# DisplayPort 1.4 RX Subsystem v2.1

*Product Guide*

**PG300 (v2.1) August 31, 2020**

# Table of Contents

# Introduction

The Xilinx® DisplayPort 1.4 RX Subsystem is a plug-in solution for serial digital video data reception in large video systems and supports video resolutions of up to Full Ultra HD (FUHD) 8K at 30 fps. The Xilinx DisplayPort subsystem provides ease of use in selecting the required mode with automated customization.

## Features

- Support for DisplayPort Sink (RX) capabilities
- Supports multi-stream transport (MST) and single stream transport (SST)
- Dynamic lane support (1, 2, or 4 lanes)
- Dynamic link rate support (1.62/2.7/5.4/8.1 Gb/s)
- Dynamic support for 6, 8, 10, 12, or 16 bits per component (BPC)
- Dynamic support for RGB/YCbCr444/YCbCr422 color formats
- Supports 16-bit Video PHY (GT) interface
- Supports 2 to 8 channel audio with 44/48 kHz sample rates
- Supports HDCP 1.3 and HDCP 2.2 decryption in SST
- AXI IIC controller for external peripheral programming
- Supports native or AXI4-Stream video input interface
- Supports single audio stream in MST mode
- Supports SDP packet for static HDR mode
- Supports eDP v1.4b

# IP Facts

| Subsystem IP Facts Table | |
|---|---|
| **Subsystem Specifics** | |
| Supported Device Family[1] | UltraScale+™ Families (GTHE4, GTYE4)<br>UltraScale™ Families (GTHE3)<br>Zynq® UltraScale+™ RFSoC (GTYE4) |
| Supported User Interfaces | AXI4-Stream, AXI4-Lite, Native video |
| Resources | Performance and Resource Use web page |
| **Provided with Subsystem** | |
| Design Files | Hierarchical subsystem packaged with DisplayPort RX core and other IP cores |
| Example Design | Vivado® IP integrator |
| Test Bench | Not Provided |
| Constraints File | IP cores delivered with XDC files |
| Simulation Model | Not Provided |
| Supported S/W Driver | Standalone, Linux[2] |
| **Tested Design Flows[3]** | |
| Design Entry | Vivado Design Suite |
| Simulation | For supported simulators, see the Xilinx Design Tools: Release Notes Guide. |
| Synthesis | Vivado Synthesis |
| **Support** | |
| Release Notes and Known Issues | Master Answer Record: 70294 |
| All Vivado IP Change Logs | Master Vivado IP Change Logs: 72775 |
| Xilinx Support web page | |

**Notes:**

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

# Overview

The DisplayPort 1.4 RX Subsystem is a full feature, hierarchically packaged subsystem with a DisplayPort (RX) core ready to use in applications in large video systems.

The DisplayPort 1.4 RX Subsystem, in both AXI4-Stream and native interfaces, operates in the following video modes:

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

**RECOMMENDED:** *For the Xilinx® DisplayPort 1.4 RX Subsystem, use a MegaChip retimer.*

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

*Table 1:* **Core Support**

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

*Table 1:* **Core Support** *(cont'd)*

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

## Unsupported Features

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

- In-band stereo
- Video AXI4-Stream interface is not scalable with dynamic pixel mode selection
- Dual-pixel splitter is not supported in native video mode
- HDCP is not supported in MST mode
- iDP
- Global Time Code (GTC)
- Non-LPCM audio
- DSC and/or FEC
- 16/32 channel audio
- 420 Colorimetry

## Licensing and Ordering

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

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

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

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

### License Checkers

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

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

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

*Chapter 3*

# Product Specification

The subsystem can operate with an AXI4-Stream video interface or a native interface using a variety of sub-cores which are described in the following sections.

## AXI4-Stream Video Interface

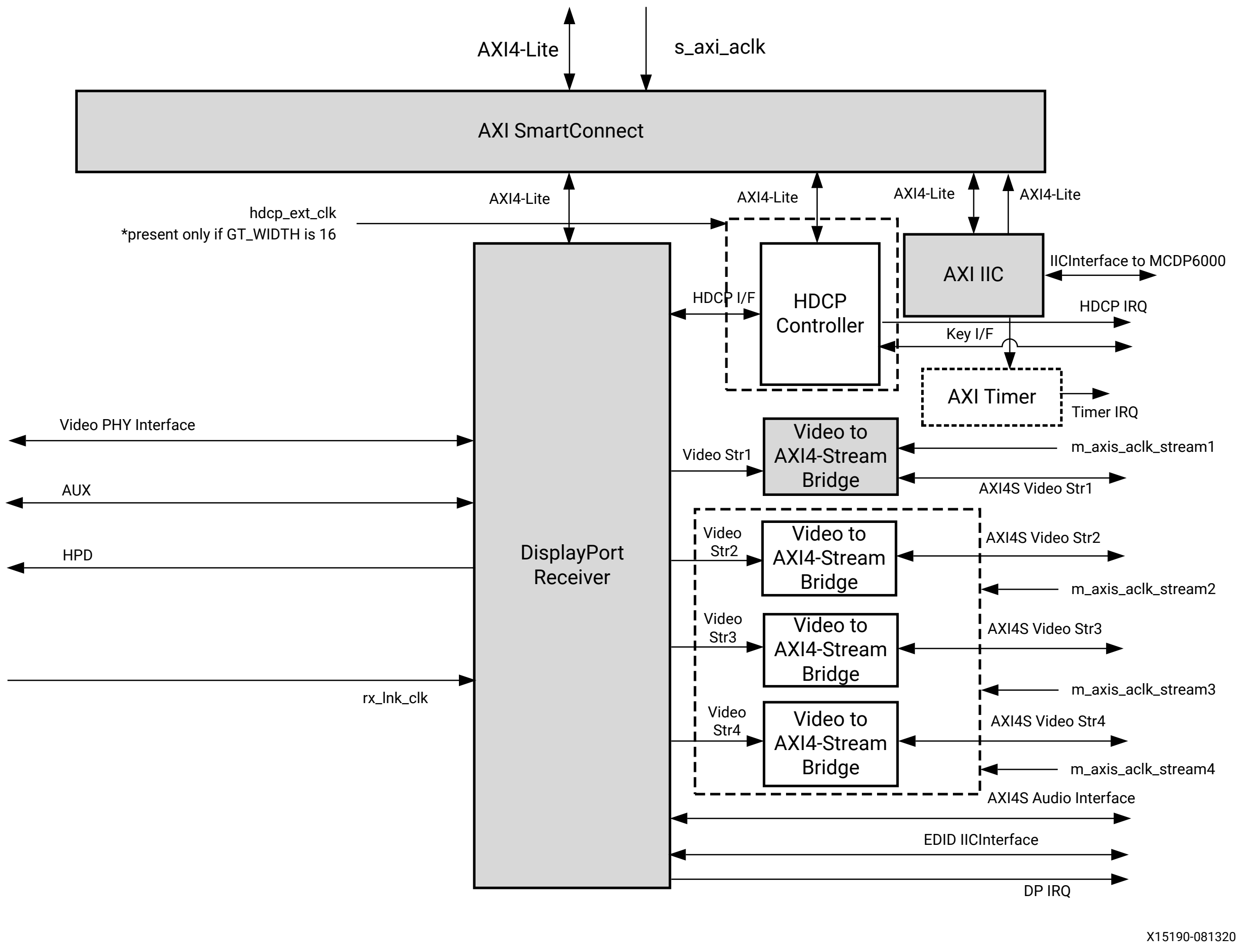
When configured with the AXI4-Stream video interface, the subsystem is packaged with these subcores:

- DisplayPort Receive core
- DisplayPort Video to AXI4-Stream Bridge
- AXI IIC controller
- HDCP core with AXI Timer when HDCP feature is enabled

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

Because the DisplayPort 1.4 RX Subsystem is hierarchically packaged, you select the parameters and the subsystem creates the required hardware. The subsystem receives video using the DisplayPort v1.4 protocol over a 16-bit video PHY interface. The subsystem works with the Video PHY Controller (*Video PHY Controller LogiCORE IP Product Guide* (PG230)) configured for the DispalyPort protocol. The subsystem outputs multi-pixel video over an AXI4-Stream interface. The following figure shows the architecture of the subsystem assuming MST with four streams.

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



**Related Information**

Pixel Mapping Examples on AXI4-Stream Interface

## Native Video Interface

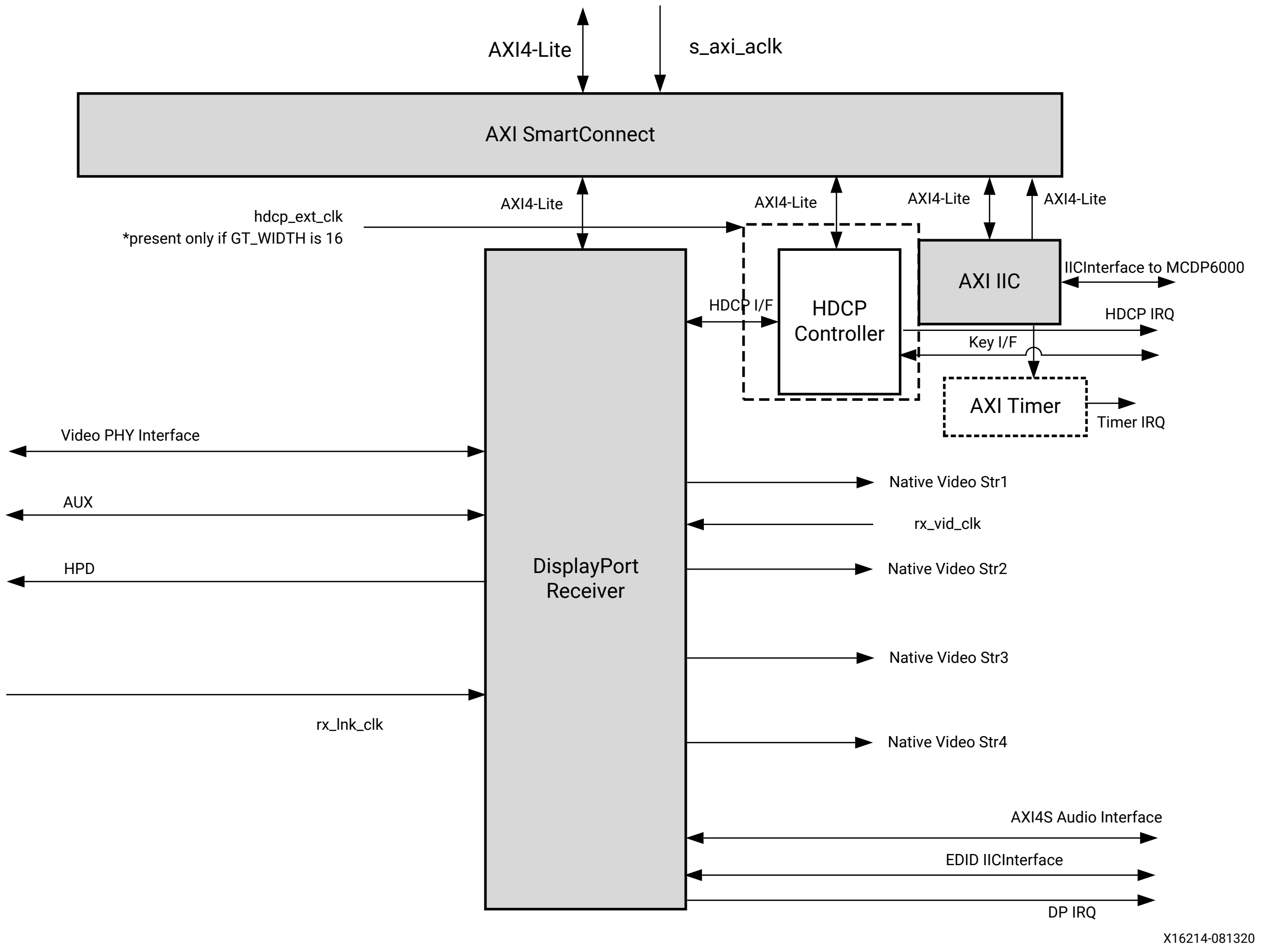
When the native video interface is selected, the subsystem is packaged with two mandatory subcores:

- DisplayPort RX core
- AXI IIC controller

Because the DisplayPort 1.4 RX Subsystem is hierarchically packaged, you select the parameters and the subsystem creates the required hardware. The following figure shows the architecture of the subsystem assuming MST with four native video streams.

The DisplayPort 1.4 RX Subsystem receives the video using the DisplayPort 1.4 protocol over a 16-bit video PHY interface. The subsystem works in conjunction with the Video PHY Controller configured for DisplayPort protocol. The subsystem outputs multi-pixel video to the AXI4-Stream interface.

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



**Related Information**

Pixel Mapping on Native Video Interface

## Subsystem Sub-core Descriptions

The subsystem is comprised of multiple sub-cores. The following sections provide a brief overview of these sub-cores.

### Video to AXI4-Stream Bridge IP Core

A video to AXI4-Stream bridge is used in DisplayPort 1.4 RX Subsystem to convert the video output of the DisplayPort RX IP core to an AXI4-Stream interface. See the *Video In to AXI4-Stream LogiCORE IP Product Guide* (PG043) for information on this core.

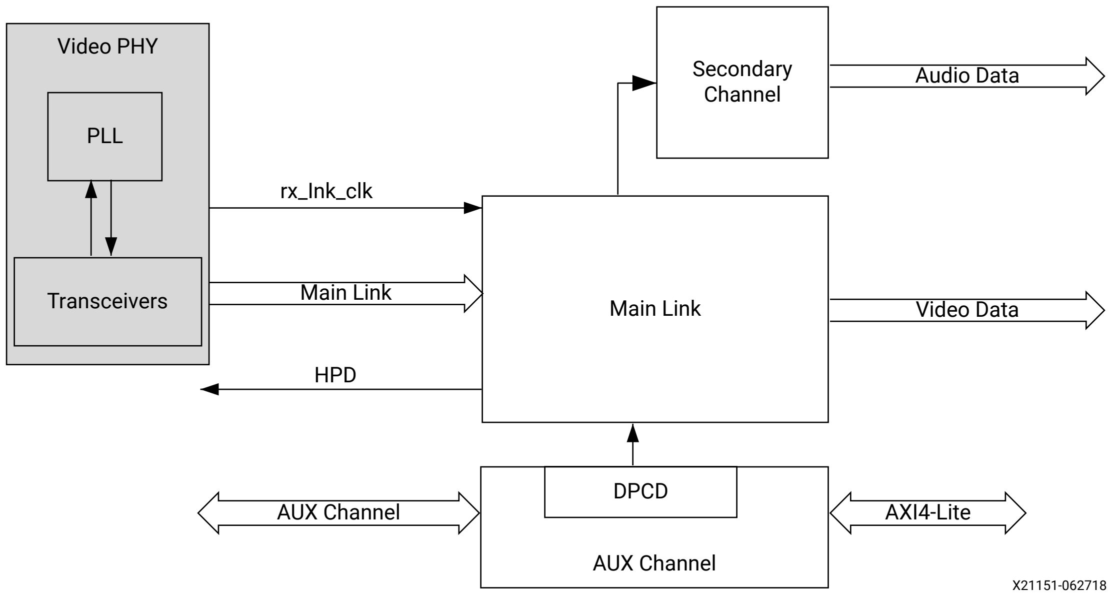
***Note***: In MST mode, there are N number of bridges in the subsystem, where N = the number of AXI4-Stream outputs to the subsystem.

## DisplayPort Receive IP Core

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

- **Main Link:** Provides 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 3:* **DisplayPort Receive Core Block Diagram**

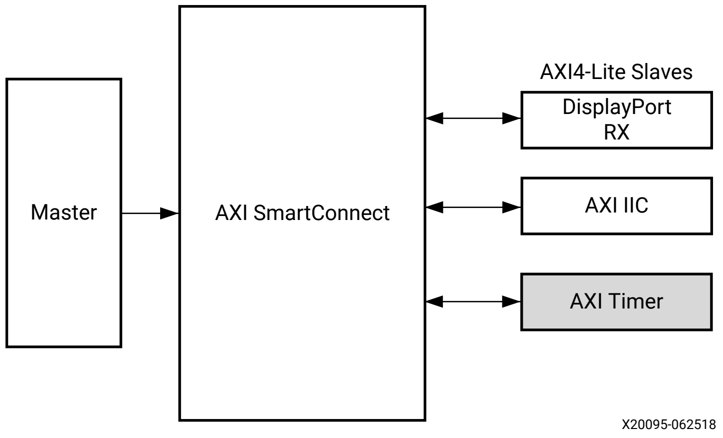


## AXI SmartConnect IP Core

The subsystem uses the Xilinx® AXI Smartconnect IP core, as a smart connect, which contains one AXI4-Lite slave interface and two AXI4-Lite master interfaces.

The following figure shows the AXI slave structure within the DisplayPort 1.4 RX Subsystem. For more details on the AXI smart connect functionality, see the *SmartConnect LogiCORE IP Product Guide* (PG247).

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



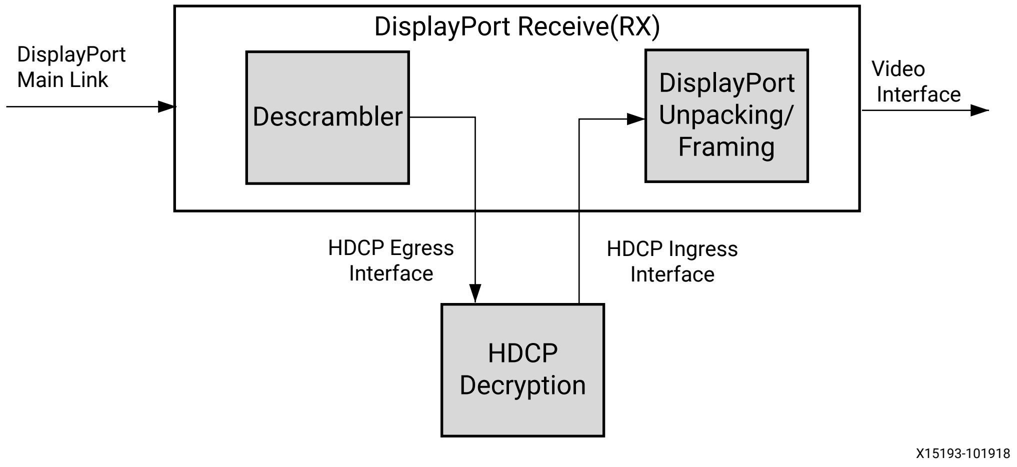
**Related Information**

Address Map Example

## HDCP Controller IP Core

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

The following figure shows the DisplayPort 1.4 RX Subsystem with the HDCP controller. For more details on HDCP v1.3, see the *HDCP 2.2 LogiCORE IP Product Guide* (PG249).



### AXI Timer IP Core

A 32-bit AXI Timer IP core is used in the DisplayPort 1.4 RX Subsystem. The AXI Timer can be accessed through the AXI4 master interface for basic timer functionality in the system.

### AXI IIC IP Core

The AXI IIC Bus Interface IP core is used in DisplayPort 1.4 RX Subsystem to configure any external active devices such as the Retimer.

## Standards

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

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

## Port Descriptions

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

### AXI4-Lite Interface

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

| Port Name | I/O | Description |
|---|---|---|
| s_axi_aclk | I | AXI Bus clock |
| s_axi_aresetn | I | AXI reset. Active-Low. |
| s_axi_awaddr[13:0] | I | Write address |

*Table 2:* **AXI4-Lite Interface** *(cont'd)*

| Port Name | I/O | Description |
|---|---|---|
| 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<br>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 |

## HDCP Key Interface

*Table 3:* **Interrupt Interface**

| Port Name | I/O | Description |
|---|---|---|
| hdcp_ext_clk | I | HDCP external clock |
| hdcp_key_aclk | I | HDCP key clock |
| hdcp_key_aresetn | I | Key Interface reset. Active-Low |
| hdcp_key_tdata[63:0] | I | AXI4-Stream Key Tdata |
| hdcp_key_last | I | AXI4-Stream Key Tlast |
| hdcp_key_tready | O | AXI4-Stream Key Tready |
| hdcp_key_tuser[7:0] | I | AXI4-Stream Key TUSER. KMB should send the Key number from 0 to 41.<br>0 corresponds to KSV and 1 to 40 are the HDCP Keys count. |
| hdcp_key_tvalid | I | AXI4-Stream Key TValid |
| reg_key_sel[2:0] | O | To select the one of the eight sets of 40 keys. |
| start_key_transmit | O | An Active-High pulse that is used to start key transmit. |

## DisplayPort Video PHY Sideband Status

*Table 4:* **DisplayPort Video PHY Sideband Status**

| Port 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 |

## DisplayPort Video PHY Sideband Control

*Table 5:* **DisplayPort Video PHY Sideband Control**

| Port 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 |

## DisplayPort Link Clock Interface

*Table 6:* **DisplayPort Link Clock Interface**

| Port Name | I/O | Description |
|---|---|---|
| rx_lnk_clk | I | Link clock |

## DisplayPort Video PHY Main Link [Lane0–Lane3]

*Table 7:* **DisplayPort Video PHY Main Link [Lane0–Lane3]**

| Port 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 |

*Table 7:* **DisplayPort Video PHY Main Link [Lane0–Lane3]** *(cont'd)*

| Port Name | I/O | Description |
|---|---|---|
| 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 |
| 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 |

## DisplayPort Receive Video Interface

*Table 8:* **DisplayPort Receive Video Interface**

| Port Name | I/O | Description |
|---|---|---|
| rx_vid_clk | I | DisplayPort 1.4 RX video clock |
| rx_vid_rst | I | DisplayPort 1.4 RX video reset |

## AXI4-Stream Video Stream 1 Interface

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

*Table 9:* **DisplayPort 1.4 RX Subsystem AXI4-Stream Video Stream 1 Interface**

| Port Name | I/O | Description |
|---|---|---|
| m_axis_aclk_stream1 | I | Stream1 video clock input |
| m_axis_ video_stream1_tdata[191:0] | O | Stream1 video data. Maximum width of 192. |
| m_axis_video_stream1_tlast | O | Stream1 video last data, End of line pixel. |
| m_axis_video_stream1_tready | I | Stream1 video data read |
| m_axis_video_stream1_tuser | O | Stream1 video user data |
| m_axis_video_stream1_tvalid | O | Stream1 video data valid |

## Native Video Stream 1 Interface

*Table 10:* **DisplayPort 1.4 RX Subsystem Native Video Stream 1 Interface**

| Port 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. If not used, this pin should be connected to 0. |
| rx_vid_stream1_tx_vid_pixel0[47:0] | O | Video data |
| rx_vid_stream1_tx_vid_pixel1[47:0] | O | Video data |

*Table 10:* **DisplayPort 1.4 RX Subsystem Native Video Stream 1 Interface** *(cont'd)*

| Port Name | I/O | Description |
|---|---|---|
| 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] | O | User pixel width/mode |
| rx_cformat[2:0] | O | Colorimetry indicator field and derived from MISC0 and MISC1 fields |

## DisplayPort MST Stream

*Table 11:* **DisplayPort 1.4 RX Subsystem MST Stream**

| Port Name | I/O | Description |
|---|---|---|
| m_axis_aclk_stream<n> | I | Stream<n> video clock input |
| m_axis_ video_stream<n>_tdata[191:0] | O | Stream<n> video data. Maximum width of 192. |
| m_axis_video_stream<n>_tlast | O | Stream<n> video last data, End of line pixel. |
| m_axis_video_stream<n>_tready | I | Stream<n> video data read |
| m_axis_video_stream<n>_tuser | O | Stream<n> video user data |
| m_axis_video_stream<n>_tvalid | O | Stream<n> video data valid |
| 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 | Indicates an odd (1) or even (0) field polarity. If not used, this pin should be connected to 0. |
| 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. |
| 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 I/O Interface – Internal Bidirectional IOB Ports

*Table 12:* **AUX I/O Interface – Internal Bidirectional IOB**

| Port Name | I/O | Description |
|---|---|---|
| aux_rx_io_p | I/O | Bidirectional AUX IO- p |
| aux_rx_io_n | I/O | Bidirectional AUX IO- n |

## AUX I/O Interface – Internal Unidirectional IOB

*Table 13:* **AUX I/O Interface – Internal Unidirectional IOB**

| Port 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 |
| 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 14:* **AUX IP Interface – External IOB**

| Port 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

*Table 15:* **HPD Interface Ports**

| Port Name | I/O | Description |
|---|---|---|
| rx_hpd | O | HPD from DisplayPort 1.4 RX |

## EDID IIC Interface

*Table 16:* **EDID IIC Interface**

| Port 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. |

### IIC Interface

*Table 17:* **IIC Interface**

| Port Name | I/O | Description |
|---|---|---|
| ext_iic_sci_i | I | IIC SCL input |
| ext_iic_sci_o | O | IIC SCL output |
| ext_iic_sci_t | O | IIC SCL enable |
| ext_iic_sda_i | I | IIC SDA input |
| ext_iic_sda_o | O | IIC SDA output |
| ext_iic_sda_t | O | IIC SDA enable |
| ext_rst | O | IIC reset through AXI IIC controller GPIO port0 |

### Interrupts

*Table 18:* **Interrupts**

| Port Name | I/O | Description |
|---|---|---|
| dprxss_dp_irq | O | DisplayPort 1.4 RX IP interrupt out |
| dprxss_iic_irq | O | AXI IIC IP interrupt out |
| dprx_timer_irq | O | AXI Timer interrupt out |

## Register Space

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

- DisplayPort RX IP
- AXI IIC
- HDCP Controller
- 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 1.4 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 1.4 RX Subsystem IP.

## DisplayPort Registers

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

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

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

### *Receiver Core Configuration*

*Table 19:* **Receiver Core Configuration**

| Offset | Access Type | Description |
|---|---|---|
| 0x000 | R/W | LINK_ENABLE. Enable the receiver<br>1 - Enables the receiver core. Asserts the HPD signal when set. |
| 0x004 | R/W | AUX_CLOCK_DIVIDER. Contains the clock divider value for generating the internal 1 MHz clock from the AXI4-Lite host interface clock. The clock divider register provides integer division only and does not support fractional AXI4-Lite clock rates (for example, set to 75 for a 75 MHz AXI4-Lite clock).<br>[27:24] - Valid values are 0-8. Non-zero value in this field issues 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.<br>[15:8] - The number of AXI4-Lite clocks (defined by the AXI4-Lite clock name: s_axi_aclk) equivalent to the recommended width of AUX pulse. Allowable values include:0 (default), 8, 16, 24, 32, 40, and 48. As per the DisplayPort protocol specifications, AUX Pulse Width range = 0.4 to 0.6 µs.<br>[7:0] - Clock divider value<br>For example, for AXI4-Lite clock of 50 MHz (= 20 ns), the filter width, when set to 24, falls in the allowable range as defined by the protocol spec.<br>(20 × 24 = 480)<br>Program a value of 24 in this register. |
| 0x008 | R/W | 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.<br>[3] - End of line reset disable to the MST video stream 4<br>[2] - End of line reset disable to the MST video stream 3<br>[1] - End of line reset disable to the MST video stream 2<br>[0] - End of line reset disable to the SST video stream or MST video stream 1 |
| 0x00C | R/W | DTG_ENABLE. Enables the display timing generator in the user interface.<br>[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. |

*Table 19:* **Receiver Core Configuration** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x010 | R/W | 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). Use quad pixel mode in MST.<br>[2:0]<br>• 1 = Single pixel wide interface. Valid for designs with 1 lane only.<br>• 2 = Dual pixel output mode. Valid for designs with 2 lanes only.<br>• 4 = Quad pixel output mode. Valid for designs with 4 lanes only. |
| 0x014 | R/W | INTERRUPT_MASK. Masks the specified interrupt sources from asserting the axi_init signal. When set to a 1, the specified interrupt source is masked. This register resets to all 1s at power up.<br>[31] - Mask for Cable disconnect/unplug interrupt<br>[30] - CRC test start interrupt<br>[29] - Mask MST Act sequence received interrupt<br>[28] - Mask interrupt generated when DPCD registers 0x1C0, 0x1C1 and 0x1C2 are written for allocation/de-allocation/partial deletion<br>[27] - Audio packet FIFO overflow interrupt<br>[18] - Training pattern 3 start interrupt<br>[17] - Training pattern 2 start interrupt<br>[16] - Training pattern 1 start interrupt<br>[15] - Bandwidth change interrupt<br>[14] - TRAINING_DONE<br>[13] - DOWN_REQUEST_BUFFER_READY<br>[12] - DOWN_REPLY_BUFFER_READ<br>[11] - VC Payload De-allocated<br>[10] - VC Payload Allocated<br>[9] - EXT_PKT_RXD: Set to 1 when extension packet is received<br>[8] - INFO_PKT_RXD: Set to 1 when info packet is received<br>[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.<br>[4] - TRAINING_LOST: Training has been lost on any one of the active lanes<br>[3] - VERTICAL_BLANKING: Start of the vertical blanking interval<br>[2] - NO_VIDEO: The no-video condition has been detected after active video received<br>[1] - POWER_STATE: Power state change, DPCD register value 0x00600<br>[0] - VIDEO_MODE_CHANGE: Resolution change, as detected from the MSA fields |
| 0x018 | R/W | MISC_CONTROL. Allows the host to instruct the receiver to pass the MSA values through unfiltered.<br>[2] - When set to 1, I2C DEFERs is sent as AUX DEFERs to the source device.<br>[1] - When set to 1, the long I2C write data transfers are responded to using DEFER instead of Partial ACKs.<br>[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.<br>[8] - Soft reset control to external HDCP FIFOs.<br>[7] - AUX Soft Reset: When set, AUX logic resets.<br>[0] - Soft Video Reset: When set, video logic resets. Reads return zeros. |
| 0x7F0 | R/W | CFG_EXT_AMX_LINE_RATE<br>[7:0] : Maximum line rate. This value is mirrored in 0x2201 DPCD register. |

### *AUX Channel Status*

*Table 20:* **AUX Channel Status**

| Offset | Access Type | Description |
|---|---|---|
| 0x020 | RO | AUX_REQUEST_IN_PROGRESS. Indicates the receipt of an AUX Channel request<br>[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.<br>[7:0] - Error count, a write to register address 0x28 clears this counter. |
| 0x028 | R/W | REQUEST_COUNT. Provides a running total of the number of AUX requests received.<br>[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.<br>[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.<br>[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.<br>[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.<br>[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.<br>[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 is limited to the seven LSBs. |
| 0x03C | RO | REQUEST_LENGTH. The length of the most recent AUX request is written to this register. The length of the AUX request is the value of this register plus one.<br>[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. |

*Table 20:* **AUX Channel Status** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 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.<br>[31] - Cable disconnect/unplug interrupt<br>[30] - CRC test start interrupt<br>[29] - MST Act sequence received interrupt<br>[28] - Interrupt generated when DPCD registers 0x1C0, 0x1C1, and 0x1C2 are written for allocation/de-allocation/partial deletion<br>[27] - Audio packet FIFO overflow interrupt<br>[18] - Training pattern 3 start interrupt<br>[17] - Training pattern 2 start interrupt<br>[16] - Training pattern 1 start interrupt<br>[15] - Bandwidth change interrupt<br>[14] - TRAINING_DONE: Set to 1 when training is done<br>[13] - DOWN_REQUEST_BUFFER_READY: Set to 1 indicating availability of Down request<br>[12] - DOWN_REPLY_BUFFER_READ: Set to 1 for a read event from Down Reply Buffer by upstream source<br>[11] - VC Payload De-allocated: Set to 0 when de-allocation event occurs in controller<br>[10] - VC Payload Allocated: Set to 1 when allocation event occurs in controller<br>[9] - EXT_PKT_RXD: Set to 1 when extension packet is received<br>[8] - INFO_PKT_RXD: Set to 1 when info packet is received<br>[7] - Reserved<br>[6] - VIDEO: Set to 1 when a valid video frame is detected on main link<br>[5] - Reserved<br>[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<br>[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<br>[2] - NO_VIDEO: Receiver has detected the no-video flags in the VBID field after active video has been received<br>[1] - POWER_STATE: Transmitter has requested a change in the current power state of the receiver core<br>[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 20:* **AUX Channel Status** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x044 | R/W | 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.<br>[31] - TPS4 Interrupt (DPCD Wr)<br>[30] - Access TP Lane Set Interrupt (DPCD Wr)<br>[29] - Access Link Qual Pattern Interrupt (DPCD Wr)<br>[28] - Access Symbol Error Counter Interrupt (DPCD Rd)<br>[17] - Video Interrupt - Stream 4<br>[16] - Vertical Blanking Interrupt - Stream4<br>[15] - No Video Interrupt - Stream 4<br>[14] - Mode Change Interrupt - Stream 4<br>[13] - Info Packet Received - Stream 4<br>[12] - Ext Packet Received - Stream 4<br>[11] - Video Interrupt - Stream 3<br>[10] - Vertical Blanking Interrupt - Stream 3<br>[9] - No Video Interrupt - Stream 3<br>[8] - Mode Change Interrupt - Stream 3<br>[7] - Info Packet Received - Stream 3<br>[6] - Ext Packet Received - Stream 3<br>[5] - Video Interrupt - Stream 2<br>[4] - Vertical Blanking Interrupt - Stream 2<br>[3] - No Video Interrupt - Stream 2<br>[2] - Mode Change Interrupt - Stream 2<br>[1] - Info Packet Received - Stream 2<br>[0] - Ext Packet Received - Stream 2 |

*Table 20:* **AUX Channel Status** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 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.<br>[31] - TPS4 Interrupt (DPCD Wr)<br>[30] - Access TP Lane Set Interrupt (DPCD Wr)<br>[29] - Access Link Qual Pattern Interrupt (DPCD Wr)<br>[28] - Access Symbol Error Counter Interrupt (DPCD Rd)<br>[17] - Video Interrupt - Stream 4<br>[16] - Vertical Blanking Interrupt - Stream4<br>[15] - No Video Interrupt - Stream 4<br>[14] - Mode Change Interrupt - Stream 4<br>[13] - Info Packet Received - Stream 4<br>[12] - Ext Packet Received - Stream 4<br>[11] - Video Interrupt - Stream 3<br>[10] - Vertical Blanking Interrupt - Stream 3<br>[9] - No Video Interrupt - Stream 3<br>[8] - Mode Change Interrupt - Stream 3<br>[7] - Info Packet Received - Stream 3<br>[6] - Ext Packet Received - Stream 3<br>[5] - Video Interrupt - Stream 2<br>[4] - Vertical Blanking Interrupt - Stream 2<br>[3] - No Video Interrupt - Stream 2<br>[2] - Mode Change Interrupt - Stream 2<br>[1] - Info Packet Received - Stream 2<br>[0] - Ext Packet Received - Stream 2 |
| 0x050 | R/W | 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.<br>[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.<br>[7:0] - HSYNC_PULSE_WIDTH: Specifies the number of clock cycles the horizontal sync pulse is asserted. The vid_hsync signal is High for the specified number of clock cycles. |
| 0x058 | R/W | 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.<br>[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 | R/W | [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 1.4. |
| 0x064 | R/W | [31] - Set to override Training Pattern 1 (TP1) score.<br>[14:0] - Training pattern1 (TP1) score to override. |
| 0x068 | R/W | [31] - Set to override the Training Pattern2/3 scores.<br>[12:0] - Training pattern2/3 (TP2/TP3) score to override. |
| 0x06C | RO | Provides the contents of DPCD registers 0x1C0, 0x1C1, and 0x1C2.<br>[21:16] - Time slot count<br>[13:8] - Starting time slot<br>[5:0] - VC Payload ID |

*Table 20:* **AUX Channel Status** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x074 | R/W | [5] - Enable CRC support. The CRC has to be calculated outside the DisplayPort IP and the values have to be provided in 0x078, 0x07C, and 0x080.<br>[3:0] - CRC change count to be configured by SW |
| 0x078 | R/W | CRC for Red color |
| 0x07C | R/W | CRC for Green color |
| 0x080 | R/W | CRC for Blue color |

## DPCD Fields

*Table 21:* **DPCD Fields**

| Offset | Access Type | Description |
|---|---|---|
| 0x084 | R/W | LOCAL_EDID_VIDEO. Indicates the presence of EDID information for the video stream.<br>[0] - Set to 1 to indicate to the transmitter through the DPCD registers that the receiver supports local EDID information. |
| 0x088 | R/W | LOCAL_EDID_AUDIO. Indicates the presence of EDID information for the audio stream.<br>[0] - Set to 1 to indicate to the transmitter through the DPCD registers that the receiver supports local EDID information |
| 0x08C | R/W | REMOTE_COMMAND. General byte for passing remote information to the transmitter.<br>[7:0] - Remote data byte. |
| 0x090 | R/W | DEVICE_SERVICE_IRQ. Indicates DPCD DEVICE_SERVICE_IRQ_VECTOR state.<br>[4] - Set to 1 to indicate a new DOWN Reply Buffer Message is ready.<br>[1] - Reflects SINK_SPECIFIC_IRQ state of DPCD 0x201 register. This bit is RO.<br>[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 | R/W | VIDEO_UNSUPPORTED. DPCD register bit to inform the transmitter that video data is not supported.<br>[0] - Set to 1 when video data is not supported. |
| 0x098 | R/W | AUDIO_UNSUPPORTED. DPCD register bit to inform the transmitter that audio data is not supported<br>[0] - Set to 1 when audio data is not supported. |
| 0x09C | R/W | 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.<br>• [4:0] - Link rate override value for DisplayPort Standard v1.4 designs<br>• [3:0] - Link rate override value for DisplayPort Standard v1.4 designs<br>• 0x6 = 1.62 Gb/s<br>• 0xA = 2.7 Gb/s<br>• 0x14 = 5.4 Gb/s<br>• 0x1E = 8.1 Gb/s |

*Table 21:* **DPCD Fields** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x0A0 | R/W | 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.<br>[7] - ENHANCED_FRAME_CAP: Capability override<br>[6] - TPS3_SUPPORTED: Capability override for DisplayPort Standard v1.4 protocol designs only. Reserved for v1.1a protocol.<br>[4:0] - Lane count override value (1, 2, or 4 lanes). |
| 0x0A4 | R/W | 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.<br>[15:8] - TRAINING_AUX_RD_INTERVAL (the values are based on DisplayPort Standard v1.4 protocol).<br>[7:6] - SYMBOL ERROR COUNT SEL Override<br>[5] - SCRAMBLING_DISABLE Override<br>[4] - RECOVERED_CLOCK_OUT_EN Override<br>[3:0] - Training Pattern Select |
| 0x0A8 | R/W | 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.<br>[7:6] - Reserved<br>[5] - MAX_PRE-EMPHASIS_REACHED override<br>[4:3] - PRE-EMPHASIS_SET override<br>[2] - MAX_SWING_REACHED override<br>[1:0] - VOLTAGE SWING SET override |
| 0x0AC | R/W | 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 | R/W | 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 | R/W | 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 * | R/W | Override DPCD Control Register. Setting this register to 0x1 enables AXI/APB write access to DPCD capability structure. |
| 0x0BC | R/W | Override DPCD DOWNSPREAD control field. Register 0x0B8 must be set to 1 to override DPCD values.<br>[0] - MAX_DOWNSPREAD Override. Set to 1'b1 by default. |
| 0x0C0 | R/W | Override DPCD LINK_QUAL_LANE0_SET field for DPCD1.4 version only. Register 0x0B8 must be set to 1 to override DPCD values.<br>[2:0] - LINK_QUAL_LANE0_SET override |
| 0x0C4 | R/W | Override DPCD LINK_QUAL_LANE1_SET field for DPCD1.4 version only. Register 0x0B8 must be set to 1 to override DPCD values.<br>[2:0] - LINK_QUAL_LANE1_SET override |

*Table 21:* **DPCD Fields** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x0C8 | R/W | Override DPCD LINK_QUAL_LANE2_SET field for DPCD1.4 version only. Register 0x0B8 must be set to 1 to override DPCD values.<br>[2:0] - LINK_QUAL_LANE2_SET override |
| 0x0CC | R/W | Override DPCD LINK_QUAL_LANE3_SET field for DPCD1.4 version only. Register 0x0B8 must be set to 1 to override DPCD values.<br>[2:0] - LINK_QUAL_LANE3_SET override |
| 0x0D0 | R/W | [8] - Clears the VCPayload Table contents. This bit is auto cleared.<br>[7:5] - Unused<br>[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<br>[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.<br>[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.<br>[0] - MST CAPABILITY: Enable or Disable MST capability. Set to 1 to enable MST capability. This bit should be set during the configuration programming stage only. |
| 0x0D4 | R/W | [6:0] - Sink device count: Recommended to be programmed during initialization of the Sink device. In SST mode, the value should be 1. |
| 0x0E0 | R/W | GUID word 0. Allows you to set up GUID if required from host interface. Valid for DPCD1.4 version only.<br>[31:0] - Lower 4 bytes of GUID DPCD field |
| 0x0E4 | R/W | GUID word 1. Allows you to set up GUID if required from host interface. Valid for DPCD1.4 version only.<br>[31:0] - Bytes 4 to 7 of GUID DPCD field |
| 0x0E8 | R/W | GUID word 2. Allows you to set up GUID if required from host interface. Valid for DPCD1.4 version only.<br>[31:0] - Bytes 8 to 11 of GUID DPCD field |
| 0x0EC | R/W | GUID word 3. Allows you to set up GUID if required from host interface. Valid for DPCD1.4 version only.<br>[31:0] - Bytes 12 to 15 of GUID DPCD field |
| 0x0F0 | R/W | GUID Override.<br>[0]: When set to 0x1, the GUID field of the DPCD reflects the data written in GUID Words 0 to 3. Valid for DPCD1.4 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 22:* **Core ID**

| Offset | Access Type | Description |
|---|---|---|
| 0x0FC | RO | CORE_ID. Returns the unique identification code of the core and the current revision level.<br>[31:24] - DisplayPort protocol major version<br>[23:16] - DisplayPort protocol minor version<br>[15:8] - DisplayPort protocol revision<br>[7:0] - Core mode of operation<br>• 0x00: Transmit<br>• 0x01: Receive<br>The CORE_ID value for the protocol and core is DisplayPort Standard v1.4 with a Receive core: 32'h01_04_00_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.<br>[11] - Video Timing FIFO_OVERFLOW_FLAG (Stream 4): 1 indicates that the Video Timing FIFO is overflowed for stream4.<br>[10] - Video Timing FIFO_OVERFLOW_FLAG (Stream 3): 1 indicates that the Video Timing FIFO is overflowed for stream3.<br>[9] - Video Timing FIFO_OVERFLOW_FLAG (Stream 2): 1 indicates that the Video Timing FIFO is overflowed for stream2.<br>[8] - Video Timing FIFO_OVERFLOW_FLAG (Stream 1): 1 indicates that the Video Timing FIFO is overflowed.<br>[7] - Video Unpack FIFO_OVERFLOW_FLAG (Stream 4): 1 indicates that the Video unpack FIFO is overflowed for stream4.<br>[6] - Video Unpack FIFO_OVERFLOW_FLAG (Stream 3): 1 indicates that the Video unpack FIFO is overflowed for stream3.<br>[5] - Video Unpack FIFO_OVERFLOW_FLAG (Stream 2): 1 indicates that the Video unpack FIFO is overflowed for stream2.<br>[4] - Video Unpack FIFO_OVERFLOW_FLAG (Stream 1): 1 indicates that the Video unpack FIFO is overflowed.<br>[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.<br>[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.<br>[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.<br>[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.<br>[3] - State of the vertical sync pulse for Stream 4.<br>[2] - State of the vertical sync pulse for Stream 3.<br>[1] - State of the vertical sync pulse for Stream 2.<br>[0] - State of the vertical sync pulse for Stream 1. |

### *PHY Configuration and Status*

*Table 23:* **PHY Configuration and Status**

| Offset | Access Type | Description |
|---|---|---|
| 0x208 | RO | PHY_STATUS. Provides status for the receiver core PHY.<br>[31:30] - RX buffer status, lane 3<br>[29:28] - RX buffer status, lane 2<br>[27:26] - RX buffer status, lane 1<br>[25:24] - RX buffer status, lane 0<br>[23] - RXBYTEISALIGNED status of PHY, lane 3<br>[22] - RXBYTEISALIGNED status of PHY, lane 2<br>[21] - RXBYTEISALIGNED status of PHY, lane 1<br>[20] - RXBYTEISALIGNED status of PHY, lane 0<br>[15:14] - RX voltage low, lanes 2 and 3<br>[13:12] - RX voltage low, lanes 0 and 1<br>[11:10] - PRBS error, lanes 2 and 3<br>[9:8] - PRBS error, lanes 0 and 1<br>[7] - Receiver Clock locked<br>[6] - FPGA general interconnect clock PLL locked<br>[5] - PLL for lanes 2 and 3 locked (Tile 1)<br>[4] - PLL for lanes 0 and 1 locked (Tile 0)<br>[3:2] - Reset done for lanes 2 and 3 (Tile 1)<br>[1:0] - Reset done for lanes 0 and 1 (Tile 0) |

*Table 23:* **PHY Configuration and Status** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x214 | R/W | 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 forces training to fail until this value is met.<br><br>***Note***: It is not recommended to change this register value.<br><br>[31] - PHY Test Mode (Advanced. Used to test PHY.)<br>[23:14] - PREEMP_TABLE (only for Advanced users)<br>15:14: Iteration 1 pre-emp request level<br>17:16: Iteration 2 pre-emp request level<br>19:18: Iteration 3 pre-emp request level<br>21:20: Iteration 4 pre-emp request level<br>23:22: Iteration 5 pre-emp request level<br>[13:12] - SET_PREEMP (only for Advanced users)<br>[11:10] - Channel Equalization options (only for Advanced users)<br>• 00: Default (pre-emphasis adjust request is incremented one by per iteration until maximum pre-emphasis limit (SET_PREEMP) is reached)<br>• 01: Hold pre-emphasis adjust request to SET_PREEMP for all iterations<br>• 10: Not applicable<br>• 11: Pick values from PREEMP_TABLE<br>[9:8] - SET_VSWING (only for Advanced users). Default value is 0x0000.<br>[6:4] - VSWING_SWEEP_CNT (only for Advanced users)<br>[3:2] - Clock Recovery options (only for Advanced users)<br>• 00: Default (Voltage swing adjust request will get incremented one by every iteration)<br>• 01: Increment adjust request every 4 or VSWING_SWEEP_CNT iterations<br>• 10: Hold adjust request to SET_VSWING value for all iterations<br>• 11: Not applicable<br>[1:0] - The minimum voltage swing setting matches the values defined in the DisplayPort Standard for the TRAINING_LANEX_SET register. |
| 0x21C | R/W | CDR_CONTROL_CONFIG.<br>[30] - Disable Training Timeout<br>[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 | R/W | 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.<br>[31:0] - The value written in this register is used in DisplayPort Sink to detect cable disconnect or unplug event. 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. Cable unplug counter is a free running counter and it reloads with BS_IDLE_TIME as and when it receives the BS character over link or when it's time out by reaching maximum count. |

### DisplayPort Audio

*Table 24:* **DisplayPort Audio**

| Offset | Access Type | Description |
|---|---|---|
| 12'h300 | R/W | RX_AUDIO_CONTROL. This register enables audio stream packets in main link.<br>[5:4]: MST Audio Stream no. N/A for SST<br>[3]: Audio Enable for STREAM 4 in MST. N/A for SST<br>[2]: Audio Enable for STREAM 3 in MST. N/A for SST<br>[1]: Audio Enable for STREAM 2 in MST. N/A for SST<br>[0]: Audio Enable for SST. In MST, Audio Enable for STREAM 1 |
| 12'h304 | RO | RX_AUDIO_INFO_DATA<br>[31:0] Word formatted as per CEA 861-C Info Frame. Total of eight words should be read.<br>1st word<br>• [31:24] = HB3<br>• [23:16] = HB2<br>• [15:8] = HB1<br>• [7:0] = HB0<br>2nd word - DB3,DB2,DB1,DB0<br>....<br>8th word - DB27,DB26,DB25,DB24<br>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).<br>[31:24] - Reserved<br>[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).<br>[31:24] - Reserved<br>[23:0] - NAUD |
| 12'h32C | RO | RX_AUDIO_STATUS.<br>[9] - Extension Packet Received. Resets automatically after all words (9) are read. Blocks new packet until host reads the data.<br>[8:3] - Reserved<br>[2:1] - RS Decoder Error Counter. Used for debugging purpose.<br>[0] - Info Packet Received. Resets automatically after all info words (eight) are read. Blocks new packet until host reads the data. |

*Table 24:* **DisplayPort Audio** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 12'h330 to 12'h350 | RO | RX_AUDIO_EXT_DATA<br>[31:0] - Word formatted as per extension packet described in protocol standard. Packet length is fixed to 32 bytes in Sink controller.<br>You should convey this information to Source using the vendor fields and ensure proper packet size transmission is done by the Source controller. Total of nine words should be read. Extension packet address space can be used to hold the audio copy management packet/ISRC packet/VSC packets.<br>1st word<br>• [31:24] = HB3<br>• [23:16] = HB2<br>• [15:8] = HB1<br>• [7:0] = HB0<br>2nd word - DB3,DB2,DB1,DB0<br>....<br>9th word - DB31,DB30,DB29,DB28<br>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*

For detailed descriptions of these registers, see the *VESA DisplayPort Standard* (VESA website) standard.

*Table 25:* **DPCD Configuration Space**

| Offset | Access Type | Description |
|---|---|---|
| 0x400 | RO | DPCD_LINK_BW_SET. Link bandwidth setting.<br>[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.<br>[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.<br>[0] - Set to 1 when enhanced framing mode is enabled. |
| 0x40C | RO | DPCD_TRAINING_PATTERN_SET. Current value of the training pattern registers.<br>[1:0] - TRAINING_PATTERN_SET: Set the link training pattern according to the 2-bit code:<br>• 000 = Training not in progress<br>• 001 = Training pattern 1<br>• 010 = Training pattern 2<br>• 011 = Training pattern 3<br>• 111 = Training pattern 4 |

*Table 25:* **DPCD Configuration Space** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x410 | RO | DPCD_LINK_QUALITY_PATTERN_SET. Current value of the link quality pattern field of the DPCD training pattern register.<br>[1:0] - transmitter is sending the link quality pattern:<br>• 00 = Link quality test pattern not transmitted<br>• 01 = D10.2 test pattern (unscrambled) transmitted<br>• 10 = Symbol Error Rate measurement pattern<br>• 11 = PRBS7 transmitted |
| 0x414 | RO | DPCD_RECOVERED_CLOCK_OUT_EN. Value of the output clock enable field of the DPCD training pattern register.<br>ixia_locid="41">[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.<br>[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.<br>[1:0] - SYMBOL_ERROR_COUNT_SEL:<br>• 00 = Disparity error and illegal symbol error<br>• 01 = Disparity error<br>• 10 = Illegal symbol error<br>• 11 = Reserved |
| 0x420 | RO | DPCD_TRAINING_LANE_0_SET. Used by the transmitter during link training to configure the receiver PHY for lane 0.<br>[5] - MAX_PRE-EMPHASIS_REACHED: Set to 1 when the maximum pre-emphasis setting is reached.<br>[4:3] - PRE-EMPHASIS_SET<br>• 00 = Training Pattern 2 without pre-emphasis<br>• 01 = Training Pattern 2 with pre-emphasis level 1<br>• 10 = Training Pattern 2 with pre-emphasis level 2<br>• 11 = Training Pattern 2 with pre-emphasis level 3<br>[2] - MAX_SWING_REACHED: Set to 1 when the maximum driven current setting is reached.<br>[1:0] - VOLTAGE_SWING_SET<br>• 00 = Training Pattern 1 with voltage swing level 0<br>• 01 = Training Pattern 1 with voltage swing level 1<br>• 10 = Training Pattern 1 with voltage swing level 2<br>• 11 = Training Pattern 1 with voltage swing level 3 |
| 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 downspreading has been enabled.<br>[0] - SPREAD_AMP: Set to 1 for 0.5% spreading or 0 for none. |

*Table 25:* **DPCD Configuration Space** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x434 | RO | DPCD_MAIN_LINK_CHANNEL_CODING_SET. 8B/10B encoding can be disabled by the transmitter through this register bit.<br>[0] - Set to 0 to disable 8B/10B channel coding. The default is 1. |
| 0x438 | RO | PCD_SET_POWER_STATE. Power state requested by the source core. On reset, power state is set to power down mode.<br>[1:0] - Requested power state<br>• 00 = Reserved<br>• 01 = state D0, normal operation<br>• 10 = state D3, power down mode<br>• 11 = Reserved |
| 0x43C | RO | DPCD_LANE01_STATUS. Value of the lane 0 and lane 1 training status registers.<br>**Write-Only Access**<br>[31] - Override Lane Set Registers<br>[27:26] - Override Pre-Emphasis Level<br>[25:24] - Override Voltage Level<br>**Read-Only Access**<br>[15:14] - Lane 1 Adjust Pre-Emphasis Value<br>[13:12] - Lane 1 Adjust Voltage Value<br>[11:10] - Lane 0 Adjust Pre-Emphasis Value<br>[9:8] - Lane 0 Adjust Voltage Value<br>[6] - LANE_1_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_1.<br>[5] - LANE_1_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_1.<br>[4] - LANE_1_CLOCK_RECOVERY_DONE. Set to 1, when D10.2 symbol is locked on LANE_1.<br>[2] - LANE_0_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_0.<br>[1] - LANE_0_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_0.<br>[0] - LANE_0_CLOCK_RECOVERY_DONE. Set to 1, when D10.2 symbol is locked on LANE_0. |
| 0x440 | RO | DPCD_LANE23_STATUS. Value of the lane 2 and lane 3 training status registers.<br>**Read-Only Access**<br>[15:14] - Lane 3 Adjust Pre-Emphasis Value<br>[13:12] - Lane 3 Adjust Voltage Value<br>[11:10] - Lane 2 Adjust Pre-Emphasis Value<br>[9:8] - Lane 2 Adjust Voltage Value<br>[6] - LANE_3_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_3.<br>[5] - LANE_3_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_3.<br>[4] - LANE_3_CLOCK_RECOVERY_DONE. Set to 1, when D10.2 symbol is locked on LANE_3.<br>[2] - LANE_2_SYMBOL_LOCKED. Set to 1 when, the 8B10B pattern is locked and TP2/3 pattern is locked on LANE_2.<br>[1] - LANE_2_CHANNEL_EQ_DONE. Set to 1, when the lane alignment is done on LANE_2.<br>[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.<br>[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. |

*Table 25:* **DPCD Configuration Space** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 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.<br>[31] - Lane 1 error count valid<br>[30:16] - Lane 1 error count<br>[15] - Lane 0 error count valid<br>[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.<br>[31] - Lane 3 error count valid<br>[30:16] - Lane 3 error count<br>[15] - Lane 2 error count valid<br>[14:0] - Lane 2 error count |
| 0x452 | RO | DPCD Value written by DP Source<br>[31:0] - {5'd0, Link_Qual_Lane3, 5'd0,Link_Qual_Lane2, 5'd0,Link_Qual_Lane2, 5'd0,Link_Qual_Lane0} |
| 0x45C | R/W | [31:16] - PRBS Error Counter - Lane 1 {Valid, 15-bit counter value}<br>[15:0] - PRBS Error Counter - Lane 0 {Valid, 15-bit counter value} |
| 0x460 | R/W | [31:16] - PRBS Error Counter - Lane 3 {Valid, 15-bit counter value}<br>[15:0] - PRBS Error Counter - Lane 2 {Valid, 15-bit counter value} |

## *MSA Values*

*Table 26:* **MSA Values**

| Offset | Access Type | Description |
|---|---|---|
| 0x500 | RO | MSA_HRES. The horizontal resolution detected in the Main Stream Attributes.<br>[15:0] - Represents the number of pixels in a line of video |
| 0x504 | RO | MSA_HSPOL. Horizontal sync polarity.<br>[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.<br>[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.<br>[15:0] - Number of blanking cycles before active data |
| 0x510 | RO | MSA_HTOTAL. Tells the receiver core how many video clock cycles occur between leading edges of the horizontal sync pulse.<br>[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.<br>[15:0] - Vertical resolution of the received video |
| 0x518 | RO | MSA_VSPOL. Specifies the vertical sync polarity requested by the transmitter.<br>[0] - A value of 1 in this register indicates an active-High vertical sync. A 0 indicates an active-Low vertical sync. |

*Table 26:* **MSA Values** *(cont'd)*

| Offset | Access Type | Description |
|---|---|---|
| 0x51C | RO | MSA_VSWIDTH. The transmitter uses this value to specify the width of the vertical sync pulse in lines.<br>[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.<br>[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.<br>[15:0] - Total number of lines per video frame is contained in this value |
| 0x528 | RO | MSA_MISC0. Contains the value of the MISC0 attribute data.<br>[7:5] - COLOR_DEPTH: Number of bits per color/component.<br>[4] - YCbCR_COLOR: Set to 1 (ITU-R BT709-5) or 0 (ITU-R BT601-5).<br>[3] - DYNAMIC_RANGE: Set to 1 (CEA range) or 0 (VESA range).<br>[2:1] - COMPONENT_FORMAT:<br>• 00 = RGB<br>• 01 = YCbCr 4:2:2<br>• 10 = YCbCr 4:4:4<br>• 11 = Reserved<br>[0] - CLOCK_MODE:<br>• 0 = Asynchronous clock mode<br>• 1 = Synchronous clock mode |
| 0x52C | RO | MSA_MISC1. Contains the value of the MISC1 attribute data.<br>[7] - Implements the attribute information contained in the DisplayPort MISC1 register described in section 2.2.4 of the standard<br>[6:3] - RESERVED: Bits are always set to 0<br>[2:1] - STEREO_VIDEO: Used only when stereo video sources are being transmitted. See the *VESA DisplayPort Standard* (VESA website) section 2.24 for more information.<br>[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.<br>[23:0] - MVID: Value of the clock recovery M value. |
| 0x534 | RO | MSA_NVID. This attribute value is used to recover the video clock from the link clock. The recovered clock frequency depends on this value as well as the CLOCK_MODE and MSA_MVID registers.<br>[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.<br>[7:0] - VBID: See Table 2-3 (pg. 44) in the *VESA DisplayPort Standard* (VESA website) for more information. The default value is 0x19. |
| 0x540 to 0x578 | RO | Bit definition is identical to that of the MSA Values registers (0x500 to 0x538), but for MST STREAM 2. |
| 0x580 to 0x5B8 | RO | Bit definition is identical to that of the MSA Values registers (0x500 to 0x538), but for MST STREAM 3. |
| 0x5C0 to 0x5F8 | RO | Bit definition is identical to that of the MSA Values registers (0x500 to 0x538), but for MST STREAM 4. |

### *Vendor-Specific DPCD*

*Table 27:* **Vendor-Specific DPCD**

| Offset | Access Type | Description |
|---|---|---|
| 0xE00 to 0xEFC | R/W | SOURCE_DEVICE_SPECIFIC_FIELD. Access to Source specific field of DPCD address space. AXI accesses are all word-based (32 bits).<br>• 0xE00 to 0xE02: Read Only (IEEE OUI Value Programmed by Source)<br>• 0xE03 to 0xEFF: Write/Read |
| 0xF00 to 0xFFC | R/W | SINK_DEVICE_SPECIFIC_FIELD. Access to Sink specific field of DPCD address space. AXI accesses are all word-based (32 bits).<br>• 0xF00 to 0xF02: Read Only (IEEE OUI Value from GUI)<br>• 0xF03 to 0xFFF: Write/Read |

## AXI IIC Registers

For details about the AXI IIC registers, see the *AXI IIC Bus Interface LogiCORE IP Product Guide* (PG090).

## AXI Timer Registers

For details about the AXI Timer registers, see the *AXI Timer LogiCORE IP Product Guide* (PG079).

## HDCP Registers

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

*Chapter 4*

# 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 proper 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 the 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 does 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 your data interface.

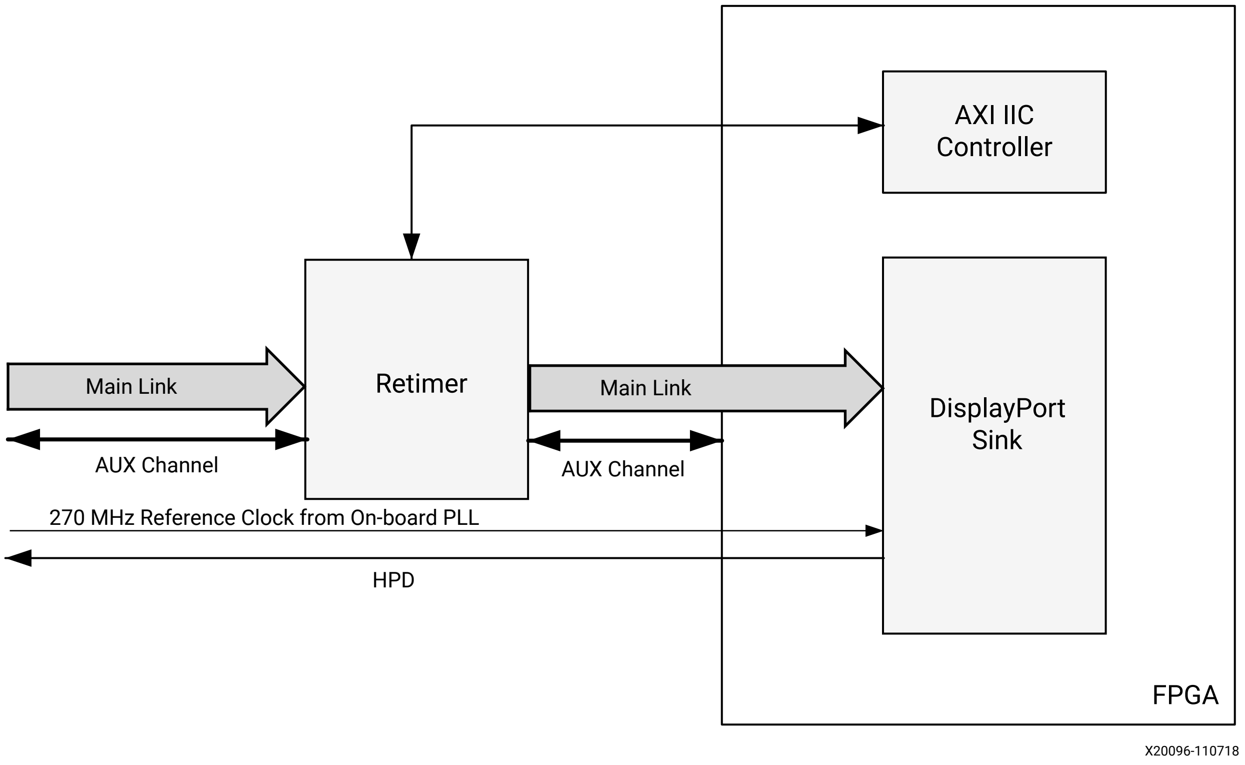
## MegaChips Retimer

Xilinx® expects the use of the MCDP6000 Retimer along with the DisplayPort 1.4 RX Subsystem solution. MCDP6000 as a retimer provides better SI features. As a retimer, the MCDP6000 removes the random and ISI jitter from video source. The MCDP6000 configuration is controlled through an I2C interface. The DisplayPort 1.4 RX Subsystem design needs an external I2C controller to configure the Retimer.

***Note***: The IIC controller must work at 400 kHz.

For details on reference clock requirements and its connectivity, see the related information.

*Figure 5:* **MCDP6000 Retimer**



Because the DisplayPort software driver handles the required MCDP6000 configuration, you do not have to control the MCDP6000 retimer. If the MCDP6000 is managed by a control other than the DisplayPort software driver, contact MegaChips (mca_support@megachips.com) for detailed programming information. The product page is http://www.megachips.com/products/displayport/MCDP60x0.

**Related Information**

Clocking

## Link Training

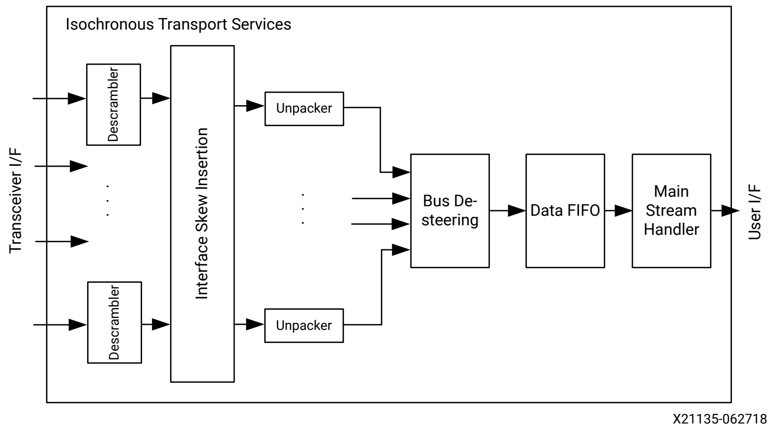
The link training commands are passed from the DPCD register block to the link training function. When set into the link training mode, the functional datapath is blocked, and the link training controller monitors the PHY and detects the specified pattern. Care must be taken to place the Sink core into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

The link training process is specified in the *VESA DisplayPort Standard* (VESA website).

The Main Link for the Sink core drives a stream of video data for you. 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.

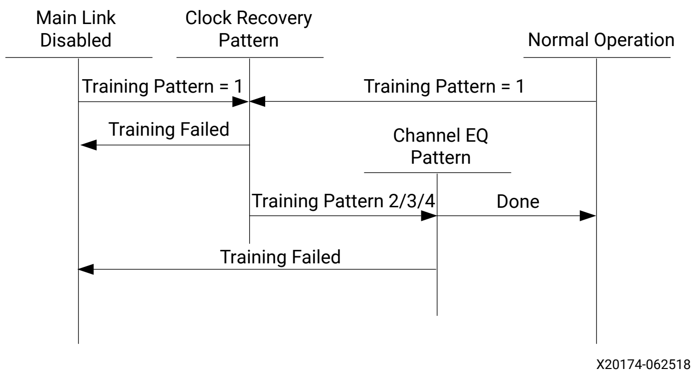
You can specify one, two, or four pixel-wide data through a register field. The bit width and format is determined from the Main Stream Attributes, which are provided as register fields.

*Figure 6:* **Sink Main Link Datapath**



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

*Figure 7:* **Link Training States**



## 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. You must either read the new MSA values from register space or use the dedicated ports provided on the Main Link to properly frame the video data.

### Cable is Unplugged, Lost Training

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

After the cable becomes 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.

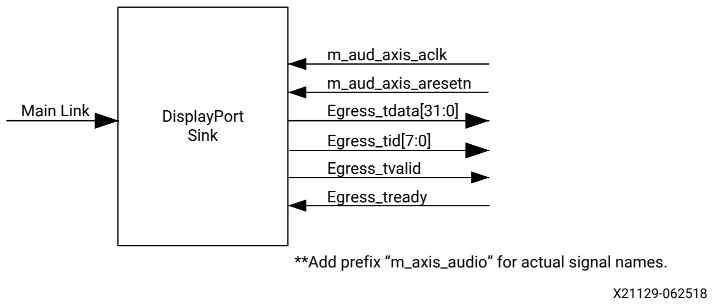
## Secondary Channel Operation

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 the following figure, 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 8:* **Audio Data Interface of DisplayPort Sink System**



### *Multi-Channel Audio*

The DisplayPort RX 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. The Stream ID received over the Info frame is also sent over the `TID[7:4]`. Samples for unallocated channels are dropped in the DisplayPort RX.

## Audio Management

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

### *Programming DisplayPort Sink*

1. Disable Audio by writing `0x00` to the RX_AUDIO_CONTROL register. The disable bit also flushes the buffers in the DisplayPort Sink. When there is a change in video/audio parameters, Xilinx® recommends following this step.
2. Enable audio by writing `0x01` to the RX_AUDIO_CONTROL register.
3. To read the 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.
5. Wait for some time (in µs).

### *Reading Info/Ext Packet*

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

#### Poll Mode

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

#### Interrupt Mode

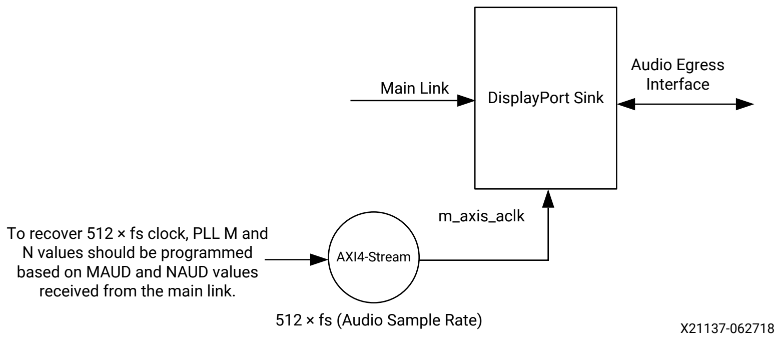
1. Ensure EXT_PKT_RXD/INFO_PKT_RXD interrupt is enabled by setting proper mask.
2. Wait for interrupt, Read interrupt cause register to check if EXT_PKT_RXD or INFO_PKT_RXD is set.
3. Based on interrupt status, read packet from appropriate buffer immediately.

### *Audio Clocking (Recommendation)*

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 9:* **Audio Clocking for Sink**


Main Link
DisplayPort Sink
Audio Egress Interface
m_axis_aclk
To recover 512 × fs clock, PLL M and N values should be programmed based on MAUD and NAUD values received from the main link.
AXI4-Stream
512 × fs (Audio Sample Rate)
X21137-062718

## Sampling Frequencies

The DisplayPort 1.4 RX Subsystem with a GT data width of 16-bit mode supports up to eight channels of audio with maximum supported sampling frequency of 192 kHz for all link rates.

## Programming the Core in MST Mode

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

### *Enabling MST*

The following steps are recommended to enable MST functionality:

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*

1. Wait for the DOWN_REQUEST_BUFFER_READY status interrupt, and read from DOWN_REQUEST_BUFFER. Continue to collect side-band messages as per *VESA DisplayPort Standard* (VESA website). 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, the Sink hardware updates the VC payload table by monitoring the AUX transactions. Alternatively, if you are an advanced user, you can use software to control the VC payload table (setting bit 1 in 0x0D0 enables it); 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. When the software finishes writing to the VC payload table, the software has to set bit 4 in 0x0D0.

   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 the VESA DisplayPort Standard (VESA website). The LINK_ADDRESS reply helps the source to identify the topology of the sink. For example, if the sink core is configured for four MST streams, it receives the multi-stream input from the DisplayPort RX and outputs four individual streams in native video format. In this case, the LINK_ADDRESS_REPLY can be modeled to contain one input and four output logical ports, with their DisplayPort device plug status set as 1 and Peer Device Type set as 3.*

## Reduced Blanking

The DisplayPort IP core 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 RX). 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 TX knows the resolution ahead of time and reset disable can be done during initialization. In the DisplayPort RX when the 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

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

## eDP Support

The DisplayPort RX subsystem supports the following eDP features:

- Reduced Aux timing
- Alternate Scrambler Seed Reset (ASSR)
- Enhanced Framing
- Fast link training
- Backlight control through Aux
- Black video

eDP can be enabled by using the following tcl command:

```
set_property -dict [list CONFIG.EDP_ENABLE {true}] [get_bd_cells
v_dp_rxss1_0]
```

## Pixel Mapping

### Pixel Mapping on AXI4-Stream Interface

***Note:*** 4PPC (QUAD pixel interface) requires 4 lanes. 2PPC (DUAL pixel interface) requires 2 lanes. 1PPC (single pixel interface) requires 1 lane. For more information on the relationship between PPC and lanes, see the description of offset address 0x010 in Table 19: Receiver Core Configuration.

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

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

- For a 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 × 3

Interface Width = Pixel Width × LANE_COUNT
```

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

## *Pixel Mapping Examples on AXI4-Stream Interface*

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

*Table 28:* **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 |
| 16 | 2 | 48 | 96 | 16 | - | - | - | - | - | - | 95:80 | 79:64 | 63:48 | 47:32 | 31:16 | 15:0 |
| 16 | 1 | 48 | 48 | 16 | - | - | - | - | - | - | - | - | - | 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 |
| 12 | 2 | 36 | 72 | 12 | - | - | - | - | - | - | 71:60 | 59:48 | 47:36 | 35:24 | 23:12 | 11:0 |
| 12 | 1 | 36 | 40 | 12 | - | - | - | - | - | - | - | - | - | 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 |
| 10 | 2 | 30 | 64 | 10 | - | - | - | - | - | - | 59:50 | 49:40 | 39:30 | 29:20 | 19:10 | 9:0 |
| 10 | 1 | 30 | 32 | 10 | - | - | - | - | - | - | - | - | - | 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 |
| 8 | 2 | 24 | 48 | 8 | - | - | - | - | - | - | 47:40 | 39:32 | 31:24 | 23:16 | 15:8 | 7:0 |
| 8 | 1 | 24 | 24 | 8 | - | - | - | - | - | - | - | - | - | 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 |
| 16 | 2 | 48 | 96 | 12 | - | - | - | - | - | - | 95:84 | 79:68 | 63:52 | 47:36 | 31:20 | 15:4 |
| 16 | 1 | 48 | 48 | 12 | - | - | - | - | - | - | - | - | - | 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 |
| 12 | 2 | 36 | 72 | 10 | - | - | - | - | - | - | 71:62 | 59:50 | 47:38 | 35:26 | 23:14 | 11:2 |
| 12 | 1 | 36 | 40 | 10 | - | - | - | - | - | - | - | - | - | 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 |
| 10 | 2 | 30 | 64 | 8 | - | - | - | - | - | - | 59:52 | 49:42 | 39:32 | 29:22 | 19:12 | 9:2 |
| 10 | 1 | 30 | 32 | 8 | - | - | - | - | - | - | - | - | - | 29:22 | 19:12 | 9:2 |

*Table 28:* **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 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 8 | 2 | 24 | 48 | 6 | - | - | - | - | - | - | 47:42 | 39:34 | 31:26 | 23:18 | 15:10 | 7:2 |
| 8 | 1 | 24 | 24 | 6 | - | - | - | - | - | - | - | - | - | 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 |
| 16 | 2 | 48 | 96 | 10 | - | - | - | - | - | - | 95:86 | 79:70 | 63:54 | 47:38 | 31:22 | 15:6 |
| 16 | 1 | 48 | 48 | 10 | - | - | - | - | - | - | - | - | - | 47:38 | 31:22 | 15:6 |
| 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 |
| 12 | 2 | 36 | 72 | 8 | - | - | - | - | - | - | 71:64 | 59:52 | 47:40 | 35:28 | 23:16 | 11:4 |
| 12 | 1 | 36 | 40 | 8 | - | - | - | - | - | - | - | - | - | 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 |
| 10 | 2 | 30 | 64 | 6 | - | - | - | - | - | - | 59:54 | 49:44 | 39:34 | 29:24 | 19:14 | 9:4 |
| 10 | 1 | 30 | 32 | 6 | - | - | - | - | - | - | - | - | - | 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 |
| 16 | 2 | 48 | 96 | 8 | - | - | - | - | - | - | 95:88 | 79:72 | 63:56 | 47:40 | 31:24 | 15:8 |
| 16 | 1 | 48 | 48 | 8 | - | - | - | - | - | - | - | - | - | 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 |
| 12 | 2 | 36 | 72 | 6 | - | - | - | - | - | - | 71:66 | 59:54 | 47:42 | 35:30 | 23:18 | 11:6 |
| 12 | 1 | 36 | 36 | 6 | - | - | - | - | - | - | - | - | - | 35:30 | 23:18 | 11:6 |
| 16 | 4 | 48 | 192 | 6 | 191:186 | 175:170 | 159:154 | 143:138 | 127:122 | 111:106 | 95:90 | 79:74 | 63:58 | 47:42 | 31:26 | 15:10 |
| 16 | 2 | 48 | 96 | 6 | - | - | - | - | - | - | 95:90 | 79:74 | 63:58 | 47:42 | 31:26 | 15:10 |
| 16 | 1 | 48 | 48 | 6 | - | - | - | - | - | - | - | - | - | 47:42 | 31:26 | 15:10 |

**Notes:**

1. The padding bits are zeros.

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

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

*Table 29:* **Pixel Mapping Examples on AXI4-Stream Interface (UG934-Compliant Mode)**

| MAX_BPC | LANES | Pixel Width | Interface Width | Video Sampling Mode | Pixel 3 | | | Pixel 2 | | | Pixel 1 | | | Pixel 0 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 4 | 24 | 96 | 444 | 95:88 | 87:80 | 79:72 | 71:64 | 63:56 | 55:48 | 47:40 | 39:32 | 31:24 | 23:16 | 15:8 | 7:0 |
| | 4 | 16 | 64 | 422 | 63:56 | | 55:48 | 47:40 | | 39:32 | 31:24 | | 23:16 | 15:8 | | 7:0 |
| | 4 | 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 |
| | 2 | 16 | 32 | 422 | - | | - | - | | - | 31:24 | | 23:16 | 15:8 | | 7:0 |
| | 2 | 8 | 16 | Y-Only | | | - | | | - | | | 15:8 | | | 7:0 |
| | 1 | 24 | 24 | 444 | - | - | - | - | - | - | - | - | - | 23:16 | 15:8 | 7:0 |
| | 1 | 16 | 16 | 422 | - | | - | - | | - | - | | - | 15:8 | | 7:0 |
| | 1 | 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 |
| | 4 | 20 | 80 | 422 | 79:72 | | 69:62 | 59:52 | | 49:42 | 39:32 | | 29:22 | 19:12 | | 9:2 |
| | 4 | 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 |
| | 2 | 20 | 40 | 422 | - | | - | - | | - | 39:32 | | 29:22 | 19:12 | | 9:2 |
| | 2 | 10 | 20 | Y-Only | | | - | | | - | | | 19:12 | | | 9:2 |
| | 1 | 30 | 30 | 444 | - | - | - | - | - | - | - | - | - | 29:22 | 19:12 | 9:2 |
| | 1 | 20 | 20 | 422 | - | | - | - | | - | - | | - | 19:12 | | 9.2 |
| | 1 | 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 |
| | 4 | 24 | 96 | 422 | 95:88 | | 83:76 | 71:64 | | 59:52 | 47:40 | | 35:28 | 23:16 | | 11:4 |
| | 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 |
| | 2 | 24 | 48 | 422 | - | | - | - | | - | 47:40 | | 35:28 | 23:16 | | 11:4 |
| | 2 | 12 | 24 | Y-Only | | | - | | | - | | | 23:16 | | | 11:4 |
| | 1 | 36 | 36 | 444 | - | - | - | - | - | - | - | - | - | 35:28 | 23:16 | 11:4 |
| | 1 | 24 | 24 | 422 | - | | - | - | | - | - | | - | 23:16 | | 11:4 |
| | 1 | 12 | 12 | Y-Only | | | - | | | - | | | - | | | 11:4 |

*Table 29:* **Pixel Mapping Examples on AXI4-Stream Interface (UG934-Compliant Mode)** *(cont'd)*

| MAX_BPC | LANES | Pixel Width | Interface Width | Video Sampling Mode | Pixel 3 | | | Pixel 2 | | | Pixel 1 | | | Pixel 0 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| | 4 | 32 | 128 | 422 | 127:120 | | 111:104 | 95:88 | | 79:72 | 63:56 | | 47:40 | 31:24 | | 15:8 |
| | 4 | 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 |
| | 2 | 32 | 64 | 422 | - | | - | - | | - | 63:56 | | 47:40 | 31:24 | | 15:8 |
| | 2 | 16 | 32 | Y-Only | | | - | | | - | | | 31:24 | | | 15:8 |
| | 1 | 48 | 48 | 444 | - | - | - | - | - | - | - | - | - | 47:40 | 31:24 | 15:8 |
| | 1 | 32 | 32 | 422 | - | | - | - | | - | - | | - | 31:24 | | 15:8 |
| | 1 | 16 | 16 | Y-Only | | | - | | | - | | | - | | | 15:8 |

**Notes:**

1. Video BPC is 8.

## Pixel Mapping on Native Video Interface

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

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

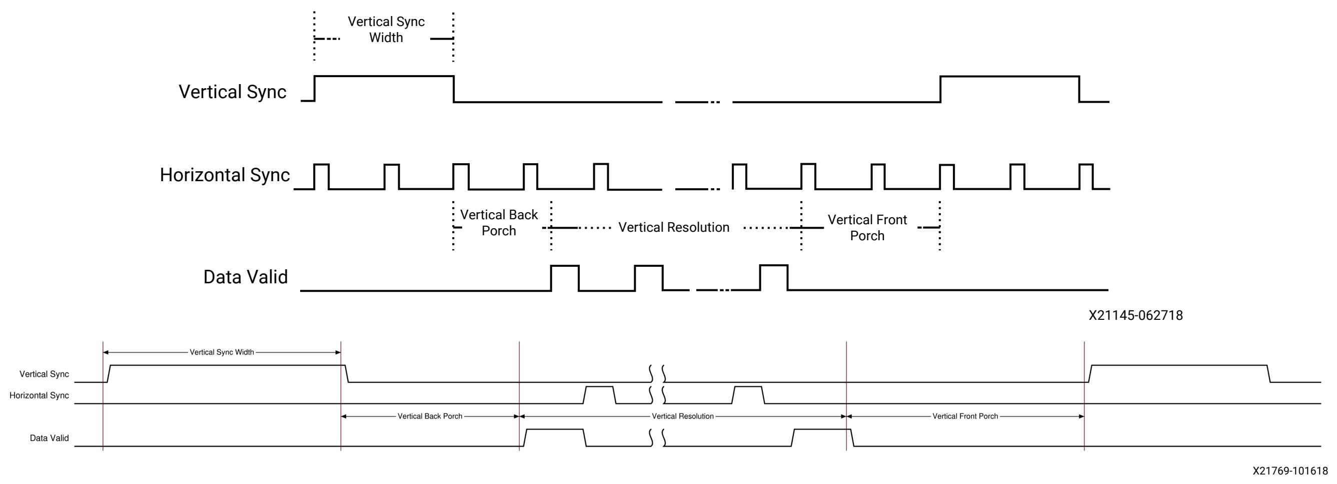
At the trailing edge of each vertical sync pulse, the user data interface resets key elements of the image datapath. This provides for a robust user interface that recovers from any kind of interface error in one vertical interval or less.

You have 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.

**RECOMMENDED:** *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.*

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

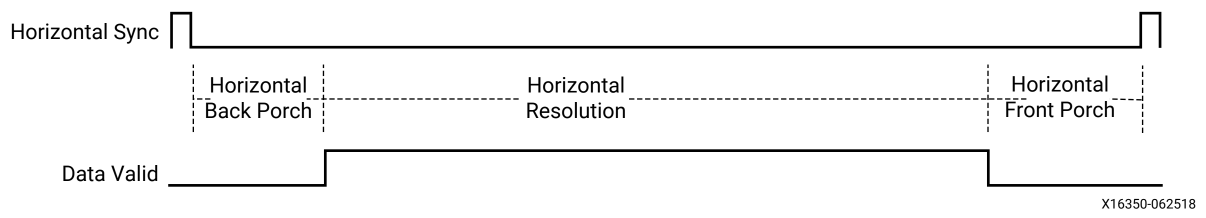
*Figure 10:* **User Interface Vertical Timing**



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

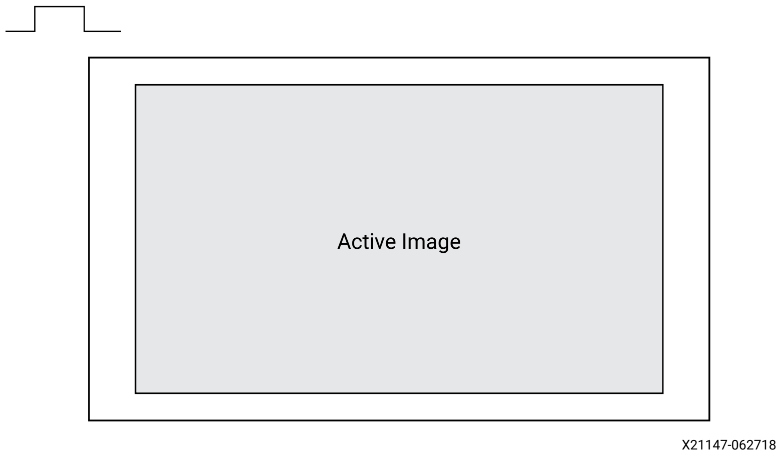
***Note*:** Data Valid might toggle if using a fast clock. Also, Data Valid signal might remain asserted for the duration of a scan line. Dropping the valid signal might result in improper operation.

*Figure 11:* **User Interface Horizontal Timing**



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

*Figure 12:* **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 > 1. The `vid_pixel` width is always 48-bits, regardless if all bits are used. For pixel mappings that do not require all 48 bits, the convention used for this core is to occupy the MSB bits first and leave the lower bits either untied or driven to zero.

## Pixel Mapping Examples on the User Data Interface

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

*Table 30:* **Pixel Mapping Examples on the User Data Interface**

| Format | BPC/BPP | R | G | B | Cr | Y | Cb | Cr/Cb | Y |
|---|---|---|---|---|---|---|---|---|---|
| RGB | 6/18 | [47:42] | [31:26] | [15:10] | - | - | - | - | - |
| RGB | 8/24 | [47:40] | [31:24] | [15:8] | - | - | - | - | - |
| RGB | 10/30 | [47:38] | [31:22] | [15:6] | - | - | - | - | - |
| RGB | 12/36 | [47:36] | [31:20] | [15:4] | - | - | - | - | - |
| RGB | 16/48 | [47:32] | [31:16] | [15:0] | - | - | - | - | - |
| YCrCb444 | 6/18 | - | - | - | [47:42] | [31:26] | [15:10] | - | - |
| YCrCb444 | 8/24 | - | - | - | [47:40] | [31:24] | [15:8] | - | - |
| YCrCb444 | 10/30 | - | - | - | [47:38] | [31:22] | [15:6] | - | - |
| YCrCb444 | 12/36 | - | - | - | [47:36] | [31:20] | [15:4] | - | - |
| YCrCb444 | 16/48 | - | - | - | [47:32] | [31:16] | [15:0] | - | - |
| YCrCb422 | 8/16 | - | - | - | - | - | - | [47:40] | [31:24] |
| YCrCb422 | 10/20 | - | - | - | - | - | - | [47:38] | [31:22] |
| YCrCb422 | 12/24 | - | - | - | - | - | - | [47:36] | [31:20] |
| YCrCb422 | 16/32 | - | - | - | - | - | - | [47:32] | [31:16] |
| YONLY | 8/8 | - | - | - | - | - | - | - | [47:40] |
| YONLY | 10/10 | - | - | - | - | - | - | - | [47:38] |

*Table 30:* **Pixel Mapping Examples on the User Data Interface** *(cont'd)*

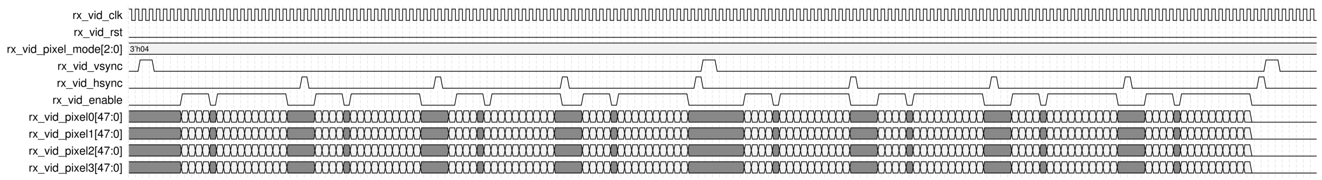
| Format | BPC/BPP | R | G | B | Cr | Y | Cb | Cr/Cb | Y |
|---|---|---|---|---|---|---|---|---|---|
| YONLY | 12/12 | - | - | - | - | - | - | - | [47:36] |
| YONLY | 16/16 | - | - | - | - | - | - | - | [47:32] |

**Notes:**

1. For a YCrCb 4:2:2, the input follows YCr, YCb, YCr, YCb, and so on. This means Cr and Cb are mapped to the same bits on the video 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. DisplayPort 1.4 RX Subsystem 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 over MSA. The interface timing in this case is shown in the following figure.

*Figure 13:* **RX Pixel Timing**



***Note***: The width of `rx_vid_vsync`, `rx_vid_hsync`, `rx_vid_enable`, and the number of hsync pulses shown in the previous figure above 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 MSA.

## AXI4-Stream Interface Color Mapping

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

*Table 31:* **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 the Pixel Mapping Examples on AXI4-Stream Interface.

**Related Information**

Pixel Mapping Examples on AXI4-Stream Interface

## Clocking

This section describes the link clock (`rx_lnk_clk`), video clock (`rx_vid_clk`), and Video Bridge to AXI4-Stream master interface clock. When the AXI4 stream interface is selected for DP RX, the output clock of Video to AXI4-stream bridge that is `m_axis_aclk_stream` should be selected such that it is greater than or equal to `rx_vid_clk`.

The `rx_lnk_clk` is a link clock input to the DisplayPort 1.4 RX Subsystem generated by the Video PHY (GT).

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

The following table shows the clock ranges.

*Table 32:* **Clock Ranges**

| Clock Domain | Min (MHz) | Max (MHz) | Description |
|---|---|---|---|
| rx_lnk_clk | 81 | 405 | Link clock |
| rx_vid_clk | 150 | 300 | Video clock |
| s_axi_aclk | 25 | 135 | Host processor clock |

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.When the lanes are running at 8.1 Gb/s, `lnk_clk` operates at 405 MHz.

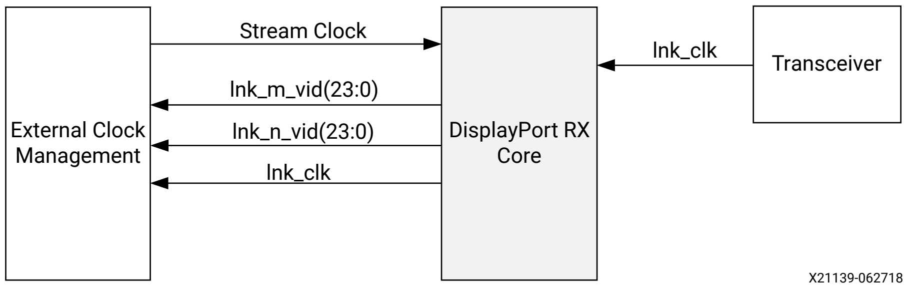
  In the DisplayPort Sink core, `lnk_clk` is derived from the recovered clock from the transceiver. When the cable is disconnected this clock becomes unstable.

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

- **vid_clk:** This is the primary user interface clock. Frequency of this clock is dependent on whether "frame buffer" is being used in the design or not.
  - **Frame Buffer:** With a frame buffer, `vid_clk` can run up to 300 MHz and must be at least equal to the pixel clock..This is the Xilinx® implemented solution as it does not require any external clock management.
  - **Without Frame Buffer:** For non-frame buffer designs, the DisplayPort RX 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 RX core places this information on dedicated signals and provides an update flag to signal a change in these values. The following figure shows how to use the M and N values from the core to generate a clock. For more details, see the *VESA DisplayPort Standard* (VESA website).

**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. For more details, see the VESA DisplayPort Standard (VESA website).*

*Figure 14:* **Receiver Clock Generation**



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

- **aud_clk:** This is the audio interface clock. The frequency will be equal to 512 × audio sample rate.
- **s_aud_axis_aclk:** This clock is used by the source audio streaming interface. This clock should be = 512 × audio sample rate.
- **m_aud_axis_aclk:** This clock is used by the sink audio streaming interface. This clock should be = 512 × audio sample rate.

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

## Resets

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

- **s_axi_aresetn**: Active-Low AXI4-Lite reset. This resets all the programming registers.
- **rx_vid_rst**: Active-High video pipe reset. MST with four streams, there are four video resets.
- **mcdp6000_rst** : Active-High soft reset to the MCDP6000 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 Bus Interface LogiCORE IP Product Guide* (PG090).

## Address Map Example

The following table shows an example based on a subsystem base address of `0x44C0_0000` (14 bits). There are no registers in the Video to AXI4-Stream bridge.

*Table 33:* **Address Map Example**

| IP Core/Subsystem | SST | MST |
|---|---|---|
| DisplayPort 1.4 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 LogiCORE IP Product Guide* (PG230).

# Design Flow Steps

This section describes customizing and generating the subsystem, constraining the subsystem, and the simulation, synthesis, and implementation steps that are specific to this IP subsystem. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

## Customizing and Generating the Subsystem

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

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

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

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

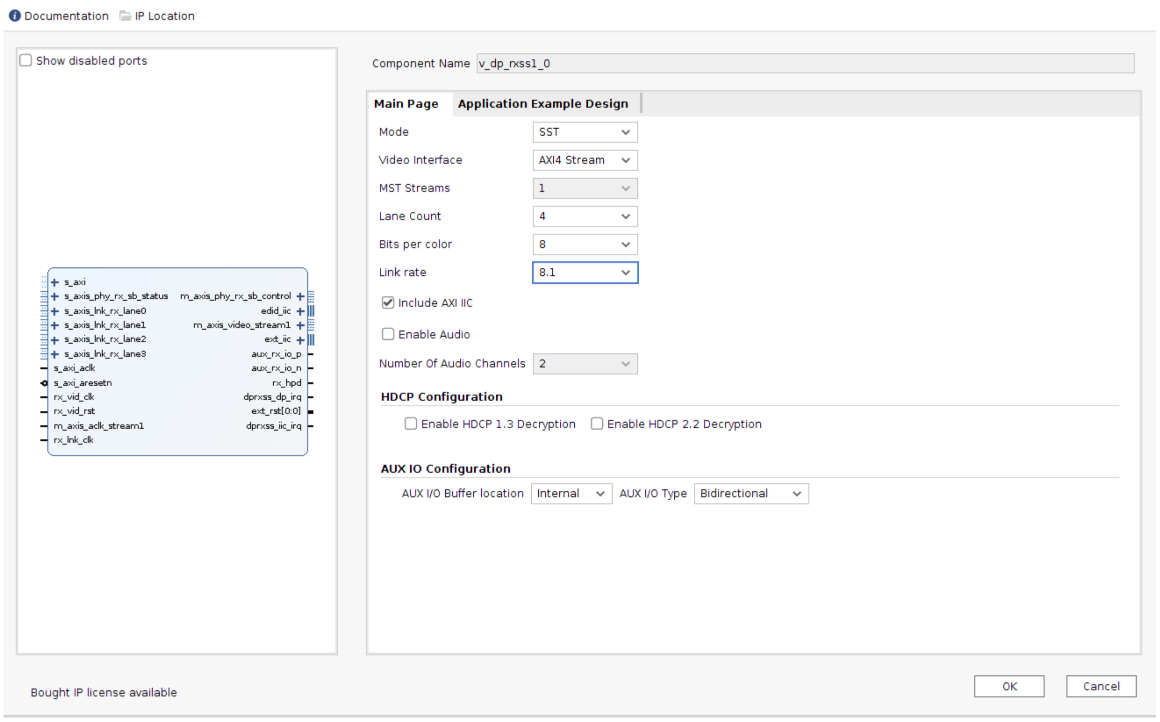
For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

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

## Subsystem Configuration Screen

The subsystem configuration screen is shown in the following figure.

*Figure 15:* **Configuration Screen**



☐ Show disabled ports

Component Name v_dp_rxss1_0

| | |
|---|---|
| Mode | SST |
| Video Interface | AXI4 Stream |
| MST Streams | 1 |
| Lane Count | 4 |
| Bits per color | 8 |
| Link rate | 8.1 |

☑ Include AXI IIC

☐ Enable Audio

Number Of Audio Channels 2

**HDCP Configuration**

☐ Enable HDCP 1.3 Decryption ☐ Enable HDCP 2.2 Decryption

**AUX IO Configuration**

AUX I/O Buffer location Internal AUX I/O Type Bidirectional

Bought IP license available

- **Component Name:** The Component Name is used as the name of the top-level wrapper file for the core. The underlying netlist still retains its original name. Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9, and "_". The name displayport_0 is used as internal module name and should not be used for the component name. The default is v_dp_rxss1_0.
- **Mode:** Selects the desired resolution for the DisplayPort IP. Options are SST or MST.
- **Video Interface:** Selects the AXI4-Streamor native for the input video interface.
- **MST Streams:** Selects the maximum number of streams in MST mode.
- **Lane Count:** Selects the maximum number of lanes.
- **Bits Per Color:** Selects the desired maximum bit per component (BPC).
- **Include AXI IIC:** Select to include AXI IIC in DisplayPort subsystem.
- **Link Rate:** Selects the desired link rate in Gb/s.
- **Enable HDCP 1.3 Encryption:** Enables the HDCP 1.3 encryption.
- **Enable HDCP 2.2 Encryption:** Enables the HDCP 2.2 encryption.

- **Enable Audio:** Enables the audio support.
- **Audio Channels:** Selects the number of audio channels.
- **AUX I/O Buffer location:** Selects the buffer location for AUX channel.
- **AUX I/O Type:** Selects the Bidirectional or Unidirectional buffer type.

## User Parameters

The following table shows the relationship between the fields in the Vivado® IDE and the user parameters (which can be viewed in the Tcl Console).

*Table 34:* **User Parameters**

| Vivado IDE Parameter/Value | User Parameter/Value | Default Value |
|---|---|---|
| Mode | MODE | SST |
| PHY Data Width | PHY_DATA_WIDTH | 16 |
| Video Interface | VIDEO_INTERFACE | AXI4-Stream |
| MST Streams | NUM_STREAMS | 1 |
| Lane Count | LANE_COUNT | 4 |
| Bits Per Color | BITS_PER_COLOR | 8 |
| Enable HDCP 1.3 | HDCP_ENABLE | 0 |
| Enable Audio | AUDIO_ENABLE | 0 |
| Enable HDCP 2.2 | HDCP22_ENABLE | 0 |
| Number Of Audio Channels | AUDIO_CHANNELS | 2 |
| Pixel Mode | PIXEL_MODE | 1/2/4 (Valid only in native mode) |
| AUX I/O Buffer Location | AUX_IO_LOC | Internal |
| AUX I/O Type | AUX_IO_TYPE | Bidirectional |
| eDP | EDP_ENABLE | 0 |
| Link Rate | LINK_RATE | 8.1 |
| Include AXI IIC | INCLUDE_AXI_IIC | 1 |

## Output Generation

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

# Constraining the Subsystem

### Required Constraints

This section is not applicable for this IP subsystem.

### Device, Package, and Speed Grade Selections

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

This section is not applicable for this IP subsystem.

### Clock Frequencies

See Clocking for more details about clock frequencies.

### Clock Management

This section is not applicable for this IP subsystem.

### Clock Placement

This section is not applicable for this IP subsystem.

### Banking

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

### Transceiver Placement

Transceiver is external to DisplayPort 1.4 RX Subsystem *Video PHY Controller LogiCORE IP Product Guide* (PG230).

### I/O Standard and Placement

This section is not applicable for this IP subsystem.

### AUX Channel

The *VESA DisplayPort Standard* (VESA website) describes the AUX channel as a bidirectional LVDS signal. For , hence there are no specific transceiver placement constraints. For more information on the specific transceiver placement constraints, see the 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 UltraScale+™ and UltraScale™ families supporting HR IO banks, use the following Sink constraints:

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

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

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

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

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

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

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

**Related Information**
IP Facts
Clocking

## Simulation

There is no simulation support for DisplayPort 1.4 RX Subsystem.

## Synthesis and Implementation

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

# Example Design

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

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

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

**Related Information**

Setting the FMC Voltage to 1.8V

## Available Example Designs

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

*Table 35:* **Available Example Designs**

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

*Table 35:* **Available Example Designs** *(cont'd)*

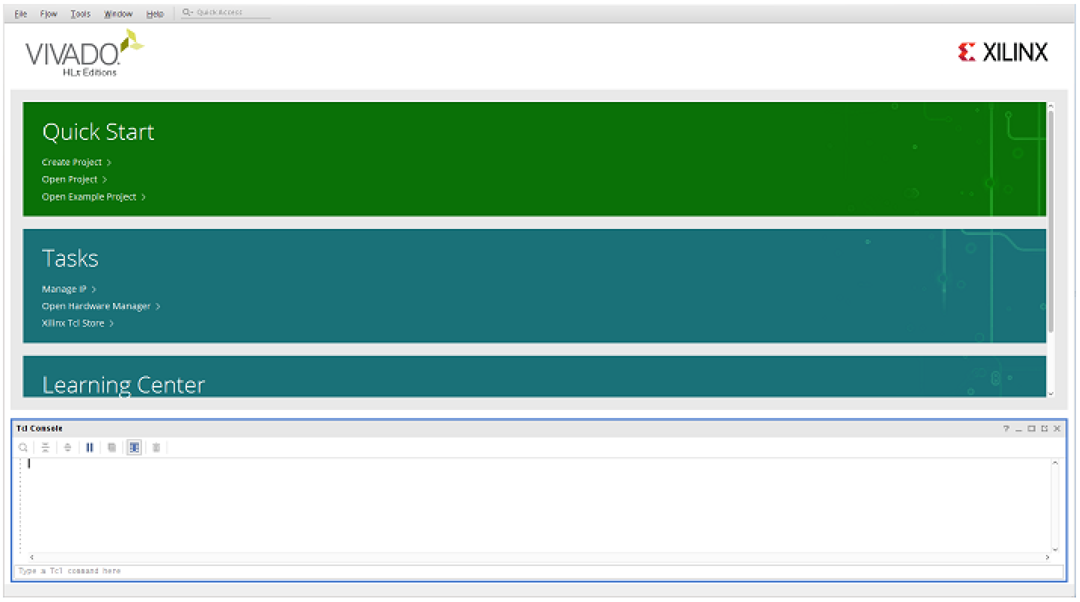
| GT Type | Topology | Video PHY Config (TXPLL) | Video PHY Config (RXPLL) | Hardware | BPC | Processor |
|---|---|---|---|---|---|---|
| GTHE4 | RX only | - | CPLL | ZCU102 + Inrevium TB-FMCH-VFMC-DP | 10 | A53 |
| | TX only | QPLL | - | ZCU102 + Inrevium TB-FMCH-VFMC-DP | 10 | A53 |
| | FB Pass-through without HDCP1.3/HDCP2.2 | QPLL | CPLL | ZCU102 + Inrevium TB-FMCH-VFMC-DP | 10 | A53 |
| | FB Pass-through with HDCP1.3 and HDCP2.2 | QPLL | CPLL | ZCU102 + Inrevium TB-FMCH-VFMC-DP | 10 | A53 |
| | MST FB Pass-through without HDCP1.3 and TX only | QPLL | CPLL | ZCU102+ Inrevium TB-FMCH-VFMC-DP | 10 | A53 |
| GTYE4 | RX only | - | CPLL | VCU118 + Inrevium TB-FMCH-VFMC-DP | 10 | MicroBlaze |
| | TX only | QPLL | - | VCU118 + Inrevium TB-FMCH-VFMC-DP | 10 | MicroBlaze |

**Notes:**

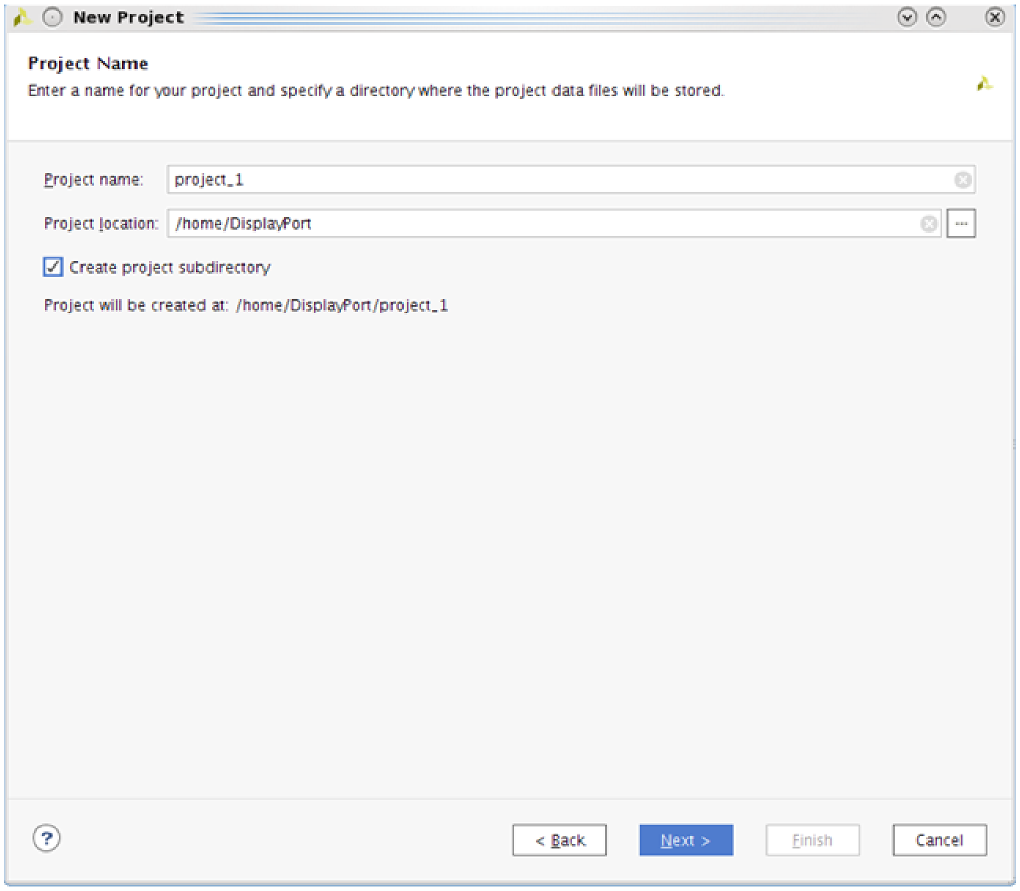
1. GT data width is 2 bytes.

## 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.



Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

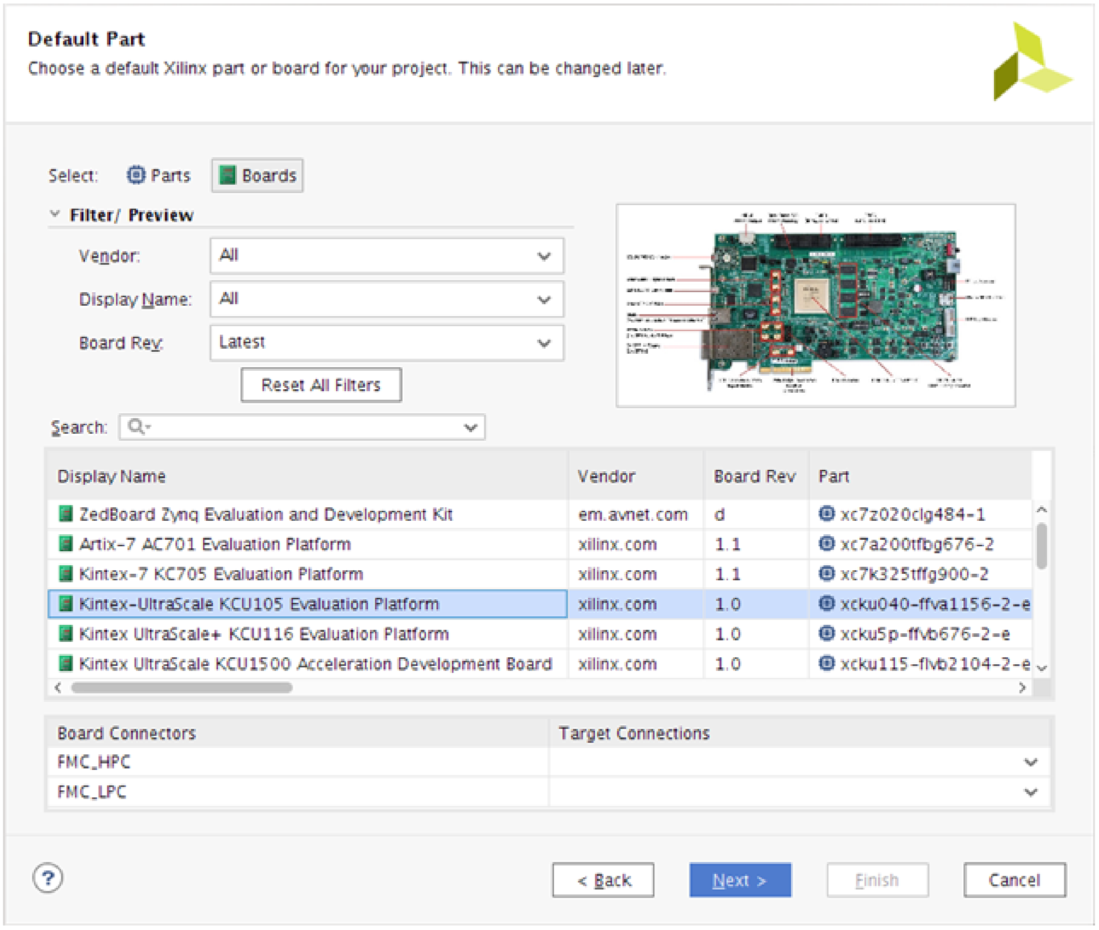
Project name: project_1

Project location: /home/DisplayPort

☑ Create project subdirectory

Project will be created at: /home/DisplayPort/project_1

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



**Default Part**

Choose a default Xilinx part or board for your project. This can be changed later.

Select: Parts | Boards

Filter/ Preview

Vendor: All

Display Name: All

Board Rev: Latest

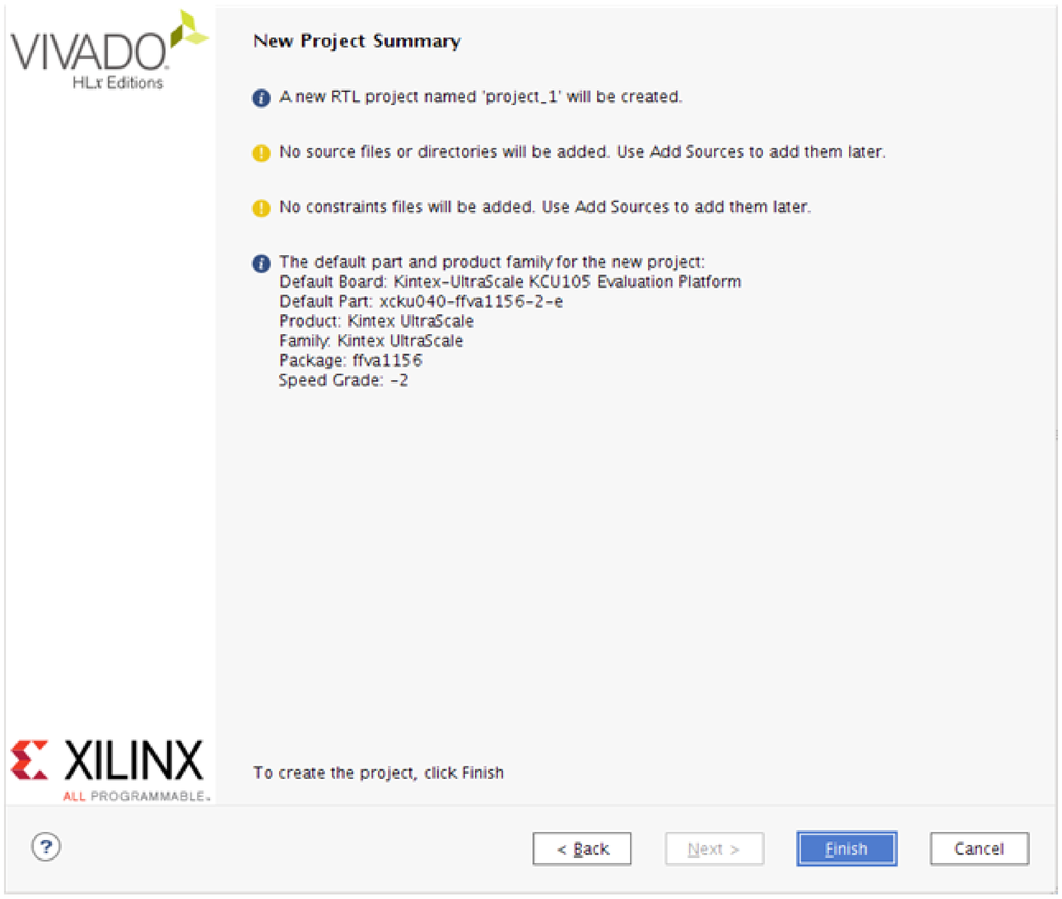
Reset All Filters

Search:

| Display Name | Vendor | Board Rev | Part |
|---|---|---|---|
| ZedBoard Zynq Evaluation and Development Kit | em.avnet.com | d | xc7z020clg484-1 |
| Artix-7 AC701 Evaluation Platform | xilinx.com | 1.1 | xc7a200tfbg676-2 |
| Kintex-7 KC705 Evaluation Platform | xilinx.com | 1.1 | xc7k325tffg900-2 |
| Kintex-UltraScale KCU105 Evaluation Platform | xilinx.com | 1.0 | xcku040-ffva1156-2-e |
| Kintex UltraScale+ KCU116 Evaluation Platform | xilinx.com | 1.0 | xcku5p-ffvb676-2-e |
| Kintex UltraScale KCU1500 Acceleration Development Board | xilinx.com | 1.0 | xcku115-flvb2104-2-e |

| Board Connectors | Target Connections |
|---|---|
| FMC_HPC | |
| FMC_LPC | |

4. Click **Finish**.


New Project Summary

A new RTL project named 'project_1' will be created.

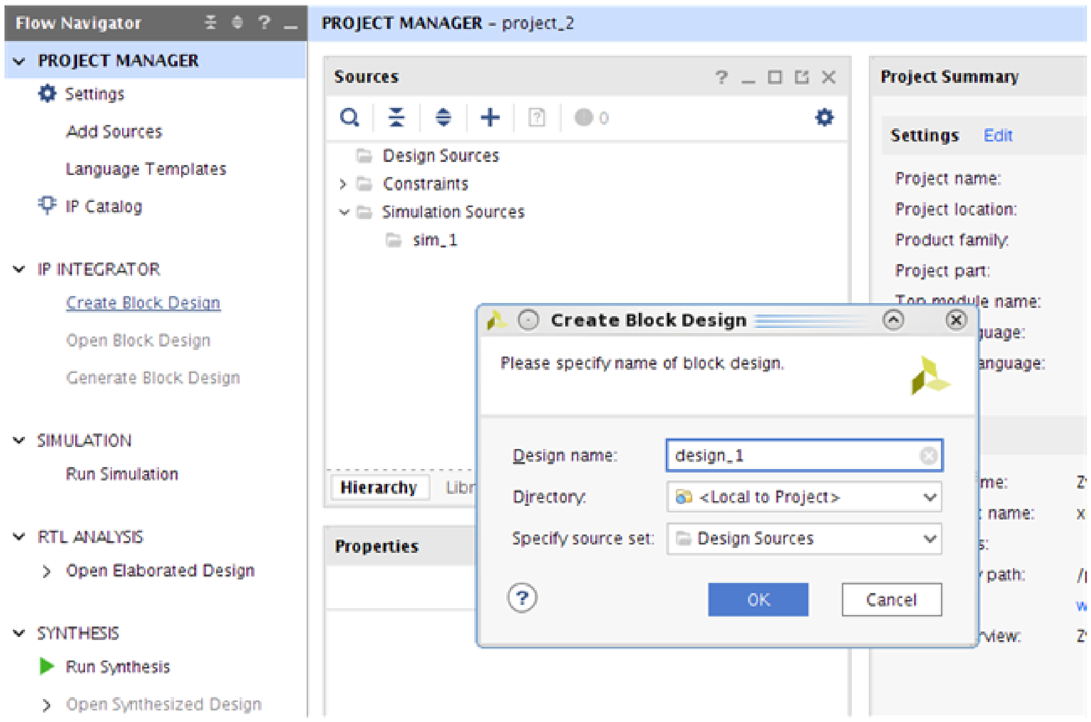
No source files or directories will be added. Use Add Sources to add them later.

No constraints files will be added. Use Add Sources to add them later.

The default part and product family for the new project:
Default Board: Kintex-UltraScale KCU105 Evaluation Platform
Default Part: xcku040-ffva1156-2-e
Product: Kintex UltraScale
Family: Kintex UltraScale
Package: ffva1156
Speed Grade: -2

To create the project, click Finish

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



Create Block Design

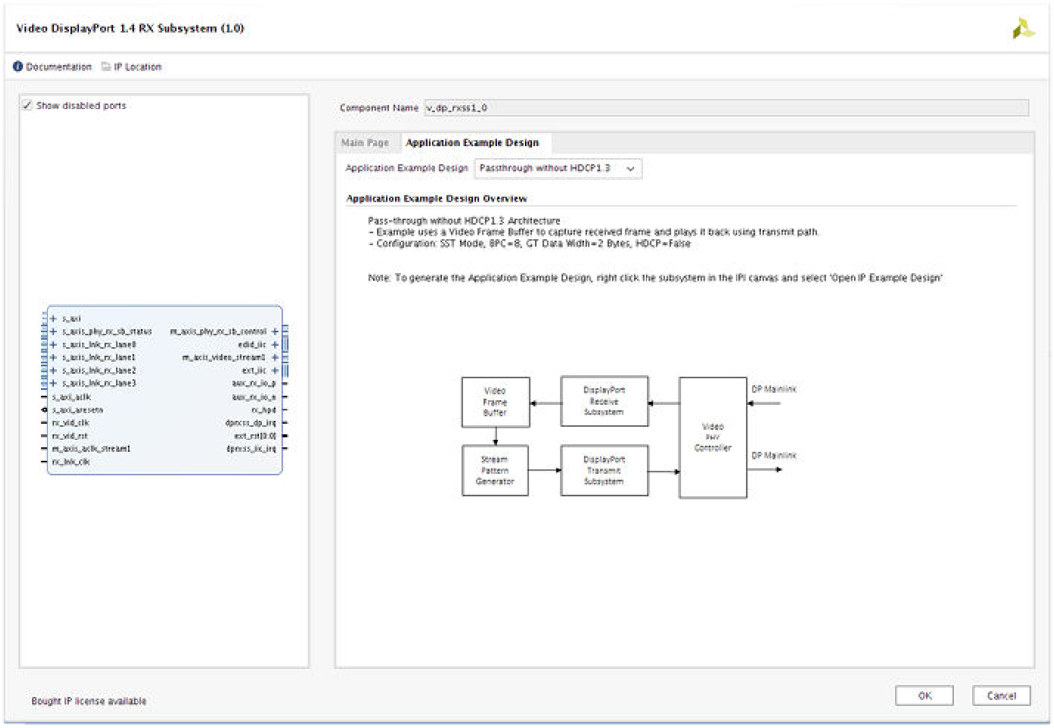
Please specify name of block design.

Design name: design_1

Directory: <Local to Project>

Specify source set: Design Sources

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


Video DisplayPort 1.4 RX Subsystem (1.0)

Component Name: v_dp_rxss1_0

Application Example Design: Passthrough without HDCP1.3

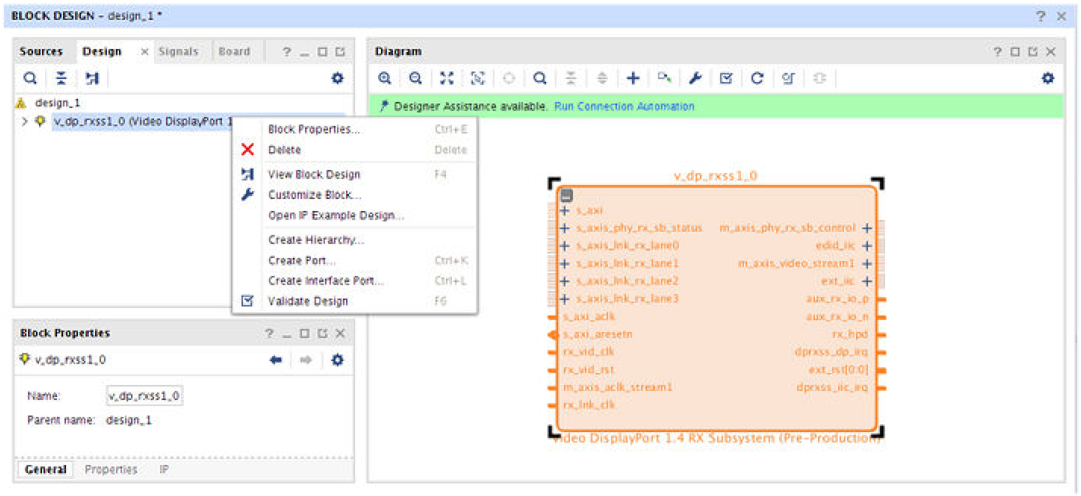
**Application Example Design Overview**

Pass-through without HDCP1.3 Architecture
- Example uses a Video Frame Buffer to capture received frame and plays it back using transmit path.
- Configuration: SST Mode, BPC=8, GT Data Width=2 Bytes, HDCP=False

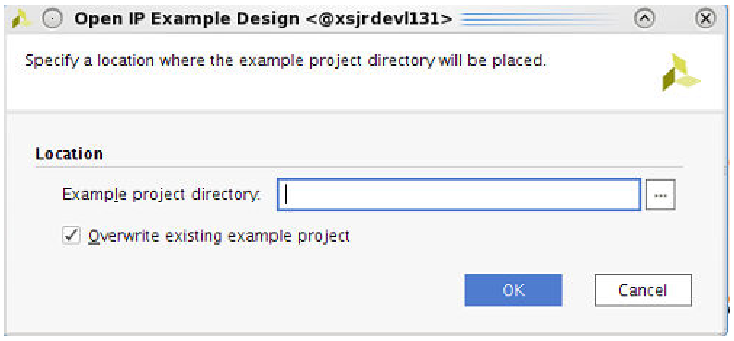
Note: To generate the Application Example Design, right click the subsystem in the IPI canvas and select 'Open IP Example Design'

Bought IP license available

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


BLOCK DESIGN - design_1 *
Sources
Design
Signals
Board
design_1
v_dp_rxss1_0 (Video DisplayPort 1
Block Properties... Ctrl+E
Delete Delete
View Block Design F4
Customize Block...
Open IP Example Design...
Create Hierarchy...
Create Port... Ctrl+K
Create Interface Port... Ctrl+L
Validate Design F6
Diagram
Designer Assistance available. Run Connection Automation
v_dp_rxss1_0
Video DisplayPort 1.4 RX Subsystem (Pre-Production)
Block Properties
v_dp_rxss1_0
Name: v_dp_rxss1_0
Parent name: design_1
General
Properties
IP

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



Open IP Example Design <@xsjrdevl131>

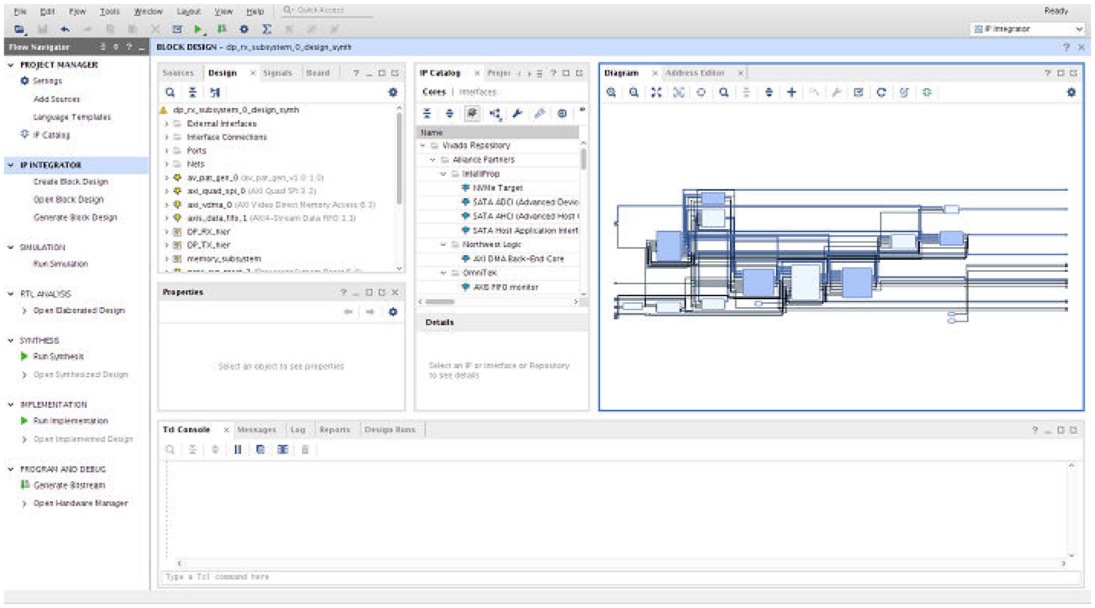
Specify a location where the example project directory will be placed.

Location

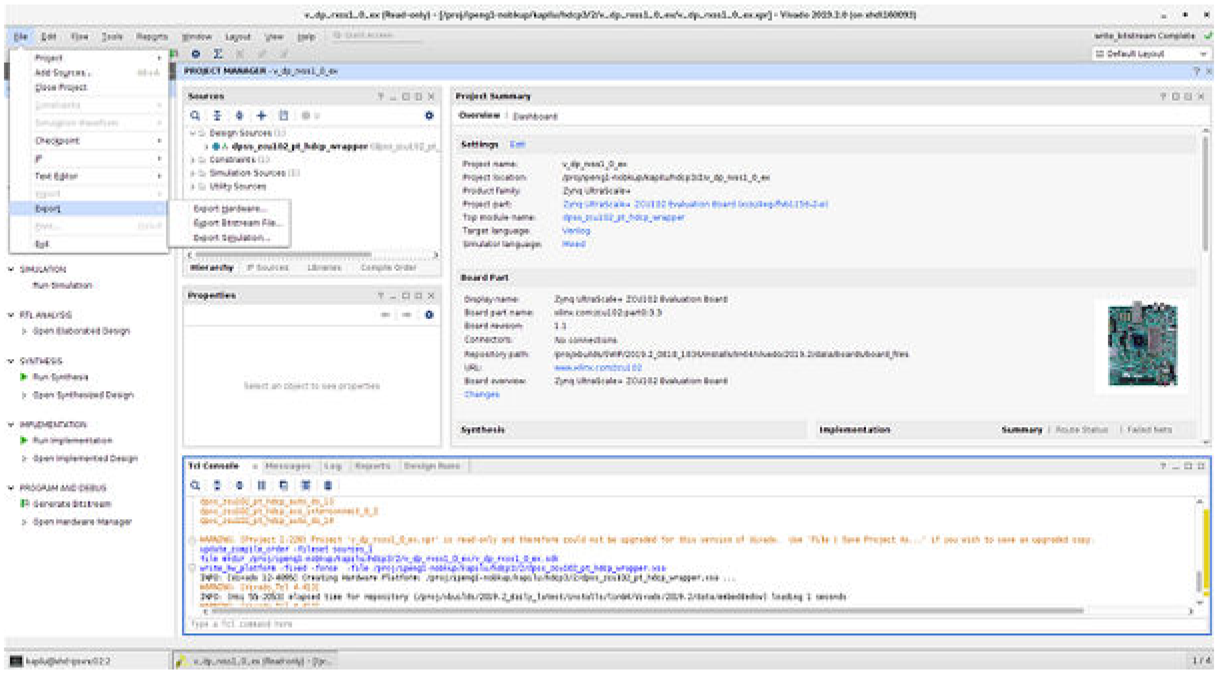
Example project directory:

✓ Overwrite existing example project

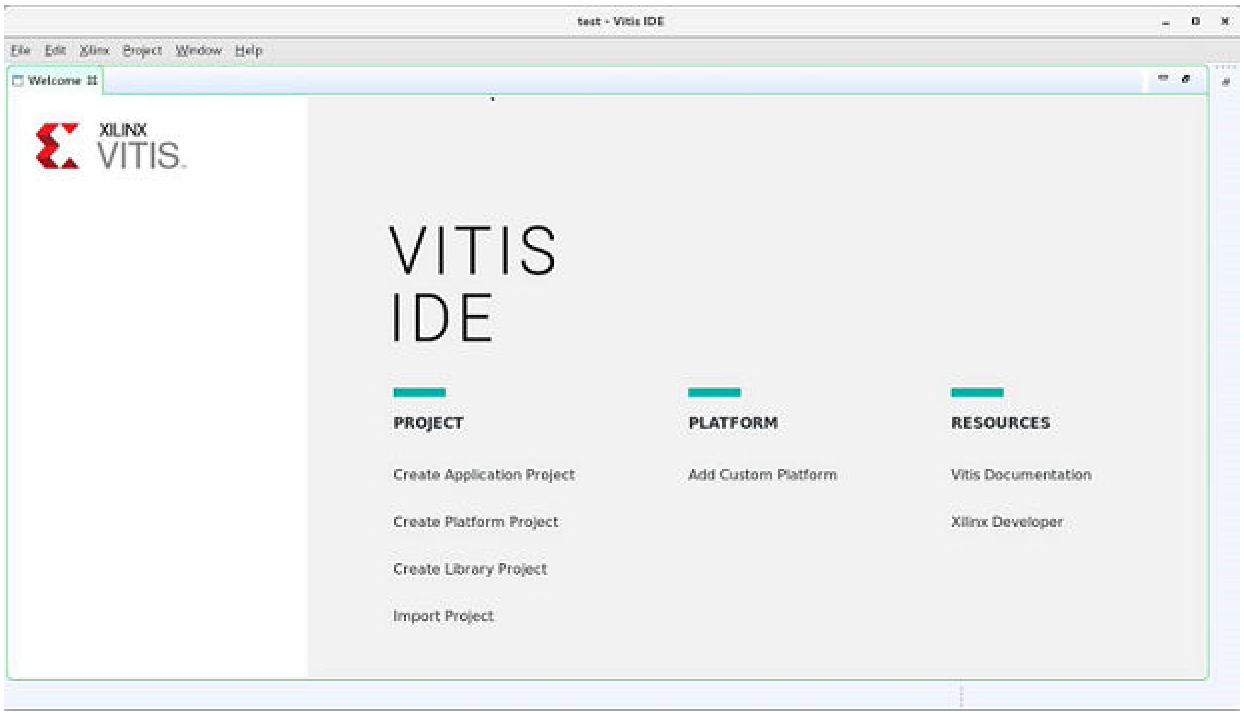
10. The following figure shows the Vivado IP integrator design. Choose the **Generate Bitstream**.



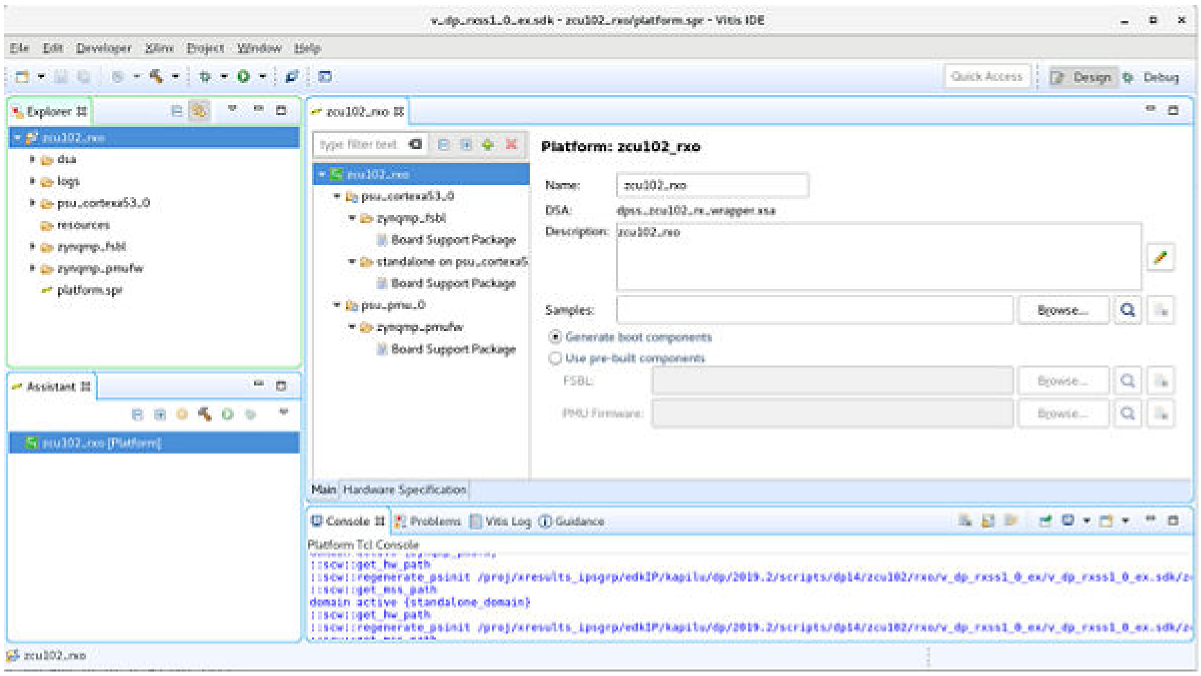
11. Export the hardware (xsa) to Vitis software platform. Click **File → Export → Export Hardware**.



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



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



**Platform: zcu102_rxo**

Name: zcu102_rxo

DSA: dpss_zcu102_rx_wrapper.xsa

Description: zcu102_rxo

Samples:

(●) Generate boot components

( ) Use pre-built components

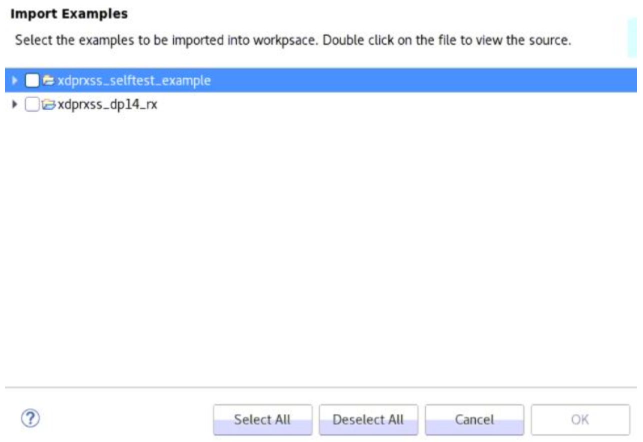
FSBL:

PMU Firmware:

Platform Tcl Console

```
::scw::get_hw_path
::scw::regenerate_psinit /proj/xresults_ipsgrp/edkIP/kapilu/dp/2019.2/scripts/dp14/zcu102/rxo/v_dp_rxss1_0_ex/v_dp_rxss1_0_ex.sdk/z
::scw::get_mss_path
domain active {standalone_domain}
::scw::get_hw_path
::scw::regenerate_psinit /proj/xresults_ipsgrp/edkIP/kapilu/dp/2019.2/scripts/dp14/zcu102/rxo/v_dp_rxss1_0_ex/v_dp_rxss1_0_ex.sdk/z
```

14. Click **Board Support Package** from Vitis. Locate the row for dp14rxss. Click **Import Examples**.



**Import Examples**

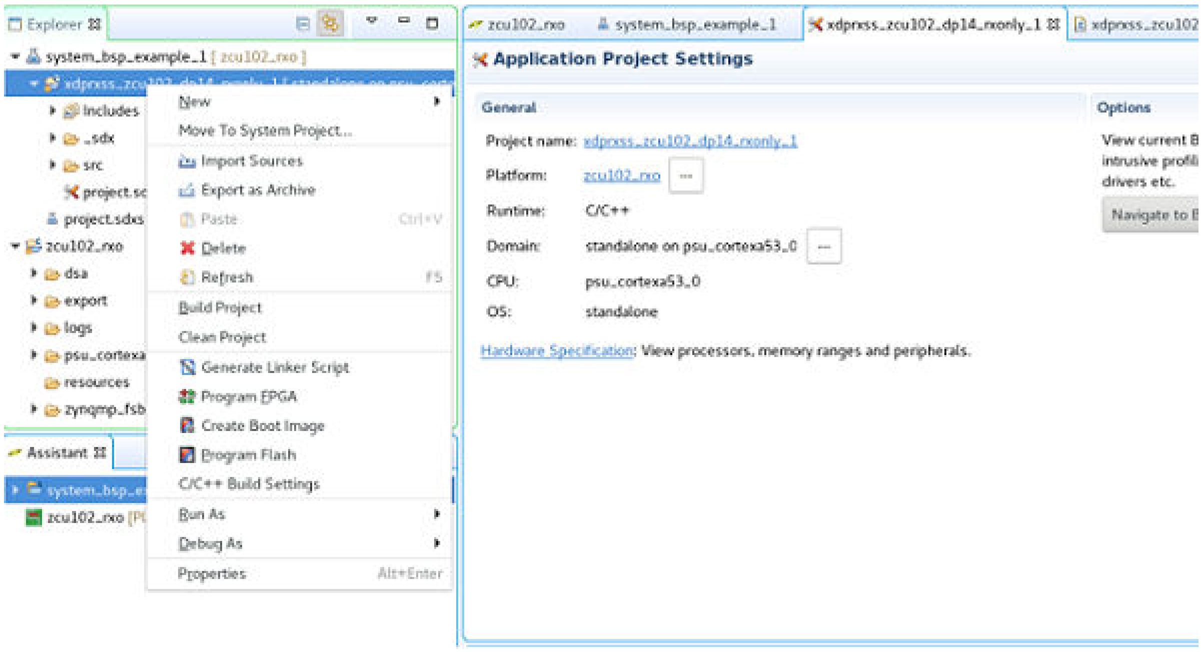
Select the examples to be imported into workpsace. Double click on the file to view the source.

- [ ] xdprxss_selftest_example
- [ ] xdprxss_dp14_rx

15. Select the Example Application corresponding to your hardware:

- For the Pass-through KCU105 project, select the `*_kcu105_dp14` option.
- For the RX-Only ZCU102/VCU118 project, select the `*_dp14_rx` option.

16. Build the example in Vitis software platform by right clicking and selecting **Build Project**.



Project name: xdprxss_zcu102_dp14_rxonly_1
Platform: zcu102_rxo
Runtime: C/C++
Domain: standalone on psu_cortexa53_0
CPU: psu_cortexa53_0
OS: standalone

Hardware Specification: View processors, memory ranges and peripherals.

## Hardware Setup and Run

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

*Figure 16:* **KCU105 Board Setup**

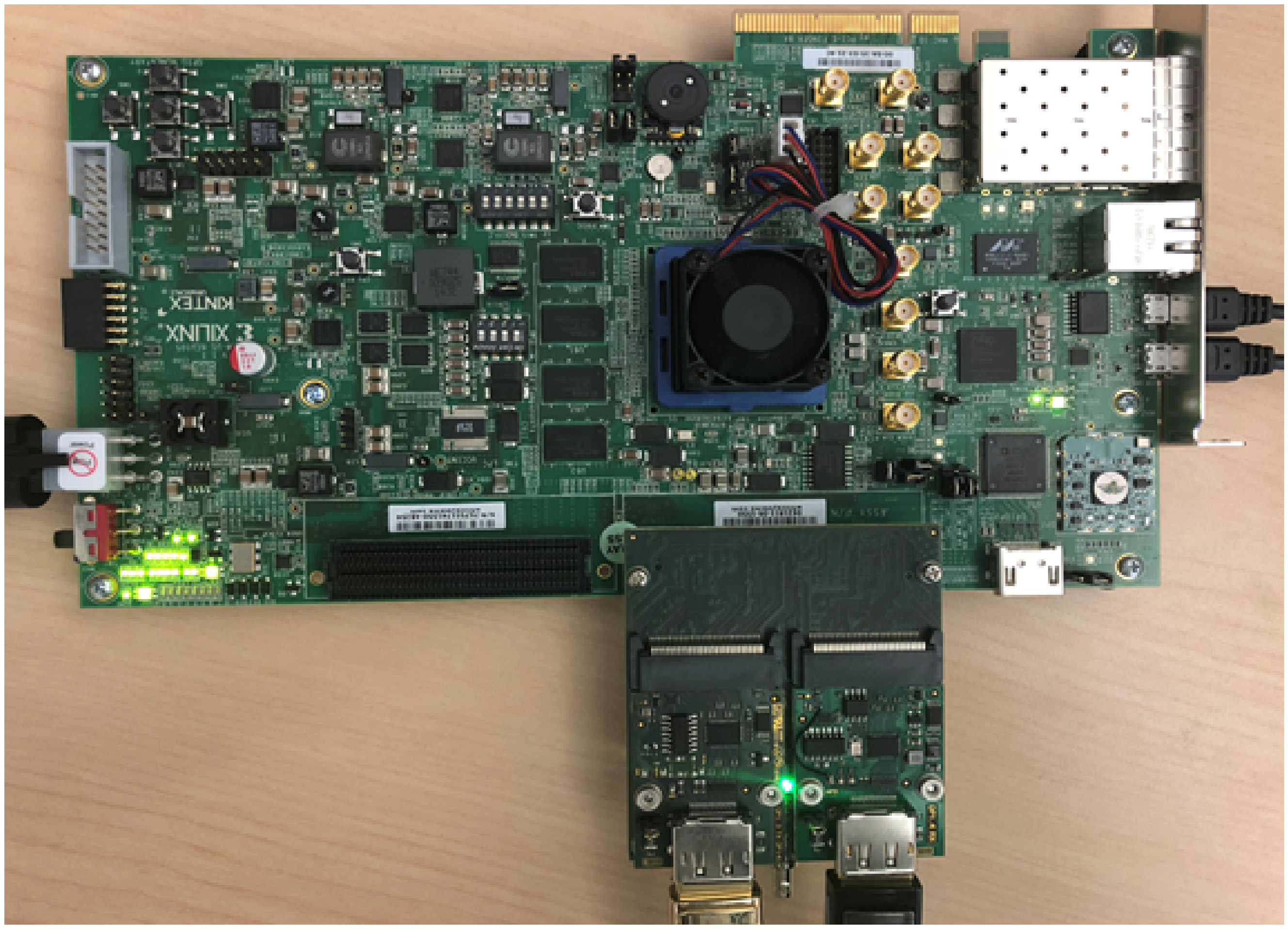
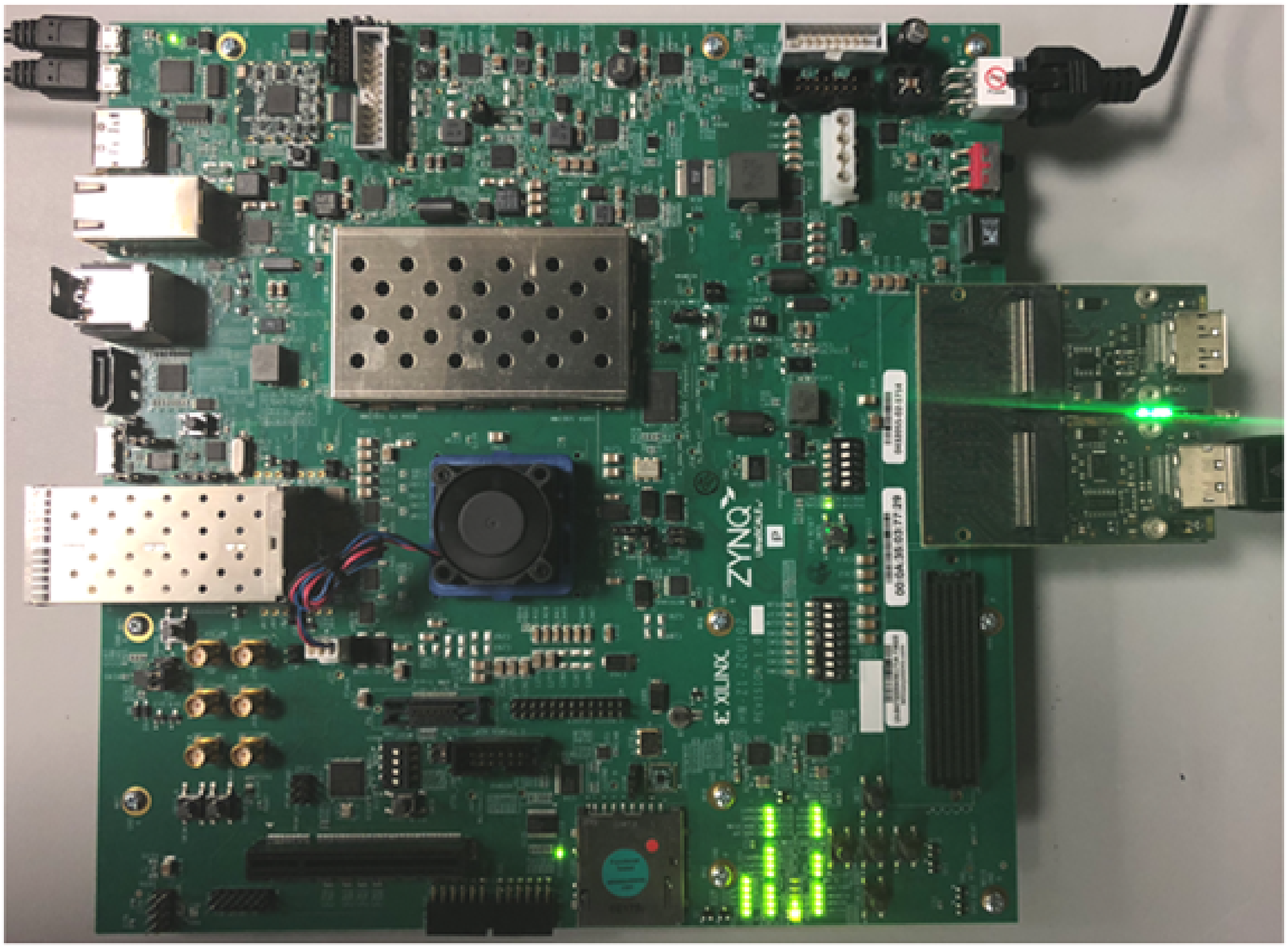
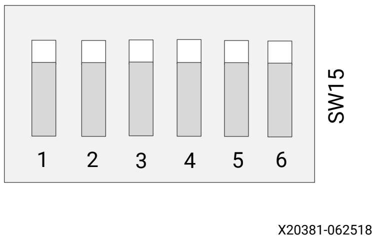


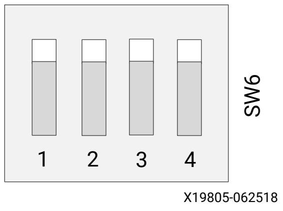
*Figure 17:* **ZCU102 Board Setup**



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



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

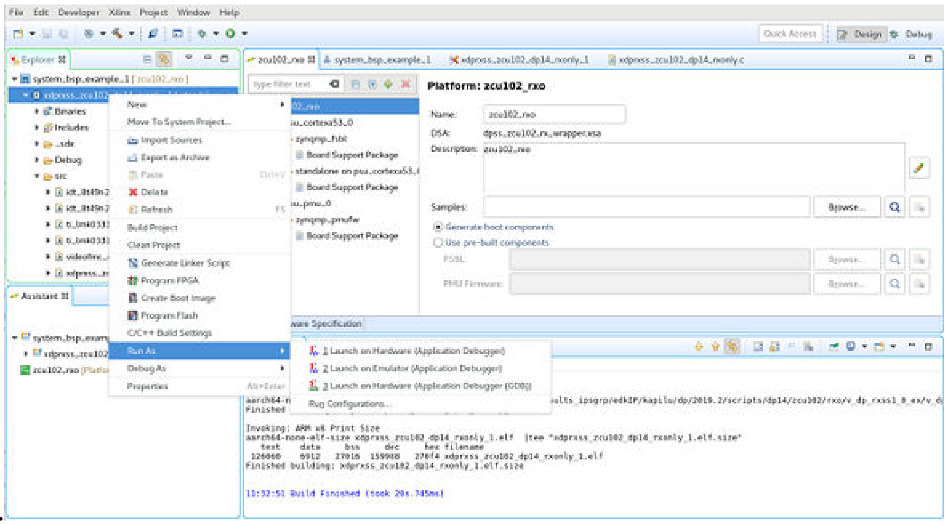


8. Connect the power supply and power on the board.

9. Start an UART terminal program such as Tera Term or Putty with the following settings:
    a. Baud rate = 115200 for KCU105/ZCU102/VCU118
    b. Data bits = 8
    c. Parity = none
    d. Stop bits = 1
    e. Flow Control = none

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

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

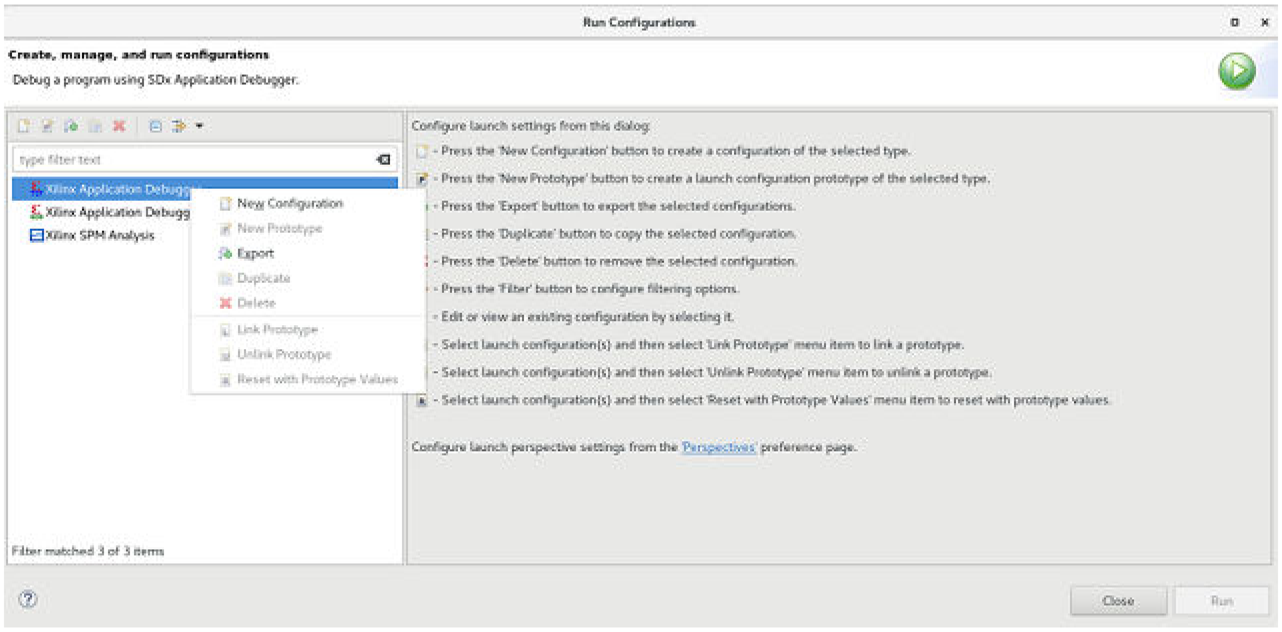


```
aarch64-n                                                    ults_ipsgrp/edkIP/kapilu/dp/2019.2/scripts/dp14/zcu102/rxo/v_dp_rxss1_0_ex/v_d
Finished

Invoking: ARM v8 Print Size
aarch64-none-elf-size xdprxss_zcu102_dp14_rxonly_1.elf  |tee "xdprxss_zcu102_dp14_rxonly_1.elf.size"
   text    data     bss     dec     hex filename
 128060    6912   27016  159988   270f4 xdprxss_zcu102_dp14_rxonly_1.elf
Finished building: xdprxss_zcu102_dp14_rxonly_1.elf.size

11:32:51 Build Finished (took 20s.745ms)
```

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

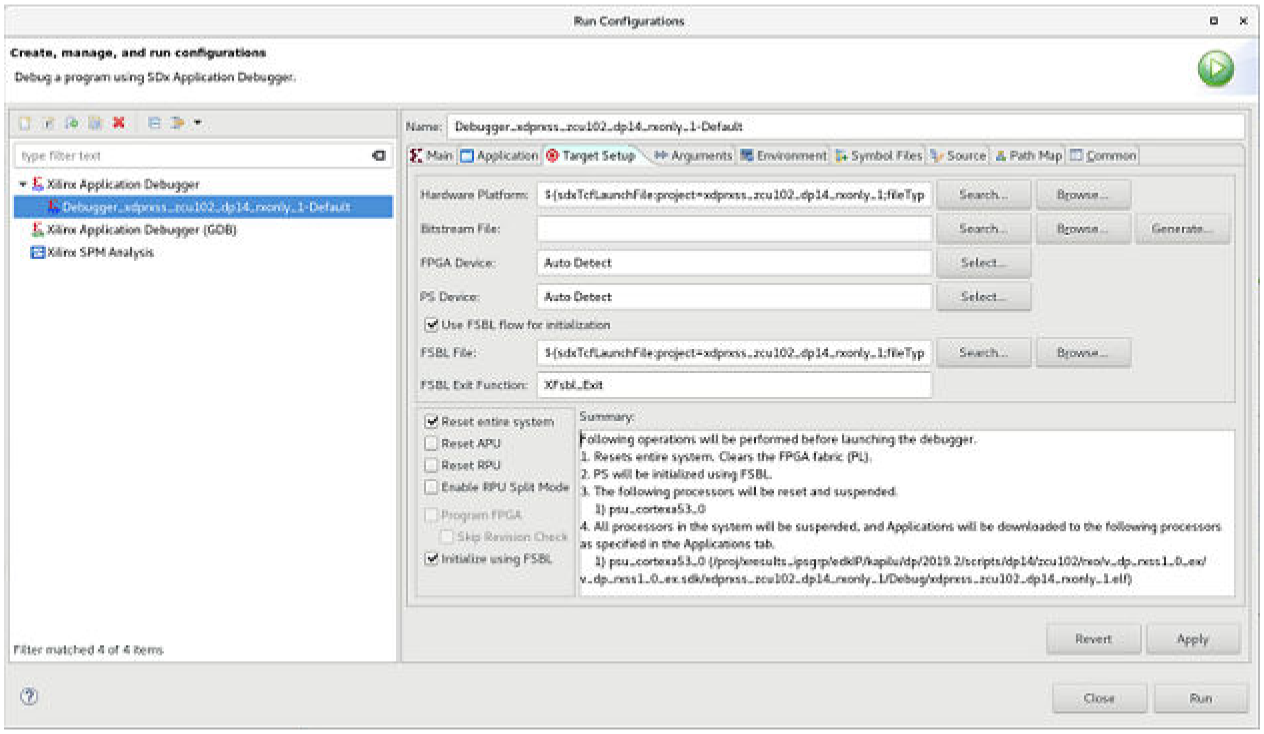

Run Configurations
Create, manage, and run configurations
Debug a program using SDx Application Debugger.
type filter text
Xilinx Application Debugger
Xilinx Application Debugger
Xilinx SPM Analysis
New Configuration
New Prototype
Export
Duplicate
Delete
Link Prototype
Unlink Prototype
Reset with Prototype Values
Filter matched 3 of 3 items
Close
Run

Configure launch settings from this dialog:

- Press the 'New Configuration' button to create a configuration of the selected type.
- Press the 'New Prototype' button to create a launch configuration prototype of the selected type.
- Press the 'Export' button to export the selected configurations.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.
- Select launch configuration(s) and then select 'Link Prototype' menu item to link a prototype.
- Select launch configuration(s) and then select 'Unlink Prototype' menu item to unlink a prototype.
- Select launch configuration(s) and then select 'Reset with Prototype Values' menu item to reset with prototype values.

Configure launch perspective settings from the 'Perspectives' preference page.

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



Run Configurations

**Create, manage, and run configurations**

Debug a program using SDx Application Debugger.

type filter text

- Xilinx Application Debugger
  - Debugger_xdprxss_zcu102_dp14_rxonly_1-Default
- Xilinx Application Debugger (GDB)
- Xilinx SPM Analysis

Filter matched 4 of 4 items

Name: Debugger_xdprxss_zcu102_dp14_rxonly_1-Default

Main | Application | Target Setup | Arguments | Environment | Symbol Files | Source | Path Map | Common

| | | | | |
|---|---|---|---|---|
| Hardware Platform: | ${sdxTcfLaunchFile:project=xdprxss_zcu102_dp14_rxonly_1;fileTyp | Search... | Browse... | |
| Bitstream File: | | Search... | Browse... | Generate... |
| FPGA Device: | Auto Detect | Select... | | |
| PS Device: | Auto Detect | Select... | | |

[✓] Use FSBL flow for initialization

| | | | |
|---|---|---|---|
| FSBL File: | ${sdxTcfLaunchFile:project=xdprxss_zcu102_dp14_rxonly_1;fileTyp | Search... | Browse... |
| FSBL Exit Function: | XFsbl_Exit | | |

- [✓] Reset entire system
- [ ] Reset APU
- [ ] Reset RPU
- [ ] Enable RPU Split Mode
- [ ] Program FPGA
  - [ ] Skip Revision Check
- [✓] Initialize using FSBL

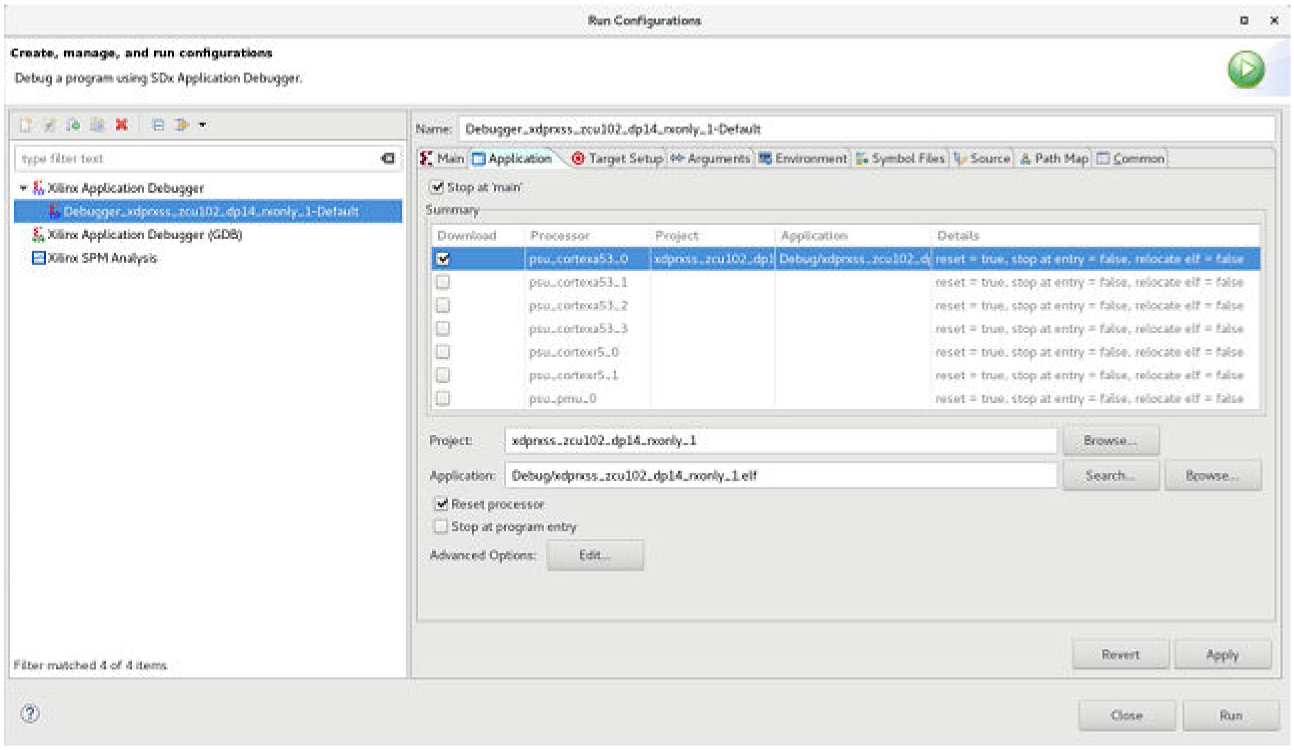
Summary:

```
Following operations will be performed before launching the debugger.
1. Resets entire system. Clears the FPGA fabric (PL).
2. PS will be initialized using FSBL.
3. The following processors will be reset and suspended.
    1) psu_cortexa53_0
4. All processors in the system will be suspended, and Applications will be downloaded to the following processors
as specified in the Applications tab.
    1) psu_cortexa53_0 (/proj/xresults_ipsgrp/edkIP/kapilu/dp/2019.2/scripts/dp14/zcu102/rxo/v_dp_rxss1_0_ex/
v_dp_rxss1_0_ex.sdk/xdprxss_zcu102_dp14_rxonly_1/Debug/xdprxss_zcu102_dp14_rxonly_1.elf)
```

Revert | Apply

Close | Run

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



Run Configurations

**Create, manage, and run configurations**

Debug a program using SDx Application Debugger.

type filter text

- Xilinx Application Debugger
  - Debugger_xdprxss_zcu102_dp14_rxonly_1-Default
- Xilinx Application Debugger (GDB)
- Xilinx SPM Analysis

Filter matched 4 of 4 items

Name: Debugger_xdprxss_zcu102_dp14_rxonly_1-Default

Main | Application | Target Setup | Arguments | Environment | Symbol Files | Source | Path Map | Common

[✓] Stop at 'main'

Summary

| Download | Processor | Project | Application | Details |
|---|---|---|---|---|
| ✓ | psu_cortexa53_0 | xdprxss_zcu102_dp1 | Debug/xdprxss_zcu102_d | reset = true, stop at entry = false, relocate elf = false |
| ☐ | psu_cortexa53_1 | | | reset = true, stop at entry = false, relocate elf = false |
| ☐ | psu_cortexa53_2 | | | reset = true, stop at entry = false, relocate elf = false |
| ☐ | psu_cortexa53_3 | | | reset = true, stop at entry = false, relocate elf = false |
| ☐ | psu_cortexr5_0 | | | reset = true, stop at entry = false, relocate elf = false |
| ☐ | psu_cortexr5_1 | | | reset = true, stop at entry = false, relocate elf = false |
| ☐ | psu_pmu_0 | | | reset = true, stop at entry = false, relocate elf = false |

| | | | |
|---|---|---|---|
| Project: | xdprxss_zcu102_dp14_rxonly_1 | Browse... | |
| Application: | Debug/xdprxss_zcu102_dp14_rxonly_1.elf | Search... | Browse... |

- [✓] Reset processor
- [ ] Stop at program entry

Advanced Options: Edit...

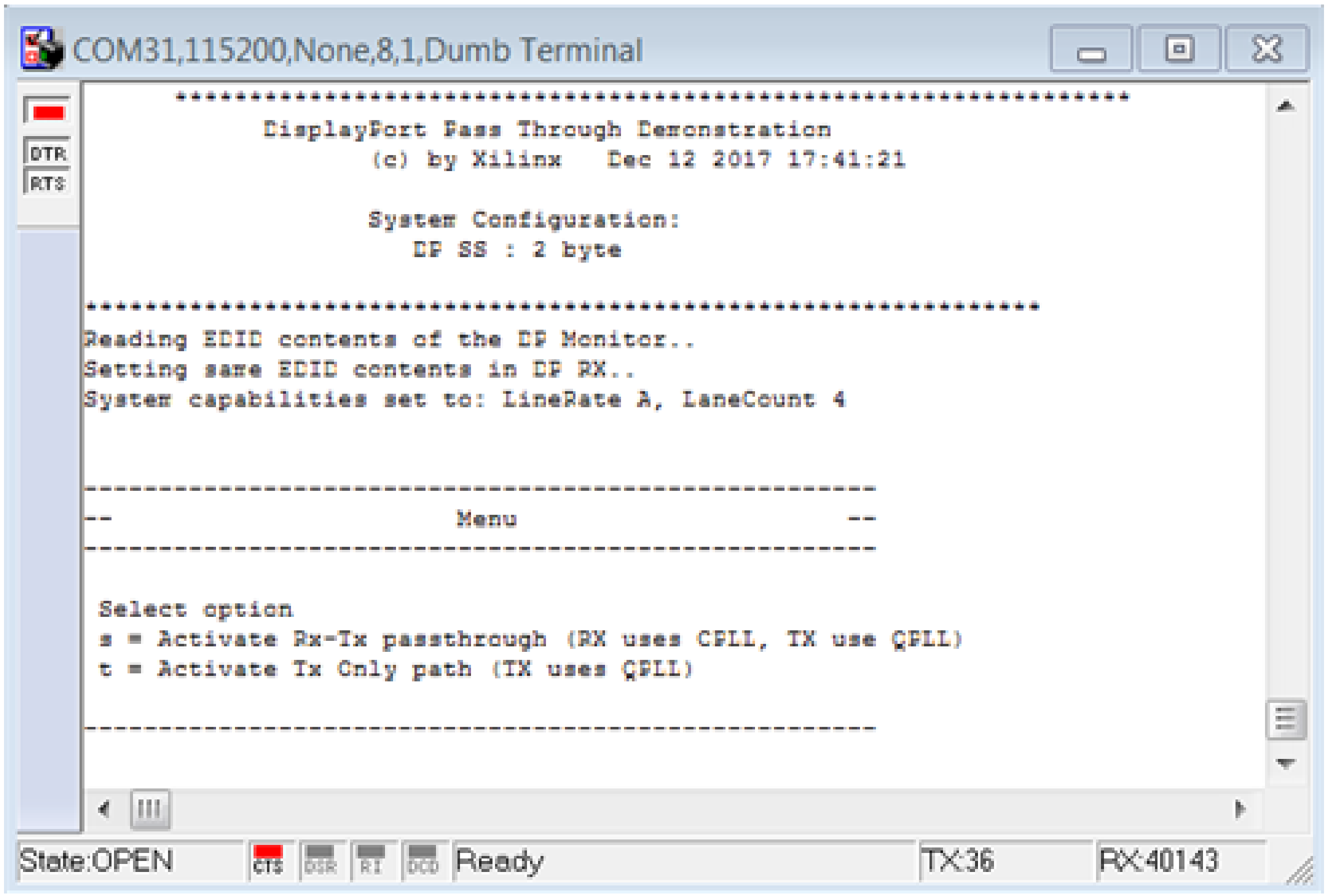
Revert | Apply

Close | Run

## Display User Console

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

*Figure 18:* **DisplayPort User Console**



```
        ****************************************************************
              DisplayPort Pass Through Demonstration
                     (c) by Xilinx    Dec 12 2017 17:41:21

                     System Configuration:
                        DP SS : 2 byte

****************************************************************
Reading EDID contents of the DP Monitor..
Setting same EDID contents in DP RX..
System capabilities set to: LineRate A, LaneCount 4


-----------------------------------------------------
--                       Menu                      --
-----------------------------------------------------

 Select option
 s = Activate Rx-Tx passthrough (RX uses CPLL, TX use QPLL)
 t = Activate Tx Only path (TX uses QPLL)

-----------------------------------------------------
```

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

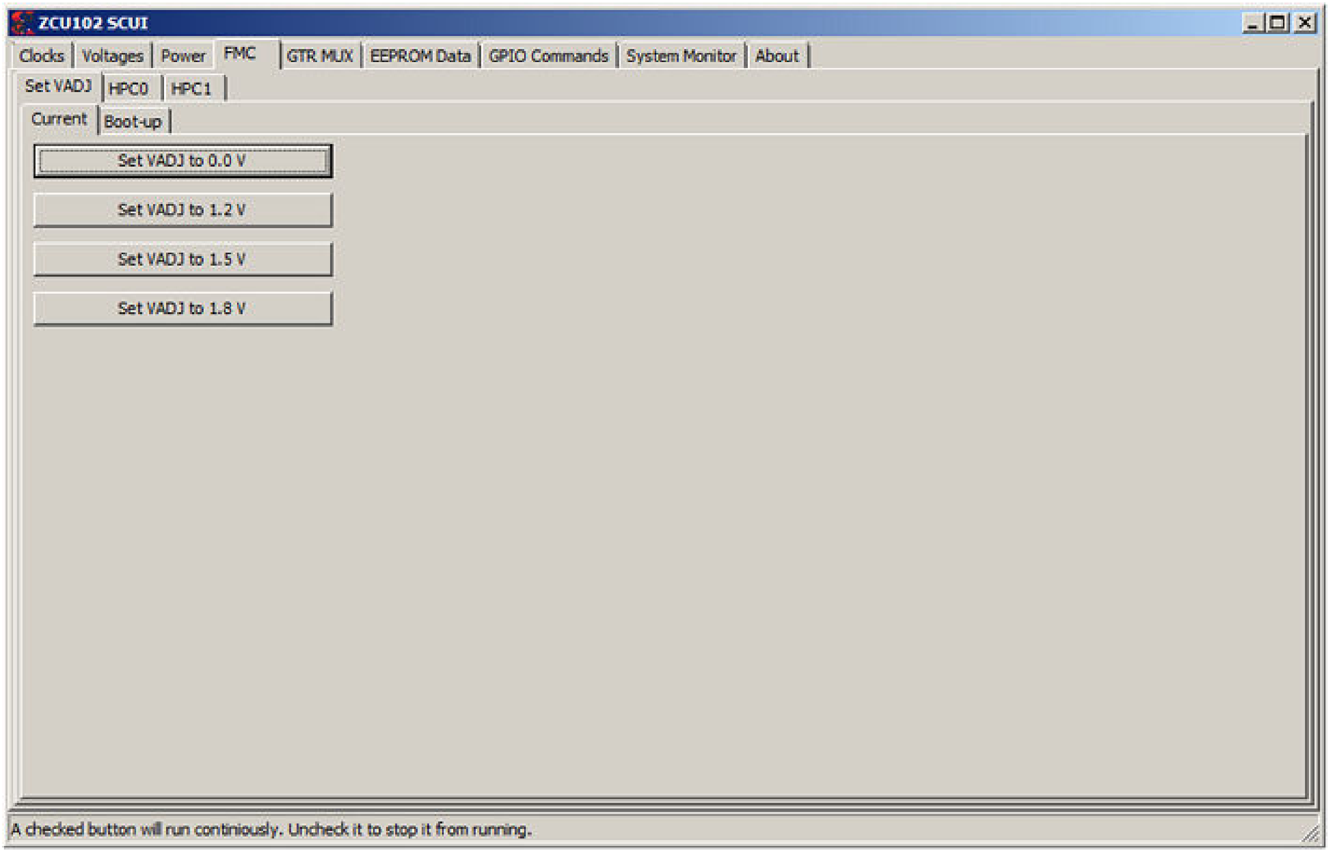
## Setting the FMC Voltage to 1.8V

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

- Random AUX failures
- Training failures

To set the FMC VADJ voltage:

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



A checked button will run continiously. Uncheck it to stop it from running.

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

# Configuring HDCP Keys and Key Management

## HDCP 1.3 Key Management

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

### Using the Encryption Software

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

- 32-byte AES key
- Valid HDCP 1.3 keys

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

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

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

   ```
   ./build.sh
   ```

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

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

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

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

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

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

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

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

7. Replace the dummy AES encrypted HDCP keys in the `KEYMGMT_ENCDATA[]` array of the example design application's source file `src/keys.c` with the encrypted keys generated in step 5.

### HDCP 2.x Key Management

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

## Tested Equipment

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

*Table 36:* **Source Equipment**

| Sink Type | Brand Name | Model Name | Platform |
|---|---|---|---|
| GPU | NVIDIA | GTX 1080 | Windows 10 |
| GPU | NVIDIA | GTX1060 | Windows 10 |
| GPU | NVIDIA | RTX2070 | Windows 10 |
| GPU | AMD | RX 460 | Windows 10 |
| GPU | AMD | R460 | - |
| GPU | AMD | 7600 | - |
| Laptop | Dell | PrecisionM4700 | - |
| Laptop | Dell | E7440 | - |
| Laptop | Apple | MBP | macOS |
| Tester | Unigraf | DPT-200 | - |
| Tester | Unigraf | DPT-323 | - |
| Tester | Unigraf | DPT-400 | - |

# Upgrading

There is no direct upgrade path due to the new retimer. Xilinx® recommends starting with a new design.

# Questions and Answers

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

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

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

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

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

A. Perform the following:

1. Verify that all relevant ARs are taken into account.
2. Increase the AUX_DEFER value in register offset `0x004`.

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:

$(H_{Total} \times V_{Total} \times \text{bits\_per\_component} \times \text{frame rate}) < (0.8 \times \text{link\_lane} \times \text{num\_lanes})$

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.

*Appendix C*

# 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.

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

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

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

## Finding Help on Xilinx.com

To help in the design and debug process when using the subsystem, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

### Documentation

This product guide is the main document associated with the subsystem. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator. Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

## Answer Records

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

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

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

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

### *Master Answer Record for the DisplayPort 1.4 RX Subsystem*

AR 70294

## Technical Support

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

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

To ask questions, navigate to the Xilinx Community Forums.

## Debug Tools

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

### Vivado Design Suite Debug Feature

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

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

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

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

## Hardware Debug

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

### Receive – Training

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

1. Try with a different source such as the DisplayPort Analyzer.
2. Change the cable and check again.
3. Put an AUX Analyzer in the Receive path and check if the various training stages match with those mentioned in DisplayPort Overview.
4. Probe the `lnk_clk` output and check the SI of the Clock is within the Phase Noise mask of the respective GT.
5. 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.
6. Check the `0x43C` and `0x440` registers for Symbol_Locked, Channel Equalization, and Clock Recovery Done.

**Related Information**
DisplayPort Overview

## Receive – Retimer Related Issues

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

- IIC checks:
  - Check if the IIC speed is 100 kHz or higher speed (1 MHz).
  - Check if the IIC writes are happening properly to the Retimer IC.
  - Check if the IIC writes are interrupt or polling based. If it is interrupt based, it is like calling an interrupt within another interrupt routine. Ensure this function is correct, or better to go with the polling mode, as Retimer IIC writes are supposed to happen at Retimer training events.
- Until the training is done ensure 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.
- 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 Retimer configuration as the configuration must be completed as quickly as possible 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.

1. If the video timing counters are reporting 0 lines, toggle the DTG enable and software-video reset and check again.
2. 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.
3. 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 have been followed correctly.

**Related Information**
Audio Management

## Receive – Sink MST

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

- Check if the connected GPU is recognizing the streams correctly. Read the MSA of all the streams and verify against the GPU data.
- Read the VC Payload table through an AXI write and check if the allocated stream IDs are sequential in the slots. The $0^{th}$ 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 might 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. Four streams of 1080p are possible only in 5.4 x 4. With HBR, only two 1080p streams are possible. The link rate downshift might be because of a training failure—ensure that a DisplayPort v1.4 cable is used.
- Make sure a DisplayPort v1.4 cable is used with MCDP6000 in between source and sink.

## 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 incoming DisplayPort data stream on the `lnk_clk` domain is too fast compared to the outgoing data stream on the `rx_vid_clk` domain.

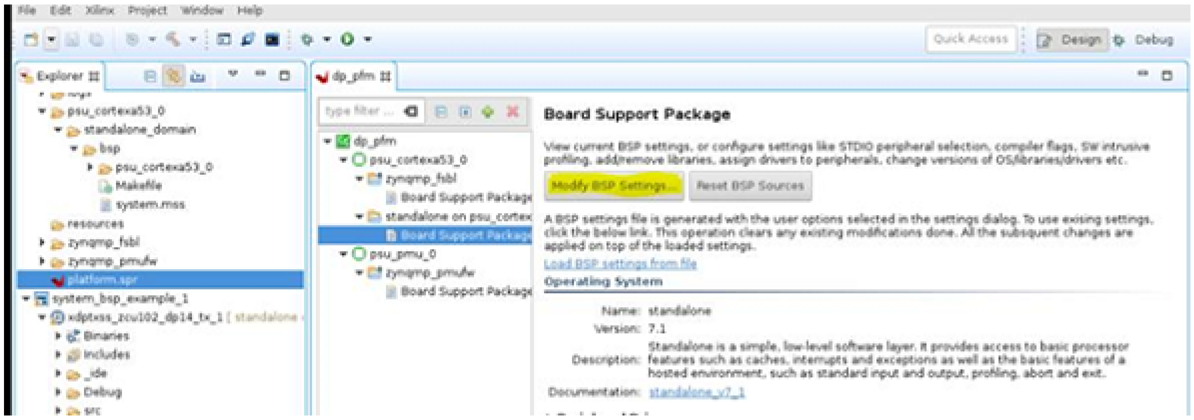
There are two ways to resolve this:

1. If possible, increase HBLANK from the source.
2. Increase the `rx_vid_clk` frequency to the maximum tested of 300 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 on **Modify BSP settings** (see figure).



**Board Support Package**

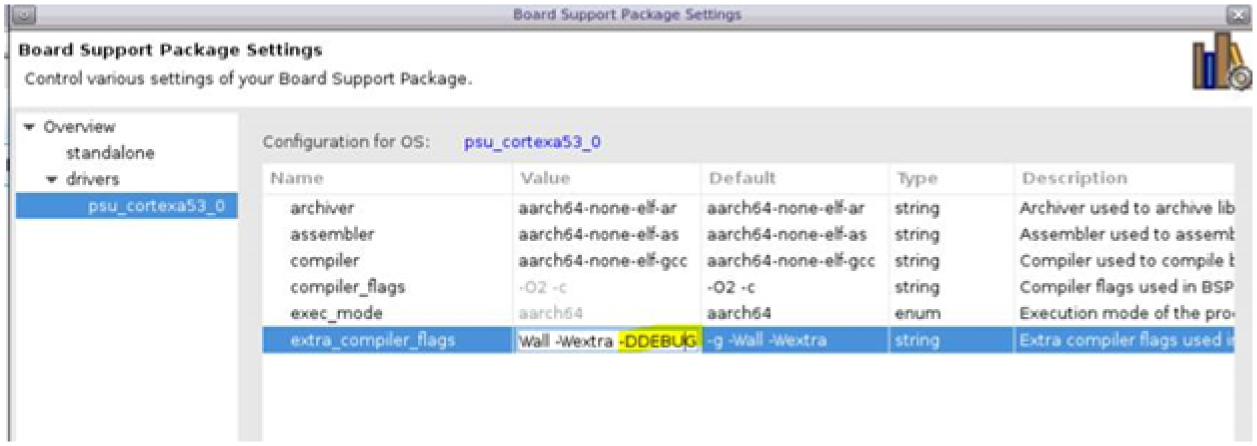
View current BSP settings, or configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/remove libraries, assign drivers to peripherals, change versions of OS/libraries/drivers etc.

A BSP settings file is generated with the user options selected in the settings dialog. To use exising settings, click the below link. This operation clears any existing modifications done. All the subsequent changes are applied on top of the loaded settings.

**Operating System**

| | |
|---|---|
| Name: | standalone |
| Version: | 7.1 |
| Description: | Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit. |
| Documentation: | standalone_v7_1 |

2. Add the option -DDEBUG to the extra compiler flags then close the BSP settings (see figure below).



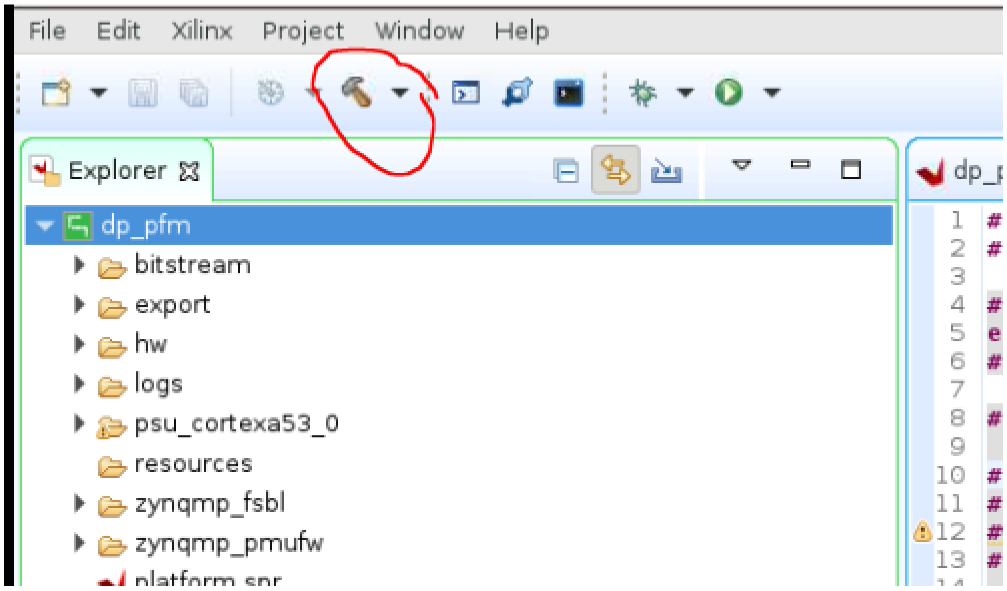
**Board Support Package Settings**

Control various settings of your Board Support Package.

Configuration for OS: psu_cortexa53_0

| Name | Value | Default | Type | Description |
|---|---|---|---|---|
| archiver | aarch64-none-elf-ar | aarch64-none-elf-ar | string | Archiver used to archive lib |
| assembler | aarch64-none-elf-as | aarch64-none-elf-as | string | Assembler used to assemb |
| compiler | aarch64-none-elf-gcc | aarch64-none-elf-gcc | string | Compiler used to compile b |
| compiler_flags | -O2 -c | -O2 -c | string | Compiler flags used in BSP |
| exec_mode | aarch64 | aarch64 | enum | Execution mode of the pro |
| extra_compiler_flags | Wall -Wextra -DDEBUG | -g -Wall -Wextra | string | Extra compiler flags used i |

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



This will enable the debug symbol.

# Video EDID

## Introduction

Video EDID (`vid_edid_v1_0`) core is used for hosting EDID information of sync device and communicate the EDID blocks through I2C bus. It is designed based on VESA EEDC v1.2 specification.

## Port Description

*Table 37:* **Port Description**

| Signal Name | Direction | Description |
|---|---|---|
| s_axi_aclk | Input | AXI Clock |
| s_axi_aresetn | Input | AXI Reset. Active-Low |
| s_axi** | Input | AXI4-Lite interface, defined in the *Vivado Design Suite: AXI Reference Guide* (UG1037) |
| ctl_clk | Input | Free running clock. `s_axi` interface clock is connected |
| ctl_reset | Input | Reset signal and external reset is connected |
| iic_scl_in | Input | I2C bus SCL input |
| iic_sda_in | Input | I2C bus SDA input |
| iic_sda_out | Output | I2C bus SDA output |

## Functional Operation

At system level, software copies the EDID of the connected sync device and writes into EDID byte array present within Video EDID core through AXI4 Lite interface. EDID byte array size is 384 B. Video EDID receives I2C commands from DisplayPort RX Subsystem through AUX transactions and responds back with EDID data back to DisplayPort RX Subsystem based on the received request. EDID memory is split into segments and each segment is 128 B in size. All transactions will be based on segment pointer within EDID memory space. Please refer the VESA EEDC v1.2 specification or later for the protocol.

*Appendix F*

# Additional Resources and Legal Notices

## Xilinx Resources

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

## Documentation Navigator and Design Hubs

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

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

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

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

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

## References

These documents provide supplemental material useful with this guide:

1. *Video PHY Controller LogiCORE IP Product Guide* (PG230)
2. *VESA DisplayPort Standard* (VESA website)
3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
4. *AXI4-Stream Video IP and System Design Guide* (UG934)
5. *SmartConnect LogiCORE IP Product Guide* (PG247)
6. *AXI IIC Bus Interface LogiCORE IP Product Guide* (PG090)
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *Vivado Design Suite User Guide: Getting Started* (UG910)
9. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
10. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
11. *Vivado Design Suite User Guide: Implementation* (UG904)
12. *Vivado Design Suite: AXI Reference Guide* (UG1037)
13. *AXI Timer LogiCORE IP Product Guide* (PG079)
14. *ZCU102 System Controller GUI Tutorial* (registration required) (XTP433)
15. *VCU118 System Controller Tutorial* (registration required) (XTP447)
16. *UltraScale Architecture and Product Data Sheet: Overview* (DS890)
17. *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892)
18. *Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS893)
19. *Defense-Grade UltraScale Architecture Data Sheet: Overview* (DS895)
20. *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)
21. *Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS923)
22. *Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925)
23. *Zynq UltraScale+ RFSoC Data Sheet: DC and AC Switching Characteristics* (DS926)

## Revision History

The following table shows the revision history for this document.

| Section | Revision Summary |
|---|---|
| **08/31/2020 v2.1** | |
| Chapter 2: Overview | Updated with device specific line rate information. |
| Chapter 4: Designing with the Subsystem | Added eDP support. |

| Section | Revision Summary |
|---|---|
| Chapter 6: Example Design | Added Configuring HDCP Keys and Key Management sections. |
| **12/02/2019 v2.1** | |
| General updates | • HDCP 2.2 support<br>• FB Pass-through with HDCP 1.3 and HDCP 2.2<br>• EDID I2C speed control added<br>• Vitis flow updated in Chapter 6<br>• Added Appendix E for Video EDID Helper Cores |
| **05/22/2019 v2.1** | |
| Chapter 6: Example Design | MST FB Pass-through example design details added |
| **12/05/2018 v2.0** | |
| HDCP Key Interface | HDCP Ports added |
| DisplayPort MST Stream | MST Ports added |
| Programming the Core in MST Mode | MST Programming added |
| Pixel Mapping Examples on AXI4-Stream Interface (UG934-Compliant) | UG934-compliant pixel mapping |
| **04/04/2018 v1.0** | |
| Initial release. | N/A |

## Please Read: Important Legal Notices

## Copyright