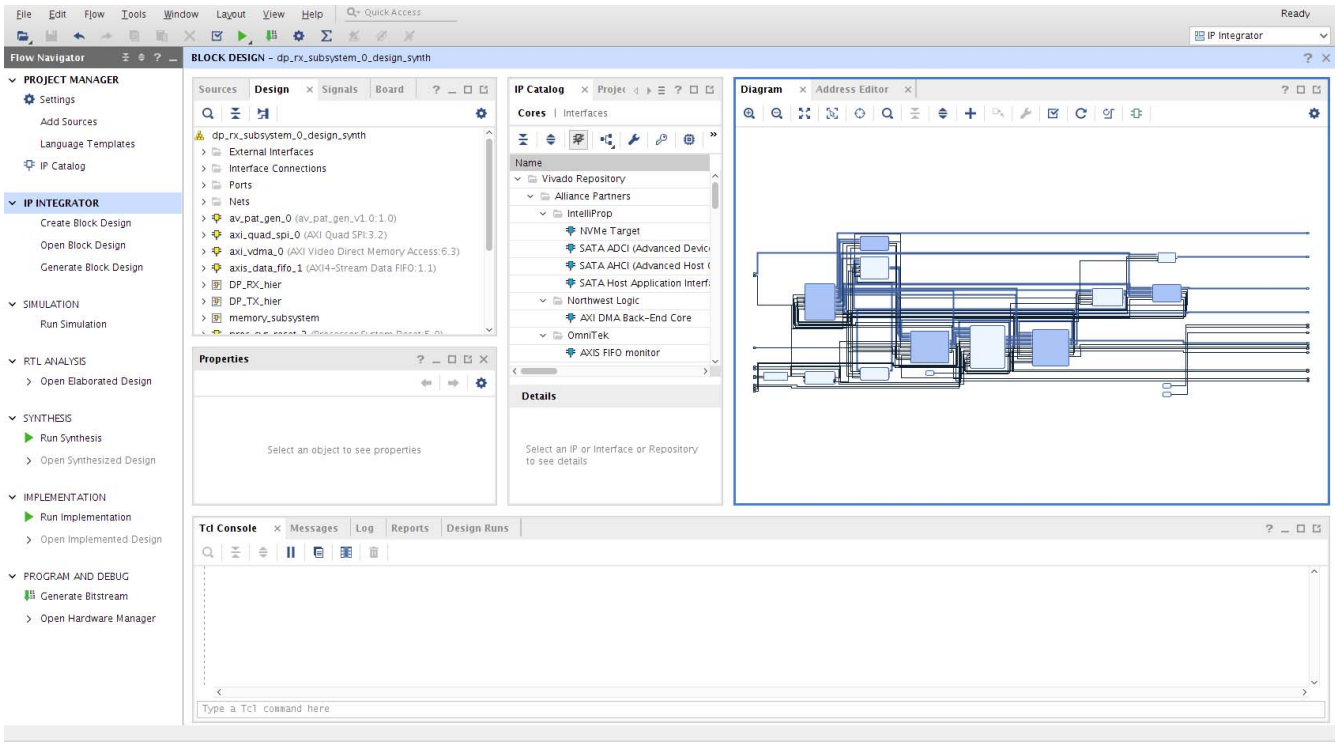
8. Right-click the **DisplayPort Subsystem** IP under Design source in the **Design** tab and click **Open IP Example Design**.



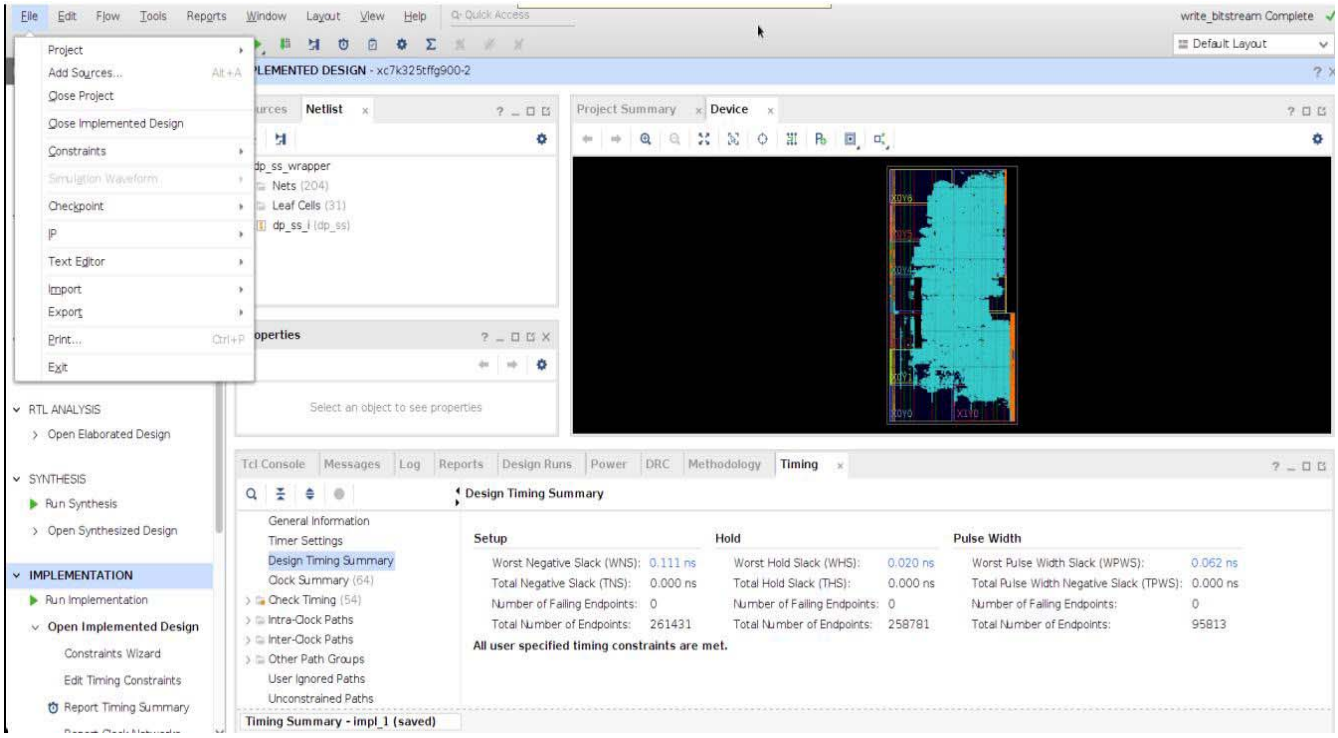
9. Choose **Example project directory** and click **OK**.



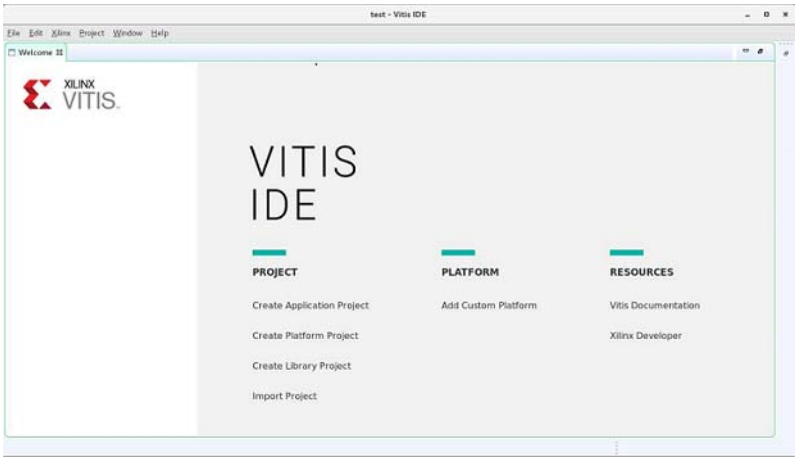
10. The following figure shows the Vivado IP integrator design. Choose the **Generate Bitstream**.



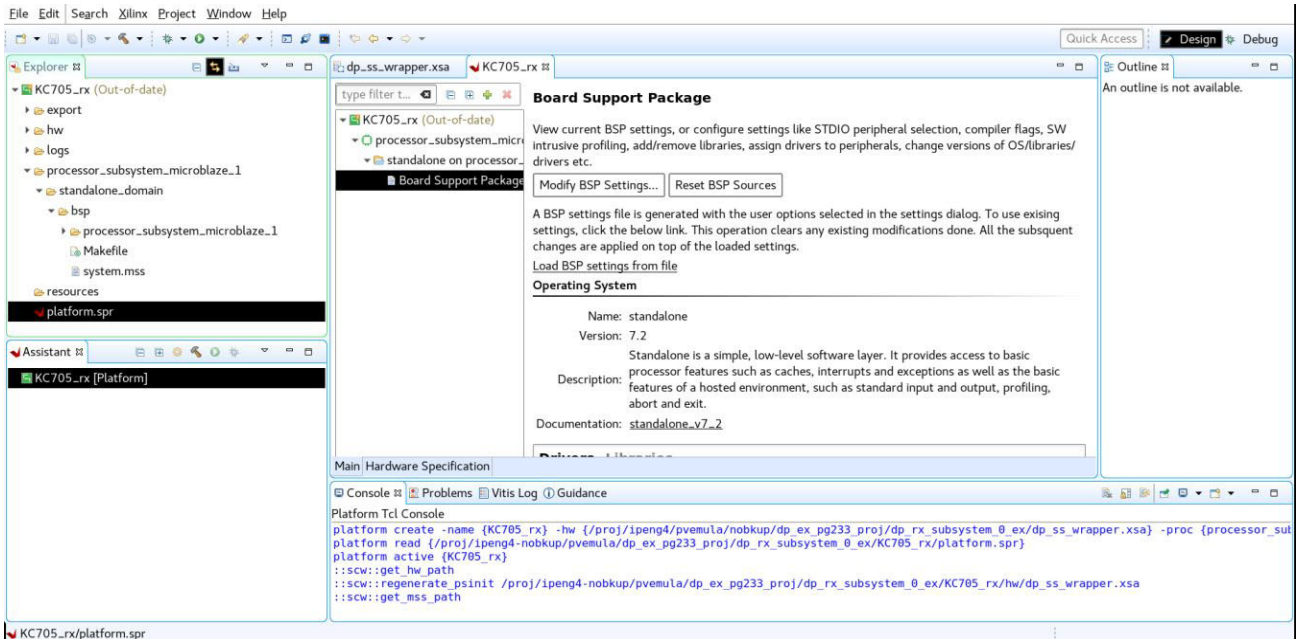
11. Export the hardware (xsa) to the Vitis™ software platform. Click **File > Export > Export Hardware**.



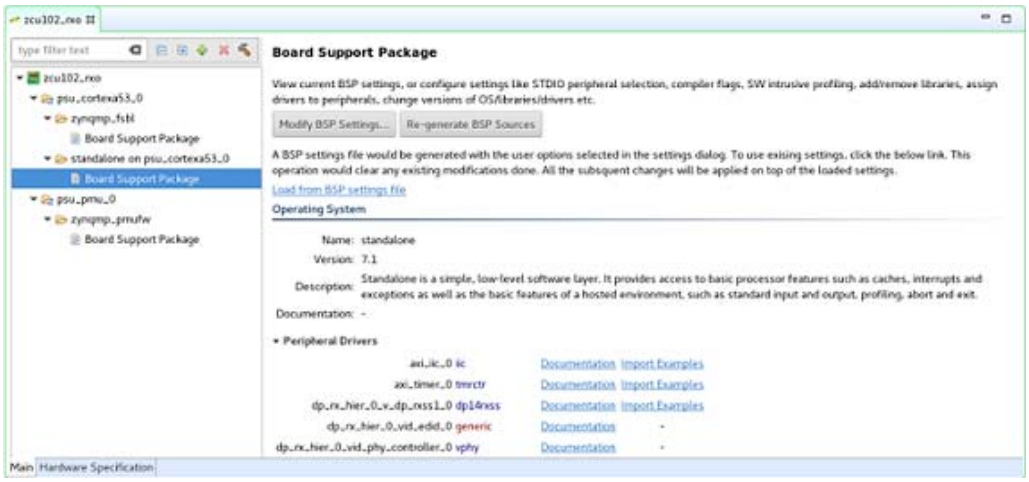
12. Launch the Vitis software platform from the command line. Set up a workspace and create a platform project using the exported xsa file.

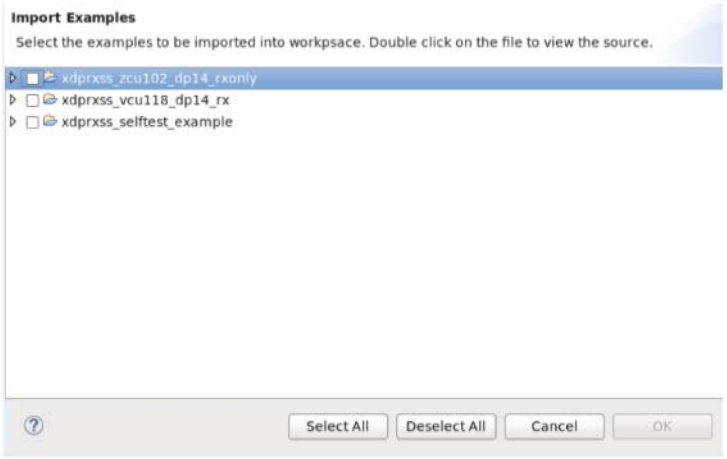


13. The following figure shows an example of when the platform is built successfully.



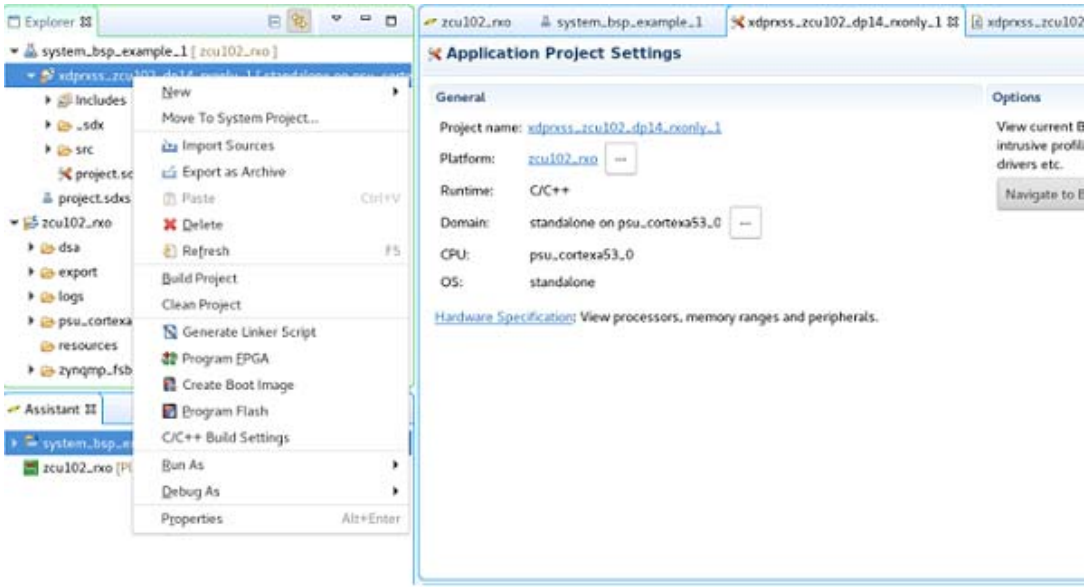
14. Click **Board Support Package** in Vitis. Click **Import Examples**.





15. Select the Example Application corresponding to your hardware:
    - For Pass-through KC705 project, select the **`*_kc705`** option.

    Note: Note: In the KC705, for HDCP Pass-through application, select the HDCP option in the Subsystem IP configuration screen before opening the Example Design.
    - For Pass-through KCU105 project, select the **`*_kcu105`** option.
    - For RX only ZCU102 project, select the **`*_zcu102_rxonly`** option in the RX subsystem driver.
    - For TX only ZCU102 project, select the **`*_zcu102_txonly`** option in the TX subsystem driver.
16. Build the example in the Vitis software platform by right-clicking and selecting **Build Project**.



Hardware Setup and Run

1. Connect the Tokyo Electron Device Limited (TED) TB-FMCH-DP3 module to the HPC FMC connector on the KC705 (or KCU105) board or to the HPC0 connector on the ZCU102 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 KC705 for serial communication. In the case of KCU105 or ZCU102, use Type A to micro B type of USB cable.
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 the pass-through or TX only applications, connect a DP cable from the TX port of the TED TB-FMCH-DP-3 module to a monitor, as shown in the following figures.
5. For the pass-through or RX only applications, connect a DP cable from the RX port of the TED TB-FMCH-DP-3 module to a DP source (GPU), as shown in the following figures.

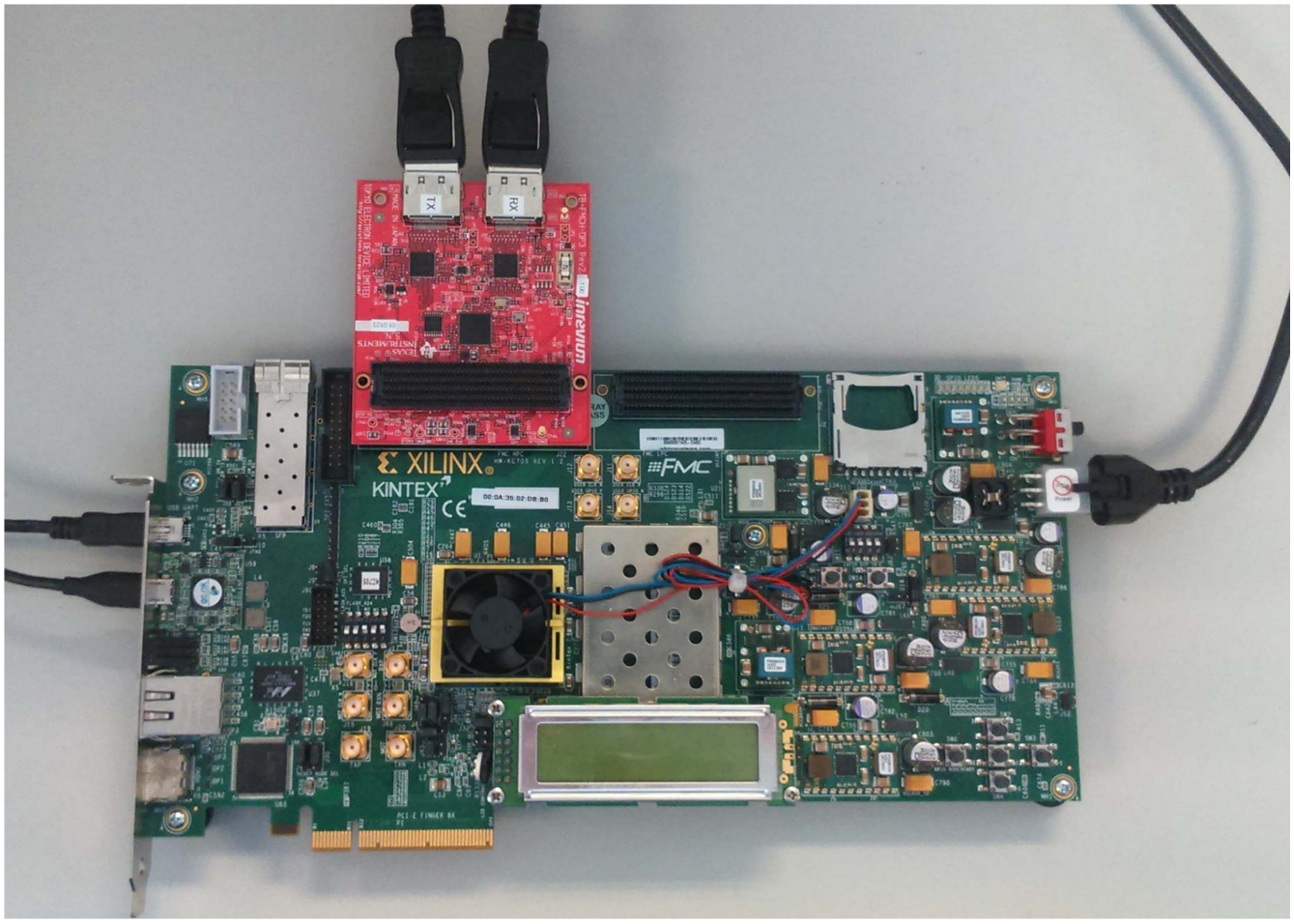


Figure 5-6: **KC705 Board Setup**

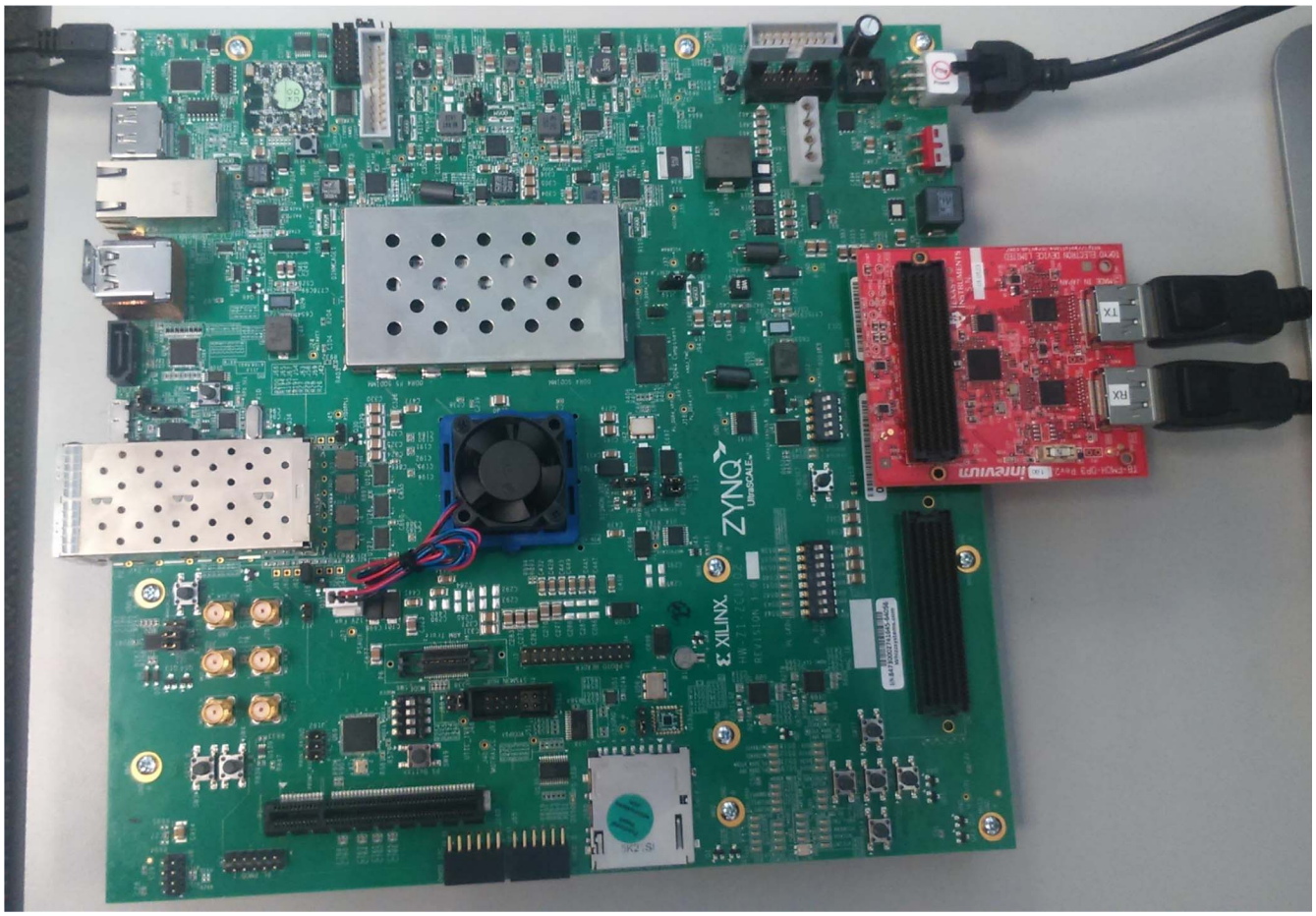
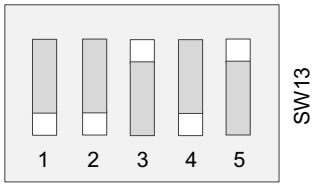


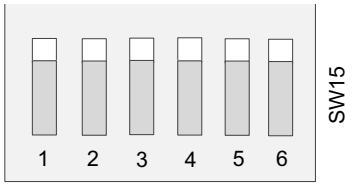
Figure 5-7: **ZCU102 Board Setup**

6. On the KC705 set the mode pin to JTAG (SW13 in 00101 position):



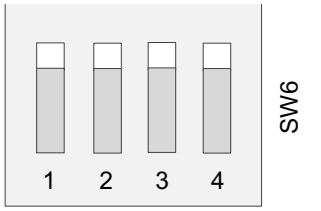
X19735-030918

On the KCU105 set SW15 to 111111:



X20381-030618

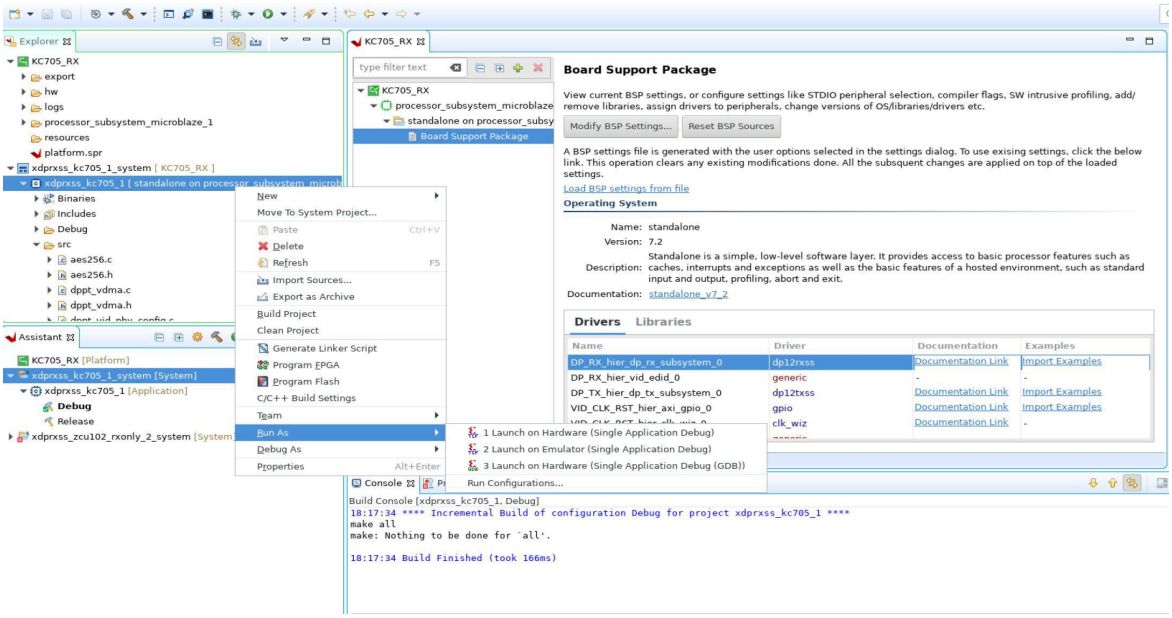
On the XCU102 set SW6 to 1111:



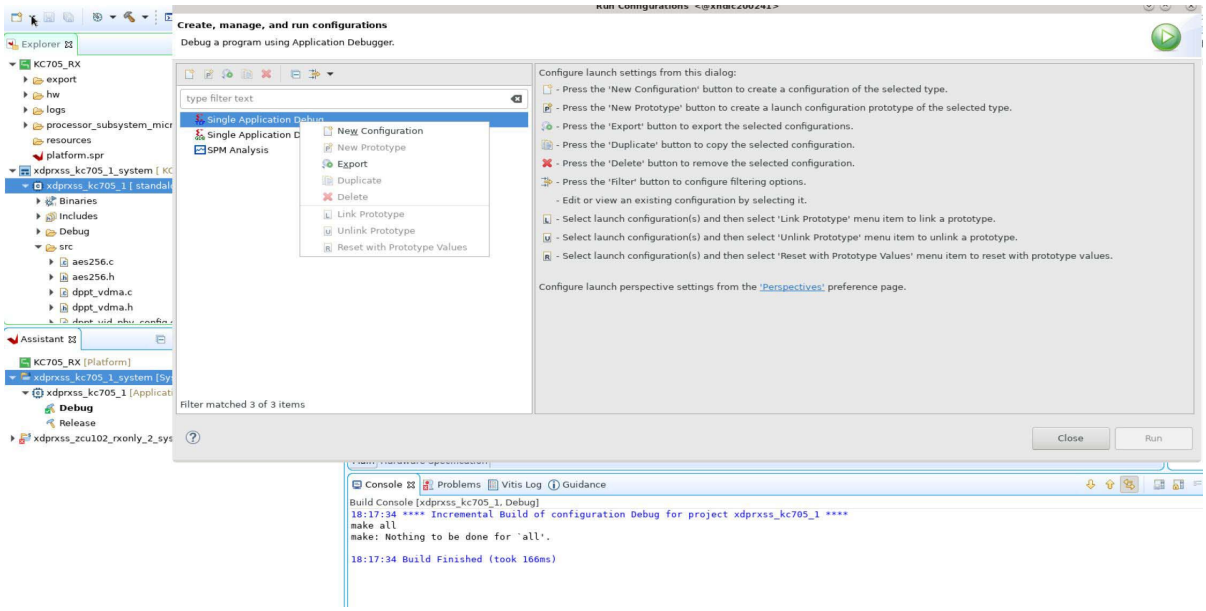
X19805-082817

7. Connect the power supply and power on the board.
8. Start an UART terminal program such as Tera Term or Putty with the following settings:
   a. Baud rate = 115200
   b. Data bits = 8
   c. Parity = none
   d. Stop bits = 1
   e. Flow Control = none

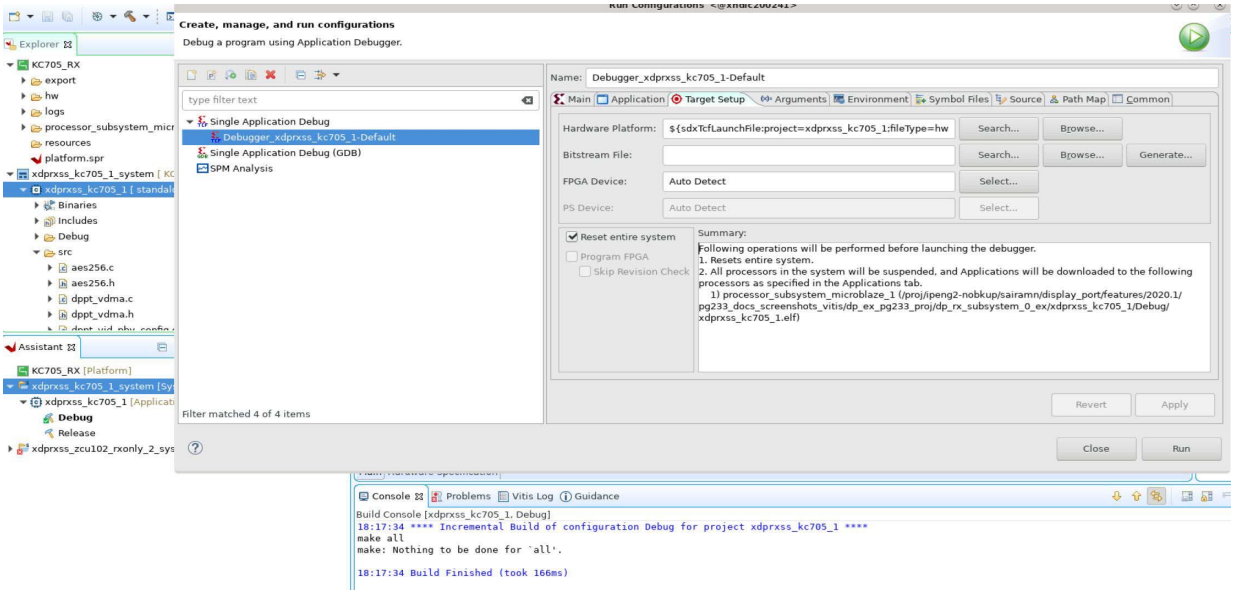
   Note: With the ZCU102 board, there are four COM ports available.
9. In the Vitis environment, under the **Project Explorer**, right-click the application and click **Run As > Run Configurations**.



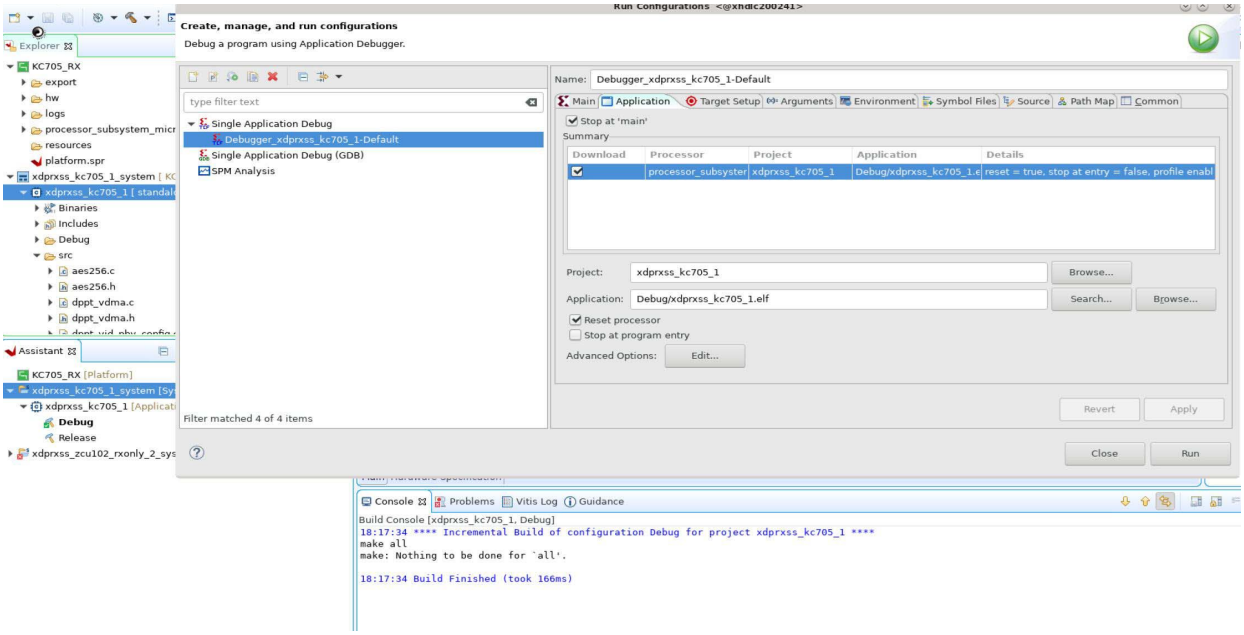
10. In the **Run Configurations** popup menu, right-click **Single Application Debug** and click **New**.



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



12. In the **Application** tab, click **Run**.


Create, manage, and run configurations
Debug a program using Application Debugger.
Explorer
KC705_RX
export
hw
logs
processor_subsystem_micr
resources
platform.spr
xdprxss_kc705_1_system [KC
xdprxss_kc705_1 [standal
Binaries
Includes
Debug
src
aes256.c
aes256.h
dppt_vdma.c
dppt_vdma.h
Assistant
KC705_RX [Platform]
xdprxss_kc705_1_system [Sy
xdprxss_kc705_1 [Applicati
Debug
Release
xdprxss_zcu102_rxonly_2_sys
type filter text
Single Application Debug
Debugger_xdprxss_kc705_1-Default
Single Application Debug (GDB)
SPM Analysis
Filter matched 4 of 4 items
Name: Debugger_xdprxss_kc705_1-Default
Main
Application
Target Setup
Arguments
Environment
Symbol Files
Source
Path Map
Common
Stop at 'main'
Summary
Download
Processor
Project
Application
Details
processor_subsyster
xdprxss_kc705_1
Debug/xdprxss_kc705_1.e
reset = true, stop at entry = false, profile enabl
Project:
xdprxss_kc705_1
Browse...
Application:
Debug/xdprxss_kc705_1.elf
Search...
Browse...
Reset processor
Stop at program entry
Advanced Options:
Edit...
Revert
Apply
Close
Run
Console
Problems
Vitis Log
Guidance
Build Console [xdprxss_kc705_1, Debug]
18:17:34 **** Incremental Build of configuration Debug for project xdprxss_kc705_1 ****
make all
make: Nothing to be done for 'all'.
18:17:34 Build Finished (took 166ms)

Display User Console

Pass-Through Application (KC705 and 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 Figure 5-8 to choose from (KC705).

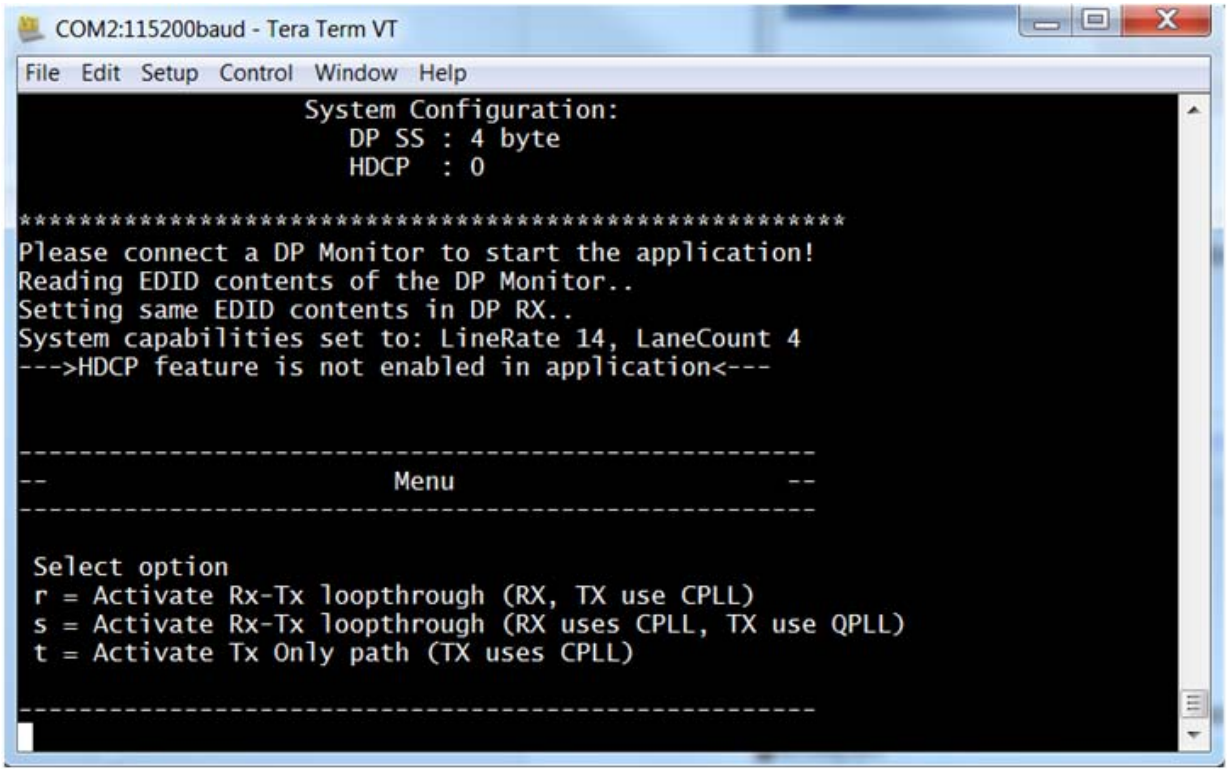


Figure 5-8: **DisplayPort User Console**

Selecting either **r** or **s** puts the system in Pass-Through mode, where the Video received by RX is forwarded to 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 DP RX, then it starts the training. Else, the training happens when the cable is plugged in. As soon as the training is completed, the application starts the DP TX Subsystem. The video should be seen on the monitor once the TX is up. Figure 5-8 shows the UART transcript. The transcript might differ based on the training done by GPU.

HDCP Support and Operation

On KC705, an application example design with HDCP support is available. HDCP is supported by the DisplayPort RX Subsystem (SST mode only). This provides two systems; one with HDCP and the other without HDCP.

Note: To have a working HDCP solution it is necessary to have valid HDCP keys and a monitor that supports HDCP. Xilinx does not provide any HDCP keys. See Configuring HDCP Keys and Key Management to set up and manage HDCP Keys.

The application detects the HDCP capabilities of the monitor that is connected to DisplayPort RX Subsystem. HDCP feature is enabled only if the monitor is capable of displaying HDCP content.

In Pass-Through mode (options **`r`** or **`s`**), the DisplayPort source (for example, GPU) detects the HDCP capability of the DisplayPort Sink and starts the authenticate process. When the Authentication is successful, the GPU can start playing the encrypted HDCP content. The DisplayPort Sink, automatically detects this and starts decrypting the encrypted content. The decrypted content has to be encrypted again before being displayed on the monitor connected to the DisplayPort RX Subsystem. This has to be done to ensure that the unencrypted content is not displayed.

The HDCP authentication process with the DisplayPort Monitor starts when application detects encrypted content on the RX. After the authentication is completed, the DisplayPort RX encrypts the video content before sending it to the monitor. The application tries 100 attempts to authenticate. If it is not able to authenticate within 100 attempts, then it switches the RX video to color bar pattern that is, the unencrypted HDCP content is not displayed.

In RX only mode (selection **`t`**), you have to manually initiate the HDCP authentication and encryption.

If the monitor does not support HDCP, the HDCP functionality is disabled in the application.

Configuring HDCP Keys and Key Management

The application software does not use the raw HDCP keys directly. To use the HDCP keys, they have to be first encrypted and then added into the application. You have to manually perform this process. This application note provides the scripts and software that helps you encrypt the HDCP keys.

Using the Encryption Software

For more information on using the encryption software, see AR: 70605. To generate the AES encrypted HDCP keys, you must have the following keys:

1. 32-byte AES key
2. Valid HDCP keys

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

Follow the steps mentioned here to generate the AES encrypted HDCP key block:

1. Unzip the project and go to the **`hdcp_util/keys`** directory. You can find the encryption block in this directory.
2. Modify the **`gDefaultKey`** array in the following file to a user specified 32 bytes unique key. This is the 32-byte AES key mentioned in point (1) above

   ```
   hdcp_util/keys/key-encryptor/common/src/keyfile.c
   ```

3. Navigate to **`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<n>_keys.dat
   ```

   Note: The user-provided **`devb<n>_keys.dat`** file is expected to have one and 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 keys, each key should be housed exclusively in one **`.dat`** file.

5. The AES encrypted HDCP key block is now created as an array in the **`keymgmt_data.c`** file.
6. Ensure that the following AES keys in the reference design matches with the keys in step 2.

   ```
   \**kc705_system_directly**\**sw_directly**\src\keys.c
   ```

   The HDCP keys are now encrypted and ready to be used. You can any of the following two ways to use the HDCP keys.

Using the AES Encryption Key Block from the EEPROM

You can use the **hdcp_util/iic_wr_util/kc705** project provided with the IP to write the keys to the EEPROM. You need to copy the AES encrypted block generated in the **keymgmt_data.c** file as described in the previous section to the HDCP_KEYS array in the **hdcp_util/iic_wr_util/kc705/sw/src/iic_keys.c** file.

Note: The size of the keys is calculated and stored in the HDCP_KEYS_SZ variable. You should ensure that size of the keys should fit in the EEPROM.

Navigate to the **hdcp_util/iic_wr_util/kc705/sw/** folder. Use the 2016.4 Vivado build in the command prompt and run the following command:

```
xsct ./all.tcl
```

This creates the **iic_eeprom.elf** file in **hdcp_util/iic_wr_util/kc705/sw/iic_eeprom/Debug** folder.

You can then use this **elf** file and the **design_1_wrapper.bit** file provided in **hdcp_util/iic_wr_util/kc705/ready_to_download** folder to program the EEPROM on the board with the AES encrypted keys.

Set **gIsKeyWrittenInEeeprom = TRUE** in the ****kc705_system_directly****software_directly**\src\xdp*xss_kc705.c** file. Ensure that the IIC device ID is correctly set in **keygen_config.h** file.

Using the AES Encryption Key Block from the Block RAM

To use keys from block RAM, you have to copy the AES Encrypted HDCP keys generated in the previous section to the **KEYMGMT_ENCDATA** array defined in the following file in the reference design:

```
\**kc705_system_directly**\software_directly\src\keys.c
```

This process enables the reference design to read keys from block RAM.

Set **gIsKeyWrittenInEeeprom = FALSE** in the ****kc705_system_directly****software_directly**\src\ xdp*xss_kc705.c** file.

Setting the FMC Voltage to 1.8V

To run the example design on the ZCU102 board, ensure that the FMC voltage is set to 1.8V. To set the FMC VADJ voltage:

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

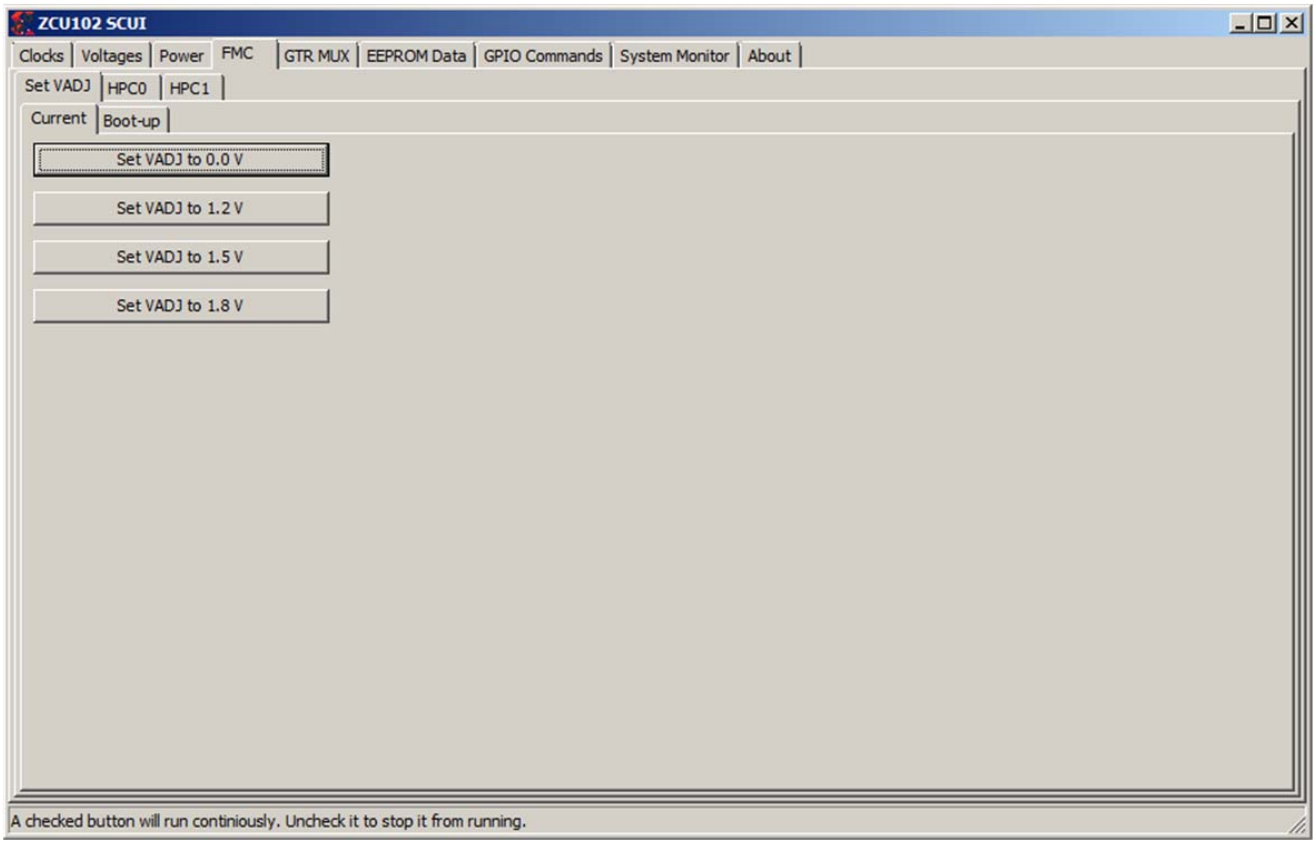


Figure 5-9: **ZCU102 SCUI**

Tested Equipment

Table 5-2 lists the tested equipment used with the example design.

Table 5-2: **Source Equipment**

Sink Type	Brand Name	Model Name	Driver Version	Platform
GPU	NVIDIA	GTX 980	21.21.13.7619	Windows 7
GPU	AMD	RX 460	21.19.384.37	Windows 10
GPU	AMD	FirePro V7900	15.201.2401	Windows 10
Laptop	Apple	MBP	–	macOS
GPU	AMD	Radeon R7 350	21.19.384.37	Windows 10
PC	AMD	Radeon HD 6450	–	CentOS
GPU	NVIDIA	GTX 1080	21.21.13.7290	Windows 10
GPU	NVIDIA	Quadro M4000	21.21.13.7849	Windows 10
Tester	Unigraf	DPT-200	–	–
Tester	Unigraf	DPT-323	–	–

Upgrading

When migrating from the LogiCORE DisplayPort IP to the DisplayPort RX Subsystem with the *Video PHY Controller Product Guide* (PG230) [Ref 1]. Xilinx recommends removing the LogiCORE DisplayPort IP in entirety and then implementing the DisplayPort RX Subsystem. Note the following to assist in the migration:

- Use the example design as a reference to ensure that all connections are correct.
- The LogiCORE DisplayPort IP integrates the transceivers whereas the transceivers reside in the *Video PHY Controller Product Guide* (PG230) [Ref 1] of the subsystem implementation.
- The DisplayPort Subsystem has the option to have a native pixel or an AXI4-Stream interface.
- All associated signals that were part of the transceivers, reference clocks and transceiver lanes, are now part of the *Video PHY Controller Product Guide* (PG230) [Ref 1].
- The parameters for the number of DisplayPort lanes and the PHY Data width need to match between the DisplayPort RX Subsystem and the *Video PHY Controller Product Guide* (PG230) [Ref 1].
- The link clock for the DisplayPort RX Subsystem is generated by the *Video PHY Controller Product Guide* (PG230) [Ref 1].
- The **`lnk_clk_[p/n]`** of the LogiCORE DisplayPort IP should be connected to the **`mgtrefclk0_pad_[p/n]_in`** of the *Video PHY Controller Product Guide* (PG230) [Ref 1].

Frequently Asked Questions

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**.

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

A. 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**$)$

Q. Can I get more information on EDID?

A. The EDID block is outside of the DisplayPort controller and is connected using the I2C interface. It is your responsibility to add the proper EDID content. If you select a different clock source for the EDID block, ensure that the I2C bus has the proper false path constraints.

Driver Documentation

The driver documentation can be found at the Xilinx GitHub page.

Debugging

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

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

Finding Help on Xilinx.com

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

Documentation

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

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

Answer Records

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

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

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

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

Master Answer Record for the DisplayPort Subsystem

AR: 65447

Technical Support

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

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

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

Debug Tools

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

Vivado Design Suite Debug Feature

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

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

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

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 15].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing.

Receive – Training

This section contains debugging steps if the clock recovery or channel equalization is not happening at sink.

- Try with a different source such as the DisplayPort Analyzer.
- Change the cable and check again.
- Put an AUX Analyzer in the receive path and check if the various training stages match with the one's mentioned in DisplayPort Overview in Chapter 3.
- Probe the **`lnk_clk`** output and check the SI of the Clock is within the Phase Noise mask of the respective GT Transceiver.
- Check the RX Initialization Status register (0x0028) and PLL Lock Status (0x0018) register of the Video PHY Controller for Reset done and PLL lock for the active lanes.
- Check the 0x43C and 0x440 registers for Symbol_Locked, Channel Equalization and Clock Recovery Done.

Receive – DP159 Related Issues

This section contains debugging steps for issues related to DP159. Proper operation of DP159 is essential for the training to complete successfully.

- IIC checks:
  - Check if the IIC speed is 400 kHz or higher speed (1 MHz).
  - Check if the IIC writes are happening properly to the DP159 IC
  - Check if the IIC writes are interrupt or polling based. If it is interrupt based, it would be like calling an interrupt within another interrupt routine. Make sure this function correct, or better to go with polling mode, as DP159-IIC writes are supposed to happen at DP training events
- Until the training is done make sure only the TP1 and TP23 interrupts are enabled.
- Ensure that you have no other software code in between TP1 interrupt to training done duration.

- Check whether the TP1 and TP23 handlers are called correctly when the TP1 and TP23 interrupts are detected.
- Ensure that the selection of the reference clock is correct in the Video PHY.
- Probe the **`lnk_clk`** output and check the SI of the Clock is within the Phase Noise mask of the respective GT.
- Avoid using PRINTF to monitor the DP159 configuration as the configuration must be completed as quickly as possible in order to meet the DisplayPort Standard requirements.

Receive – Issues 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.

- If the video timing counters are reporting 0 lines, toggle the DTG enable and software-video reset and check again.
- Check the symbol and disparity error counters 0x448 and 0x44C through AXI reads. If the errors are accumulating, the alignment bit might go off eventually. Perform **`dprx_init`** once and toggle HPD so the source can train the sink again.
- Training lost can occur:
  - When there is change in link configuration and RX is in previously trained state
  - Either symbol lock/channel equalization/clock recovery failure
  - Lane inactivity

Receive – Audio

If the audio is not played at the Sink device or the audio is noisy, check if the programming steps mentioned in Audio Management in Chapter 3 have been followed correctly.

Receive – Sink MST

This section contains debugging steps for issues with the Sink device.

- Check if the GPU connected is recognizing the streams properly. Read the MSA of all the streams and verify against the GPU data.
- Read the VC Payload table through AXI write and check if the allocated stream IDs are sequential in the slots. The 0th slot is not used and should not contain any of the allocated stream IDs.
- Check the symbol and disparity error counters through AXI reads. If there are a lot of errors, there could be video defects.

- Check with AUX Analyzer to see if all the sideband messages are decoded properly.
- Check the link rate and lane count at which it is trained. Only in 5.4 x 4, four streams of 1080p will be possible. With HBR, only two 1080p streams will be possible. The link rate downshift could be because of training failure—make sure that a DisplayPort v1.2a cable is used.
- Make sure a DisplayPort v1.2a cable is used with DP159 in between.

Receive – FIFO Overflow

How do I resolve the USER_FIFO_OVERFLOW interrupts (0x110) when I am using the DisplayPort in Receiver mode?

This is caused when the outgoing data stream on the **`rx_vid_clk`** domain is not fast enough compared to the incoming Display Port data stream on the **`lnk_clk`** domain.

This causes the DisplayPort data stream to get pushed into the FIFO faster than the speed the FIFO can be read out.

There are two ways to resolve this:

1. If possible, increase HBLANK from the source.
2. Increase the **`rx_vid_clk`** frequency to the maximum tested of 200 MHz.

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 **Modify BSP Settings** (Figure D-1).

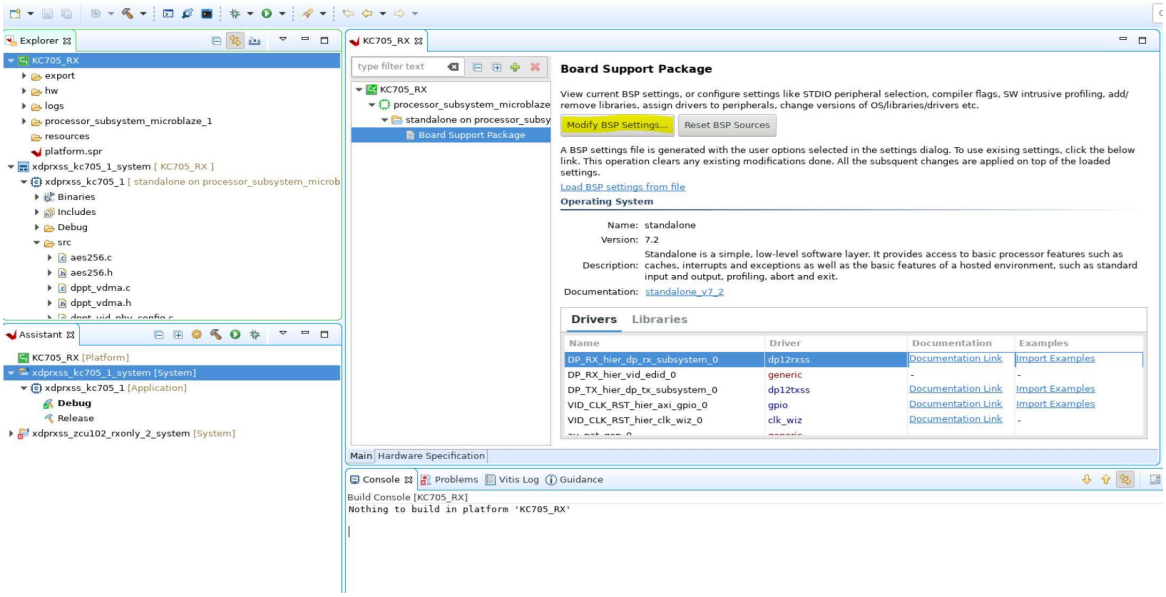


Figure D-1: **Path to Board Support Package Settings**

2. Add the option -DDEBUG to the extra compiler flags then close the BSP settings (Figure D-2).

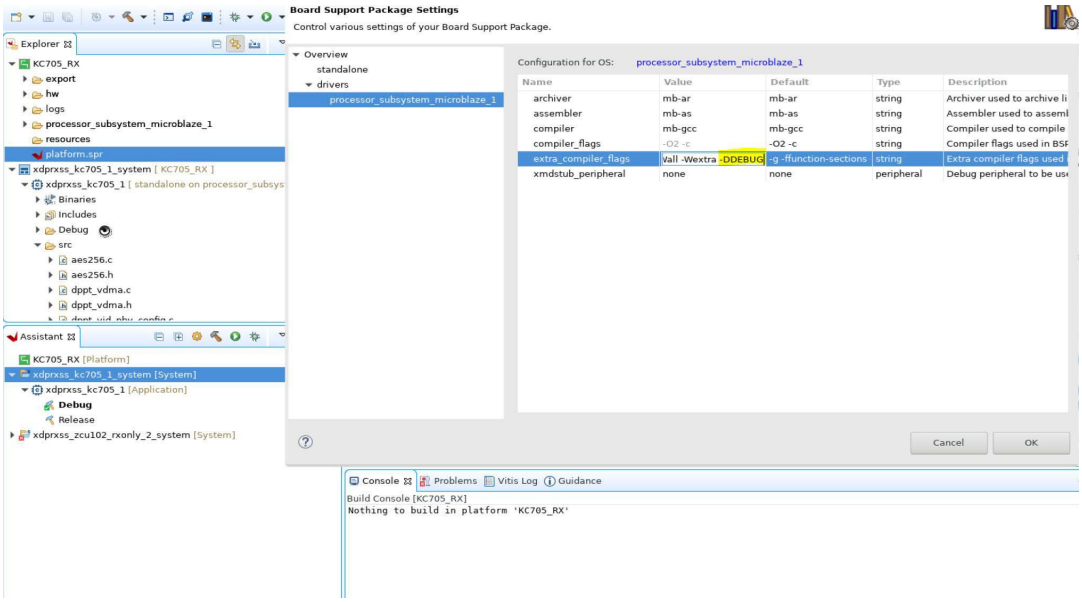


Figure D-2: **Path to Board Support Package Settings**

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

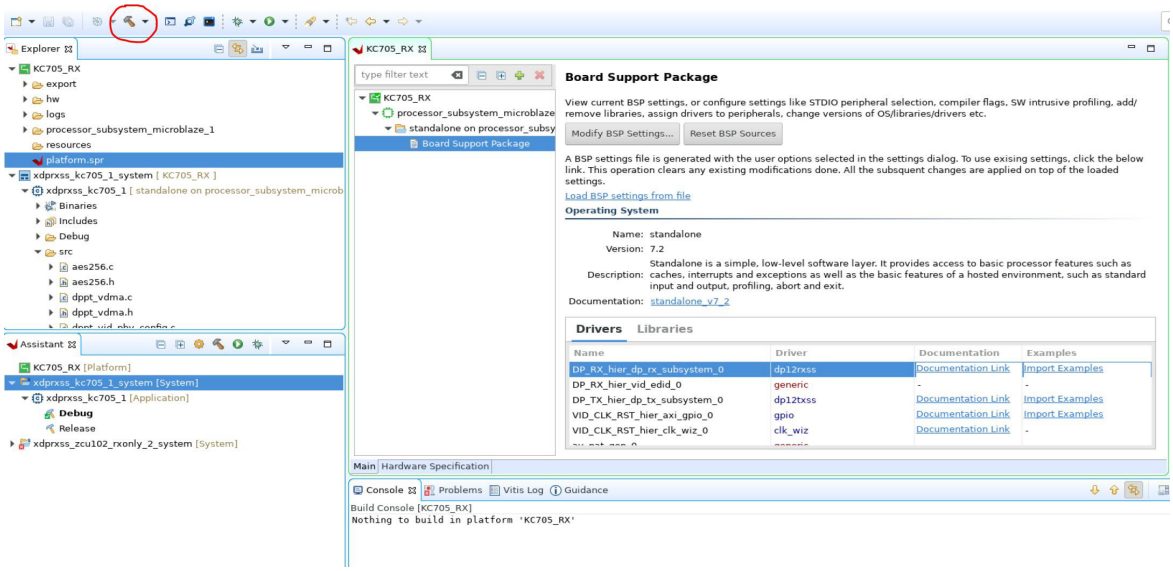


Figure D-3: **Board Support Package Settings Window**

Application Software Development

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

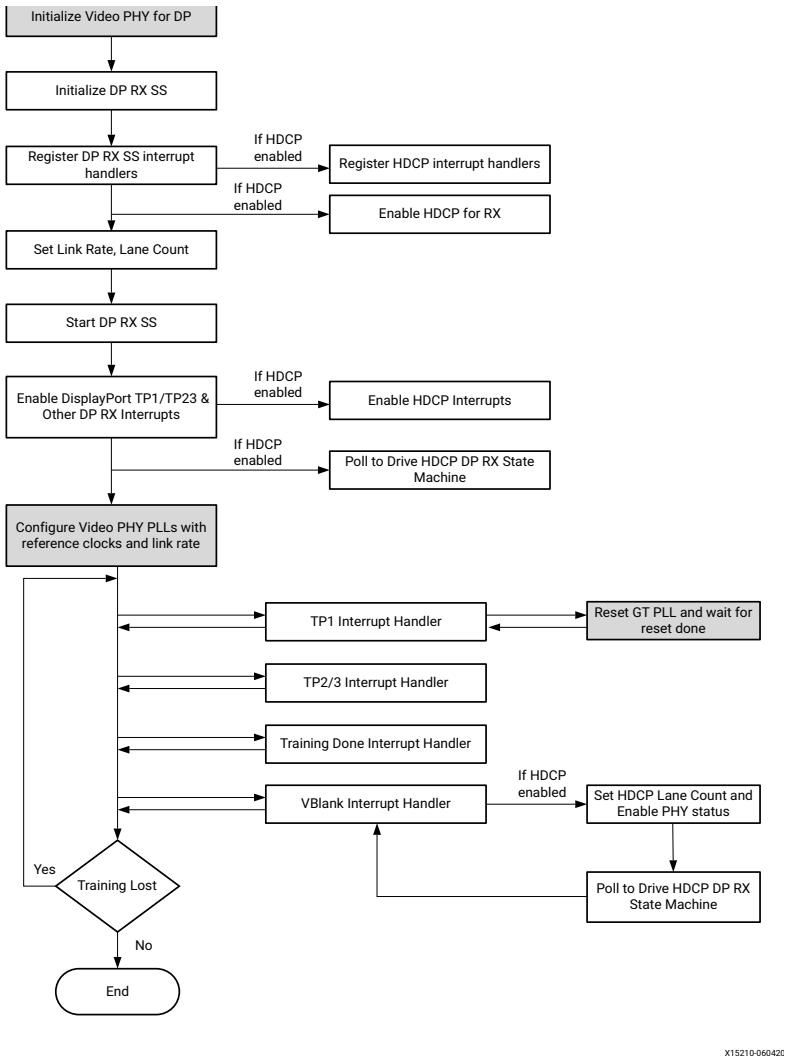


Figure E-1: **DisplayPort RX Subsystem Software Flow for SST Mode**

Note: Video PHY is external to the DisplayPort RX Subsystem and must be configured for the subsystem to work as expected. For more details on Video PHY configuration, see the *Video PHY Product Guide* (PG230) [Ref 1].

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

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

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

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

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

References

These documents provide supplemental material useful with this product guide:

1. *Video PHY Controller Product Guide* (PG230)
2. *AXI4-Stream Video IP and System Design Guide* (UG934)
3. *AXI Interconnect Product Guide* (PG059)
4. *HDCP Controller Product Guide* (PG224)
5. *AXI IIC Bus Interface Product Guide* (PG090)
6. *AXI Timer Product Guide* (PG079)
7. *SNx5DP159_Product_Preview*
8. *DP159 as a DisplayPort Retimer* (SLLA358)
9. VESA *DisplayPort Standard v1.2a*, December 22, 2009
10. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
11. *Vivado Design Suite User Guide: Designing with IP* (UG896)
12. *Vivado Design Suite User Guide: Getting Started* (UG910)
13. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
14. *ISE to Vivado Design Suite Migration Guide* (UG911)
15. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
16. *Vivado Design Suite User Guide: Implementation* (UG904)
17. *AXI Reference Guide* (UG1037)
18. *AXI4-Stream Video IP and System Design Guide* (UG934)
19. *ZCU102 System Controller GUI Tutorial* (XTP433) (registration required)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/03/2020	2.1	• AUX Clock Divider description updated. • EDID I2C Speed Control added. • Updated steps 11 and 13 figure in Building the Example Design. • Updated steps 9-12 in Hardware Setup and Run. • Updated Software Debug.
06/17/2019	2.1	Minor updates
12/05/2018	2.1	• Table 2-1 updated • Table 2-2 and Table 2-3 added • Figure 2-3 updated • Table 2-14 updated
04/04/2018	2.1	• Added Native Video support and dynamic lane supports in IP facts table. • Added DP159 retimer in Overview chapter. • Updated Unsupported Features section in Overview chapter. • Updated Overview section in Product Specification chapter. • Added AXI4-Stream Interface Data Mapping table in Product Specification chapter. • Updated Pixel Mapping Examples on AXI4-Stream Interface table. • Added Audio Streaming Signals section. • Updated DisplayPort RX Subsystem Ports table in Product Specification chapter. • Added IP integrator description in Register Space section in Product Specification chapter. • Updated Clock section and Address Map Example table in Designing with the Core chapter. • Updated Customizing the IP section in Design Flow Steps chapter. • Updated Constraining the Core section in Design Flow Steps chapter. • Added available example designs in Example Design chapter. • Added JTAG mode pin position KCU105 figure in Example Design chapter. • Added HDCP Support and Operation and Configuring HDCP Keys and Key Management sections in the Example Design chapter. • Added Software Debug section in Debugging appendix.
12/20/2017	2.1	• Added Vivado IP Integrator to IP Facts table. • Updated Setting the FMC Voltage to 1.8V section. • Added EDID in FAQ appendix.

Date	Version	Revision
10/04/2017	2.1	• Updated Supported Device Family in IP Facts. • Added MCCS over DDC/CI is not supported in Unsupported Features section. • Added Pixel Mapping for Stream Interface table and Pixel Mapping to Native Interface section in Product Specification chapter. • Added offset description in 0x09C and 0x0A0 in DisplayPort Sink Core Configuration Space table. • Added Example Design chapter. • Added Upgrading appendix. • Added FAQ appendix. • Added new Driver Documentation appendix.
07/14/2017	2.0	Updated 7 series (GTHE2) and Artix-7 support in IP Facts.
06/07/2017	2.0	Vivado Design Suite release for DisplayPort RX v2.0.
04/05/2017	2.0	• Updated Supported Device Family in IP Facts table. • Added In-band stereo in Unsupported Features section. • Added 0x01C, Bit[8] and 0x21C, Bit[30] description to DisplayPort Sink Core Configuration Space table in Product Specification chapter. • Added DisplayPort Registers Sink Core in Product Specification chapter. • Added DisplayPort Overview in Designing with the Core chapter. • Updated rx_vid_clk clock min/max range in Clock Ranges table in Designing with the Core chapter. • Added Reduced Blanking in Designing with the Core chapter. • Updated Hardware Debug section in Debug Appendix.
12/20/2016	2.0	Added HDCP note for Supported Device Family in IP Facts table.
11/30/2016	2.0	Added Important note in Standards section.
10/05/2016	2.0	Updated HDCP features.
04/06/2016	2.0	Added support for 16 bit GT interface and native with pixel mode.
11/18/2015	1.0	Initial Xilinx release.

Please Read: Important Legal Notices

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.