SMPTE UHD-SDI Receiver Subsystem v2.0

Product Guide

Vivado Design Suite

PG290 April 26, 2022

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

AMD
XILINX

AMD
XILINX

Table of Contents

IP Facts

Chapter 1: Overview

Navigating Content By Design Process 5
Introduction 6
Subcore Details 7
Applications 8
Unsupported Features 8
Licensing and Ordering 8

Chapter 2: Product Specification

Standards 9
Performance 9
Resource Utilization 10
Port Descriptions 10
Transport Format Detection 18
High Dynamic Range Data 21
Register Space 22
Versal ACAP Block Automation in UHD-SDI RX Subsystem 35

Chapter 3: Designing with the Subsystem

General Design Guidelines 39
Clocking 41
Resets 42
UHD-SDI Audio Extract Use Case 43

Chapter 4: Design Flow Steps

Customizing and Generating the Subsystem 46
Core Configuration Tab 46
Application Example Design Tab 48
Constraining the Subsystem 50
Simulation 51
Synthesis and Implementation 51

Chapter 5: Example Design

ZCU106 SMPTE UHD-SDI RX-Only Example Design 53
ZCU106 SMPTE UHD-SDI Pass-Through Example Design. 63
ZCU106 SMPTE UHD-SDI Audio-Video Pass-Through Example Design. 76
KCU116 SMPTE UHD-SDI Audio-Video Loopback Example Design 87
VCK190 SMPTE UHD-SDI Audio-Video Pass-Through Example Design. 99

Appendix A: Verification, Compliance, and Interoperability

Interoperability 114
Hardware Validation 115
Video Resolutions 116

Appendix B: Debugging

Finding Help on Xilinx.com 132
Debug Tools 134
Hardware Debug 134
Interface Debug 137

Appendix C: Audio Test Pattern Generator IP

Feature Summary. 139
Unsupported Features, Limitations 139
Port Descriptions 140
Register Space 140
Clocking. 145
Resets 145
Programming Sequence. 146

Appendix D: Additional Resources and Legal Notices

Xilinx Resources 147
Documentation Navigator and Design Hubs 147
References 148
Revision History 149
Please Read: Important Legal Notices 150

Introduction

The Society of Motion Picture and Television Engineers (SMPTE) UHD-SDI receiver subsystem implements an SDI receive interface in accordance with the serial digital interface (SDI) family of standards. The subsystem receives video from a native SDI and generates AXI4-Stream video. The subsystem allows fast selection of the top level parameters and automates most of the lower level parameterization. The AXI4-Stream video interface allows a seamless interface to other AXI4-Stream-based subsystems.

Features

- Supported configurations:
  - AXI4-Stream, Native Video, Native SDI data interfaces
  - 2 Pixels per clock (PPC)
  - 10 bit, 12 bits per color component
  - YCbCr/RCB 4:4:4, YCbCr 4:2:2, YCbCr 4:2:0 color spaces
  - Block automation for Versal™ ACAP device family
- High Frame Rates at Native SDI
- 6G/12G 10 bit High Frame Rates Support for AXI stream RX Subsystem configurations
- AXI4-Lite interface for register access to configure different subsystem options
- Audio support
- Supports HLG HDR video
- Standards compliance:
  - SMPTE ST 259: SD-SDI at 270 Mb/s
  - SMPTE RP 165: EDH for SD-SDI
  - SMPTE ST 292: HD-SDI at 1.485 Gb/s and 1.485/1.001 Gb/s
  - SMPTE ST 372: Dual Link HD-SDI
  - SMPTE ST 424: 3G-SDI with data mapped by any ST 425-x mapping at 2.97 Gb/s and 2.97/1.001 Gb/s
  - SMPTE ST 2081-1: 6G-SDI with data mapped by any ST 2081-x mapping at 5.94 Gb/s and 5.94/1.001 Gb/s
  - SMPTE ST 2082-1: 12G-SDI with data mapped by any ST 2082-x mapping at 11.88 Gb/s and 11.88/1.001 Gb/s
  - Dual link and quad link 6G-SDI and 12G-SDI supported by instantiating two or four SMPTE UHD-SDI RX subsystems
  - SMPTE ST 352: Payload ID packets
- Designed and supported to work directly with the UHD-SDI GT core [Ref 18]
- Supports Versal ACAP (GTYE5) product family

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ (GTHE4, GTYE4) Versal™ ACAP (GTYE5, GTYP) Zynq® UltraScale+ MPSoC (GTHE4, GTYE4) Zynq UltraScale+ RFSoC
Supported User Interfaces	AXI4-Lite, AXI4-Stream, native video, and native SDI
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Hierarchical subsystem packaged with SMPTE UHD-SDI RX core and other IP cores
Example Design	Vivado® IP integrator
Test Bench	N/A
Constraints File	IP cores delivered with XDC files
Simulation Model	N/A
Supported S/W Driver(2)	Standalone and Linux
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: 68766
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. Standalone driver details can be found in the Vitis™ directory (<install_directory>/Vitis/<release>/data/embeddedsw/doc/xilinx_drivers.htm). Linux OS and driver support information is available from http://www.wiki.xilinx.com/Xilinx+V4L2+SDI+Rx+driver.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Navigating Content By Design Process

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

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

For driver information, see the Xilinx GIT hub.

For IP patch information, see the SMPTE UHD-SDI Receiver (RX) Subsystem Release Notes and Known Issues AR# 68766.

Introduction

The SMPTE UHD-SDI Receiver (RX) Subsystem allows you to quickly create systems based on SMPTE SDI protocols. It receives unaligned native SDI streams from the UHD-SDI GT [Ref 18] and outputs an AXI4-Stream video stream, native video, or native SDI using Xilinx® transceivers as the physical layer. The top-level customization parameters select the required hardware blocks required to build the subsystem. Figure 1-1 shows the subsystem architecture using the AXI4-Stream user interface. The SMPTE UHD-SDI (RX) IP core, SDI RX to Video Bridge IP core and Video In to AXI4-Stream are present in the subsystem and the subsystem outputs AXI4-Stream.

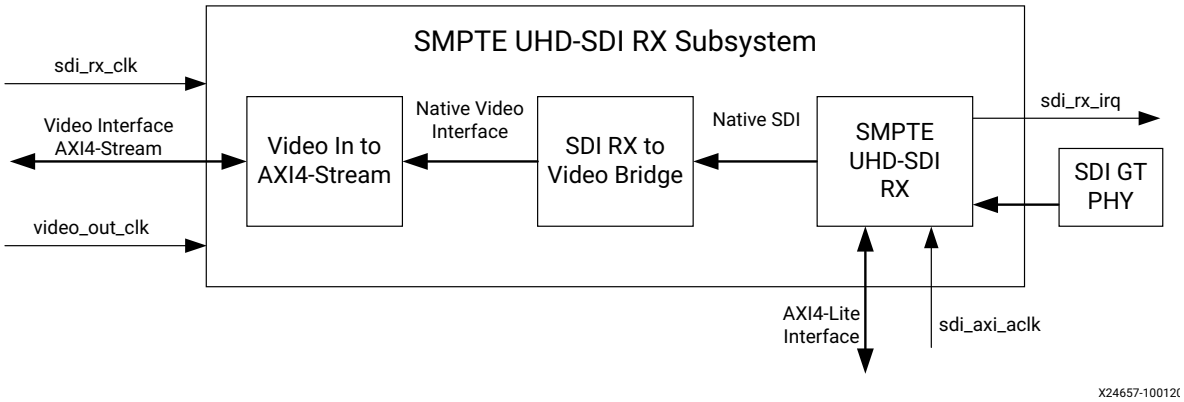

Figure 1-1: **SMPTE UHD-SDI RX Subsystem AXI4-Stream Architecture**

Figure 1-2 shows the SMPTE UHD-SDI RX Subsystem with native video as the user interface. The SMPTE UHD-SDI (RX) IP core and SDI RX to Video Bridge IP core are present in the subsystem and the subsystem outputs native video.

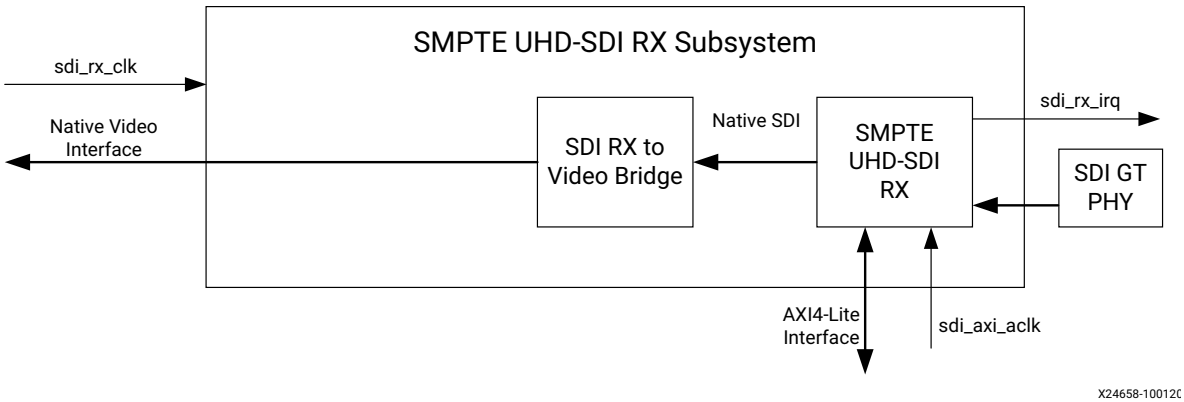

Figure 1-2: **SMPTE UHD-SDI RX Subsystem Native Video Architecture**

Figure 1-3 shows the SMPTE UHD-SDI RX Subsystem with native SDI as the user interface. The subsystem accepts native SDI input and consists only of the SMPTE UHD-SDI (RX) IP core.

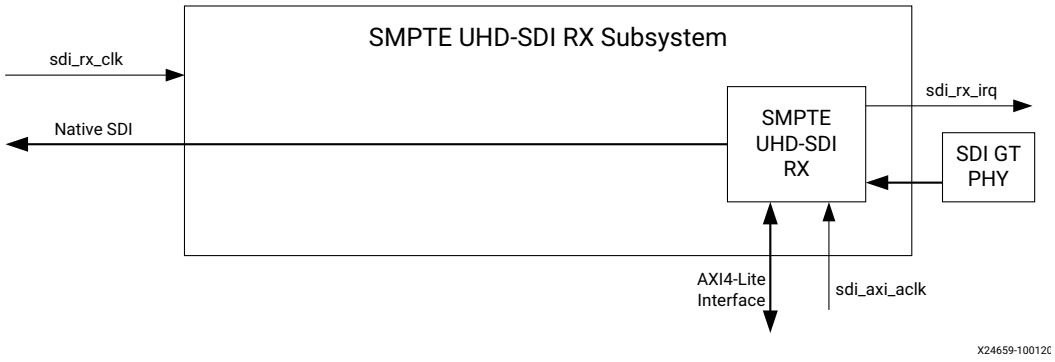

Figure 1-3: **SMPTE UHD-SDI RX Subsystem Native SDI Architecture**

Depending on the data interface connection the subsystem consists of the following cores:

- SMPTE UHD-SDI (RX)
- SDI RX to Video Bridge
- Video In to AXI4-Stream

Subcore Details

SMTPE UHD-SDI (RX)

The SMPTE UHD-SDI core receives multiplexed native SDI data streams and generates a non-multiplexed SDI data stream.

SDI RX to Video Bridge

The SDI RX to Video Bridge LogiCORE IP core is designed to interface between native SDI and native video. The input is an SDI virtual interface that has one to eight data streams with embedded synchronization. The output is video data with explicit synchronization signals. This core extracts synchronization signals, reformats the video data, and provides clock enables.

The core extracts embedded synchronization signals from the SDI data stream. It supports SD-SDI, HD-SDI, 3G-SDI Level A, 3G-SDI Level B, 6G-SDI and 12G-SDI with YCbCr data format at 10 bits and 12 bits per component. For 3G-SDI Level B, it automatically reorders two lines of parallel data to sequential lines of video data out. It supports both interlaced and progressive line standards.

Video In to AXI4-Stream

The Video In to AXI4-Stream core acts as an interface from a video source (clocked parallel video data with synchronization signals - active video with either syncs, blanks or both) to the AXI4-Stream Video Protocol interface. See the *Video In to AXI4-Stream LogiCORE IP Product Guide* (PG043) [Ref 10] for details.

Applications

- Professional broadcast cameras
- Professional digital video recorders
- Professional video processing equipment
- Medical imaging

Unsupported Features

- 16-way data stream interleaving is not supported for AXI4-Stream and native video interface subsystem configurations.
- YCbCr/RGB 4:4:4, YCbCr 4:2:2, and YCbCr 4:2:0 formats are supported. This IP supports RGB format. RGB 4:4:4 color format does not support AXI4-Stream or native video interface configuration.
- SMPTE 2082-10: 2018 HFR is supported only for 6G/12G SDI modes for 10 bit but all other HFR resolutions are not supported for AXI4-Stream Interface subsystem configurations.

Licensing and Ordering

The SMPTE UHD-SDI RX Subsystem is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License.

For more information, visit the UHD-Serial Digital Interface (SDI) product web page.

Information about this and 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.

Product Specification

Standards

The core supports the following SMPTE standards:

- SMPTE ST 259: SD-SDI at 270 Mb/s
- SMPTE RP 165: EDH for SD-SDI
- SMPTE ST 292: HD-SDI at 1.485 Gb/s and 1.485/1.001 Gb/s
- SMPTE ST 372: Dual Link HD-SDI (by instantiation of two UHD-SDI cores)
- SMPTE ST 424: 3G-SDI with data mapped by any ST 425-x mapping at 2.97 Gb/s and 2.97/1.001 Gb/s
- SMPTE ST 2081-1: 6G-SDI with data mapped by any ST 2081-x mapping at 5.94 Gb/s and 5.94/1.001 Gb/s (including multi-link 6G-SDI)
- SMPTE ST 2082-1: 12G-SDI with data mapped by any ST 2082-x mapping at 11.88 Gb/s and 11.88/1.001 Gb/s (including multi-link 12G-SDI)
- Dual link and quad link 6G-SDI and 12G-SDI are supported by instantiating two or four UHD-SDI cores.

Note: Data stitching is done by user at the System level. The UHD-SDI TX cores do not provide stitching functionality.

- SMPTE ST 352: Payload ID packets are fully supported.

Performance

Maximum Frequencies

In 12G-SDI mode, the maximum frequency of the RX clock is 297 MHz. In 6G-SDI, 3G-SDI, and SD-SDI modes, the maximum frequency of the RX clock is 148.5 MHz. In HD-SDI mode, the maximum frequency of the RX clock is 74.25 MHz.

Resource Utilization

For full details about performance and resource utilization, visit the Performance and Resource Utilization web page.

Port Descriptions

The SMPTE UHD-SDI RX Subsystem I/O signals are described in the following port tables.

AXI4-Lite Interface Ports

These signals are enabled when the AXI4-Lite interface option is selected.

Table 2-1: **AXI4-Lite Interface Port Descriptions**

Signal	I/O	Description
s_axi_aclk	I	AXI4-Lite clock
s_axi_arstn	I	AXI4-Lite reset. Active-Low
S_AXI_CTRL*		AXI4-Lite interface, defined in the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 16].

AXI4-Stream (Video) Interface Ports

These signals are enabled when the AXI4-Stream option is selected for the video interface.

Table 2-2: **Video-Over-AXIS Interface Port Descriptions**

Signal	I/O	Description
video_out_clk	I	Video output clock
video_out_arstn	I	Video output active-Low reset.
VIDEO_OUT_tdata[n-1:0]	O	Video data carrying YCbCr 4:4:4 / YCbCr 4:2:2 / YCbCr 4:2:0 video with 10 or 12-bit per component, based on the color depth configuration. For 10bpc, n=64 For 12bpc, n=72 (For details see the *AXI4-Stream Video IP and System Design* (UG934) [Ref 8])
VIDEO_OUT_tlast	O	AXI4-Stream TLAST. End of Line
VIDEO_OUT_tready	I	AXI4-Stream TREADY.
VIDEO_OUT_tuser	O	AXI4-Stream TUSER. Start of Frame

Table 2-2: **Video-Over-AXIS Interface Port Descriptions *(Cont'd)***

Signal	I/O	Description
VIDEO_OUT_tvalid	O	AXI4-Stream TVALID. Active video data enable
fid	O	Field ID. Connected field of Video-in to AXI4 Stream IP. For Interlaced videos, fid will toggle based on the field selected 0: Even field 1: Odd field For Progressive videos- fid will be set as 0. ***Note:*** The fid is expected to toggle in psf and 3G-level B progressive videos. In these modes, one progressive frame is split into two fields which are transported independently (similar to interlaced video) and then combined and correctly paired together at the receiver side with the help of the fid signal to construct the original progressive frame. For example, when 1080p 60 Hz video is transported on 3G-SDI level B-DL, the video transport is actually 1080i 60 Hz – the transport is interlaced, but the picture is progressive.

S_AXIS_STS_SB_RX Interface Ports

Table 2-3: **S_AXIS_STS_SB_RX Interface Port Descriptions**

Signal	I/O	Description
S_AXIS_STS_SB_RX_tready	O	Core ready
S_AXIS_STS_SB_RX_tvalid	I	Data valid
S_AXIS_STS_SB_RX_tdata[31:0]	I	Sideband signal information from transceiver block bit 0: rx_change_done—Indicates that SDI line rate is successful bit 2: gtrxresetdone bit 3: rx_m (Integer = 0, Fractional = 1) bit 8: rx_fabric_rst—SMPTE UHD-SDI RX IP is reset when this bit set to 1 All other bits are not used.

S_AXIS_RX Interface Ports

Table 2-4: **S_AXIS_RX Port Descriptions**

Signal	I/O	Description
sdi_rx_clk	I	SMPTE SDI RX core clock
sdi_rx_rst	I	Active-High reset
S_AXIS_RX_tready	O	SMPTE SDI RX core ready
S_AXIS_RX_tvalid	I	Data valid

Table 2-4: **S_AXIS_RX Port Descriptions *(Cont'd)***

Signal	I/O	Description
S_AXIS_RX_tdata[n-1:0]	I	Parallel data received from transceiver. n varies with SDI standard selection: n=40 for 6G-SDI and 12G-SDI n=20 for 3G-SDI
S_AXIS_RX_tuser[31:0]	I	TUSER Information. Not used.

M_AXIS_CTRL_SB_RX Interface Ports

Table 2-5: **M_AXIS_CTRL_SB_RX Port Descriptions**

Signal	I/O	Description
M_AXIS_CTRL_SB_RX__tready	I	Core Ready
M_AXIS_CTRL_SB_RX_tvalid	O	Data valid
M_AXIS_CTRL_SB_RX_tdata[31:0]	O	Sideband signal information from transceiver block bit 2:0: rx_mode bit 3: rx_mode_locked bit 4: rx_level_b_3g bit 5: rx_ce bit 31–6: unused

Interrupt Ports

Table 2-6: **Interrupt Port Descriptions**

Signal	I/O	Description
sdi_rx_irq	O	SMPTE UHD-SDI RX core interrupt

SDI_TS_DET_OUT Interface Ports

Table 2-7: **SDI_TS_DET_OUT Port Descriptions**

Signal[(1)]	I/O	Description
SDI_TS_DET_OUT_rx_t_locked	O	This output is High when the transport detection function in the receiver has identified the transport format of the SDI signal (that is, transport locked)
SDI_TS_DET_OUT_rx_t_family[3:0]	O	This output indicates which family of video signals is being used as the transport of the SDI. This output is only valid when rx_t_locked is High. This port does not necessarily identify the video format of the picture being transported. It only identifies the transport characteristics.
SDI_TS_DET_OUT_rx_t_rate[3:0]	O	This output indicates the frame rate of the transport. This is not necessarily the same as the frame rate of the actual picture. This output is only valid when rx_t_locked is High.

Table 2-7: **SDI_TS_DET_OUT Port Descriptions *(Cont'd)***

Signal[1]	I/O	Description
SDI_TS_DET_OUT_rx_t_scan	O	This output indicates whether the transport is interlaced (Low) or progressive (High). This is not necessarily the same as the scan mode of the actual picture. This output is only valid when rx_t_locked is High.

Notes:
1. See Transport Format Detection for more detailed signal descriptions.

Native SDI Ports

These signals are enabled only when Native SDI is selected as the video interface.

Table 2-8: **Native SDI Port Descriptions**

Signal	I/O	Description
sdi_rx_ctrl[63:0]	I	Bit0: module_enable Bit1: not used Bit3:bit2: Reserved Bit4: rx_frame_en Bit5: rx_mode_detect_en Bit6: rx_edh_clr_errcnt Bit7: Reserved Bit13–bit8: rx_mode_enable Bit8: enable HD-SDI mode Bit9: enable SD-SDI mode Bit10: enable 3G-SDI mode Bit11: enable 6G-SDI mode Bit12: enable 12G-SDI 11.88 Gb/s mode Bit13: enable 12G-SDI 11.88/1.001 Gb/s mode. Bit15–bit14: Reserved Bit18–bit16: rx_forced_mode: 000-HD; 001-SD; 010-3G; 100-6G; 101-12G 11.88 Gb/s 110-12G 11.88/1.001 Gb/s Bit31–bit19: Reserved Bit47–bit32: rx_edh_errcnt_en[15:0] Bit63–bit48: Reserved
ST352_DATA_OUT_rx_st352_0[31:0][1]	O	ST352 data for data stream 1 (Y stream of channel 0)
ST352_DATA_OUT_rx_st352_1[31:0][1]	O	ST352 data for data stream 3 (Y stream of channel 1)
ST352_DATA_OUT_rx_st352_2[31:0][1]	O	ST352 data for data stream 5 (Y stream of channel 2)
ST352_DATA_OUT_rx_st352_3[31:0][1]	O	ST352 data for data stream 7 (Y stream of channel 3)
ST352_DATA_OUT_rx_st352_4[31:0][1][2]	O	ST352 data for data stream 9 (Y stream of channel 4)
ST352_DATA_OUT_rx_st352_5[31:0][1][2]	O	ST352 data for data stream 11 (Y stream of channel 5)
ST352_DATA_OUT_rx_st352_6[31:0][1][2]	O	ST352 data for data stream 13 (Y stream of channel 6)
ST352_DATA_OUT_rx_st352_7[31:0][1][2]	O	ST352 data for data stream 15 (Y stream of channel 7)

Table 2-8: **Native SDI Port Descriptions *(Cont'd)***

Signal	I/O	Description
ST352_DATA_OUT_rx_st352_0_valid[(1)]	O	ST352 data valid for data stream 1 (Y stream of channel 0)
ST352_DATA_OUT_rx_st352_1_valid[(1)]	O	ST352 data valid for data stream 3 (Y stream of channel 1)
ST352_DATA_OUT_rx_st352_2_valid[(1)]	O	ST352 data valid for data stream 5 (Y stream of channel 2)
ST352_DATA_OUT_rx_st352_3_valid[(1)]	O	ST352 data valid for data stream 7 (Y stream of channel 3)
ST352_DATA_OUT_rx_st352_4_valid[(1)(2)]	O	ST352 data valid for data stream 9 (Y stream of channel 4)
ST352_DATA_OUT_rx_st352_5_valid[(1)(2)]	O	ST352 data valid for data stream 11 (Y stream of channel 5)
ST352_DATA_OUT_rx_st352_6_valid[(1)(2)]	O	ST352 data valid for data stream 13 (Y stream of channel 6)
ST352_DATA_OUT_rx_st352_7_valid[(1)(2)]	O	ST352 data valid for data stream 15 (Y stream of channel 7)
ST352_DATA_C_STR_OUT_rx_st352_0[31:0]	O	ST352 data valid for data stream 2 (C stream of channel 0)
ST352_DATA_C_STR_OUT_rx_st352_1[31:0]	O	ST352 data valid for data stream 4 (C stream of channel 1) ***Note:*** ST352_DATA_C_STR_OUT_rx_st352_1[31:0] has the same value as ST352_DATA_OUT_rx_st352_1[31:0] for 3G-SDI Level A
ST352_DATA_C_STR_OUT_rx_st352_2[31:0]	O	ST352 data for data stream 6 (C stream of channel 2)
ST352_DATA_C_STR_OUT_rx_st352_3[31:0]	O	ST352 data for data stream 8 (C stream of channel 3)
ST352_DATA_C_STR_OUT_rx_st352_4[31:0][(2)]	O	ST352 data for data stream 10 (C stream of channel 4)
ST352_DATA_C_STR_OUT_rx_st352_5[31:0][(2)]	O	ST352 data for data stream 12 (C stream of channel 5)
ST352_DATA_C_STR_OUT_rx_st352_6[31:0][(2)]	O	ST352 data for data stream 14 (C stream of channel 6)
ST352_DATA_C_STR_OUT_rx_st352_7[31:0][(2)]	O	ST352 data for data stream 16 (C stream of channel 7)
ST352_DATA_C_STR_OUT_rx_st352_0_valid	O	ST352 data valid for data stream 2 (C stream of channel 0)
ST352_DATA_C_STR_OUT_rx_st352_1_valid	O	ST352 data valid for data stream 4 (C stream of channel 1)
ST352_DATA_C_STR_OUT_rx_st352_2_valid	O	ST352 data valid for data stream 6 (C stream of channel 2)
ST352_DATA_C_STR_OUT_rx_st352_3_valid	O	ST352 data valid for data stream 8 (C stream of channel 3)
ST352_DATA_C_STR_OUT_rx_st352_4_valid	O	ST352 data valid for data stream 10 (C stream of channel 4)
ST352_DATA_C_STR_OUT_rx_st352_5_valid	O	ST352 data valid for data stream 12 (C stream of channel 5)

Table 2-8: **Native SDI Port Descriptions *(Cont'd)***

Signal	I/O	Description
ST352_DATA_C_STR_OUT_rx_st352_6_valid	O	ST352 data valid for data stream 14 (C stream of channel 6)
ST352_DATA_C_STR_OUT_rx_st352_7_valid	O	ST352 data valid for data stream 16 (C stream of channel 7)
SDI_DS_OUT_ds1[9:0]	O	SDI data stream 1
SDI_DS_OUT_ds2[9:0]	O	SDI data stream 2
SDI_DS_OUT_ds3[9:0]	O	SDI data stream 3
SDI_DS_OUT_ds4[9:0]	O	SDI data stream 4
SDI_DS_OUT_ds5[9:0]	O	SDI data stream 5
SDI_DS_OUT_ds6[9:0]	O	SDI data stream 6
SDI_DS_OUT_ds7[9:0]	O	SDI data stream 7
SDI_DS_OUT_ds8[9:0]	O	SDI data stream 8
SDI_DS_OUT_ln_num_1[10:0] to SDI_DS_OUT_ln_num_8	O	SDI data stream line number
SDI_DS_OUT_rx_ce_out	O	Clock enable
SDI_DS_OUT_rx_active_streams	O	This port indicates the number of data streams that are active for the current video format being received. The number of active data streams is 2^active_streams. 000: 1 active stream 001: 2 active streams 010: 4 active streams 011: 8 active streams
SDI_DS_OUT_rx_mode_locked	O	When this output is Low, the receiver is actively searching for the SDI mode that matches the input data stream. During this time, the rx_mode_locked output port changes frequently. When the receiver locks to the correct SDI mode, the rx_mode_locked output goes High.
SDI_DS_OUT_rx_eav	O	This output is asserted High when the XYZ word of an EAV is present on the data stream output ports.
SDI_DS_OUT_rx_sav	O	This output is asserted High when the XYZ word of a SAV is present on the data stream output ports.
SDI_DS_OUT_rx_trs	O	This output is asserted High while the four consecutive words of any EAV or SAV are present on the data stream output ports, from the 3FF word through the XYZ word.
SDI_DS_OUT_rx_mode_hd	O	High when RX is locked in HD-SDI mode
SDI_DS_OUT_rx_mode_sd	O	High when RX is locked in SD-SDI mode
SDI_DS_OUT_rx_mode_3g	O	High when RX is locked in 3G-SDI mode
SDI_DS_OUT_rx_mode_6g	O	High when RX is locked in 6G-SDI mode

Table 2-8: **Native SDI Port Descriptions *(Cont'd)***

Signal	I/O	Description
SDI_DS_OUT_rx_mode_12g	O	High when RX is locked in 12G-SDI mode
SDI_DS_OUT_rx_level_b_3g	O	IN 3G-SDI mode, this output is asserted High when the input signal is level B and Low when it is level A. This output is only valid when rx_mode_3g is High.
SDI_DS_OUT_sdi_mode[2:0]	O	This output port indicates the current SDI mode of the receiver: 000 = HD 001 = SD 010 = 3G 100 = 6G 101 = 12G 1000/1000 110 = 12G 1000/1001 When the receiver is not locked, the sdi_mode port changes values as the receiver searches for the correct SDI mode. During this time, the rx_mode_locked output is Low. When the receiver detects the correct SDI mode, the rx_mode_locked output goes High.
sdi_rx_err[31:0]	O	Bit15-bit0: rx_crc_err_ds16 to rx_crc_err_ds1 Bit31-bit16: Reserved

Notes:

1. Enabled only when the Enable AxiLite interface option is *not* selected in the Vivado IDE.
2. Enabled only when 12G-SDI 16DS is selected as the SDI standard in the Vivado IDE.

Native Video Interface Ports

These ports are enable when the video interface is native video.

Table 2-9: **VID_IO_OUT Interface Port Descriptions**

Signal	I/O	Description
VID_IO_OUT_data[n-1:0]	O	Native video output data Where n=bpc*ppc*3. For 10bpc at 2ppc, n=10*2*3=60 For 12bpc at 2ppc, n=12*2*3=72
VID_IO_OUT_active_video	O	Native Video active video output

Table 2-9: **VID_IO_OUT Interface Port Descriptions *(Cont'd)***

Signal	I/O	Description
VID_IO_OUT_field	O	Native video field output For interlaced videos, fid toggles based on the field selected: 0- even field 1- odd field For progressive videos, fid is set to 0. ***Note:*** The fid is expected to toggle in psf and 3G-level B progressive videos, as in these modes, one progressive frame is split into two fields. These two fields are transported independently, similar to interlaced video, and again combined and correctly paired together at the receiver side with the help of the fid signal to construct the original progressive frame. For example, when 1080p 60 Hz video is transported on 3G-SDI level B-DL, the video transport is actually 1080i 60 Hz. The transport is interlaced, but the picture is progressive.
VID_IO_OUT_hblank	O	Native Video hblank output
VID_IO_OUT_vblank	O	Native Video vblank output

SDI_RX_ANC_DS_OUT Interface Ports

These ports are enabled when the Ancillary (ANC) Data Insertion I/F option is selected.

Table 2-10: **SDI_RX_ANC_DS_OUT Interface Port Descriptions**

Signal	I/O	Description
SDI_RX_ANC_DS_OUT_ds1[9:0] to SDI_TX_ANC_DS_OUT_ds8[9:0]	O	SDI stream outputs from 1 to 8
SDI_RX_ANC_DS_OUT_ds9[9:0] to SDI_TX_ANC_DS_OUT_ds16[9:0]	O	SDI stream outputs from 9 to 16. Enabled only when 12 SDI 16 DS option is selected.
sdi_rx_anc_ctrl_out[31:0]	O	Bit 0: rx_ce_out Bit 3: Bit1: rx_mode Bit 7: Bit4: rx_t_family Bit 8: rx_mode_locked Bit 9: rx_t_locked Bit 10 to 13: rx_t_rate Bit 14: t_scan Bit 15: SMPTE UHD-SDI RX IP internal reset Bit 31:16: Reserved Available only when ANC data Insertion I/F option is selected

Transport Format Detection

The SDI receiver has a transport format detector function. This function examines the timing of the video signals in the SDI data streams and determines which video format is being received. The operation of this function is independent of and not dependent on the ST 352 payload ID packets. This function determines the transport format, not the picture format. Usually these are the same, but not always. For example, when 1080p 60 Hz video is transported on 3G-SDI level B-DL, the video transport is actually 1080i 60 Hz – the transport is interlaced, but the picture is progressive.

The transport format detection function determines the transport format by looking only at the video timing. It cannot distinguish between video formats that have exactly the same timing. For example, progressive segmented frame (PsF) video formats are intentionally designed to work with identical timing to corresponding interlaced formats. This means the receiver cannot distinguish interlaced formats by examining the timing. The transport format detection function reports PsF video formats as interlaced formats (rx_t_scan is Low). Your application must examine the ST 352 payload ID packets to discover whether the actual video format is PsF or interlaced.

The 6G-SDI and 12G-SDI mappings defined by SMPTE always subdivide the images into multiple sub-images. Each sub-image is formatted as a regular 1080p image. The transport format detection function examines the timing of data stream 1 (ds1) only. Thus, it reports the video transport of 6G-SDI and 12G-SDI signals as 1080p signals (rx_t_family = 0000 and rx_t_scan = 1).

The rx_t_family output port provides a 4-bit code that indicates the matching video format family. The encoding of this output port is shown in Table 2-11. The transport detection function also determines whether or not the transport is interlaced or progressive and reports this on the rx_t_scan output port.

Table 2-11: **Video Format Encoding**

Code	Transport Video Format	Active Pixels
0000	SMPTE ST 274	1920 x 1080
0001	SMPTE ST 296	1280 x 720
0010	SMPTE ST 2048-2	2048 x 1080
0011	SMPTE ST 295	1920 x 1080
1000	NTSC	720 x 486
1001	PAL	720 x 576
1111	Unknown	
Others	Reserved	

The transport detector also determines the frame rate of the transport signal. This feature is dependent on the **rx_bit_rate** input in HD-SDI and 3G-SDI modes to distinguish

between integer and non-integer frame rates. The **rx_t_rate** port indicates the frame rate of the transport signal as shown in Table 2-12 The encoding of the frame rate matches the encoding used in the picture rate field of SMPTE ST 352 payload ID packets. However, **rx_t_rate** shows the transport frame rate, not the picture rate. Also, the **rx_t_rate** port value is always the frame rate, even for interlaced transports.

Table 2-12: **Frame Rate Encoding**

Code	Frame Rate (Hz)
0000	None
0001	96
0010	23.98
0011	24
0100	47.95
0101	25
0110	29.97
0111	30
1000	48
1001	50
1010	59.94
1011	60
1100	95.9
1101	100
1110	120
1111	119.88

Software Structure

This section describes how transport information is handled in software drivers.The driver contains 2 important structures that are returned to the application interface.

XVidC_VideoStream and XSdiVid_Transport.

Both of these structures are required to determine the received picture and framerate information.

XSdiVid_Transport

This contains information about the transport layer of SDI (The values of the members in this structure are filled by reading the IP registers 0x44 & 0x48).

Each member description is given as comment for that member.

```
typedef struct {
  XSdiVid_TransMode      TMode;    /* The Transport mode, see the definition of this enum
below */
  u8        ActiveStreams;              /* Number active streams */
  u8        TScan;                      /* Transport scan of the current transport mode */
  u8        TFamily;                    /* The family of the video signals */
  u8        TRate;                      /* Transport frame rate */
  u8       IsFractional;               /* Indicates if the transport frame rate is fractional or
not */
  u8        IsLevelB3G;                 /* If the Transport mode is 3GB */
} XSdiVid_Transport;
```

The below structure gives information regarding the available transport modes:

```
typedef enum {
        XSDIVID_MODE_HD = 0,
        XSDIVID_MODE_SD = 1,
        XSDIVID_MODE_3GA = 2,
        XSDIVID_MODE_3GB = 3,
        XSDIVID_MODE_6G = 4,
        XSDIVID_MODE_12G = 5,
        XSDIVID_MODE_12GF = 6
} XSdiVid_TransMode;
```

XVidC_VideoStream

This contains information about the video stream. This is filled up from SDI payload and sdi video transport layer data.

Each member description is given as comment for that member.

```
typedef struct {
        XVidC_ColorFormat       ColorFormatId;  /* The received color format */
        XVidC_ColorDepth        ColorDepth;     /* The received color/bit depth */
        XVidC_PixelsPerClock    PixPerClk;      /* Pixels per clock */
        XVidC_FrameRate         FrameRate;      /* Frame rate of the video stream*/
        XVidC_AspectRatio       AspectRatio;    /* Aspect ratio of the frame */
        u8                      IsInterlaced;   /* If the stream is interlaced or not */
        u8                      Is3D;           /* NA for SDI */
        XVidC_3DInfo            Info_3D;        /* NA for SDI */
        XVidC_VideoMode         VmId;           /* Video mode ID */
        XVidC_VideoTiming       Timing;         /* Video timings of the VmId */
        XVidC_Eotf              Eotf;           /* Electro Optical Transfer Function */
        XVidC_ColorStd          ColorStd;       /* Color space conversion standard */
        XVidC_FrameRate         BaseFrameRate; /* NA for SDI */
        XVidC_VideoTiming       BaseTiming;     /* NA for SDI */
} XVidC_VideoStream;
```

High Dynamic Range Data

This SMPTE UHD-SDI RX IP supports only HLG HDR. The IP extracts the ST 352 payload information from the data. Refer ST 2082-10_2018 specification for the payload details. Bits 4,5 from byte 2 on ST352 are used to determine HLG HDR.

If AXI4-Lite registers are enabled (C_INCLUDE_AXILITE is true), IP passes the information to user through registers.

If AXI4-Lite registers are disabled (C_INCLUDE_AXILITE is false), IP uses the Native SDI interface ports for sending ST352 payload information. Refer Table 2-8 for the ST352 related ports.

Register Space

This section details registers available in the SMPTE UHD-SDI RX Subsystem. The SMPTE UHD-SDI RX core has an address space of 64K.

SMPTE UHD-SDI RX Registers

The SMPTE UHD-SDI RX registers are available when the AXI4-Lite Interface is selected in the Vivado IDE. The SMPTE UHD-SDI RX IP core register space is shown in Table 2-13.

IMPORTANT: *This memory space must be aligned to an AXI word (32-bit) boundary.*

All registers are in little endian format as shown in Figure 2-1.

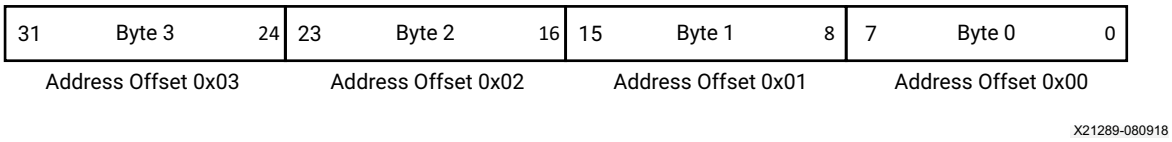

Figure 2-1: **32-bit Little Endian Example**

Table 2-13: **SMPTE UHD-SDI RX IP Core Register Space**

Offset	Name	Width	Access	Description
0x00	RST_CTRL	32-bit	R/W	Enable and soft reset controls for the IP core
0x04	MODULE_CTRL	32-bit	R/W	Module control register
0x08	RESERVED	32-bit	N/A	N/A
0x0C	GLBL_IER	32-bit	R/W	Global interrupt enable register
0x10	ISR	32-bit	R/W1C	Interrupt status register
0x14	IER	32-bit	R/W	Interrupt enable register
0x18	RX_ST352_VALID	32-bit	RO	ST352 packet valid indication
0x1C	RX_ST352_DATA_DS1	32-bit	RO	Data stream 1 ST352 packet data
0x20	RX_ST352_DATA_DS3	32-bit	RO	Data stream 3 ST352 packet data
0x24	RX_ST352_DATA_DS5	32-bit	RO	Data stream 5 ST352 packet data
0x28	RX_ST352_DATA_DS7	32-bit	RO	Data stream 7 ST352 packet data
0x2C	RX_ST352_DATA_DS9	32-bit	RO	Data stream 9 ST352 packet data
0x30	RX_ST352_DATA_DS11	32-bit	RO	Data stream 11 ST352 packet data
0x34	RX_ST352_DATA_DS13	32-bit	RO	Data stream 13 ST352 packet data
0x38	RX_ST352_DATA_DS15	32-bit	RO	Data stream 15 ST352 packet data
0x3C	VERSION	32-bit	RO	Version Register
0x40	SS_CONFIG	32-bit	RO	IP core Configuration

Table 2-13: **SMPTE UHD-SDI RX IP Core Register Space *(Cont'd)***

Offset	Name	Width	Access	Description
0x44	MODE_DET_STS	32-bit	RO	Mode detect status
0x48	TS_DET_STS	32-bit	RO	Transport Stream detect status
0x4C	RX_EDH_STS	32-bit	RO	EDH check status
0x50	RX_EDH_ERRCNT_EN	32-bit	R/W	Enable EDH error count
0x54	RX_EDH_ERRCNT	32-bit	RO	RX EDH error count
0x58	RX_CRC_ERR	32-bit	RO	RX CRC error indication
0x5C	VIDEO_LOCK_WINDOW	32-bit	R/W	Video lock window
0x60	RESERVED	32-bit	N/A	N/A
0x64	RESERVED	32-bit	N/A	N/A
0x68	RESERVED	32-bit	N/A	N/A
0x6C	RESERVED	32-bit	N/A	N/A
0x70	RX_ST352_DATA_DS2	32-bit	RO	Data stream 2 ST352 packet data
0x74	RX_ST352_DATA_DS4	32-bit	RO	Data stream 4 ST352 packet data
0x78	RX_ST352_DATA_DS6	32-bit	RO	Data stream 6 ST352 packet data
0x7C	RX_ST352_DATA_DS8	32-bit	RO	Data stream 8 ST352 packet data
0x80	RX_ST352_DATA_DS10	32-bit	RO	Data stream 10 ST352 packet data
0x84	RX_ST352_DATA_DS12	32-bit	RO	Data stream 12 ST352 packet data
0x88	RX_ST352_DATA_DS14	32-bit	RO	Data stream 14 ST352 packet data
0x8C	RX_ST352_DATA_DS16	32-bit	RO	Data stream 16 ST352 packet data

Notes:

1. Access type and reset value for all the reserved bits in the registers is read-only with value 0.
2. Register accesses should be word aligned and there is no support for a write strobe. WSTRB is not used internally.
3. Only the lower 7 bits (6:0) of the read and write address of the AXI4-Lite interface are decoded. This means that accessing address 0x00 and 0x80 results in reading the same address of 0x00.
4. Reads and writes to addresses outside this table do not return an error.

RST_CTRL Register (0x00)

This register allows you to enable and disable the SMPTE UHD-SDI RX IP core and apply a soft reset during core operation.

Table 2-14: **RST_CTRL Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:13	Reserved	RO	0	Reserved
12:10	SDIRX_BRIDGE_CH_FORMAT_AXI	R/W	0	Selection of chroma format 001 - YCbCr 4:4:4 format selection Others - Reserved

Table 2-14: **RST_CTRL Register Bit Mapping *(Cont'd)***

Bits	Name	Access	Default Value	Description
9	VID_IN_AXI4S_MOD_EN	R/W	0	Enable bit for Video-in-AXI4S core 1 – Video-in-AXI4S core is enabled 0 – Video-in-AXI4S core is disabled This bit is available only in AXI4-Stream is selected as the video interface
8	SDIRX_BRIDGE_EN	R/W	0	Enable bit for SDI RX Bridge 1 – SDI RX bridge is enabled 0 – SDI RX bridge is disabled This bit is not available if Native SDI is selected as video interface
7:4	Reserved	RO	0	Reserved
3	RST_EDH_ERRCNT	R/W	0	Clear rx_edh_errcnt register
2	RST_CRC_ERRCNT	R/W	0	Clear rx_crc_errcnt register
1	SRST	R/W	0	Soft reset for SDI RX IP core Writing a 1 to this bit resets all registers of the SDI RX IP.
0	SDIRX_IP_EN	R/W	0	Enable bit for SDI RX IP core 1 – SDI RX IP core is enabled 0 – SDI RX IP core is disabled

MODULE_CTRL Register (0x04)

This register provides control of the SMPTE UHD-SDI RX IP core and core functional modes.

Table 2-15: **MODULE_CTRL Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:19	Reserved	RO	0	Reserved
18:16	RX_FORCED_MODE	R/W	0	RX forced mode 3'b000: HD mode 3'b001: SD mode 3'b010: 3G mode 3'b100: 6G mode 3'b101: 12G mode with 11.88 Gb/s line rate 3'b110: 12G mode with 11.88/1.001 Gb/s line rate
15:14	Reserved	RO	0	Reserved

Table 2-15: **MODULE_CTRL Register Bit Mapping *(Cont'd)***

Bits	Name	Access	Default Value	Description
13:8	RX_MODE_EN	R/W	0	RX mode enable Bit8: enable HD-SDI mode Bit9: enable SD-SDI mode Bit10: enable 3G-SDI mode Bit11: enable 6G-SDI mode Bit12: enable 12G-SDI 11.88 Gb/s mode Bit13: enable 12G-SDI 11.88/1.001 Gb/s mode
7:6	Reserved	RO	2'b01	Reserved
5	RX_MODE_DET_EN	R/W	0	RX mode detection enable
4	RX_FRM_EN	R/W	0	RX frame enable
3:0	Reserved	RO	0	Reserved

Global Interrupt Enable Register (GLBL_IER) (0x0C)

Table 2-16: **GLBL_IER Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:1	Reserved	RO	0	Reserved
0	GLBL_INTRUPT_EN	R/W	0	Master enable for the device interrupt output to the system 1: Enabled—the corresponding Interrupt Enable register (IER) bits are used to generate interrupts 0: Disabled—Interrupt generation blocked irrespective of IER bits

Interrupt Status Register (ISR) (0x10)

This register captures the error and status information for the IP core.

Table 2-17: **ISR bit mapping**

Bits	Name	Access[(1)]	Default Value	Description
31:11	Reserved	RO	0	Reserved
10	UNDERFLOW_INTR	R/W1C	0	Video in to AXI4-Stream core underflow indication. This bit is available only in AXI4-Stream is selected as Video Interface.
9	OVERFLOW_INTR	R/W1C	0	Video in to AXI4-Stream core overflow indication. This bit is available only in AXI4-Stream is selected as Video Interface.

Table 2-17: **ISR bit mapping *(Cont'd)***

Bits	Name	Access[1]	Default Value	Description
8:3	Reserved	RO	0	Reserved
2	VSYNC_VALID_INTR	R/W1C	0	Asserted when Video sync has been detected at the start of each frame.
1	VIDEO_UNLOCK_INTR	R/W1C	0	Asserted when incoming video pattern is unlocked
0	VIDEO_LOCK_INTR	R/W1C	0	Asserted when incoming video pattern is locked and ST352 Valid bit of Data Stream1 (RX_ST352_VLD_DS1 field) is asserted.

Notes:
1. W1C = Write 1 to clear.

Interrupt Enable Register (IER) (0x14)

This register allows you to selectively generate an interrupt at the output port for each error/status bit in the ISR. An IER bit set to 0 does not inhibit an error/status condition from being captured, but inhibits it from generating an interrupt.

Table 2-18: **IER bit mapping**

Bits	Name	Access	Default Value	Description
31:11	Reserved	RO	0	Set bits in this register to 1 to generate the required interrupts. Set to 0 to disable the interrupt. For a description of the specific interrupt you are enabling/disabling in this register see the ISR descriptions in Table 2-17.
10	UNDERFLOW_INTR_EN	R/W	0	
9	OVERFLOW_INTR_EN	R/W	0	
8:3	Reserved	RO	0	
2	VSYNC_VALID_IE	R/W	0	
1	VIDEO_UNLOCK_INTR_EN	R/W	0	
0	VIDEO_LOCK_INTR_EN	R/W	0	

RX_ST352_VALID Register (0x18)

Table 2-19: **RX_ST352_VALID Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:16	Reserved	RO	0	Reserved
15	RX_ST352_VLD_DS16	RO	0	Asserted high when ST352 is valid on data stream 16 (C stream of Channel 7)
14	RX_ST352_VLD_DS14	RO	0	Asserted high when ST352 is valid on data stream 14 (C stream of Channel 6)

Table 2-19: **RX_ST352_VALID Register Bit Mapping *(Cont'd)***

Bits	Name	Access	Default Value	Description
13	RX_ST352_VLD_DS12	RO	0	Asserted high when ST352 is valid on data stream 12 (C stream of Channel 5)
12	RX_ST352_VLD_DS10	RO	0	Asserted high when ST352 is valid on data stream 10 (C stream of Channel 4)
11	RX_ST352_VLD_DS8	RO	0	Asserted high when ST352 is valid on data stream 8 (C stream of Channel 3)
10	RX_ST352_VLD_DS6	RO	0	Asserted high when ST352 is valid on data stream 6 (C stream of Channel 2)
9	RX_ST352_VLD_DS4	RO	0	Asserted high when ST352 is valid on data stream 4 (C stream of Channel 1)
8	RX_ST352_VLD_DS2	RO	0	Asserted high when ST352 is valid on data stream 2 (C stream of Channel 0)
7	RX_ST352_VLD_DS15	RO	0	Asserted high when ST352 is valid on data stream 15 (Y stream of Channel 7)
6	RX_ST352_VLD_DS13	RO	0	Asserted high when ST352 is valid on data stream 13 (Y stream of Channel 6)
5	RX_ST352_VLD_DS11	RO	0	Asserted high when ST352 is valid on data stream 11 (Y stream of Channel 5)
4	RX_ST352_VLD_DS9	RO	0	Asserted high when ST352 is valid on data stream 9 (Y stream of Channel 4)
3	RX_ST352_VLD_DS7	RO	0	Asserted high when ST352 is valid on data stream 7 (Y stream of Channel 3)
2	RX_ST352_VLD_DS5	RO	0	Asserted high when ST352 is valid on data stream 5 (Y stream of Channel 2)
1	RX_ST352_VLD_DS3	RO	0	Asserted high when ST352 is valid on data stream 3 (Y stream of Channel 1)
0	RX_ST352_VLD_DS1	RO	0	Asserted high when ST352 is valid on data stream 1 (Y stream of Channel 0)

RX_ST352_DATA_DS1 Register (0x1C)

Table 2-20: **RX_ST352_DATA_DS1 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS1	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 1

RX_ST352_DATA_DS3 Register (0x20)

Table 2-21: **RX_ST352_DATA_DS3 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS3	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 3

RX_ST352_DATA_DS5 Register (0x24)

Table 2-22: **RX_ST352_DATA_DS5 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS5	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 5

RX_ST352_DATA_DS7 Register (0x28)

Table 2-23: **RX_ST352_DATA_DS7 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS7	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 7

RX_ST352_DATA_DS9 Register (0x2C)

Table 2-24: **RX_ST352_DATA_DS9 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS9	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 9

RX_ST352_DATA_DS11 Register (0x30)

Table 2-25: **RX_ST352_DATA_DS11 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS11	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 11

RX_ST352_DATA_DS13 Register (0x34)

Table 2-26: **RX_ST352_DATA_DS13 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS13	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 13

RX_ST352_DATA_DS15 Register (0x38)

Table 2-27: **RX_ST352_DATA_DS15 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS15	RO	0	The ST 352 payload ID packet data bytes captured from Y stream of Channel 15

VERSION Register (0x3C)

Table 2-28: **VERSION Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	VERSION	32-bit	32'h02_00_0_0_00	For uhd_sdi_rx_ss_v1_0, VERSION REGISTER is 32'h01_00_0_0_00. For uhd_sdi_rx_ss_v2_0, VERSION REGISTER is 32'h02_00_0_0_00. • [31:24] - Subsystem major version. • [23:16] - Subsystem minor version. • [15:12] - Subsystem version revision. • [11:8] - Subsystem Patch details. • [7:0] - Internal revision.

SS_CONFIG Register (0x40)

Table 2-29: **SS_CONFIG Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:5	Reserved	RO	0	Reserved
4	ANC_IF	RO	0	ANC Interface Enable
3:2	VID_INTF	RO	0	Video Interface 2'b00: AXI4-Stream Interface 2'b01: Native Video Interface 2'b10: Native SDI
1	INC_RX_EDH_PROC	RO	1	This bit will be set if the IP core generated with INCLUDE_RX_EDH_PROCESSOR
0	Reserved	RO	0	Reserved

MODE_DET_STS Register (0x44)

This register provides the SDI mode detection status.

Table 2-30: **ODE_DET_STS Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:8	Reserved	RO	0	Reserved
7	RX_3G_LEVEL_B	RO	0	Asserted High when incoming stream is 3G-SDI level B
6:4	RX_ACT_STREAMS	RO	1	RX active data streams 3'b000: 1 active stream; 3'b001: 2 active streams (Default); 3'b010: 4 active streams; 3'b011: 8 active streams; 3'b100: 16 active streams;
3	RX_MODE_LOCKED	RO	1	RX mode locked indication If bit 5 of MODULE_CTRL is not set, then this bit will be set to 1'b1
2:0	RX_MODE	RO	0	3'b000: HD-SDI Mode (default); 3'b001: SD-SDI Mode; 3'b010: 3G-SDI Mode; 3'b100: 6G-SDI Mode; 3'b101: 12G-SDI 11.88 Gb/s Mode; 3'b110: 12G-SDI 11.88/1.001 Gb/s Mode;

TS_DET_STS Register (0x48)

This register provides the transport stream detection status.

Table 2-31: **TS_DET_STS Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:12	Reserved	RO	0	Reserved
11:8	RX_T_RATE	RO	0	This bit indicates the frame rate of the transport. This is not necessarily the same as the frame rate of the actual picture. This bit is only valid when RX_T_LOCKED is High. See Table 2-8 in the *SMPTE UHD-SDI Product Guide* (PG205) [Ref 9] for details on the encoding of the bits.
7:4	RX_T_FAMILY	RO	4'hF	This bit indicates which family of video signals is being used as the transport of the SDI. This bit is only valid when RX_T_LOCKED is High. This bit does not necessarily identify the video format of the picture being transported. It only identifies the transport characteristics. See Table 2-7 in the *SMPTE UHD-SDI Product Guide* (PG205) [Ref 9] for details on the encoding of the bits.
3	Reserved	RO	0	Reserved

Table 2-31: **TS_DET_STS Register Bit Mapping *(Cont'd)***

Bits	Name	Access	Default Value	Description
2	RX_is_HFR	RO	1	This bit indicates that the resolution detected if a HFR resolution.
1	RX_T_SCAN	RO	0	This bit indicates whether the transport is interlaced (Low) or progressive (High). This is not necessarily the same as the scan mode of the actual picture. This bit is only valid when RX_T_LOCKED is High
0	RX_T_LOCKED	RO	0	Asserted High when the transport detection function in the receiver has identified the transport format of the SDI signal.

RX_EDH_STS Register (0x4C)

Table 2-32: **RX_EDH_STS Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:23	Reserved	RO	0	Reserved
22:19	RX_EDH_PKT_FLAGS	RO	0	This four error flags related to the most recently received EDH packet. See Table 2-5 in the *SMPTE UHD-SDI Product Guide* (PG205) [Ref 9] for details on the encoding of the bits.
18:14	RX_EDH_ANC_FLAGS	RO	0	The ancillary error flag bits from the most recently received EDH packet. See Table 2-5 in the *SMPTE UHD-SDI Product Guide* (PG205) [Ref 9] for details on the encoding of the bits.
13:9	RX_EDH_FF_FLAGS	RO	0	The full frame error flag bits from the most recently received EDH packet. See Table 2-5 in the *SMPTE UHD-SDI Product Guide* (PG205) [Ref 9] for details on the encoding of the bits.
8:4	RX_EDH_AP_FLAGS	RO	0	The active picture error flag bits from the most recently received EDH packet. See Table 2-5 in the *SMPTE UHD-SDI Product Guide* (PG205) [Ref 9] for details on the encoding of the bits.
3	Reserved	RO	0	Reserved
2	RX_EDH_ANC	RO	0	This output is asserted High when an ancillary data packet checksum error is detected.
1	RX_EDH_FF	RO	0	This bit is asserted High when the full field CRC calculated for the previous field does not match the FF CRC value in the EDH packet.
0	RX_EDH_AP	RO	0	This bit is asserted High when the active picture CRC calculated for the previous field does not match the AP CRC value in the EDH packet.

RX_EDH_ERRCNT_EN Register (0x50)

Table 2-33: **RX_EDH_ERRCNT_EN Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:16	Reserved	RO	0	Reserved
15	EDH_PKT_CHKSUM_ERR	R/W	0	EDH packet checksum-error
14	AP_UES_ERR_EN	R/W	0	AP UES error
13	AP_IDA_ERR_EN	R/W	0	AP IDA error
12	AP_IDH_ERR_EN	R/W	0	AP IDH error
11	AP_EDA_ERR_EN	R/W	0	AP EDA error
10	AP_EDH_ERR_EN	R/W	0	AP EDH error
9	FF_UES_ERR_EN	R/W	0	FF UES error
8	FF_IDA_ERR_EN	R/W	0	FF IDA error
7	FF_IDH_ERR_EN	R/W	0	FF IDH error
6	FF_EDA_ERR_EN	R/W	0	FF EDA error
5	FF_EDH_ERR_EN	R/W	0	FF EDH error
4	ANC_UES_ERR_EN	R/W	0	ANC UES error
3	ANC_IDA_ERR_EN	R/W	0	ANC IDA error
2	ANC_IDH_ERR_EN	R/W	0	ANC IDH error
1	ANC_EDA_ERR_EN	R/W	0	ANC EDA error
0	ANC_EDH_ERR_EN	R/W	0	ANC EDH error

RX_EDH_ERRCNT Register (0x54)

Table 2-34: **RX_EDH_ERRCNT Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:16	Reserved	RO	0	Reserved
15:0	RX_EDH_ERRCNT	RO	0	SD-SDI mode EDH error counter. It increments once per field when any of the error conditions enabled by the RX_EDH_ERRCNT_EN register bit(s) occur during that field.

RX_CRC_ERR Register (0x58)

Table 2-35: **RX_CRC_ERR Register Bit Mapping**

Bits	Name	Access[(1)]	Default Value	Description
31:16	RX_CRC_ERR_CNT	RO	0	Cumulative CRC error count of data stream 1 to 16
15	RX_CRC_ERR_DS16	R/W1C	0	CRC error indicator for each data stream 16
14	RX_CRC_ERR_DS15	R/W1C	0	CRC error indicator for each data stream 15

Table 2-35: **RX_CRC_ERR Register Bit Mapping *(Cont'd)***

Bits	Name	Access[1]	Default Value	Description
13	RX_CRC_ERR_DS14	R/W1C	0	CRC error indicator for each data stream 14
12	RX_CRC_ERR_DS13	R/W1C	0	CRC error indicator for each data stream 13
11	RX_CRC_ERR_DS12	R/W1C	0	CRC error indicator for each data stream 12
10	RX_CRC_ERR_DS11	R/W1C	0	CRC error indicator for each data stream 11
9	RX_CRC_ERR_DS10	R/W1C	0	CRC error indicator for each data stream 10
8	RX_CRC_ERR_DS9	R/W1C	0	CRC error indicator for each data stream 9
7	RX_CRC_ERR_DS8	R/W1C	0	CRC error indicator for each data stream 8
6	RX_CRC_ERR_DS7	R/W1C	0	CRC error indicator for each data stream 7
5	RX_CRC_ERR_DS6	R/W1C	0	CRC error indicator for each data stream 6
4	RX_CRC_ERR_DS5	R/W1C	0	CRC error indicator for each data stream 5.
3	RX_CRC_ERR_DS4	R/W1C	0	CRC error indicator for each data stream 4
2	RX_CRC_ERR_DS3	R/W1C	0	CRC error indicator for each data stream 3
1	RX_CRC_ERR_DS2	R/W1C	0	CRC error indicator for each data stream 2
0	RX_CRC_ERR_DS1	R/W1C	0	CRC error indicator for each data stream 1

Notes:
1. W1C = Write 1 to clear.

VIDEO_LOCK_WINDOW Register (0x5C)

Table 2-36: **VIDEO_LOCK_WINDOW Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	VIDEO_LOCK_WINDOW	R/W	0	Number of rx_clk cycles of stable video before asserting video is locked

RX_ST352_DATA_DS2 Register (0x70)

Table 2-37: **RX_ST352_DATA_DS2 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS2	RO	0	The ST 352 payload ID packet data bytes captured from data stream 2 (C stream of data channel 0)

RX_ST352_DATA_DS4 Register (0x74)

Table 2-38: **RX_ST352_DATA_DS4 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS4	RO	0	The ST 352 payload ID packet data bytes captured from data stream 4 (C stream of data channel 1)

RX_ST352_DATA_DS6 Register (0x78)

Table 2-39: **RX_ST352_DATA_DS6 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS6	RO	0	The ST 352 payload ID packet data bytes captured from data stream 6 (C stream of data channel 2)

RX_ST352_DATA_DS8 Register (0x7C)

Table 2-40: **RX_ST352_DATA_DS8 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS8	RO	0	The ST 352 payload ID packet data bytes captured from data stream 8 (C stream of data channel 3)

RX_ST352_DATA_DS10 Register (0x80)

Table 2-41: **RX_ST352_DATA_DS10 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS10	RO	0	The ST 352 payload ID packet data bytes captured from data stream 10 (C stream of data channel 4)

RX_ST352_DATA_DS12 Register (0x84)

Table 2-42: **RX_ST352_DATA_DS12 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS12	RO	0	The ST 352 payload ID packet data bytes captured from data stream 12 (C stream of data channel 5)

RX_ST352_DATA_DS14 Register (0x88)

Table 2-43: **RX_ST352_DATA_DS14 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS14	RO	0	The ST 352 payload ID packet data bytes captured from data stream 14 (C stream of data channel 6)

RX_ST352_DATA_DS16 Register (0x8C)

Table 2-44: **RX_ST352_DATA_DS16 Register Bit Mapping**

Bits	Name	Access	Default Value	Description
31:0	RX_ST352_DATA_DS16	RO	0	The ST 352 payload ID packet data bytes captured from data stream 16 (C stream of data channel 7)

Versal ACAP Block Automation in UHD-SDI RX Subsystem

The Block Automation feature is provided in IP integrator to help you put together a basic system that connects the UHDSDI RX Subsystem IP with GT Quad using a GT Controller Bridge and a connection between Subsystem IP ports and external I/O ports.

To set up block automation:

1. Click on **Run Block Automation**, as shown in the following figure. Block automation connects the TX and RX data paths of the Parent IP to GT Wizard. (A new GT Wizard quad base is launched if it cannot pack the Parent IP with existing GT Quad resources.)

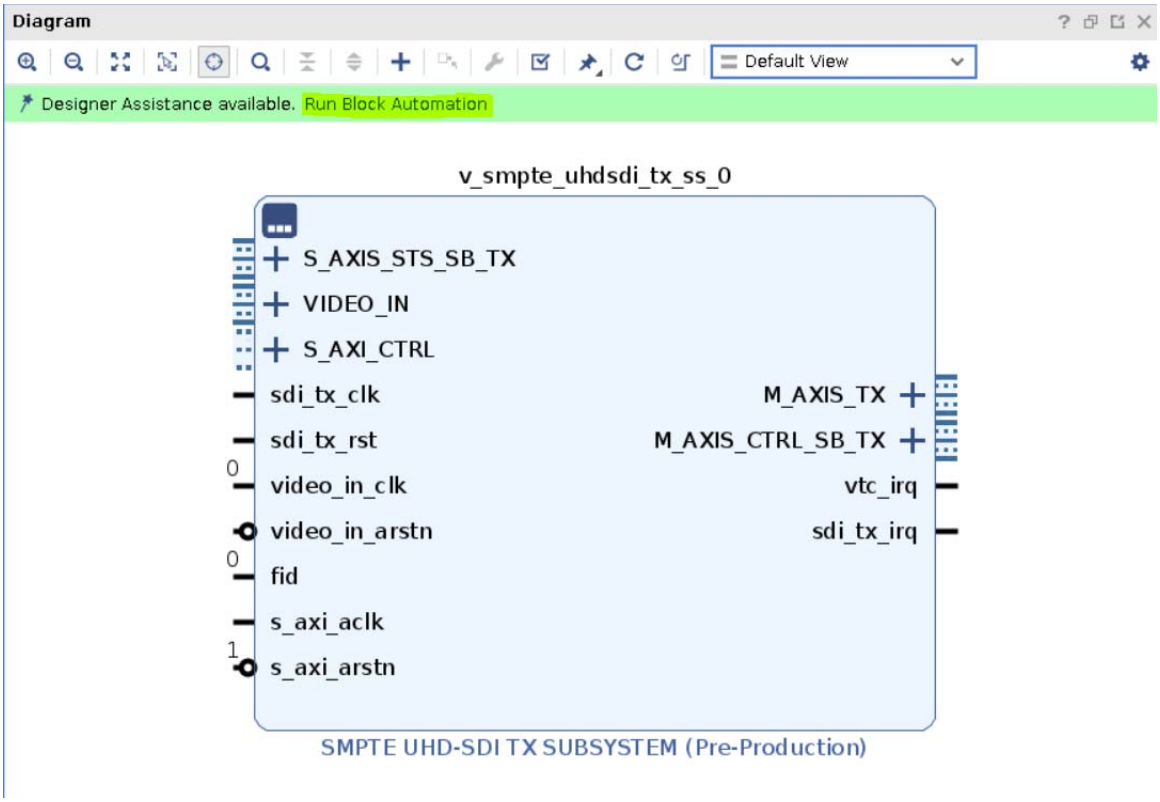

Figure 2-1: **Run Block Automation**

2. Use the Run Block Automation dialog box to specify the basic options that the UHDSDI RX subsystem IP needs. (Auto is selected by default.)

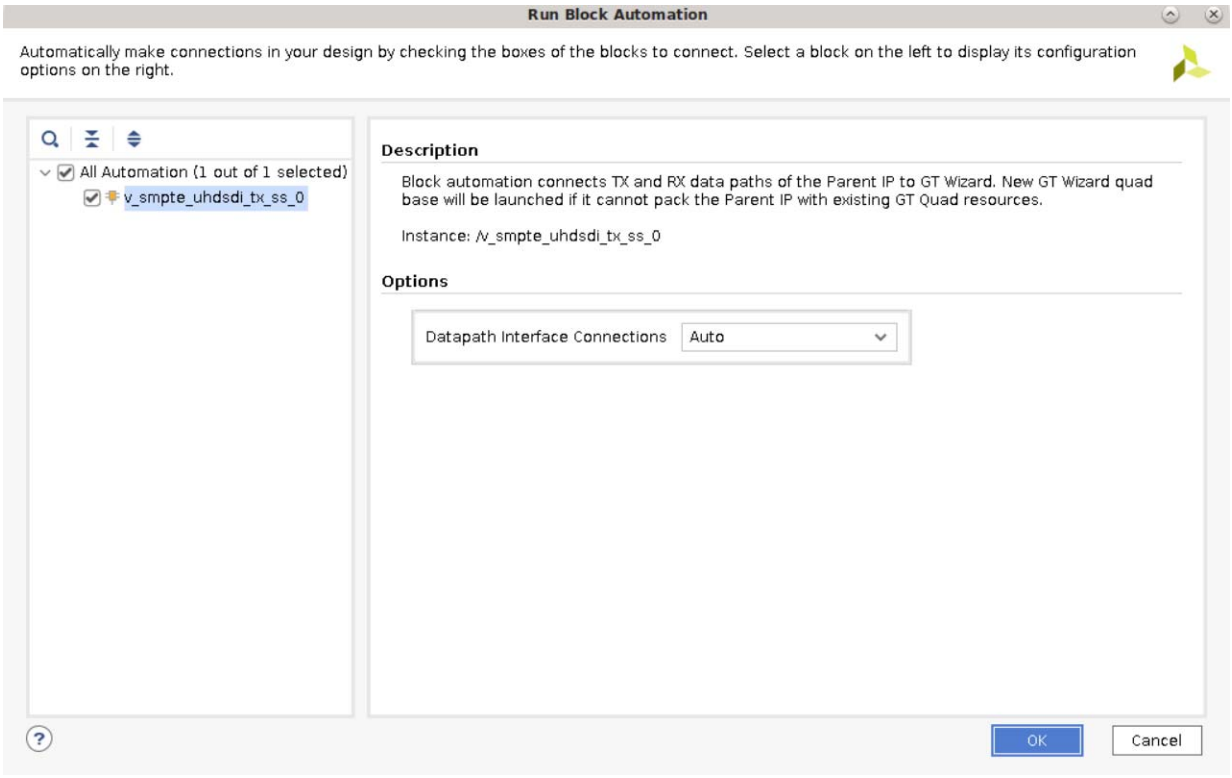

Figure 2-2: **Block Automation Options**

3. Click **OK**. The Block Automation feature then automatically creates a basic system as shown in the figure below. This example shows a basic system that consists of a UHDSDI RX Subsystem IP, a Versal GT controller Bridge IP for SDI, the clock buffer, and the GT Quad for Versal ACAP devices. Both clocks get connected to a clock source. (Because the design does not connect to any external I/O at this point, the IP integrator provides the Connection Automation feature as shown by the highlighted area in the figure.)

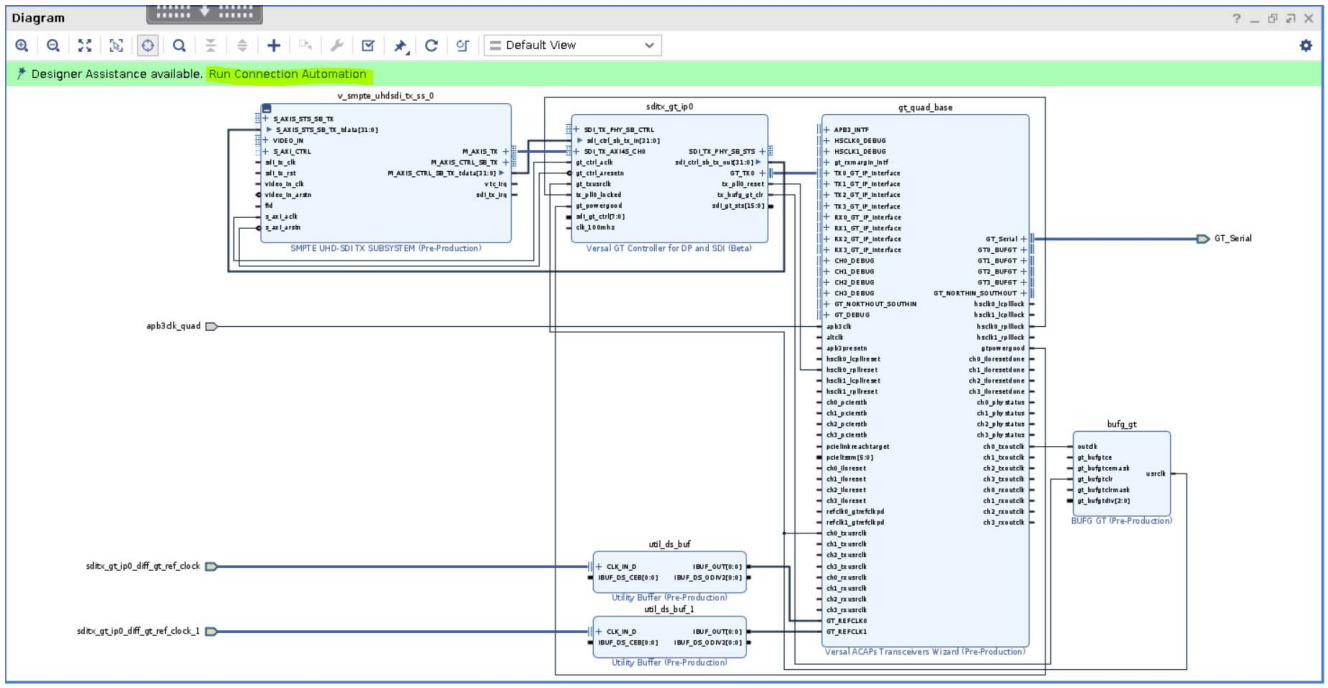

Figure 2-3: **Auto-connected Block, Basic System**

4. Click on **Run Connection Automation** to get assistance in hooking interfaces and/or ports to external I/O ports. The following figure shows a list of the ports/interfaces that can use the Connection Automation feature.)

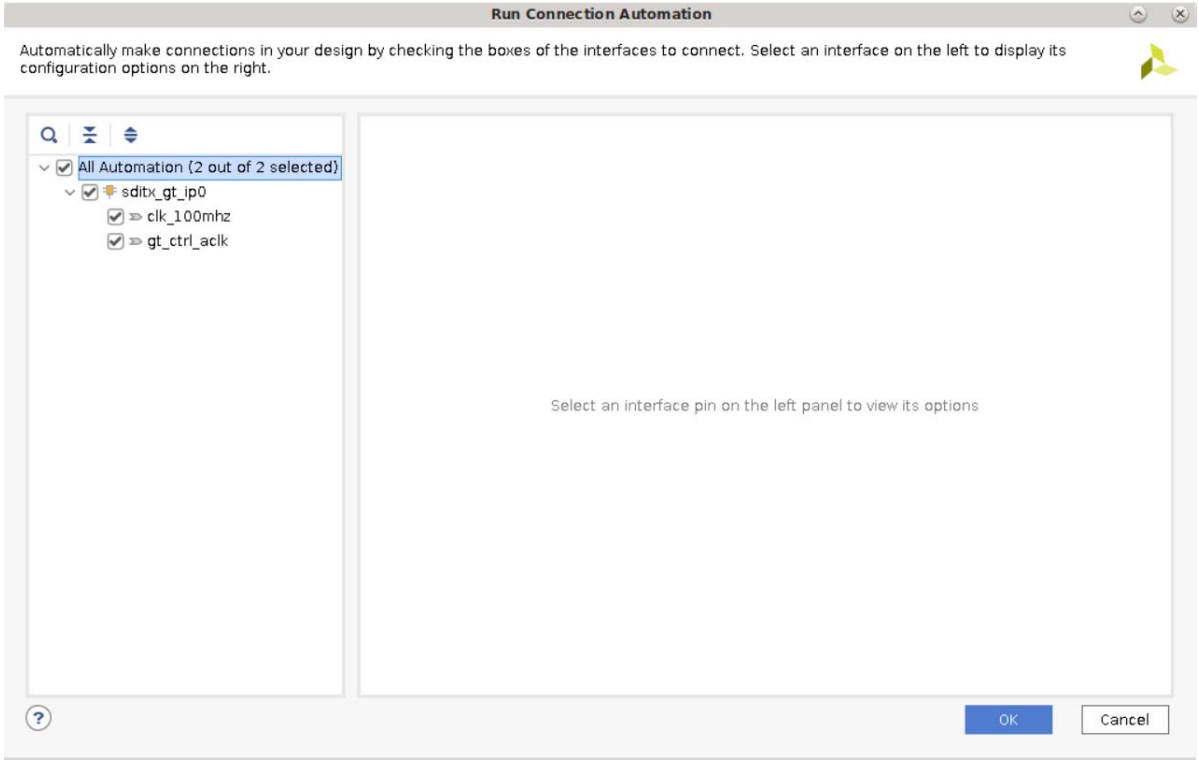

Figure 2-4: **Run Connection Automation**

5. Click **OK**. Both the clocks as shown in step 3 above are then connected to a clock source.

Designing with the Subsystem

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

General Design Guidelines

This section describes the steps required to turn a SMPTE UHD-SDI RX Subsystem into a fully functioning design with user-application logic.

IMPORTANT: *Not all implementations require all of the design steps listed here. Follow the logic design guidelines in this manual carefully.*

Use the Example Design as a Starting Point

Each instance of a SMPTE UHD-SDI RX Subsystem that is created is delivered with an example design that can be implemented in Xilinx FPGA. This design can be used as a starting point for your own design or can be used to troubleshoot the user application, if necessary.

Know the Degree of Difficulty

SMPTE UHD-SDI RX Subsystem design is challenging to implement in any technology, and the degree of difficulty is further influenced by:

- Maximum system clock frequency
- Targeted device architecture
- Nature of the user application

All SMPTE UHD-SDI RX Subsystem implementations require careful attention to system performance requirements. Pipelining, logic mappings, placement constraints and logic duplications are all methods that help boost system performance.

Keep It Registered

To simplify timing and increase system performance in an FPGA design, keep all inputs and outputs registered with flip-flops between the user application and the subsystem. Registering signals might not be possible for all paths, but doing so simplifies timing analysis and makes it easier for the Xilinx tools to place-and-route the design.

Recognize Timing Critical Signals

The XDC file provided with the example design for the core identifies the critical signals and the timing constraints that should be applied.

Make Only Allowed Modifications

The SMPTE UHD-SDI RX Subsystem is not user modifiable. Any modifications might have adverse effects on the system timings and protocol compliance. Supported user configurations of the SMPTE UHD-SDI RX Subsystem can only be made by selecting options from the Vivado® Integrated Design Environment (IDE).

Clock Frequency Selection

The SMPTE UHD-SDI RX Subsystem has multiple clock domains and has many CDC paths across the core. It is recommended to use the maximum allowed clock frequency to reduce the uncertainty due to clock domain crossing paths.

Clocking

The subsystem clocks are described in Table 3-1. Clock frequencies should be selected to match the throughput requirement and SDI standard.

Table 3-1: **Subsystem Clocks**

Clock Name	Description
s_axi_aclk	AXI4-Lite clock used by the register interface of all IP cores in the subsystem. Frequency range could be 50 MHz to 150 MHz
sdi_rx_clk	Core clock for SMPTE UHD-SDI RX core. See Table 3-2 for more details.
video_out_clk	Clock used for Video data conversion from SDI data stream. In order to support 12G-SDI for 10-bit YCbCr 4:2:2 in 2 PPC(1), clock must set to maximum 300 MHz. 2*(BPC)(2)*(PPC)*clock = 2*10*2*300 MHz = 12 Gb/s video_out_clk for SMPTE UHD-SDI RX Subsystem must not be less than sdi_rx_clk (will cause underflow). User designs must make sure no underflow occurs when using a clock speed faster than the sdi_rx_clk value.

Notes:

1. BPC can be set to either 10 or 12 because the subsystem supports 10-bit or 12-bit
2. PPC is Pixel Per Clock that is set to 2 by SDI bridge.

The frequency of **`sdi_rx_clk`** of SMPTE UHD-SDI RX core is given Table 3-2.

Table 3-2: **SMPTE UHD-SDI RX Clock**

SMPTE Standard	Supported Data Stream	Clock Frequency (MHz)
SD-SDI	1	148.5 (27 MHz sampling at rx_sd_ce with 5-6-5-6 cadence)
HD-SDI	2	74.25
3G-SDI Level A	2	148.5
3G-SDI Level B	4	148.5
6G-SDI	8	148.5
12G-SDI	8	297

More clocking details can be referenced in the *Clocking* section the *SMPTE UHD-SDI Product Guide* [Ref 9].

Resets

The subsystem has three reset ports:

- **s_axi_arstn**: Active-Low reset for the AXI4-Lite register interface and synchronous with **s_axi_aclk**.
- **video_out_arstn**: Active-Low reset for the subsystem blocks and synchronous with **video_out_clk**.
- **sdi_rx_rst**: Active-High reset for the SMPTE UHD-SDI RX core and synchronous with **sdi_rx_clk**.

Table 3-3 summarizes all resets available to the SMPTE UHD-SDI RX Subsystem and the components affected by them.

Table 3-3: **Core Resets**

Sub-Core	s_axi_arstn	video_out_arstn	sdi_rx_rst
SMPTE UHD-SDI RX	Connected to s_axi_aresetn core port	Connected to axis_rstn core port	Connected to rx_rst core port
SDI RX to Video Bridge	N/A	N/A	Connected to rst core port
Video In to AXI4-Stream	N/A	N/A	N/A

Note: The effect of each reset (`s_axi_arstn`, `video_out_arstn`, `sdi_rx_rst`) is determined by the ports of the sub-cores to which they are connected. See the individual sub-core product guides for the effect of each reset signal. All the resets should be active until the associated clocks are stable.

UHD-SDI Audio Extract Use Case

The SMPTE UHD-SDI RX Subsystem provides ANC I/F controlled by "Enable ANC Data (Incl. Audio) Extraction I/F" parameter in Vivado XGUI and can be used to extract the SDI audio from the SDI video stream. This section provides overview of extracting SDI audio from native SDI stream(s) using Xilinx UHD-SDI Audio IP core. For more information, see the *UHD-SDI Audio LogiCORE IP Product Guide* (PG309) [Ref 17]. The following figure shows the UHD-SDI Audio Extract use case.

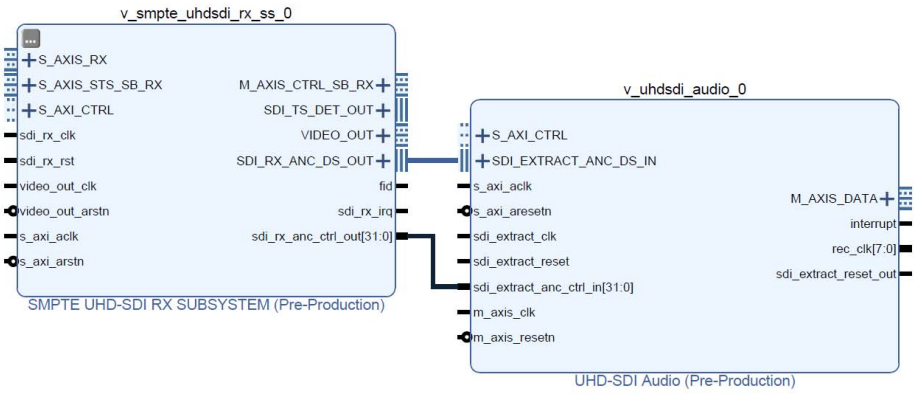

Figure 3-1: **SMPTE UHD-SDI Audio Extract Use Case**

The Xilinx UHD SDI Audio IP core is configurable as an audio embedder or an audio extractor. When configured as an audio extractor, it can extract up to 32 channels (16 channel pairs) of synchronous or asynchronous audio from the SDI stream.

It is designed in accordance with SMPTE ST 272 for SD-SDI and SMPTE ST 299 for HD SDI/ 3G-SDI/6G-SDI/12G-SDI to extract audio. It supports audio extraction at multiple audio sample rates (32 kHz, 44.1 kHz, and 48 kHz).

Functional Description

The SD interface to the audio extractor consists of data streams, mode and status signals from the SMPTE UHD-SDI RX Subsystem. The audio interface is a 32-bit AXI4-Stream master bus. Figure 3-2 shows the top level block diagram of the SDI audio extractor in 32 channel configuration.

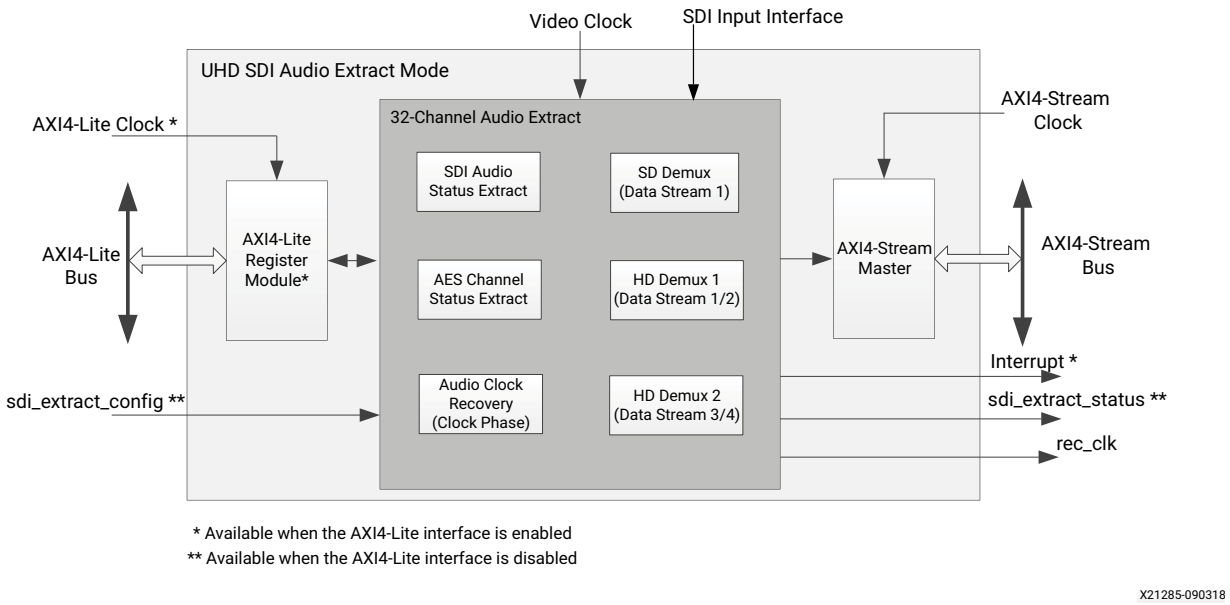

Figure 3-2: **SDI Audio Extractor Block Diagram**

In SD-SDI mode, as per SMPTE ST 272, up to 16 channels of audio are extracted from data stream 1 (Y Video In). In HD-SDI mode, as per SMPTE ST 299-1, up to 16 channels of audio are extracted from data stream 1 and 2. In 3G-SDI, 6G-SDI and 12G-SDI modes, as per SMPTE ST 299-1 and SMPTE ST 299-2, up to 32 channels of audio are extracted from data streams 1, 2, 3, and 4. Audio control packets are extracted from data stream 1 and 3 (Y Video In) and data packets are extracted from data stream 2 and 4 (C_BC_R Video In).

The Xilinx AXI4-Stream audio interface carries audio samples in Audio Engineering Society (AES3) format. The data width over the AXI4-Stream audio interface is fixed at 32 bits to carry one AES3 sub-frame as shown in the following figure.

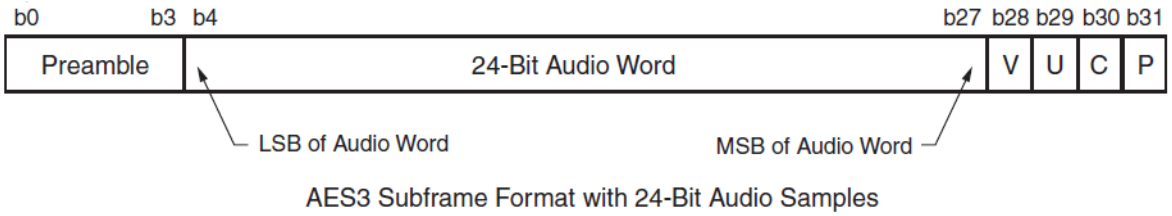

Figure 3-3: **AES3 Subframe Format with 24-Bit Audio Sample**

The TID indicates the channel number of the audio sample data.

Figure 3-4 shows the interface timing diagram for the audio extractor core.

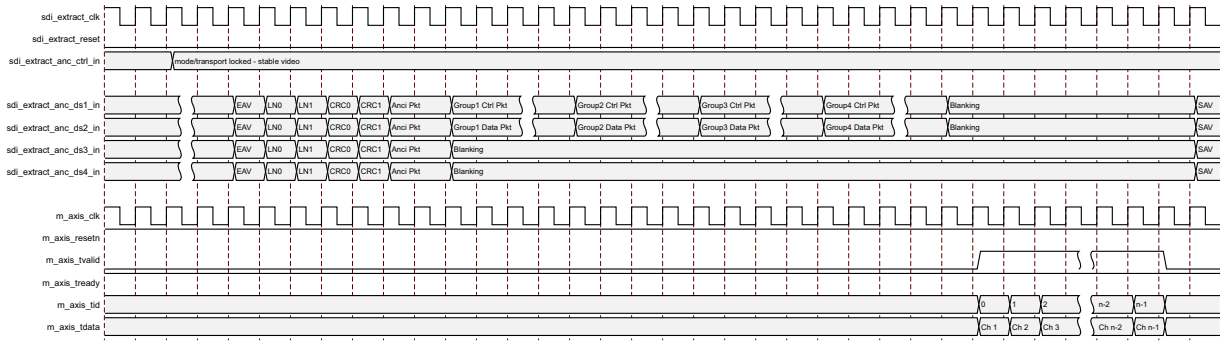

Figure 3-4: **Audio Extract Timing Diagram**

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 1]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4]

Customizing and Generating the Subsystem

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

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core 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) [Ref 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

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

Core Configuration Tab

Figure 4-1 shows the Core Configuration tab for customizing the SMPTE UHD-SDI RX Subsystem.

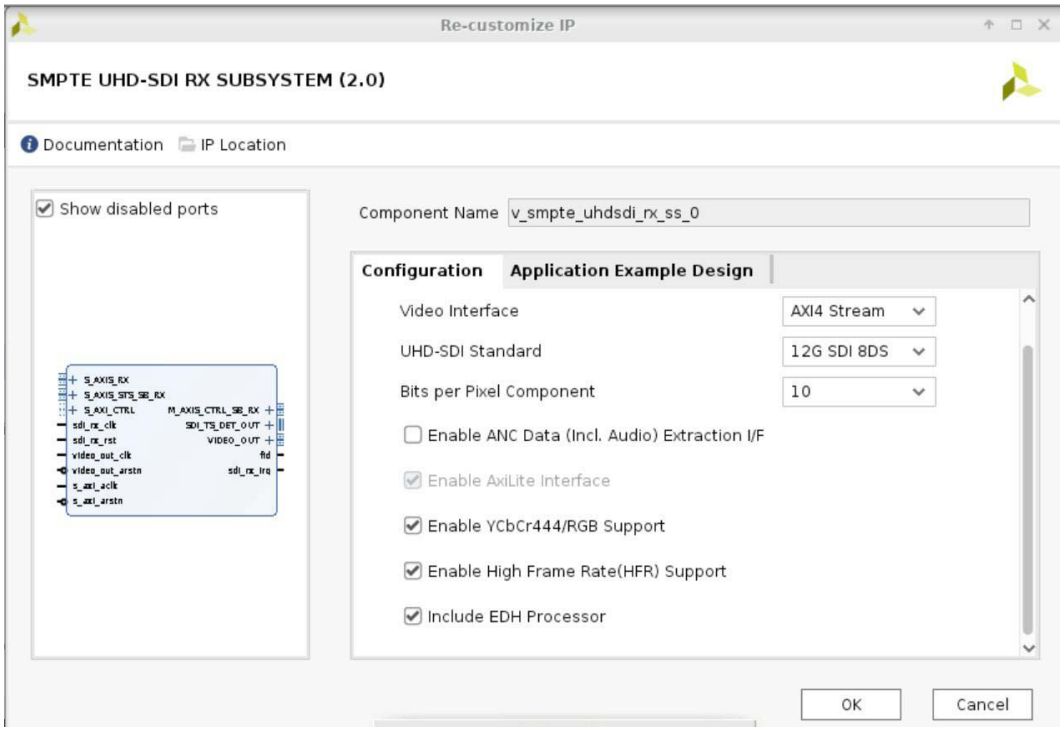

Figure 4-1: **Subsystem Configuration Tab**

Component Name: The Component Name is the base name of the output files generated for this core.

IMPORTANT: *The name must begin with a letter and be composed of the following characters: a to z, A to Z, 0 to 9 and "_."*

Core Parameters

- **Video Interface**: Specifies user interface. The available options are:
  - AXI4-Stream (Default)
  - Native Video
  - Native SDI
- **SDI Standard**: Specifies the SDI standard. The available options are:
  - 3G SDI
  - 6G SDI
  - 12G SDI 8DS (Default): 8 data streams
  - 12G SDI 16DS (available only for Native SDI): 16 data streams

- **Bits per Pixel Component:**
  - 10
  - 12
- **Enable YCbCr444/RGB Support:** The IP Subsystem supports YCbCr Video Format only when this is enabled.
  - **Enable High FrameRate (HFR) Support:** Includes HFR logic in the Core

Application Example Design Tab

Figure 4-2 shows the Application Example Design tab for using the SMPTE UHD-SDI RX example design. See Chapter 5, Example Design for more information on the example design.

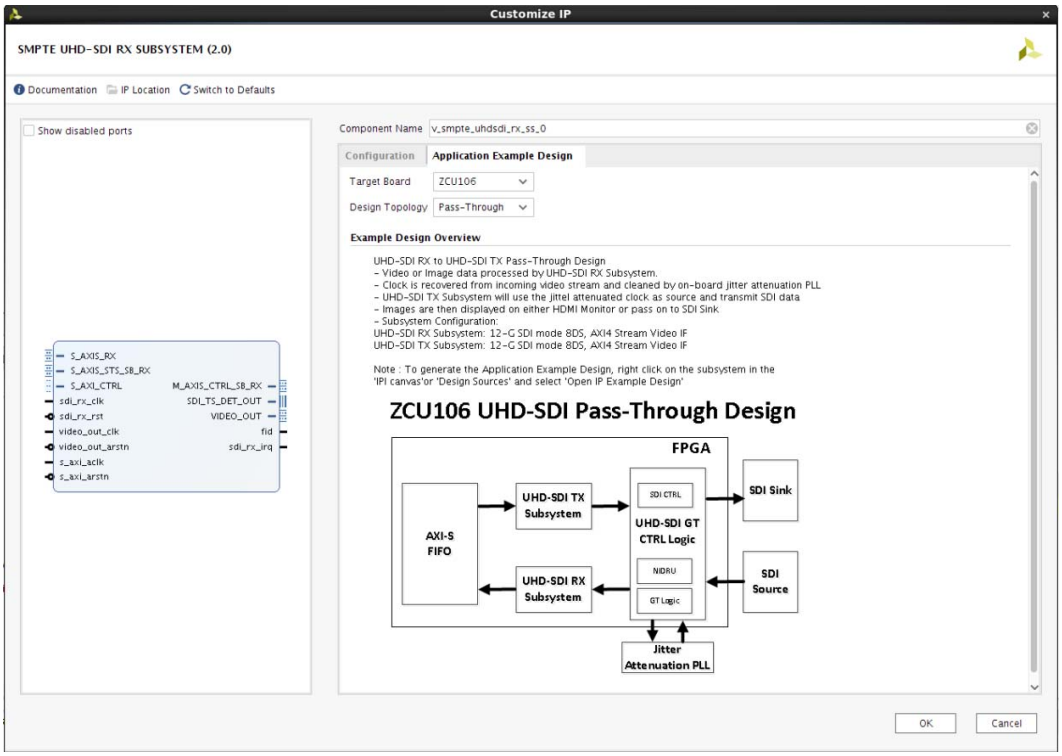

Figure 4-2: **ZCU106 Pass-Through Application Example Design**

Target Board: Target board on which the Application example design to be built. The available options are:

- ZCU106 (Default)
- KCU116

- VCK190

Design Topology: Type of configuration for application example design. The available options are:

- Pass-Through (Default)
- Audio-Video Pass-Through
- Audio-Video Loopback
- RX-Only
- Versal Audio-Video Pass-Through

Board Part Name: This option is only available for VCK190 Example design which gives users the flexibility to choose between the board parts available for VCK190. The available options are:

- xilinx.com:vck190:part0:2.0

User Parameters

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

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

Vivado IDE Parameter to User Parameter Relationship		
Vivado IDE Parameter/Value(1)	**User Parameter/Value**	**Default Value**
Core Parameters		
Video Standard	C_VIDEO_INTF	AXI4-Stream
SDI Standard	C_LINE_RATE	12G SDI 8DS
Enable ANC Data (Incl. Audio) Insertion Interface	C_INCLUDE_ADV_FEATURES	FALSE
Enable AxiLite Interface	C_INCLUDE_AXILITE	TRUE
Include EDH Processor	C_INCLUDE_EDH	FALSE
Target Board	C_EXDES_BOARD	ZCU106
Design Topology	C_EXEDES_CONFIG	Pass-through
Bit per pixel component	C_BPP	10

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Constraining the Subsystem

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

Required Constraints

This section defines the additional constraint requirements for the subsystem. Constraints are provided with a Xilinx Design Constraints (XDC) file. An XDC is provided with the HDL example design to give a starting point for constraints for your design.

Device, Package, and Speed Grade Selections

This section is not applicable for this subsystem.

Clock Frequencies

See Clocking in Chapter 3.

Clock Management

The SMPTE UHD-SDI RX Subsystem generates the required clock constraints when generated using out-of-context mode with <component_name>_ooc.xdc. You can use these or update as required for other clock constraints.

Clock Placement

This section is not applicable for this subsystem.

Banking

This section is not applicable for this subsystem.

Transceiver Placement

This section is not applicable for this subsystem.

I/O Standard and Placement

This section is not applicable for this subsystem.

Simulation

This section contains information about simulating IP in the Vivado Design Suite. For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

Note: The SMPTE UHD-SDI RX Subsystem does not support simulation.

Synthesis and Implementation

This section contains information about synthesis and implementation in the Vivado Design Suite. For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Controlling the Core Using the Xilinx Provided Drivers

- Xilinx provides a BareMetal driver intended to control the core. Refer to Chapter 5, Example Design for examples of use of this driver. This driver is delivered with the Vivado installation and is also available from the Xilinx Git repository.
- Xilinx also provides a V4L2 driver for use in PetaLinux environments. For driver documentation, see the Xilinx Wiki (https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841996/Linux) and search for **V4L2 SDI Rx driver**.

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

Table 5-1: **Example Design Hardware Requirements**

Topology	Required Hardware	Processor	UHD-SDI GT Configuration		GT Type	GT Data Width	BPC
			TXPLL	RXPLL			
(Video) Pass-Through	• ZCU106 • 2 HD-BNC to BNC cables • SDI source and sink devices	A53	QPLL1	QPLL0	GTHE4	40-bit	10
Audio-Video Pass-Through	SDI Audio Pass-Through ◦ ZCU106 ◦ 2 HD-BNC to BNC cables ◦ SDI source and sink devices SDI Audio Playback to AES ◦ ZCU106 ◦ 3 HD-BNC to BNC cables ◦ SDI source and sink devices ◦ AES Sink AES capture to SDI TX ◦ ZCU106 ◦ 3 HD-BNC to BNC cables ◦ SDI source and sink devices ◦ AES source	A53	QPLL1	QPLL0	GTHE4	40-bit	10
Audio-Video Loopback	• KCU116 • Fidus TB-FMCH-12GSDI SDI FMC • 1 HD-BNC to HD-BNC cable	MicroBlaze™	CPLL	QPLL0(1)/ QPLL1(2)	GTYE4	40-bit	10
RX-Only	• ZCU106 • 1 HD-BNC to BNC cable • Sink device	A53	QPLL1	QPLL0	GTHE4	40-bit	10
Versal Audio-Video Pass-Through	SDI Versal Video Pass-Through ◦ VCK190 ◦ Fidus TB-FMCH-12GSDI SDI FMC ◦ 2 HD-BNC to BNC cables ◦ SDI source and sink devices	A72	RPLL	LCPLL	GTYES	40-bit	10

ZCU106 SMPTE UHD-SDI RX-Only Example Design

The SMPTE UHD-SDI RX-only example design is built using the SMPTE UHD-SDI RX Subsystem. Video or image data is generated by test equipment and received by the SMPTE UHD-SDI RX Subsystem. The example design application software runs on the Zynq® UltraScale+™ MPSoC Arm processor system (PS) and is fully software controlled.

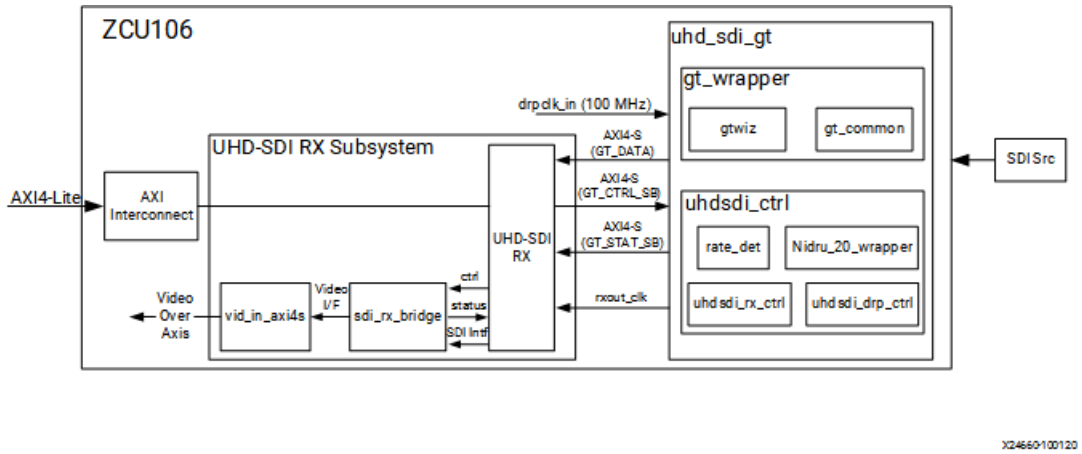

Figure 5-1: **ZCU106 SMPTE UHD-SDI RX-Only Design**

Note: The Zynq UltraScale+ MPSoC Arm PS is not shown in Figure 5-1.

Clocking

QPLL0 is allocated for the SMPTE UHD-SDI RX transceiver in this RX-only example design. Figure 5-2 shows the clocking used in the SMPTE UHD-SDI RX-Only example design. QPLL0 is given a single reference clock frequency, which can be either 148.5 MHz or 148.5/1.001 MHz, depending on which SDI line rate is to be supported (148.5 MHz for integer line rates or 148.5/1.001 MHz for fractional line rates) and comes from the on-board si570 chip.

In the example shown in the following figure, the design only requires an Integer Line rate, so the reference clock frequency is 148.5 MHz.

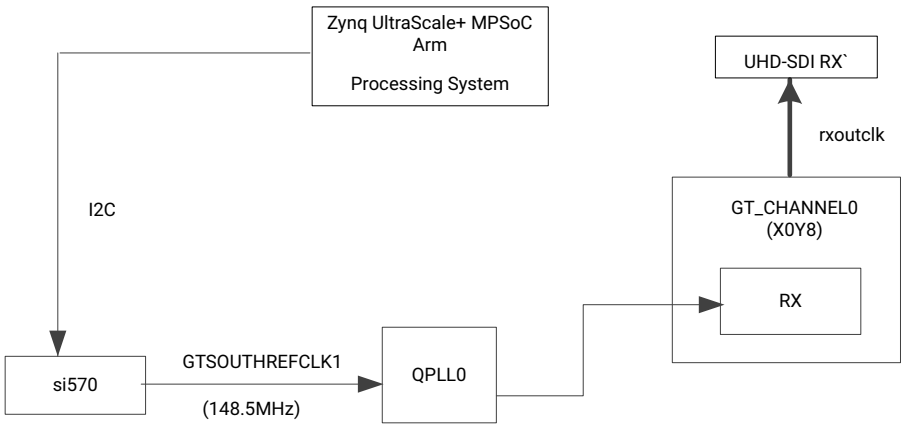

Figure 5-2: **ZCU106 SMPTE UHD-SDI RX-Only Example Design Clocking**

Table 5-2 shows the clock frequency at various parts of the system for different SDI modes.

Table 5-2: **ZCU106 RX-Only Example Design Clock Frequency Ranges**

SDI Mode	Rx_m	QPLL0 Refclk (MHz)	rxoutclk (MHz)
SD-SDI	N/A	148.5	148.5
HD-SDI	0	148.5	74.25
HD-SDI	1	148.35	74.25/1.001
3G-SDI/6G-SDI	0	148.5	148.5
3G-SDI/6G-SDI	1	148.35	148.5/1.001
12G-SDI	0	148.5	297
12G-SDI	1	148.35	297/1.001

Notes:

1. rx_m or tx_m represents integer line rate when it is set to 0 and it represents fractional line rate when it is set to 1.

Transceiver Configuration in Example Design

The UHD-SDI example design uses the UHD-SDI GT core to configure Xilinx® UltraScale+ GTH transceivers and provides options to the users to select the transceiver reference clocking. This core also generates control modules that are required to program the transceiver using the DRP interface.

Running the ZCU106 RX-Only Example Design

1. Open the Vivado Design Suite and create a new project.
2. In the pop-up window, press **Next** five times.
3. Select the ZCU106 board, as shown in Figure 5-3.

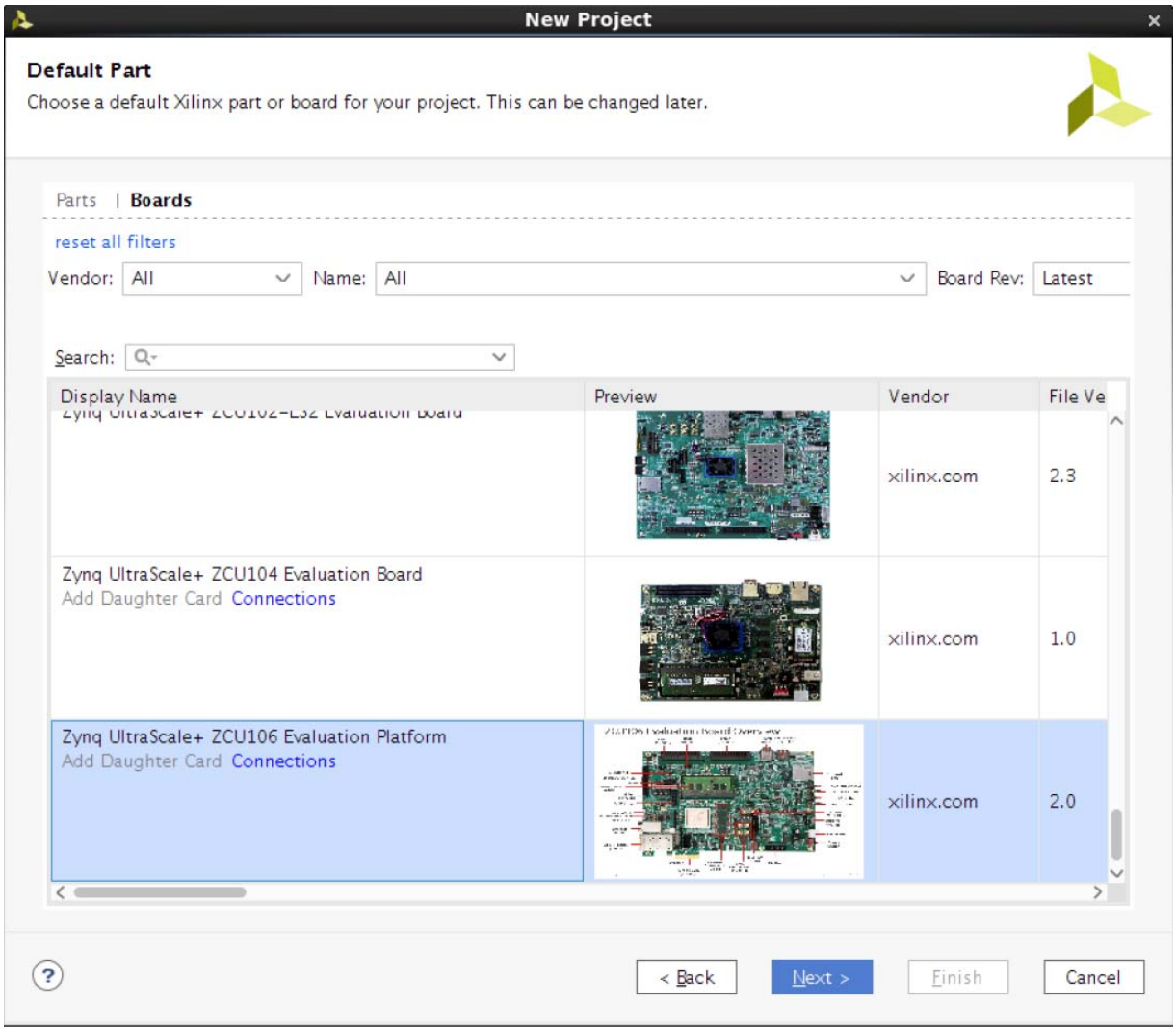

Figure 5-3: **Select the ZCU106 Board**

4. Click **Finish**.

5. Click **IP Catalog** and select **SMPTE UHD-SDI RX Subsystem** under Video Connectivity, and then double click **SMPTE UHD-SDI RX Subsystem**, as shown in Figure 5-4.

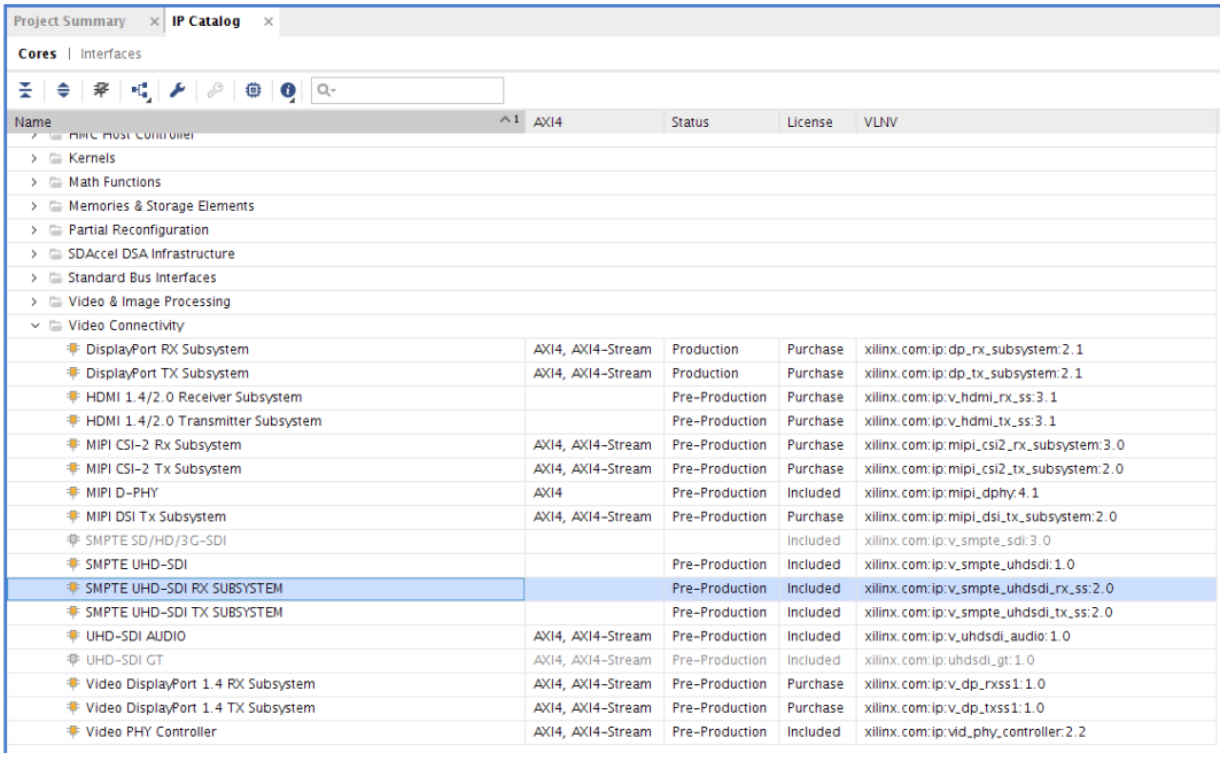

Figure 5-4: **Select the UHD-SDI RX Subsystem**

Note: For the application example design flow, the IP configuration is based on options selected in Application Example Design tab. You can rename the IP component name, which is used as the application example design project name.

6. Configure the SMPTE UHD-SDI RX Subsystem **Application Example Design** tab by selecting **ZCU106** as the Target Board followed by **RX-Only** as Design Topology and click **OK**.

7. Click **Generate**.

   Note: You can optionally click **Skip** if you only want to generate the application example design.

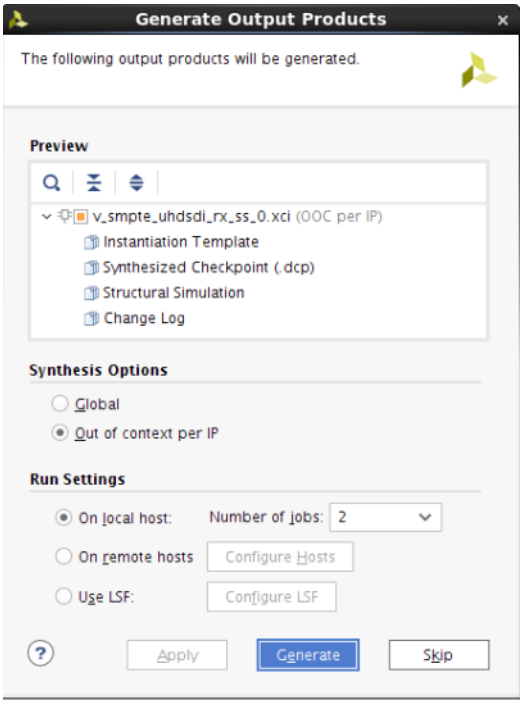

Figure 5-5: **Generate Output Products**

8. Right-click the **SMPTE UHD-SDI RX Subsystem** component under Design source, and click **Open IP Example Design**.

The IP integrator design is then generated and creates the Vitis™ software platform. You can choose to Run Synthesis, Implementation, or Generate Bitstream. Figure 5-6 shows an overall system IP integrator block diagram of the ZCU106-based application example design.

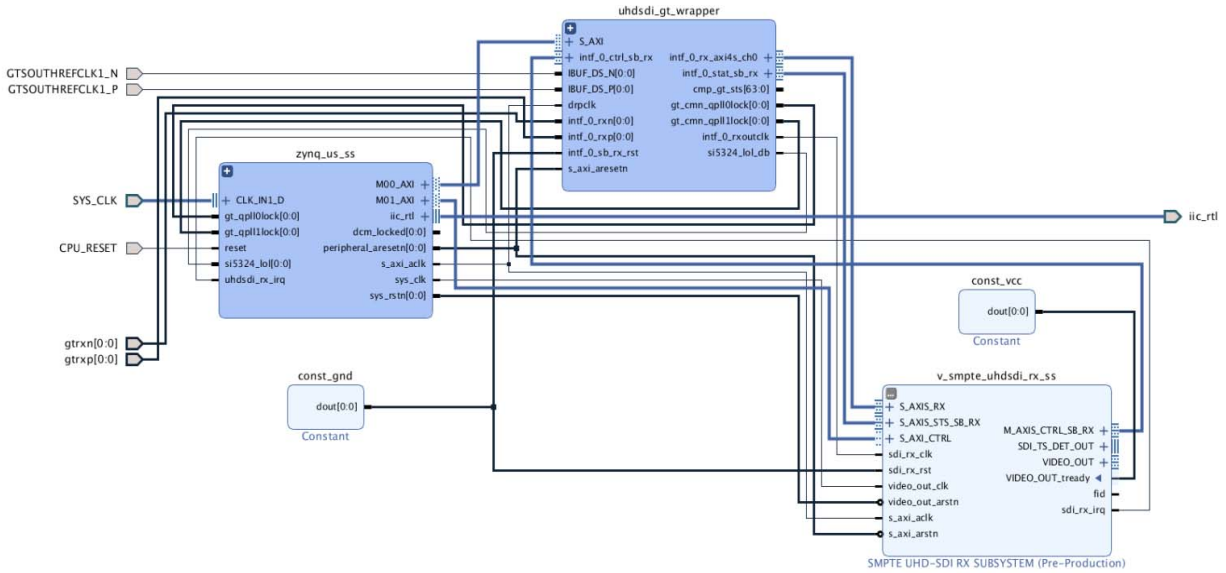

Figure 5-6: **IP Integrator Diagram of the ZCU106 RX-Only Application Example Design**

Requirements

Hardware

The hardware requirements for this reference system are:

- One Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit
- One SDI source equipment
- One HD-BNC to BNC converter cable

Software

This section includes any software requirements:

- Vivado Design Suite 2019.2 or later
- The Vitis software platform 2019.2 or later
- Software terminals (for example, Tera Term, HyperTerminal or PuTTY)
- SCUI may be required for some versions of the FMC card to set the power rail to 1.8V.

Setup

This reference design runs on the Zynq UltraScale+ MPSoC board (ZCU106) using SDI connectors available on the board. Additional information about the ZCU106 board can be found on the product page.

Note: The numbers in the parenthesis in the instructions correspond to the callouts in Figure 5-7.

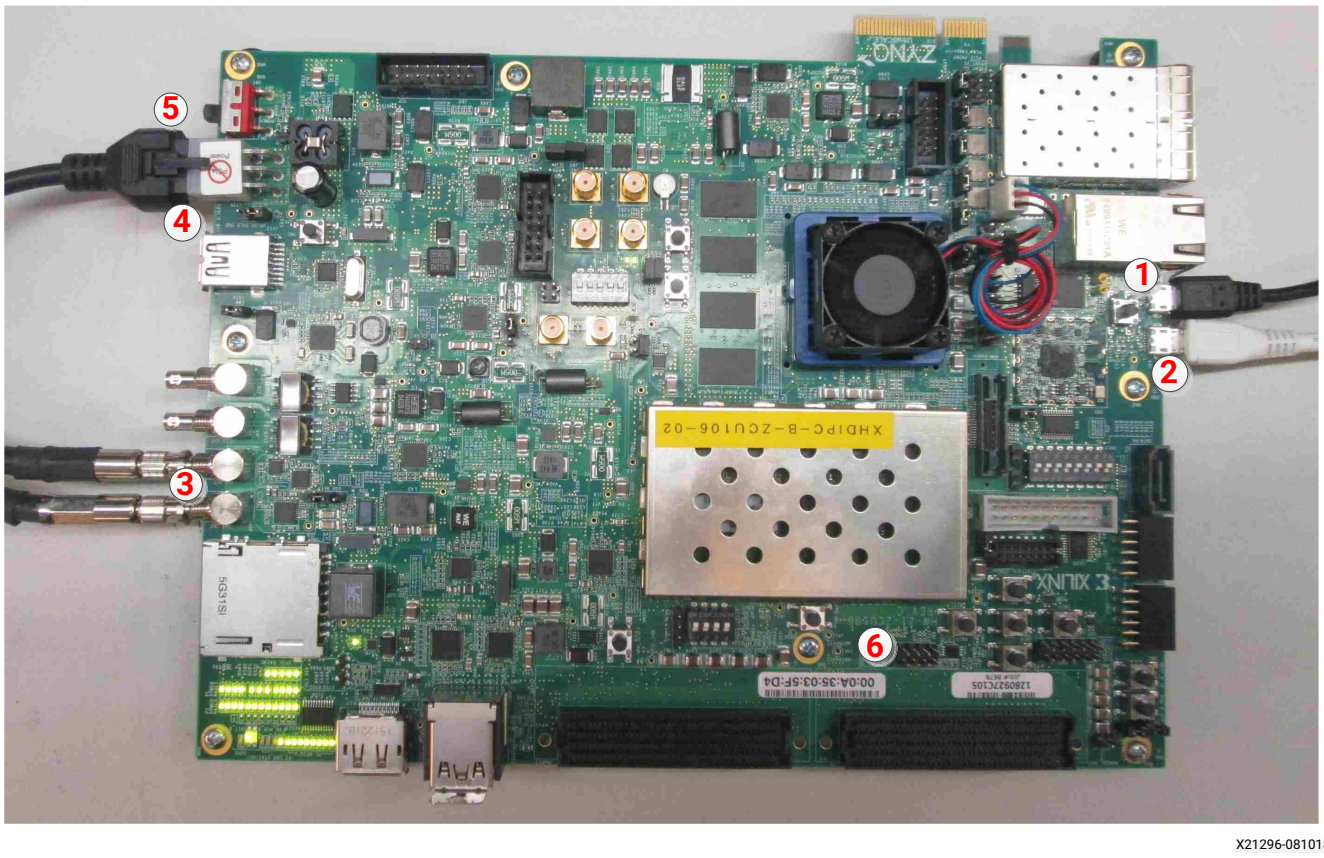

Figure 5-7: **ZCU106 Board Setup**

Follow these instructions to set up the board:

1. Connect a USB cable from the host PC to the USB JTAG port (1). Ensure the appropriate device drivers are installed.
2. Connect a second USB cable from the host PC to the USB UART port (2). Ensure that the USB UART drivers described in Hardware have been installed.
3. Connect the SDI_IN link of ZCU106 (3) to the SDI source device.
4. Connect the ZCU106 board to a power supply slot (4).
5. Switch on the ZCU106 board (5).
6. Make sure that the HW-ZCU106 board revision (6) is Rev C.

7. Start Tera Term or PuTTY to connect to the COM port interface 0 on the Host PC with 115200 bps, 8 bits, No parity, 1 stop bit, and no flow control as configuration.
8. When using FMC card, set the FMC_VADJ voltage to 1.8V.

Compiling Software in the Vitis Software Platform

The UHD-SDI application example design generates a **`.elf`** file automatically. Follow the steps if you want to open the Vitis software platform project for the UHD-SDI example design from the Vivado Design Suite:

1. In the Vivado Design Suite, click **Tools** > **Launch Vitis**.
2. Select **Exported location** and **workspace** as **`<Proj Dir>/SW/xsdi_app`** and click **OK** to launch and open the Vitis software platform project.

The Vitis software platform project is now open.

Running the design on the hardware

The following steps are used to run the BIT and ELF files on the hardware setup:

1. Connect the JTAG cable and USB-UART cable to the board.
2. Go to **`<Component Name>_ex/imports`**.
3. Start the Xilinx Software Debugger (**`XSDB`**) by sourcing xsdb from the build area to the command prompt.
4. Run the following command, **`source xsdb.tcl`**, to program FPGA and to execute the application.
5. To observe the results, open Tera Term or PUTTY, and configure the serial port (Interface 0) to 115200 baud with the default configuration.

Note: Make sure that the UART cable is connected to the board and the PC.

The UART console now displays the SDI stream details on the console.

UART Console Screens

The following screen capture shows initial UART console output along with menu options.

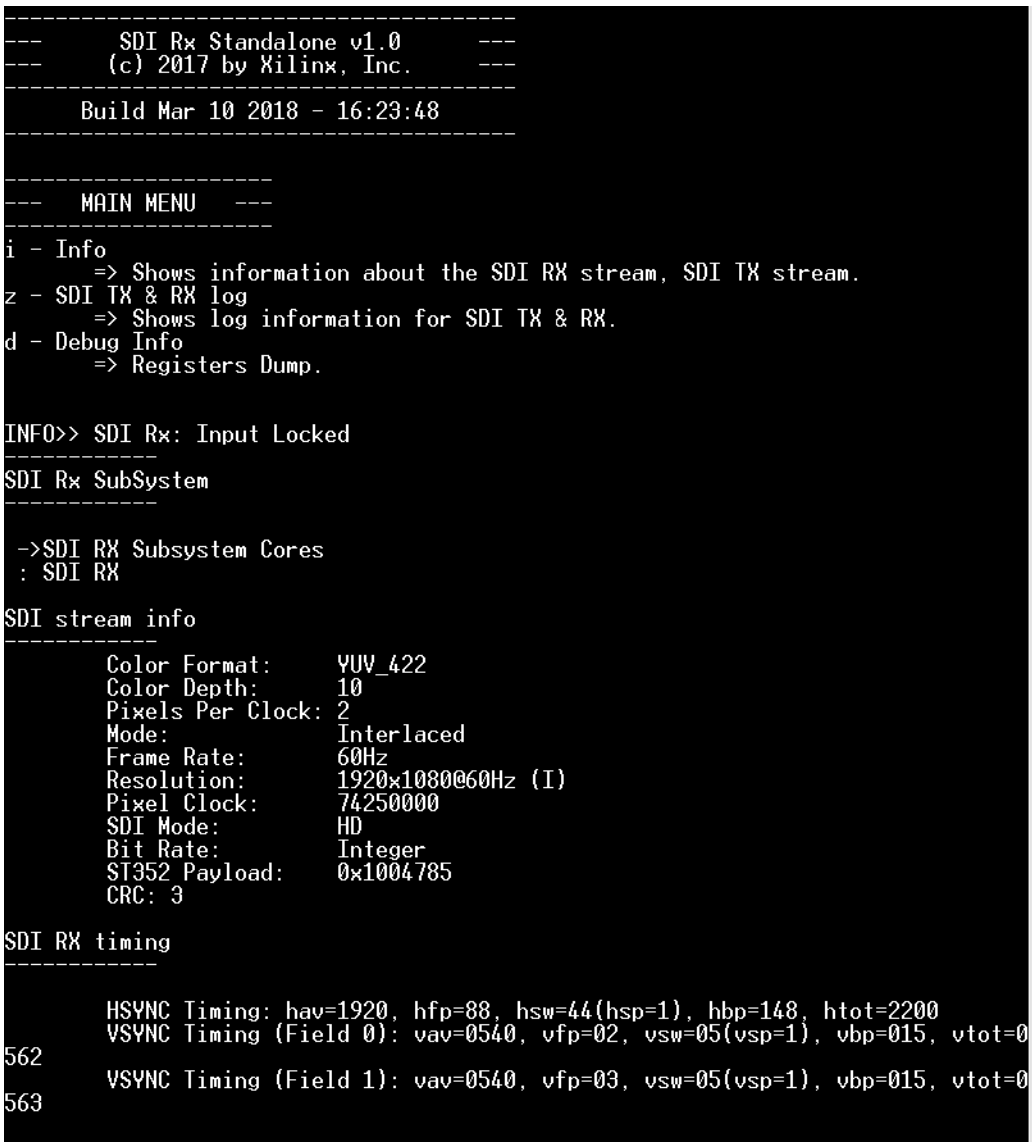

Figure 5-8: **UART Console - Received UHD SDI Stream Information**

The following screen capture shows UART console output when resolution is changed at SDI source.

```
------------
SDI Rx SubSystem
------------

 ->SDI RX Subsystem Cores
 : SDI RX

SDI stream info
------------
        Color Format:     YUV_422
        Color Depth:      10
        Pixels Per Clock: 2
        Mode:             Interlaced
        Frame Rate:       60Hz
        Resolution:       1920x1080@60Hz (I)
        Pixel Clock:      74250000
        SDI Mode:         HD
        Bit Rate:         Integer
        ST352 Payload:    0x1004785
        CRC: 3

SDI RX timing
------------

        HSYNC Timing: hav=1920, hfp=88, hsw=44(hsp=1), hbp=148, htot=2200
        VSYNC Timing (Field 0): vav=0540, vfp=02, vsw=05(vsp=1), vbp=015, vtot=0
562
        VSYNC Timing (Field 1): vav=0540, vfp=03, vsw=05(vsp=1), vbp=015, vtot=0
563


INFO>> SDI Rx: Lock Lost
INFO>> SDI Rx: Input Locked
------------
SDI Rx SubSystem
------------

 ->SDI RX Subsystem Cores
 : SDI RX

SDI stream info
------------
        Color Format:     YUV_422
        Color Depth:      10
        Pixels Per Clock: 2
        Mode:             Progressive
        Frame Rate:       48Hz
        Resolution:       3840x2160@48Hz
        Pixel Clock:      594000000
        SDI Mode:         12G
        Bit Rate:         Integer
        ST352 Payload:    0x100C8CE
        CRC: 3

SDI RX timing
------------

        HSYNC Timing: hav=3840, hfp=1276, hsw=88(hsp=1), hbp=296, htot=5500
        VSYNC Timing: vav=2160, vfp=08, vsw=10(vsp=1), vbp=072, vtot=2250
```

Figure 5-9: **UART Console - Received UHD SDI Stream Information after RX Mode Change**

ZCU106 SMPTE UHD-SDI Pass-Through Example Design

The UHD-SDI pass-through example design, shown in Figure 5-10, is built using the SMPTE UHD-SDI TX and RX subsystems. Video or image data is received and processed by the SMPTE UHD-SDI RX Subsystem. The clock is recovered by the Xilinx® UltraScale™+ GTH transceiver (RX) and fed to the on-board PLL for jitter attenuation. A jitter-attenuated clock is used as a reference clock by the GTH transceiver for the TX data path. An AXI4-Stream FIFO is used for synchronization and temporary storage between the SMPTE UHD-SDI RX Subsystem and the SMPTE UHD-SDI TX Subsystem. The SMPTE UHD-SDI TX Subsystem transmits SDI data from the AXI4-Stream FIFO after the application programs the SMPTE UHD-SDI TX Subsystem sub-core registers based on the received SDI stream and ST-352 payload packet data. The example design application software runs on the Zynq UltraScale+ MPSoC Arm processor subsystem (PS) and is fully software controlled.

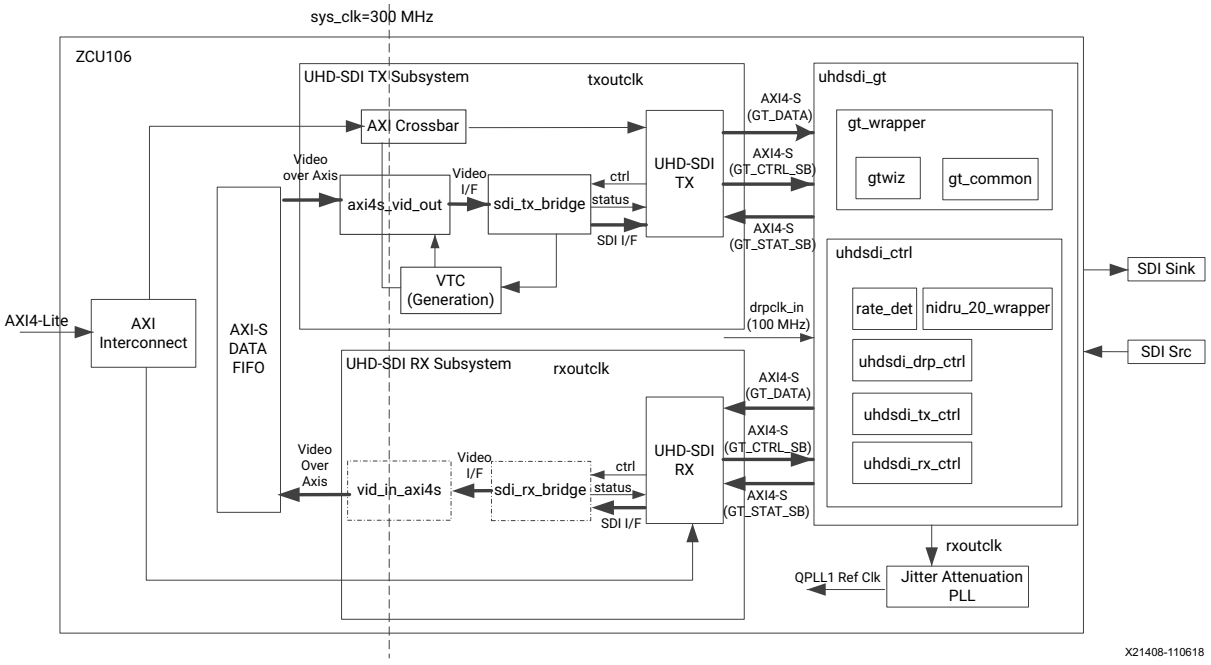

Figure 5-10: **ZCU106 Pass-Through Example Design**

Note: The Zynq UltraScale+ MPSoC PS is not shown in Figure 5-10 for simplicity.

Clocking

QPLL0 is allocated for the SMPTE UHD-SDI RX transceiver and QPLL1 for the SMPTE UHD-SDI TX in this pass-through design. The reference clock for QPLL1 comes from the si5328 chip output. Thus, the QPLL1 reference clock connection is fixed. The QPLL0 reference clock is fixed at 148.5 MHz and comes from the on-board si570 chip. Figure 5-11 shows the clocking used in the SMPTE UHD-SDI example design.

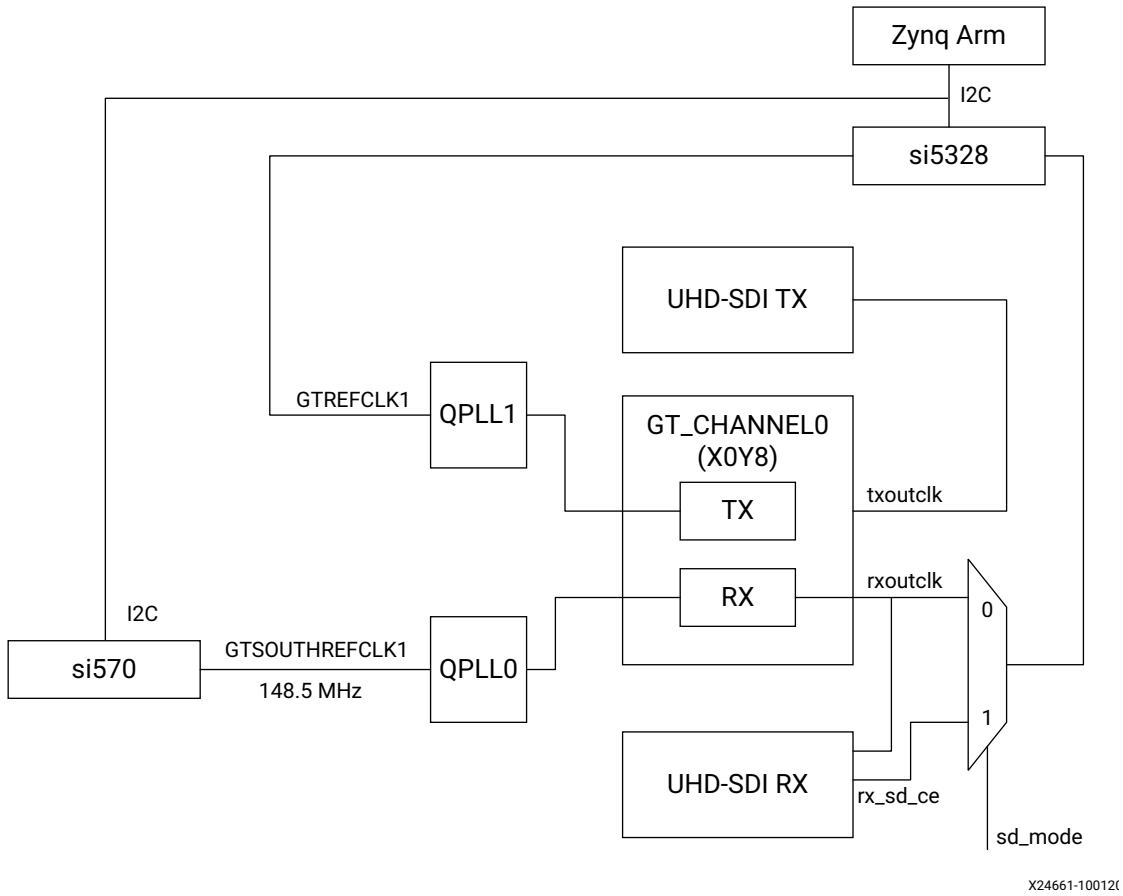

Figure 5-11: **ZCU106 Pass-Through Example Design Clocking**

Table 5-3 shows the clock frequency at different part of the system for different SDI modes:

Table 5-3: **ZCU106 Pass-Through Example Design Clock Frequency Ranges**

SDI Mode	Tx_m/ Rx_m	QPLL0 Ref clk (MHz)	QPLL1 Ref Clk (MHz)	rxoutclk (MHz)	si5328 Input (MHz)	si5328 Output (MHz)	txoutclk (MHz)
SD-SDI	N/A	148.5	148.5	148.5 rx_sd_ce=27	rx_sd_ce=27	148.5	148.5
HD-SDI	0	148.5	148.5	74.25	74.25	74.25	74.25
HD-SDI	1	148.5	148.5/1.001	74.25/1.001	74.25/1.001	74.25/1.001	74.25/1.001
3G-SDI/6G-SDI	0	148.5	148.5	148.5	148.5	148.5	148.5
3G-SDI/6G-SDI	1	148.5	148.5/1.001	148.5/1.001	148.5/1.001	148.5/1.001	148.5/1.001
12G-SDI	0	148.5	148.5	297	297	148.5	297
12G-SDI	1	148.5	148.5/1.001	297/1.001	297/1.001	148.5/1.001	297/1.001

Note: For 6G-SDI and 12G-SDI, 8 native SDI Data Streams (DS) is assumed.

For GT TX and RX data path, the reference clock requirement for data paths are different. For GT TX, for integer and fractional frame rate, PLL reference clock must be different

frequency, clock/1.000 for integer frame rate and clock/1.001 for fractional frame rate. For RX data path PLL reference clock can be same for integer and fractional frame rate.

Transceiver Configuration in Example Design

The UHD-SDI example design uses **uhdsdi_gt_v2_0** core to configure UltraScale+ GTH transceivers and provide options to select the transceiver reference clocking. The core also generates control modules that are required to program the transceiver using DRP interface and NI-DRU modules for RX SD-SDI mode.

Running the ZCU106 Pass-Through Example Design

1. Open the Vivado Design Suite and create a new project.
2. In the pop-up window, press **Next** 3 times as shown in Figure 5-12.

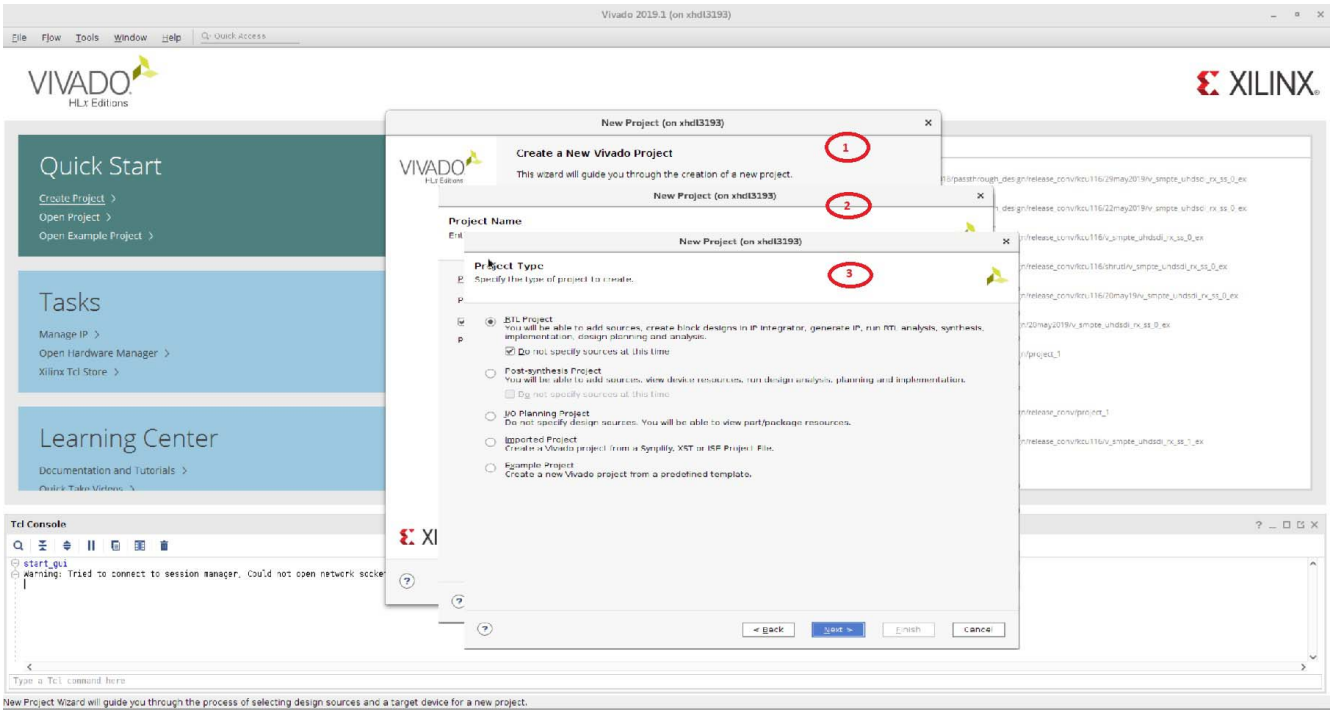

Figure 5-12: **ZCU106 Pass-Through New Project**

3. Select the ZCU106 Board as shown in Figure 5-13.

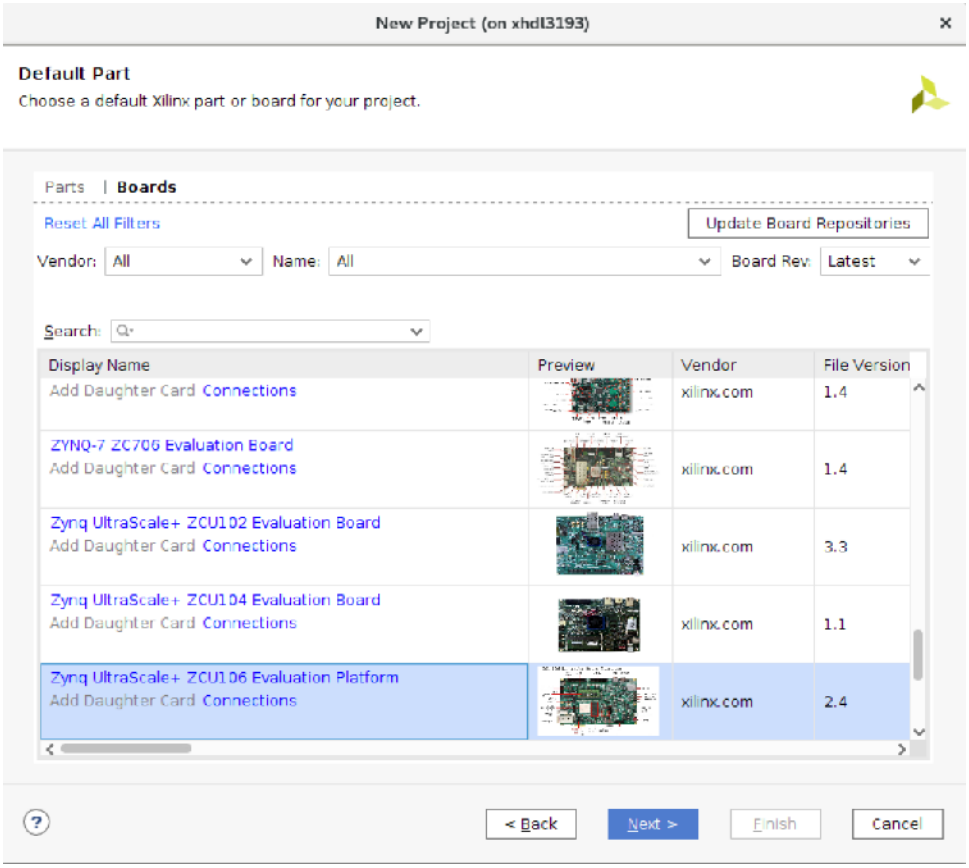

Figure 5-13: **Select the ZCU106 Board**

4. Click **Finish**.

5. Click **IP Catalog** and double-click **SMPTE UHD-SDI RX Subsystem** under Video Connectivity as shown in Figure 5-14.

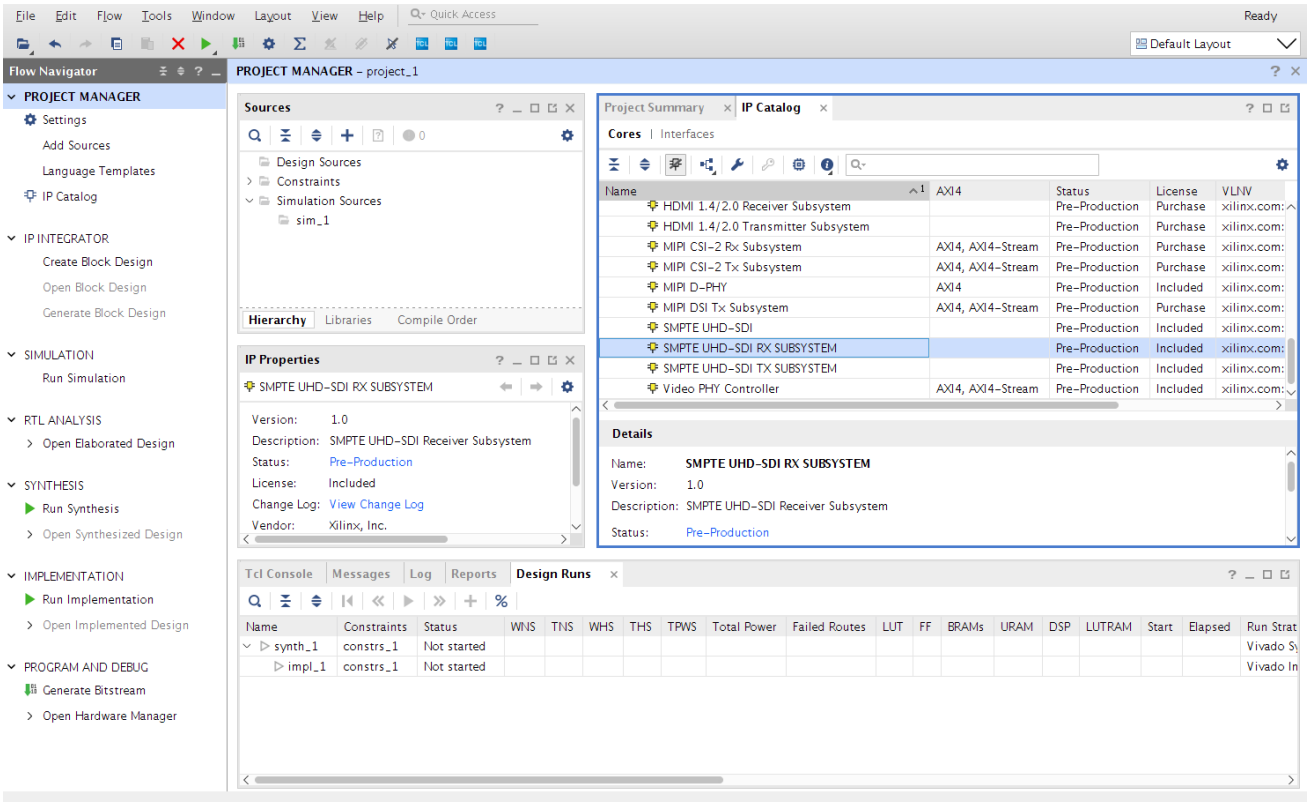

Figure 5-14: **Select the SMPTE UHD-SDI RX Subsystem**

Note: For the application example design flow, the IP configuration is based on options selected in the Application Example Design tab. You can rename the IP component name, which is used as the application example design project name.

6. Configure the SMPTE UHD-SDI RX Subsystem **Application Example Design** tab by selecting ZCU106 as the Target Board followed by **Audio-Video Pass-Through** as Design Topology, and then click **OK**.

   The Generate Output Products dialog box appears.

7. Click **Generate**. (See Figure 5-15.)

   Note: Click **Skip** if you only want to generate the application example design.

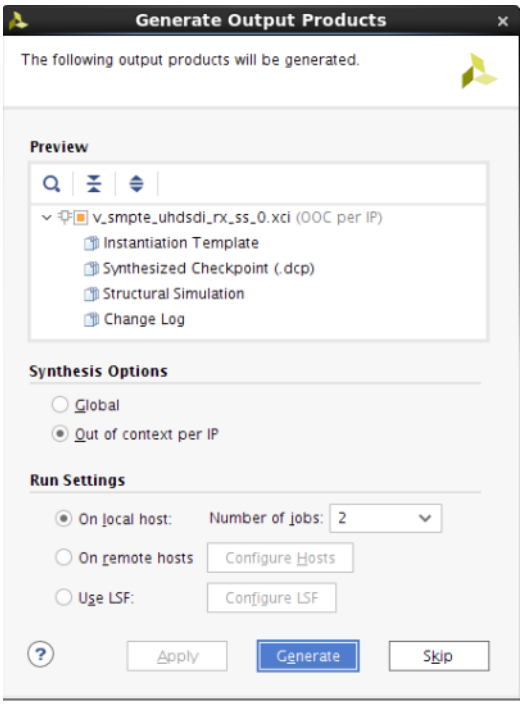

Figure 5-15: **Generate Output Products**

8. Right-click the SMPTE UHD-SDI RX Subsystem component under Design source, and click **Open IP Example Design**, as shown in Figure 5-16.

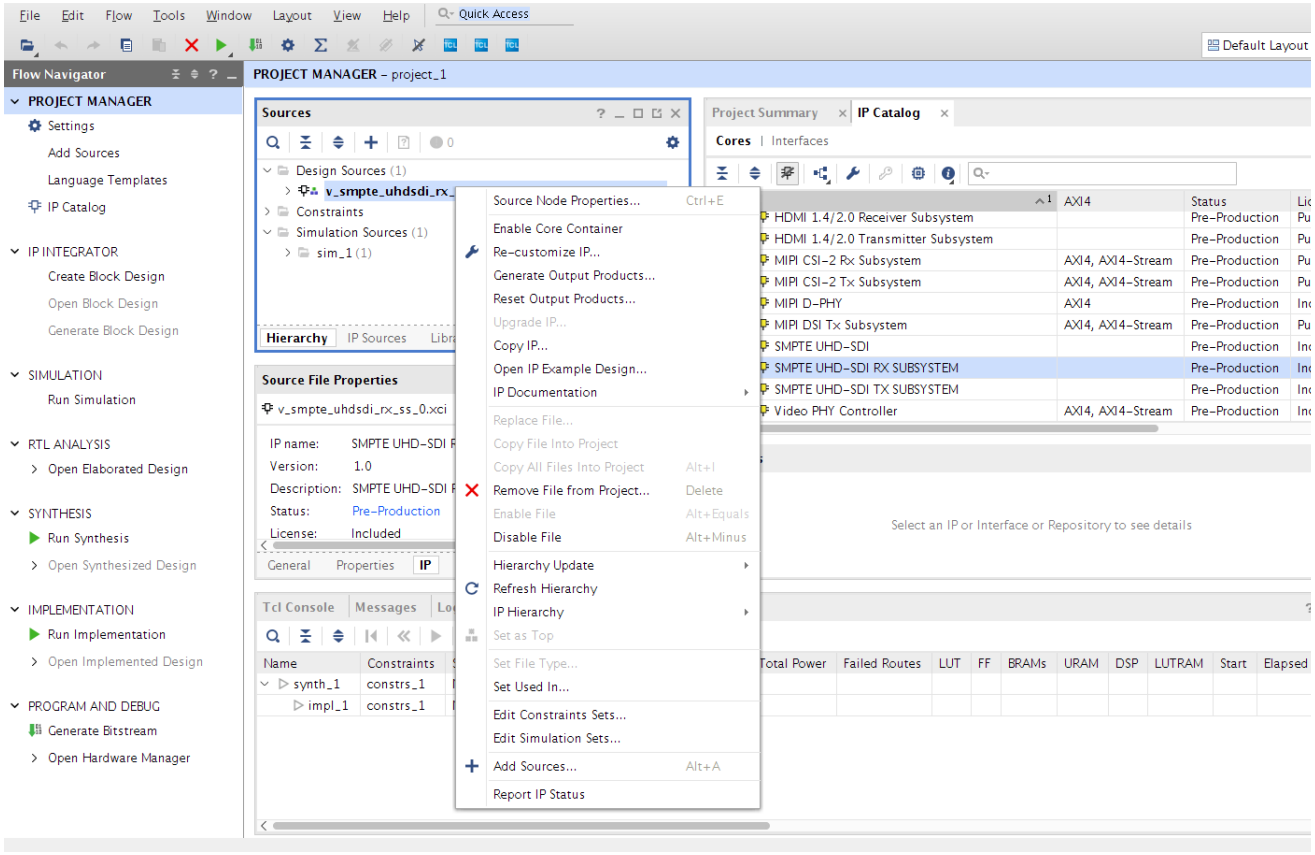

Figure 5-16: **Open IP Example Design**

9. Choose the target project location, then click **OK**.

   The IP integrator design is then generated and creates the Vitis software platform generates a `.elf` file. You can choose to Run Synthesis, Implementation, or Generate Bitstream. An overall system IP integrator block diagram of the ZCU106-based application example design is shown in Figure 5-17.

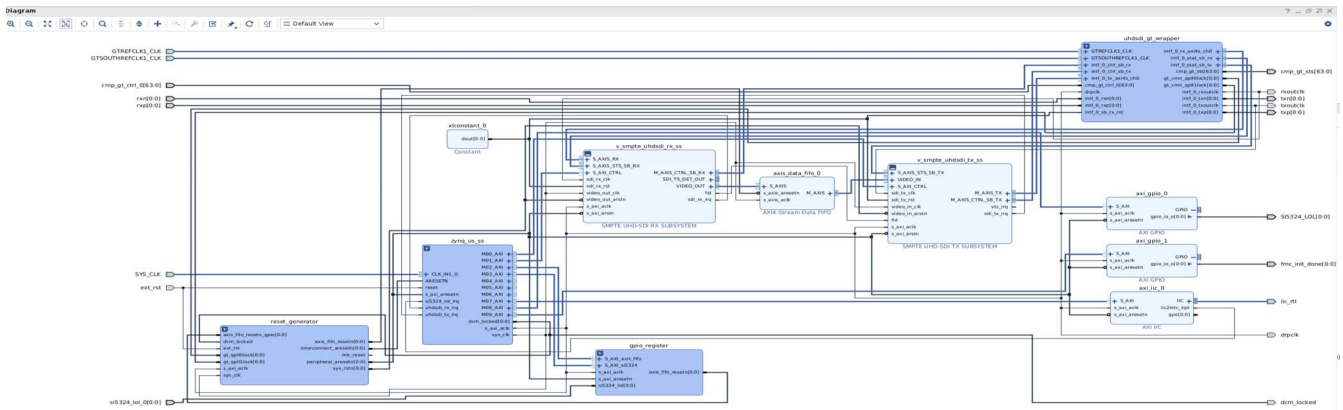

Figure 5-17: **IP Integrator Diagram of the ZCU106 Pass-Through Example Design**

Requirements

Hardware

The hardware requirements for this reference system are:

- Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit
- SDI source equipment
- SDI sink equipment
- Two HD_BNC to BNC converter cables

Software

This section includes any software requirements:

- Vivado Design Suite 2019.2 or later
- The Vitis software platform 2019.2 or later
- Software terminals (for example, Tera Term, HyperTerminal or PuTTY)

Setup

The reference design runs on the Zynq UltraScale+ MPSoC board (ZCU106) using SDI connectors available on the board.

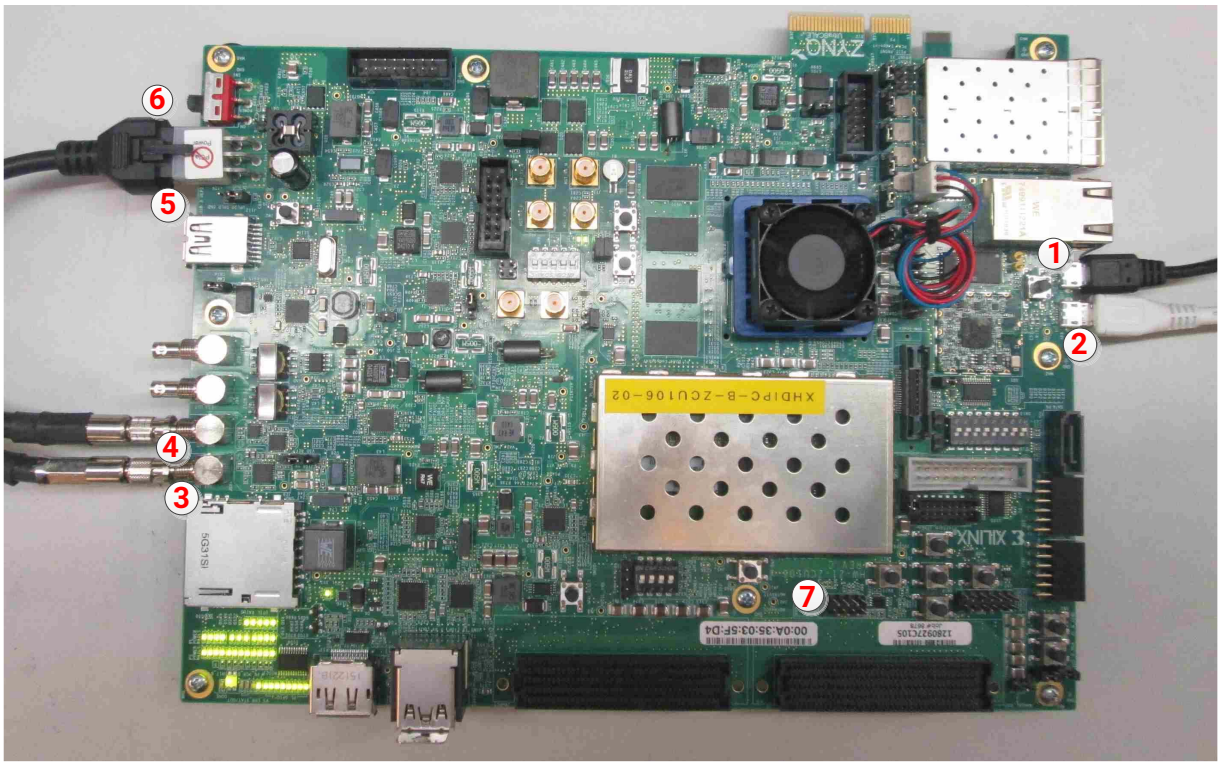

Figure 5-18: **ZCU106 Board Setup**

Note: In the following procedure, the numbers in parentheses correspond to the callout numbers in Figure 5-18.

1. Connect a USB cable from the host PC to the USB JTAG port (1). Ensure the appropriate device drivers are installed.
2. Connect a second USB cable from the host PC to the USB UART port (2). Ensure that the USB UART drivers described in Hardware have been installed.
3. Connect the SDI_INT link of ZCU106 (3) to the SDI source device.
4. Connect the SDI_OUT link of ZCU106 (4) to the SDI sink device.
5. Connect the ZCU106 board to a power supply slot (5).
6. Switch on the ZCU106 board (6).
7. Make sure that the HW-ZCU106 board revision (7) is Rev C.
8. Ensure that the SMPTE 352/Payload ID is enabled in the SDI Stream connected to the SDI input link of ZCU106.
9. Start Tera Term or PuTTY to connect to the COM port interface 0 on the Host PC with 115200 bps, 8 bits, No parity, 1 stop bit, and no flow control as configuration.

Compiling Software in Vitis

The UHD-SDI application example design generates a .elf file automatically. Use the following procedure if you want to open the SDK project for the UHD-SDI example design from the Vivado® Design Suite:

1. In the Vivado Design Suite, click **Tools > Launch Vitis**.
2. Select **Exported location** and **workspace** as **<Proj Dir>/SW/xsdi_app** and click **OK** to launch and open the Vitis™ software platform project.

   Vitis software platform project opens.

Running the Design on the Hardware

The following steps are used to run the BIT and ELF files on the hardware setup:

1. Connect the JTAG cable and USB-UART cable to the board.
2. Navigate to **<Component Name>_ex/imports**.
3. Start the Xilinx Software Debugger (XSDB) by sourcing xsdb from the build area using command prompt.
4. Run the following command to program FPGA and to execute the application.

   ```
   source xsdb.tcl
   ```

5. To observe the results, open Tera Term or PUTTY and configure its serial port (Interface 0) to 115200 baud with the default configuration. Make sure that the UART cable is connected to the board and the PC.

   The UART console displays SDI stream details on console.

UART Console Screens

The following figure shows the initial UART console output along with menu options.

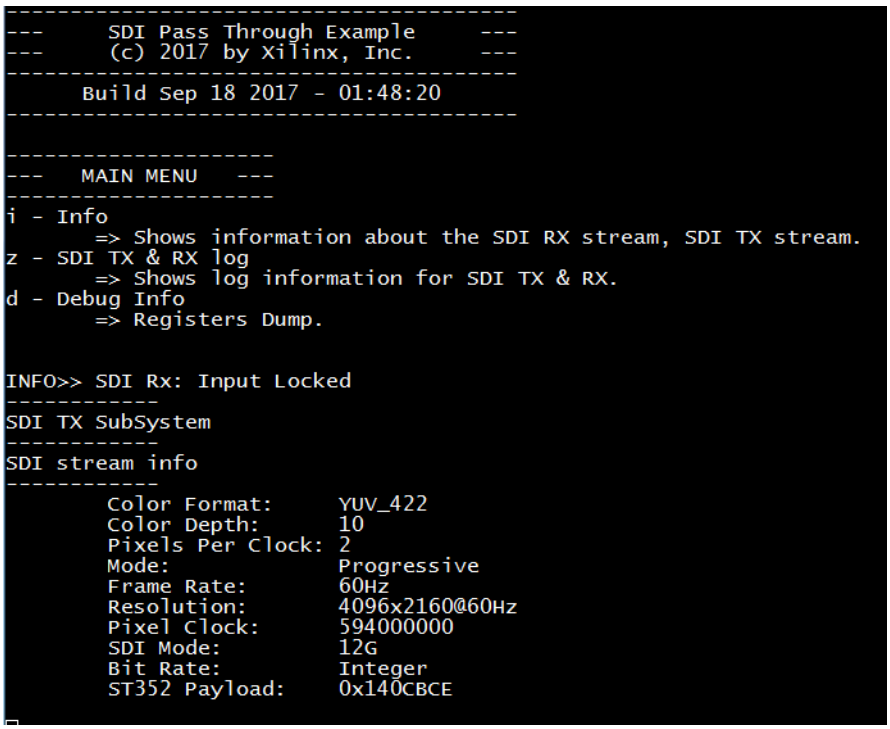

Figure 5-19: **UART Console - Menu with Transmit Stream Information**

The following figure shows the UART console output when **i** is pressed.

```
-----
Info
-----

------------
SDI TX SubSystem
------------

  ->SDI TX Subsystem Cores
    : SDI TX
    : VTC Core

SDI stream info
------------
        Color Format:     YUV_422
        Color Depth:      10
        Pixels Per Clock: 2
        Mode:             Progressive
        Frame Rate:       60Hz
        Resolution:       4096x2160@60Hz
        Pixel Clock:      594000000
        SDI Mode:         12G
        Bit Rate:         Integer
        ST352 Payload:    0x140CBCE
        No Error Detected

SDI TX timing
------------

        HSYNC Timing: hav=4096, hfp=88, hsw=88(hsp=1), hbp=128, htot=4400
        VSYNC Timing: vav=2160, vfp=08, vsw=10(vsp=1), vbp=072, vtot=2250


------------
SDI Rx SubSystem
------------

 ->SDI RX Subsystem Cores
 : SDI RX

SDI stream info
------------
        Color Format:     YUV_422
        Color Depth:      10
        Pixels Per Clock: 2
        Mode:             Progressive
        Frame Rate:       60Hz
        Resolution:       4096x2160@60Hz
        Pixel Clock:      594000000
        SDI Mode:         12G
```

Figure 5-20: **UART Console - Receive and Transmit UHD SDI Stream Information**

The following screen appears in the console output when **z** is pressed.

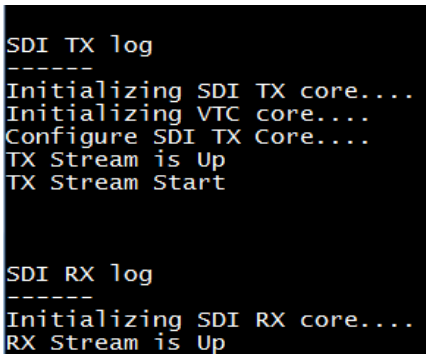

Figure 5-21: **UART Console - Initialization Screen**

The following screen appears in the console output when **d** is pressed:

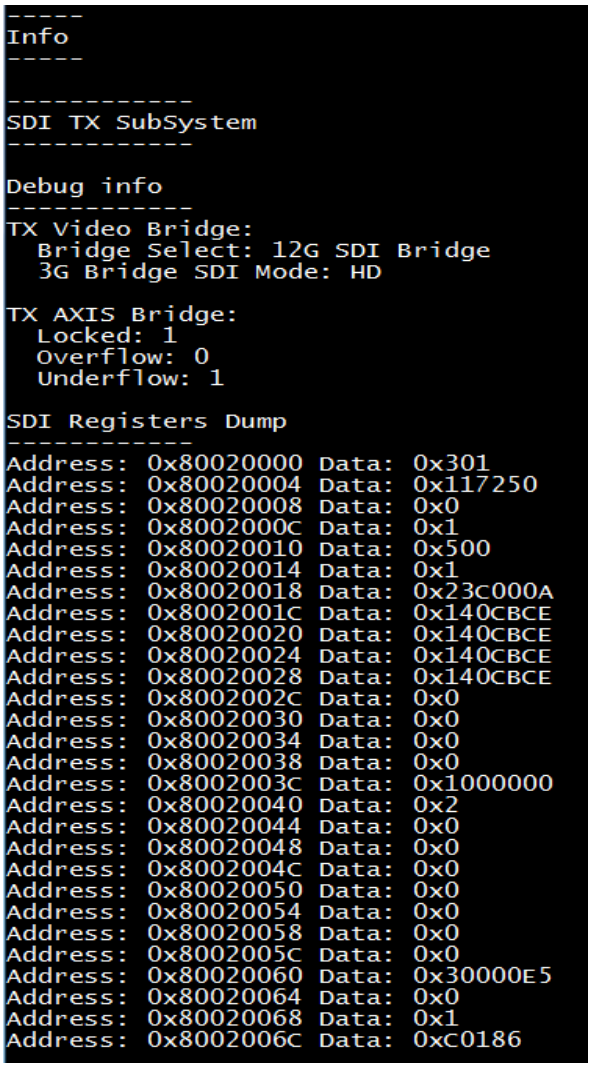

Figure 5-22: **UART Console - Register Dump for Debugging**

ZCU106 SMPTE UHD-SDI Audio-Video Pass-Through Example Design

The UHD-SDI Audio-Video Pass-through example design, shown in Figure 5-23, is built using UHD-SDI TX and RX Subsystems along with UHD-SDI Audio and SPDIF/AES3 IP cores.

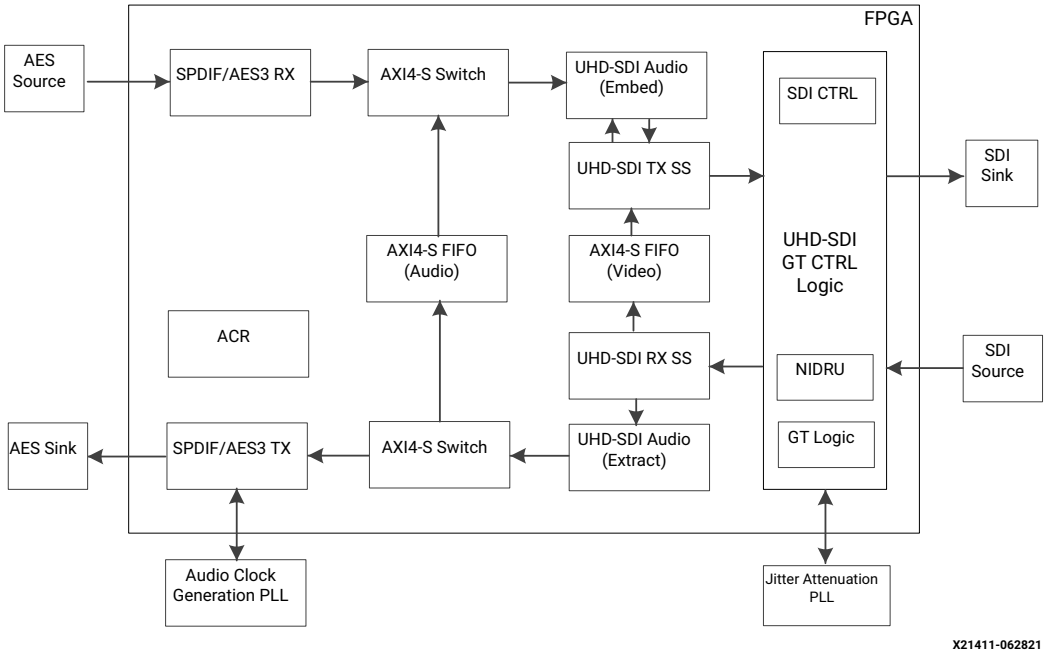

Figure 5-23: **ZCU106 Audio-Video Pass-Through Design**

Note: The Zynq UltraScale+ MPSoC PS is not shown in Figure 5-11 for simplicity.

Video Processing Path

Video or image data with audio is received and processed by the SMPTE UHD-SDI RX Subsystem. The clock is recovered by the Xilinx® UltraScale™+ GTH transceiver (RX) and fed to the on-board PLL for jitter attenuation. A jitter-attenuated clock is used as a reference clock by the GTH transceiver for the TX data path. An AXI4-Stream FIFO is used for synchronization and temporary storage between the SMPTE UHD-SDI RX Subsystem and the UHD-SDI TX Subsystem. The UHD-SDI TX Subsystem transmits SDI data from the AXI4-Stream FIFO after the application programs the UHD-SDI TX Subsystem sub-core registers based on the received SDI stream and the ST-352 payload packet data.

Audio Processing Path

This example design supports one of the two modes listed below based on the user input. Initially, the system comes up with SDI Audio Pass-through mode. You can switch to AES3 Audio Capture and Playback mode by pressing **a** on the UART terminal.

SDI Audio Pass-through

In this mode, up to 32 channels of audio can be extracted by the UHD-SDI Audio (Extract) IP. Audio received from the RX data path is embedded into the native SDI stream by the UHD-SDI Audio (Embed) IP. An AXI4-Stream FIFO is used for temporary storage between the UHD-SDI Audio (Extract) and the UHD-SDI Audio (Embed) IP.

AES3 Audio Capture and Playback

In this mode, up to 32 channels of audio can be extracted by the UHD-SDI Audio (Extract) IP. Channels 1 and 2 of the extracted audio are sent to the SPDIF/AES3 TX IP for AES playback.

The Audio Clock Recovery (ACR) IP is used to generate the audio sample rate clock (48 kHz, 44.1 kHz, or 32 kHz) from the SDI RX video clock. Application software programs the N and CTS values in the ACR IP, based on the SDI RX mode and the audio sample rate provided by the UHD-SDI Audio (Extract) IP. The ACR IP tracks the SPDIF/AES3 TX FIFO data count to adjust the sample rate clock to avoid FIFO overflow or underflow resulting in dropping or insertion of null samples. The sample rate clock from the ACR is fed to the on-board PLL for generating the audio clock (F_s*512) for the SPDIF/AES3 TX IP. On the other hand, Audio captured from the AES input is embedded on to the SDI TX.

The example design application software runs on the Zynq UltraScale+ MPSoC Arm processor subsystem (PS) and is fully software controlled.

Clocking

QPLL0 is allocated for UHD-SDI RX transceiver and QPLL1 for UHD-SDI TX in this pass-through design. The reference clock for QPLL1 comes from si5328 chip output. Thus, QPLL1 reference clock connection is fixed. QPLL0 reference clock is fixed to 148.5 MHz which comes from on-board si570 chip. Figure 5-11 shows the clocking used in the UHD-SDI example design.

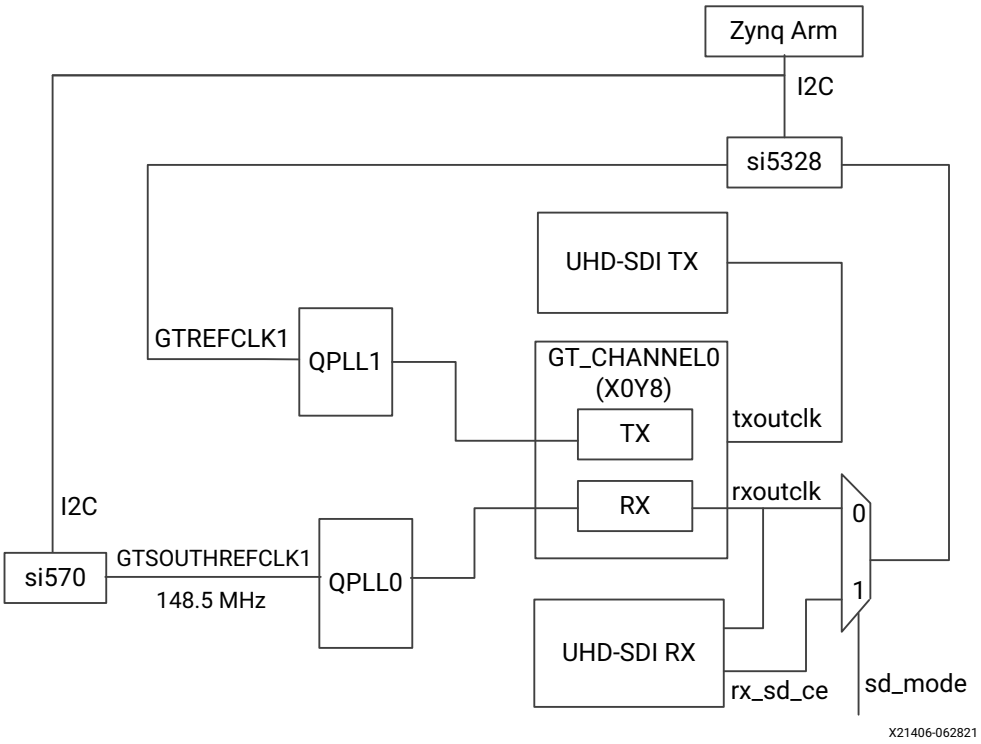

Figure 5-24: **ZCU106 Audio-Video Pass-Through Example Design Clocking**

Table 5-3 shows the clock frequency at different part of the system for different SDI modes:

Table 5-4: **ZCU106 Audio-Video Example Design Clock Frequency Ranges**

SDI Mode	Tx_m/ Rx_m	QPLL0 Ref clk (MHz)	QPLL1 Ref Clk (MHz)	txoutclk (MHz)	si5328 Input (MHz)	si5328 Output (MHz)	txoutclk (MHz)
SD-SDI	N/A	148.5	148.5	148.5 rx_sd_ce=27	rx_sd_ce=27	148.5	148.5
HD-SDI	0	148.5	148.5	74.25	74.25	148.5	74.25
HD-SDI	1	148.5	148.5/1.001	74.25/1.001	74.25/1.001	148.5/1.001	74.25/1.001
3G-SDI/6G-SDI	0	148.5	148.5	148.5	148.5	148.5	148.5
3G-SDI/6G-SDI	1	148.5	148.5/1.001	148.5/1.001	148.5/1.001	148.5/1.001	148.5/1.001
12G-SDI	0	148.5	148.5	297	297	148.5	297
12G-SDI	1	148.5	148.5/1.001	297/1.001	297/1.001	148.5/1.001	297/1.001

Note: For 6G-SDI and 12G-SDI, 8 native SDI Data Streams (DS) is assumed.

For GT TX and RX data path, the reference clock requirement for data paths are different. For GT TX, for integer and fractional frame rate, PLL reference clock must be different frequency, clock/1.000 for integer frame rate and clock/1.001 for fractional frame rate. For RX data path PLL reference clock can be same for integer and fractional frame rate up to 6-G SDI and separate reference clock is required for 12-G SDI integer and fractional frame rate.

Transceiver Configuration in Example Design

The UHD-SDI example design uses the **uhdsdi_gt_v2_0** core to configure UltraScale+ GTH transceivers and provide options to select the transceiver reference clocking. The core also generates control modules that are required to program the transceiver using DRP interface and NI-DRU modules for RX SD-SDI mode.

Running the ZCU106 Audio-Video Pass-Through Example Design

1. Open the Vivado Design Suite and create a new project.
2. In the pop-up window, press **Next** 5 times.
3. Select the **ZCU106** Board as shown in Figure 5-25.

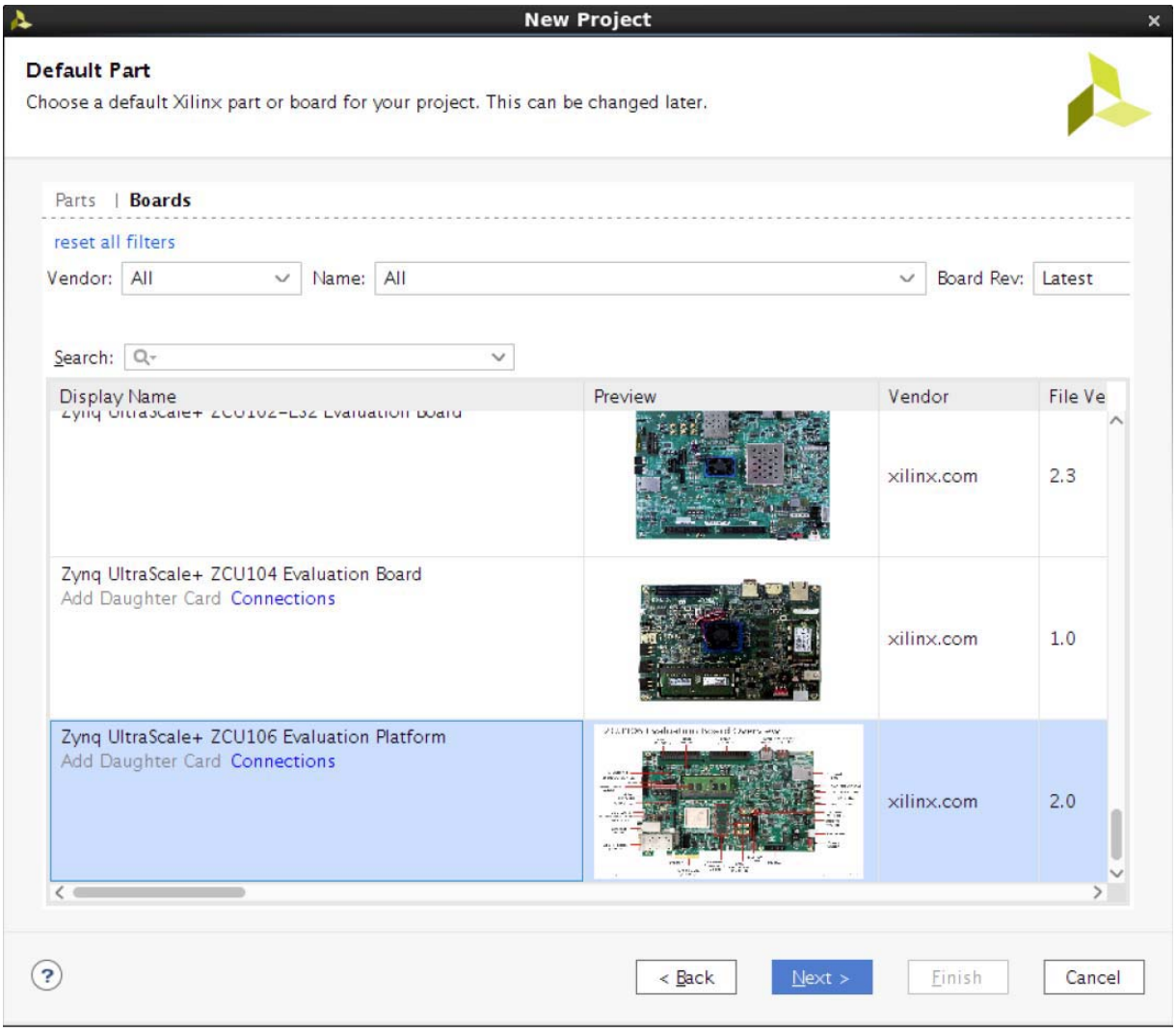

Figure 5-25: **Select the ZCU106 Board**

4. Click **Finish**.
5. In the IP Catalog double-click **SMPTE UHD-SDI RX Subsystem** under Video Connectivity.

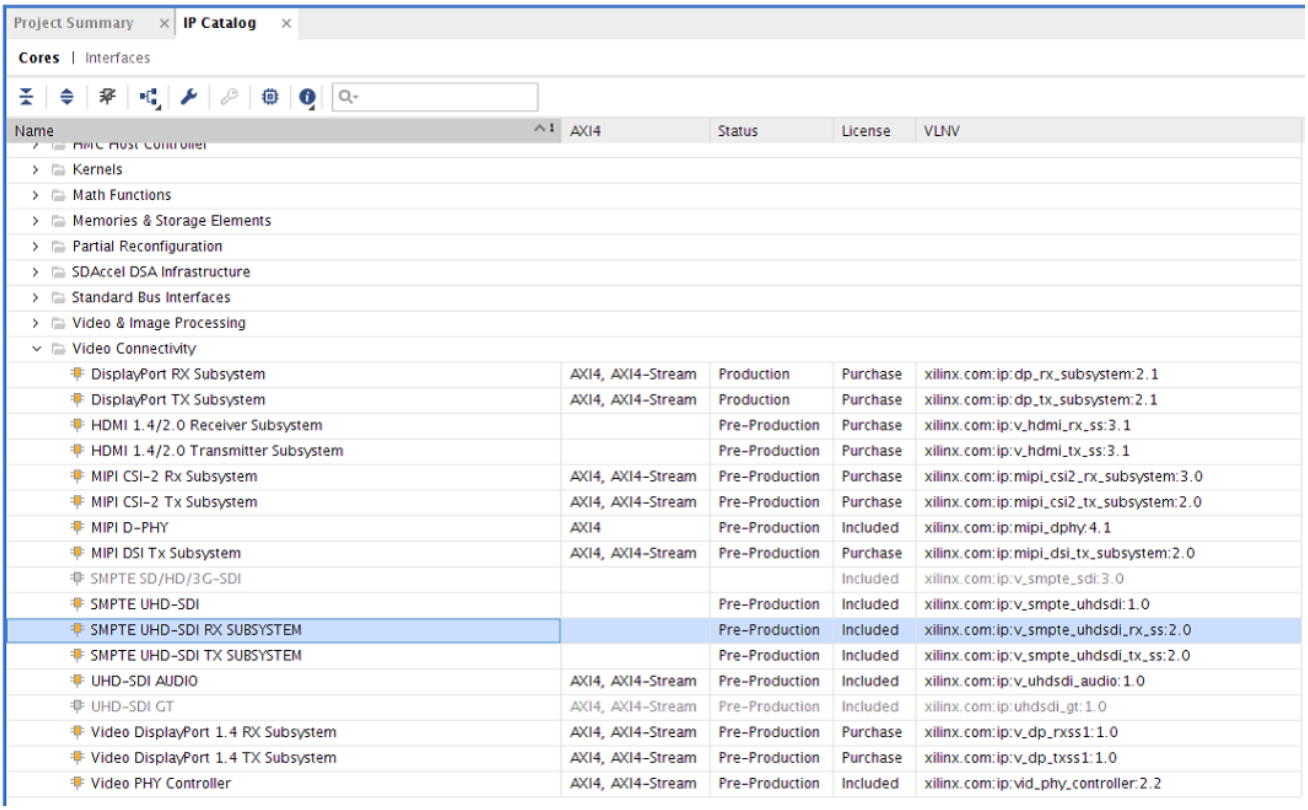

Figure 5-26: **Select the SMPTE UHD-SDI RX Subsystem**

Note: For the application example design flow, the IP configuration is based on options selected in the Application Example Design tab. You can rename the IP component name, which is used as the application example design project name.

6. Configure the SMPTE UHD-SDI RX Subsystem **Application Example Design** tab by selecting ZCU106 as the Target Board followed by **Audio-Video Pass-Through** as Design Topology and click **OK**.

   The Generate Output Products dialog box appears.

7. Click **Generate** (See Figure 5-27).

   Note: Click **Skip** if you only want to generate the application example design.

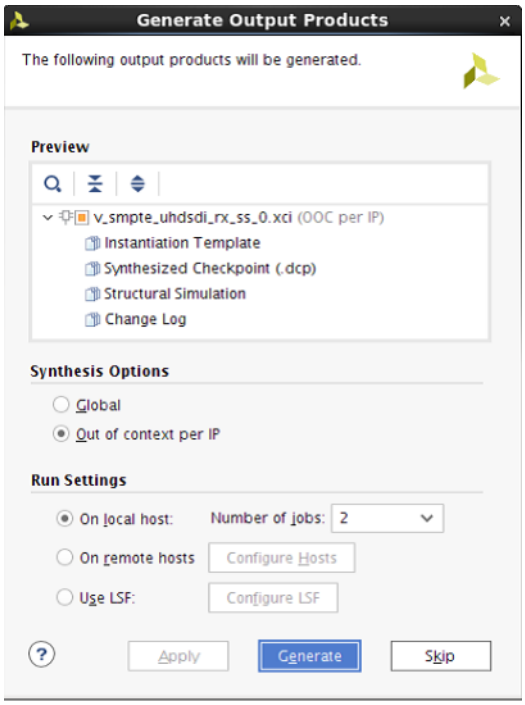

Figure 5-27: **Generate Output Products**

8. Right-click the **SMPTE UHD-SDI RX Subsystem** component under Design source, and click **Open IP Example Design**.
9. Choose the target project location and click **OK**.

   The IP integrator design is then generated and creates the Vitis software platform and generates a .elf file. You can choose to Run Synthesis, Implementation, or Generate Bitstream. An overall system IP integrator block diagram of the ZCU106-based application example design is shown here (Figure 5-28).

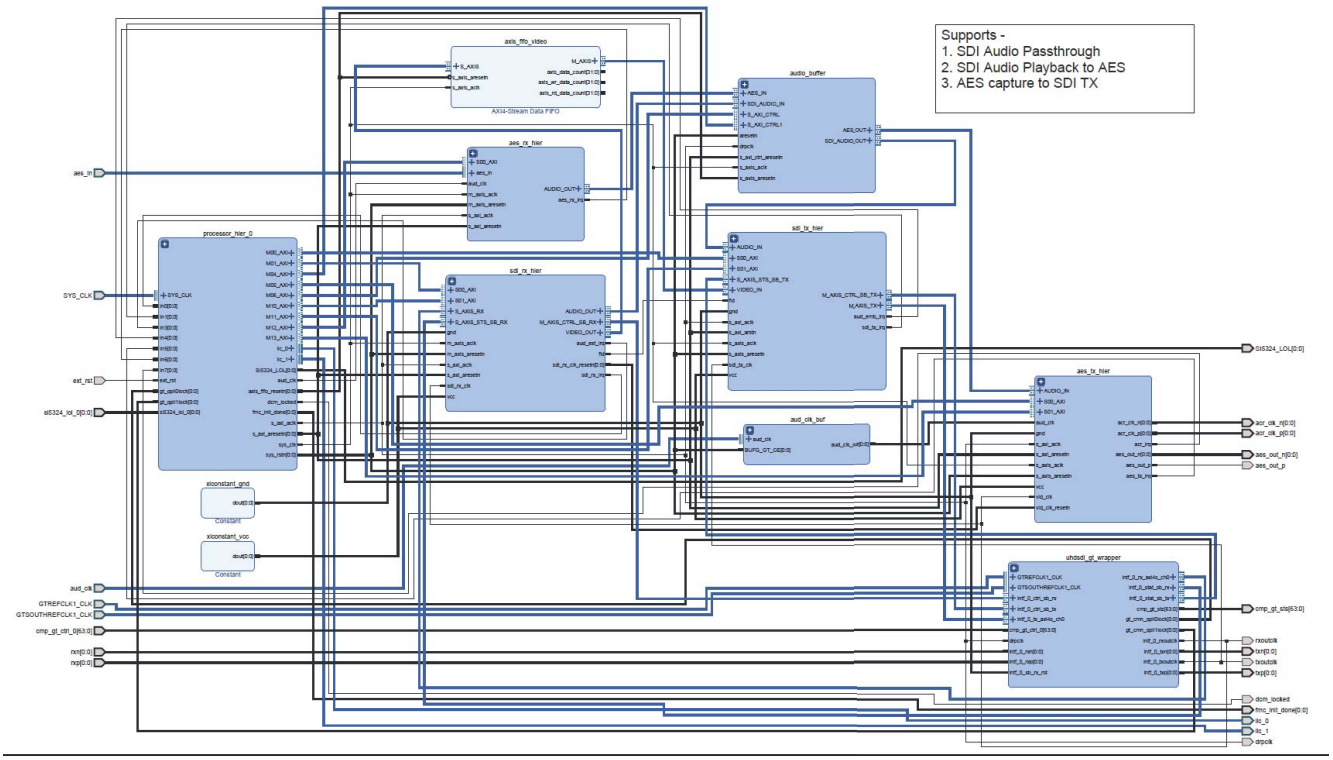

Figure 5-28: **IP Integrator Diagram of the ZCU106 Audio-Video Pass-Through Example Design**

Requirements

Hardware

The hardware requirements for this reference system are:

- Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit
- SDI source equipment
- SDI sink equipment
- Four HD-BNC to BNC converter cables
- AES source equipment
- AES sink equipment

Software

This section includes any software requirements:

- Vivado Design Suite 2019.2 or later

- The Vitis software platform 2019.2 or later
- Software terminals (for example, Tera Term, HyperTerminal or PuTTY)

Setup

The reference design runs on the Zynq UltraScale+ MPSoC board (ZCU106) using SDI and AES3 connectors available on the board.

Note: In the following procedure, the numbers in parentheses correspond to the callout numbers in Figure 5-29.

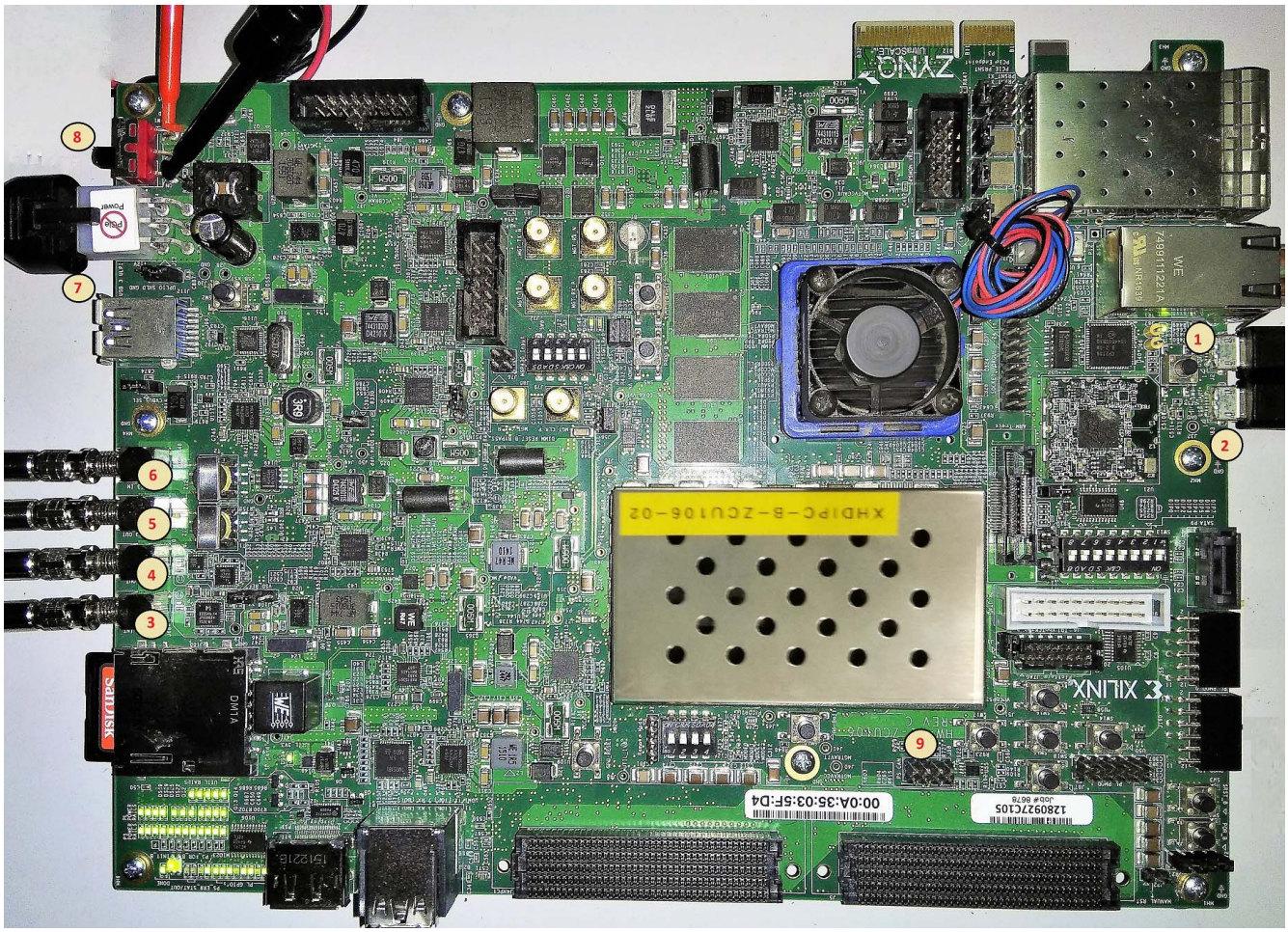

Figure 5-29: **ZCU106 Board Setup**

In the following procedure, the numbers in parentheses correspond to the callout numbers in Figure 5-18.

1. Connect a USB cable from the host PC to the USB JTAG port (1). Ensure the appropriate device drivers are installed.
2. Connect a second USB cable from the host PC to the USB UART port (2). Ensure that the USB UART drivers described in Hardware have been installed.

3. Connect the SDI_INT link of ZCU106 (3) to the SDI source device.
4. Connect the SDI_OUT link of ZCU106 (4) to the SDI sink device.
5. Connect the AES3_OUT link of ZCU106 (5) to the AES sink device.
6. Connect the AES3_IN link of ZCU106 (6) to the AES source device.
7. Connect the ZCU106 board to a power supply slot (7).
8. Switch on the ZCU106 board (8).
9. Make sure that the HW-ZCU106 board revision (9) is Rev C.
10. Ensure that the SMPTE ST 352/Payload ID is enabled in the SDI Stream connected to the SDI input link of ZCU106.
11. Start Tera Term or PuTTY to connect to the COM port interface 0 on the Host PC configured with 115200 bps, 8 bits, No parity, 1 stop bit, and no flow control.

Compiling Software in Vitis Software Platform

The UHD-SDI application example design generates a .elf file automatically. Use the following procedure if you want to open the Vitis software platform project for UHD-SDI example design from the Vivado Design Suite:

1. In the Vivado Design Suite, click **Tools > Launch Vitis**.
2. Select **Exported location** and **workspace** as **`<Proj Dir>/SW/xsdi_app`** and click **OK** to launch and open the Vitis software platform project.

   Vitis software platform project opens.

Running the Design on the Hardware

The following steps are used to run the BIT and ELF files on the hardware setup:

1. Connect the JTAG cable and USB-UART cable to the board.
2. Navigate to **`<Component Name>_ex/imports`**.
3. Start the Xilinx Software Debugger (XSDB) by sourcing XSDB from the build area from the command prompt.
4. Run the following command to program FPGA and to execute the application.

   ```
   source xsdb.tcl
   ```

5. To observe the results, open Tera Term or PUTTY and configure its serial port (Interface 0) to 115200 baud with the default configuration. Make sure that the UART cable is connected to the board and the PC.

   The UART console displays SDI stream details on console.

UART Console Screens

The following figure shows the initial UART console output along with menu options.

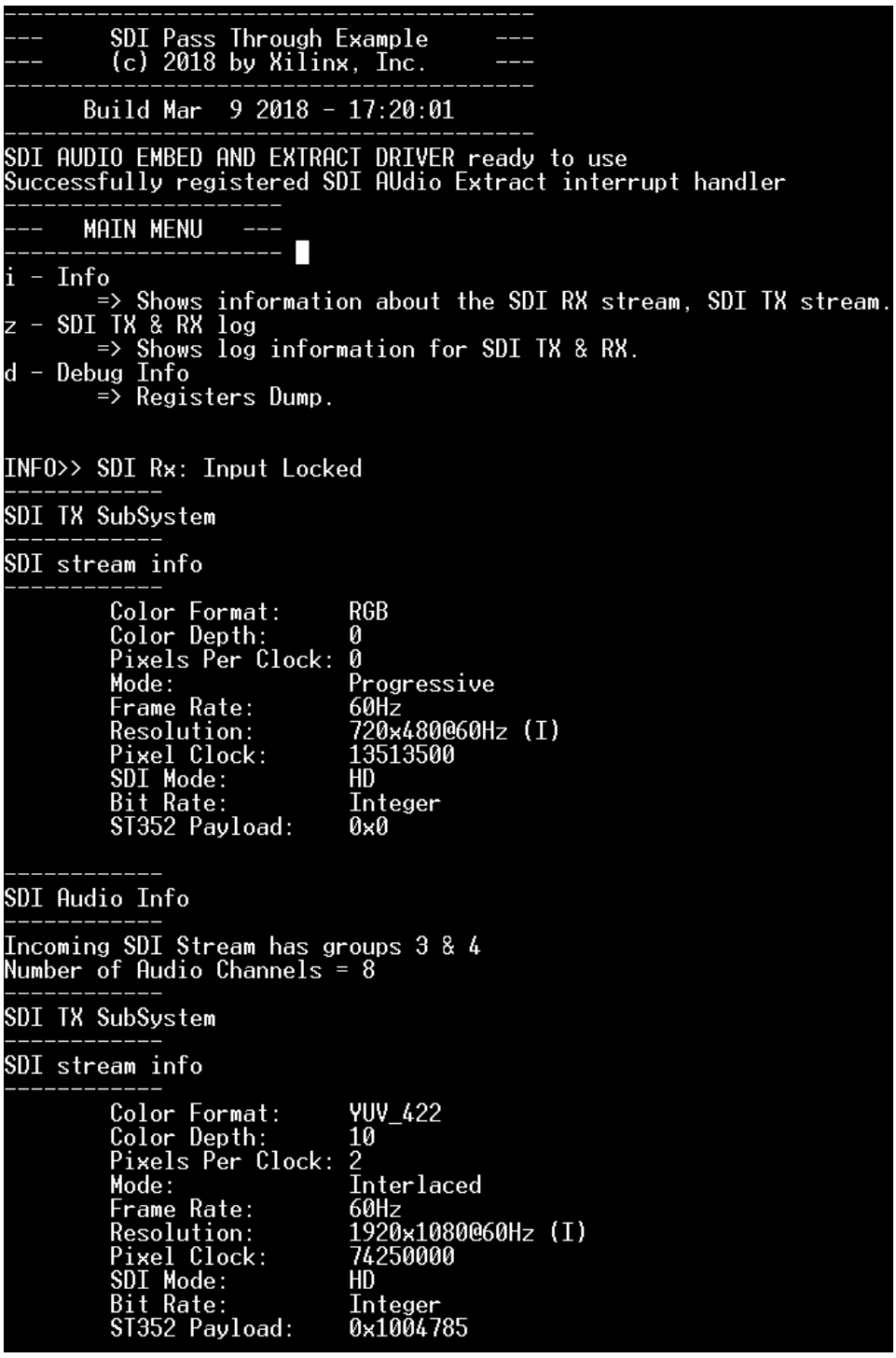

Figure 5-30: **UART Console - UHD SDI Example Design UART Menu**

The following figure shows the UART console output when the i key is pressed.

```
------------
SDI TX SubSystem
------------

  ->SDI TX Subsystem Cores
    : SDI TX
    : VTC Core

SDI stream info
------------
        Color Format:     YUV_422
        Color Depth:      10
        Pixels Per Clock: 2
        Mode:             Interlaced
        Frame Rate:       60Hz
        Resolution:       1920x1080@60Hz (I)
        Pixel Clock:      74250000
        SDI Mode:         HD
        Bit Rate:         Integer
        ST352 Payload:    0x1004785
        No Error Detected

SDI TX timing
------------

        HSYNC Timing: hav=1920, hfp=88, hsw=44(hsp=1), hbp=148, htot=2200
        VSYNC Timing (Field 0): vav=0540, vfp=02, vsw=05(vsp=1), vbp=015, vtot=0
562
        VSYNC Timing (Field 1): vav=0540, vfp=03, vsw=05(vsp=1), vbp=015, vtot=0
563


------------
SDI Rx SubSystem
------------

 ->SDI RX Subsystem Cores
 : SDI RX

SDI stream info
------------
        Color Format:     YUV_422
        Color Depth:      10
        Pixels Per Clock: 2
        Mode:             Interlaced
        Frame Rate:       60Hz
        Resolution:       1920x1080@60Hz (I)
        Pixel Clock:      74250000
        SDI Mode:         HD
        Bit Rate:         Integer
        ST352 Payload:    0x1004785
        CRC: 3

SDI RX timing
------------

        HSYNC Timing: hav=1920, hfp=88, hsw=44(hsp=1), hbp=148, htot=2200
        VSYNC Timing (Field 0): vav=0540, vfp=02, vsw=05(vsp=1), vbp=015, vtot=0
562
        VSYNC Timing (Field 1): vav=0540, vfp=03, vsw=05(vsp=1), vbp=015, vtot=0
563
```

Figure 5-31: **UART Console - Receive and Transmit UHD SDI Stream Information**

KCU116 SMPTE UHD-SDI Audio-Video Loopback Example Design

The UHD-SDI audio-video loopback example design is built using the UHD-SDI TX and RX subsystems along with the UHD-SDI Audio IP core. Video data generated by the video test pattern generator is passed to the UHD-SDI TX Subsystem. The UHD-SDI TX Subsystem outputs SDI stream after insertion of the ST352 payload (if any) and passed to the UHD-SDI Audio (Embed) IP for audio data insertion. To do so, bit 18 (**use_anc_in**) of **sdi_tx_ctrl[31:0]** has to be set to **1`b1** in the UHD-SDI TX Subsystem. Audio data is generated by the audio test pattern generator. See Appendix C to for more information on the audio test pattern generator. Native SDI video stream with audio insertion is passed to the UHD-SDI TX IP for transmission. The UHD-SDI RX Subsystem receives video and audio and passes the native video stream to the UHD-SDI Audio (Extract) IP. Audio is extracted by UHD-SDI Audio IP. The UHD-SDI TX Subsystem, UHD-SDI RX Subsystem, UHD-SDI Audio (Embed) IP, and UHD-SDI Audio (Extract) IP are configured through ports and does not include the AXI4-Lite interface for this example design. The example design application software runs on a MicroBlaze™ processor.

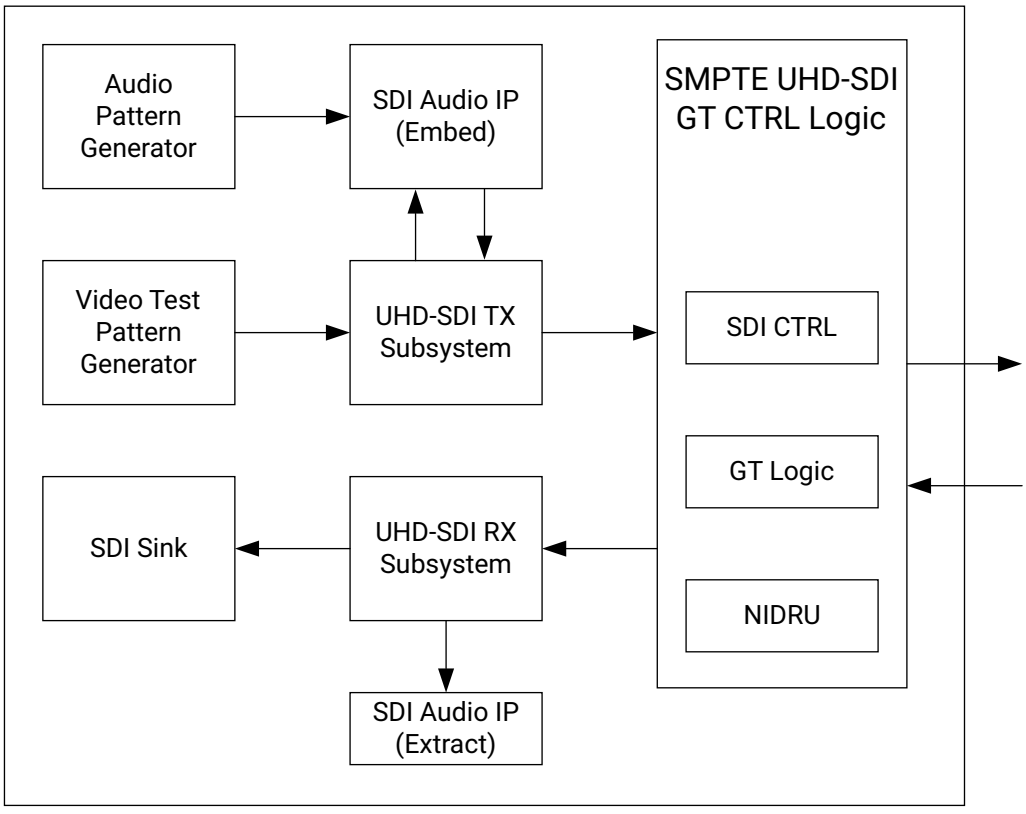

Figure 5-32: **KCU116 Audio-Video Loopback Example Design**

Note: The MicroBlaze processor is not shown in the above figure for simplicity.

Clocking

The KCU116 UHD-SDI loopback example design GT clocking architecture is shown in the following figure.

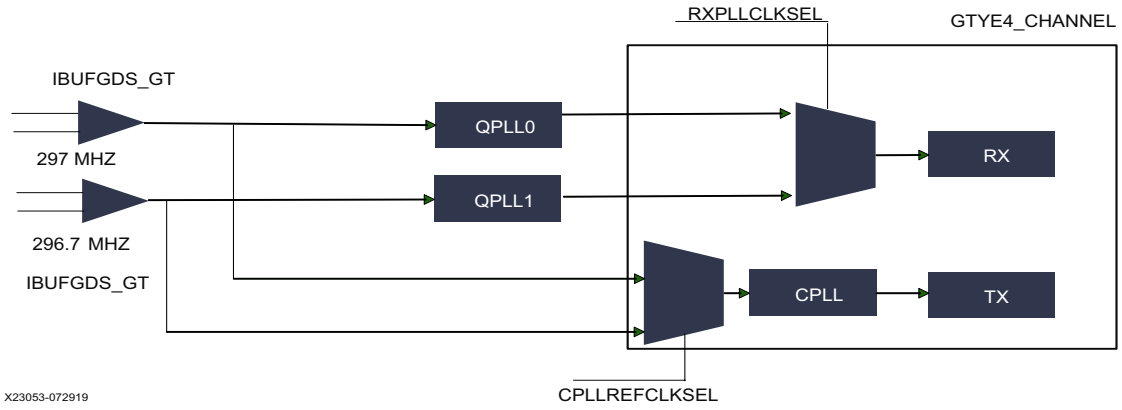

Figure 5-33: **Loopback Example Design GT Clocking Architecture**

Two reference clocks are used to support integer and fractional line rates of SDI. The reference clock for QPLL0 is fixed to 297 MHz from the on-board Si570 chip. The reference clock for CPLL is fixed at 296.7 MHz from Si5328 chip output. CPLL switches between 297 MHz and 296.7 MHz reference clocks using the CPLLREFCLKSEL.

Note: When using QPLL0 and QPLL1 for 12G-SDI integer and fractional (1/1.001) rate change, switching between rates on the SDI-RX can introduce a glitch on the clock which in turn introduces CRC errors on the TX channel. CRC errors do not occur in SD-SDI/HD-SDI/3-G SDI/6-G SDI integer/fractional modes with QPLL0 and QPLL1 clocking combination. For more information, see Answer Record 72254 and 72449. Therefore, it is not recommended to use this clocking configuration when both transmit and receive 12G-SDI integer and fractional modes use the same transceiver. If required, Xilinx recommends to use CPLL-QPLL combination with CPLL for TX and QPLL0/1 for RX as shown in Figure 5-33.

The integer and fractional rates for TX can be selected using CPLL reference clock input selection with 297 MHz and 296.7 MHz respectively. This CPLL/QPLL clocking combination is not feasible with -1 speed grade devices since CPLL does not support 12G-SDI line rates. You need to select an UltraScale+ GTH/GTY -2 speed grade or faster rate with >0.85V. Refer the respective FPGA device data sheets for CPLL line rate limits. The UHD-SDI example designs are updated to use the CPLL and QPLL clocking combination. The UHD-SDI GT IP is updated to provide CPLL support starting from 2019.2.

Running the KCU116 Audio-Video Loopback Example Design

1. Open the Vivado Design Suite and create a new project.
2. In the pop-up window, press **Next** 5 times.

3. Select the **KCU116** Board.

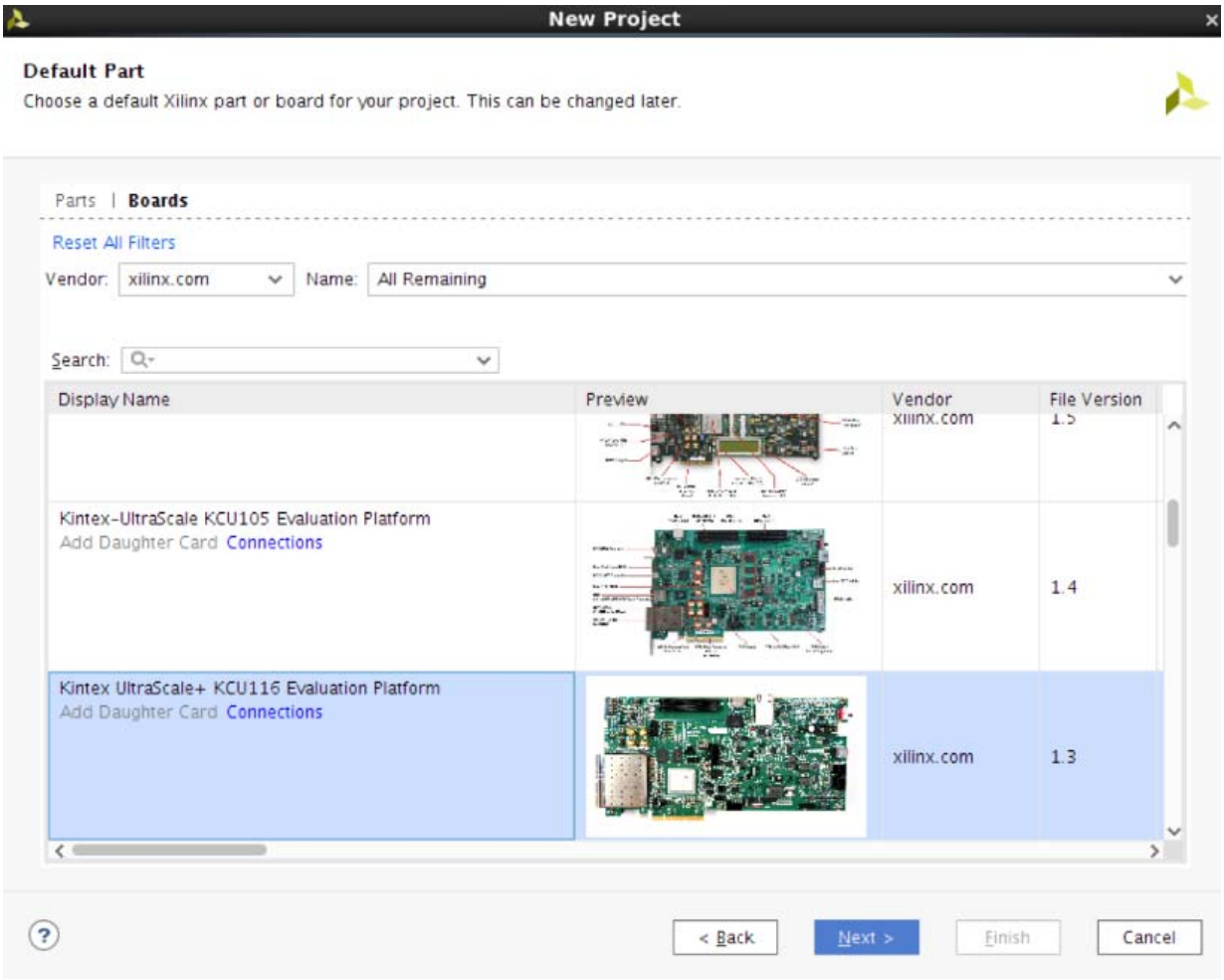

Figure 5-34: **Select the KCU116 Board**

4. Click **Finish**.

5. Click IP Catalog and double-click **SMPTE UHD-SDI RX Subsystem** under Video Connectivity, as shown in Figure 5-35.

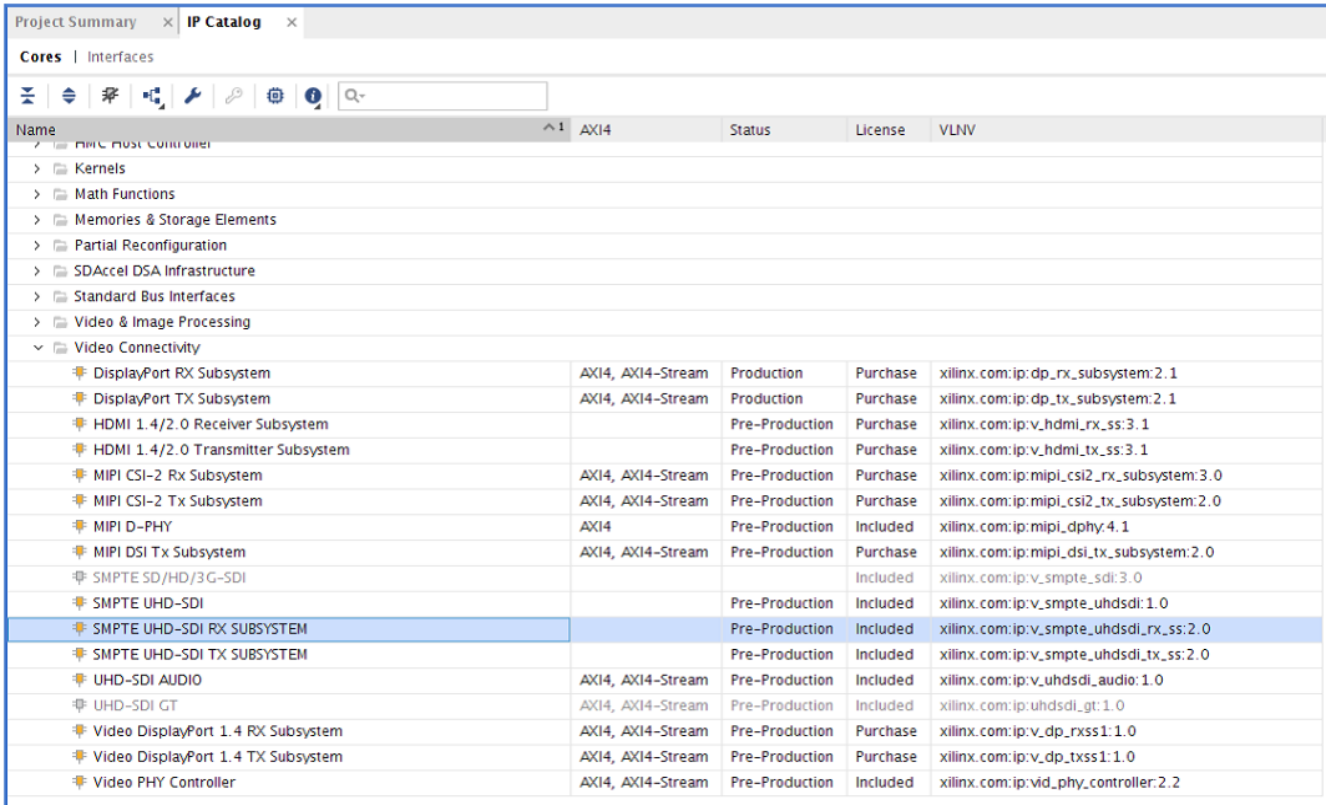

Figure 5-35: **Select the SMPTE UHD-SDI RX Subsystem**

Note: For the application example design flow, the IP configuration is based on the options selected in the Application Example Design tab. You can rename the IP component name, which is used as the application example design project name.

6. Configure the SMPTE UHD-SDI RX Subsystem **Application Example Design** tab by selecting KCU116 as the Target Board followed by **Audio-Video Pass-Through** as the Design Topology and click **OK**.

   The Generate Output Products dialog box appears

7. Click **Generate**.

   Note: Click **Skip** if you only want to generate the application example design.

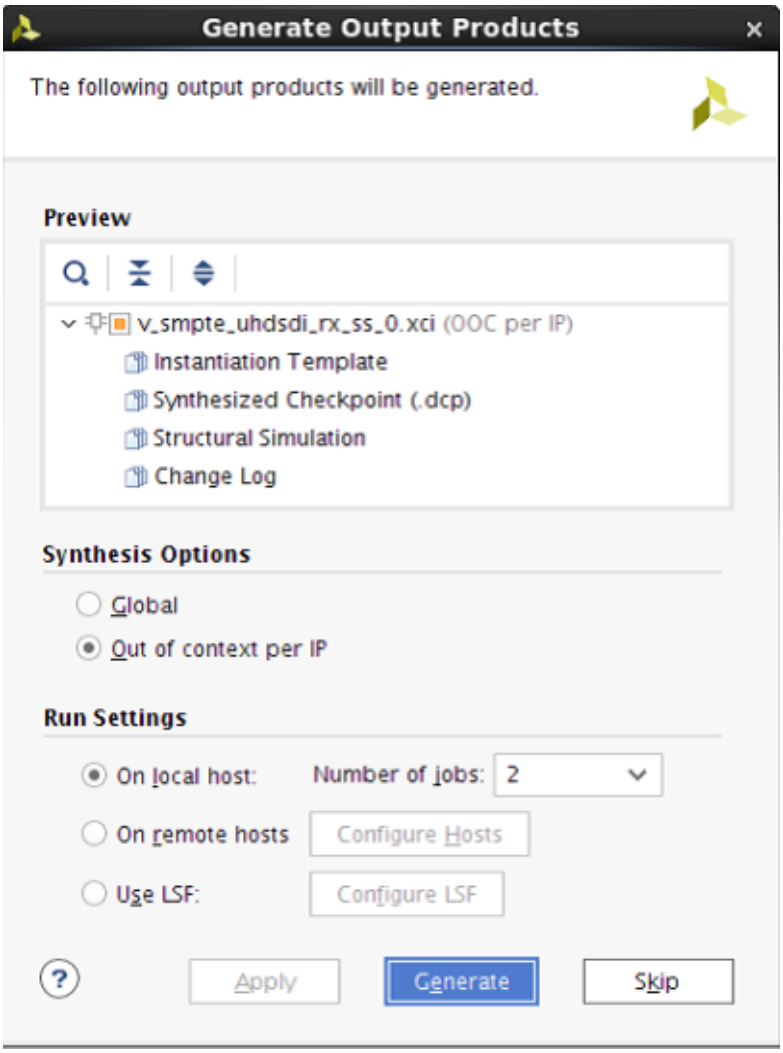

Figure 5-36: **Generate Output Products**

8. Right-click the **SMPTE UHD-SDI RX Subsystem** component under Design source, and select **Open IP Example Design**, as shown in Figure 5-37.

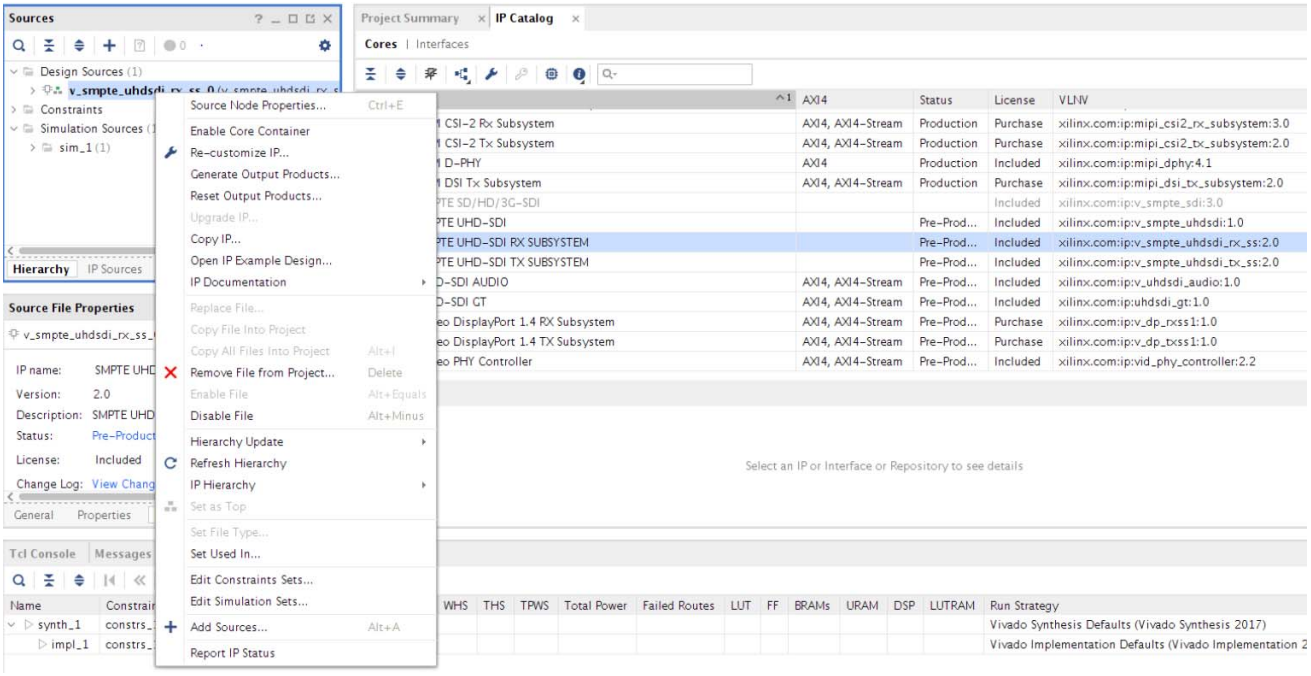

Figure 5-37: **Open the IP Example Design**

9. Choose the target project location and click **OK**.

   The IP integrator design is then generated and creates the Xilinx Vitis software platform and generates a `.elf` file. You can choose to Run Synthesis, Implementation, or Generate Bitstream. An overall system IP integrator block diagram of the KCU116-based application example design is shown in Figure 5-38.

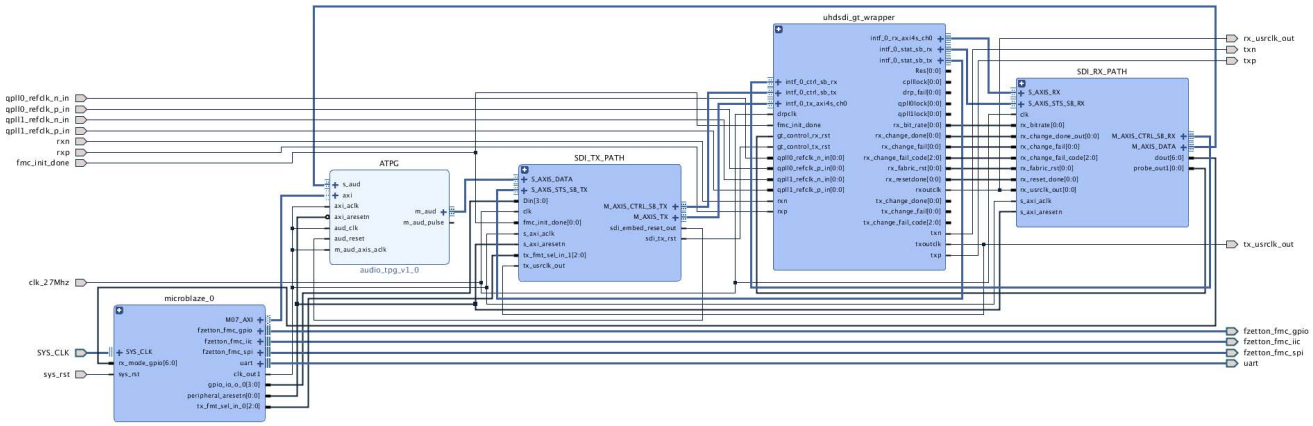

Figure 5-38: **IP Integrator Diagram of the KCU116 Audio-Video Loopback Example Design**

Requirements

Hardware

The hardware requirements for this reference system are:

- Xilinx Kintex UltraScale+ FPGA KCU116 Evaluation Kit
- Fidus TB-FMCH-12GSDI SDI FMC card
- One HD-BNC to HD-BNC cable

Software

This section lists the software requirements:

- Vivado Design Suite 2019.2 or later
- The Vitis software platform 2019.2 or later
- Software terminals (for example, Tera Term, HyperTerminal or PuTTY)

Setup

This example runs on KCU116 board and more information about the board can be found on the product page.

Note: In the following procedure, the numbers in parentheses correspond to the call out numbers in Figure 5-39.

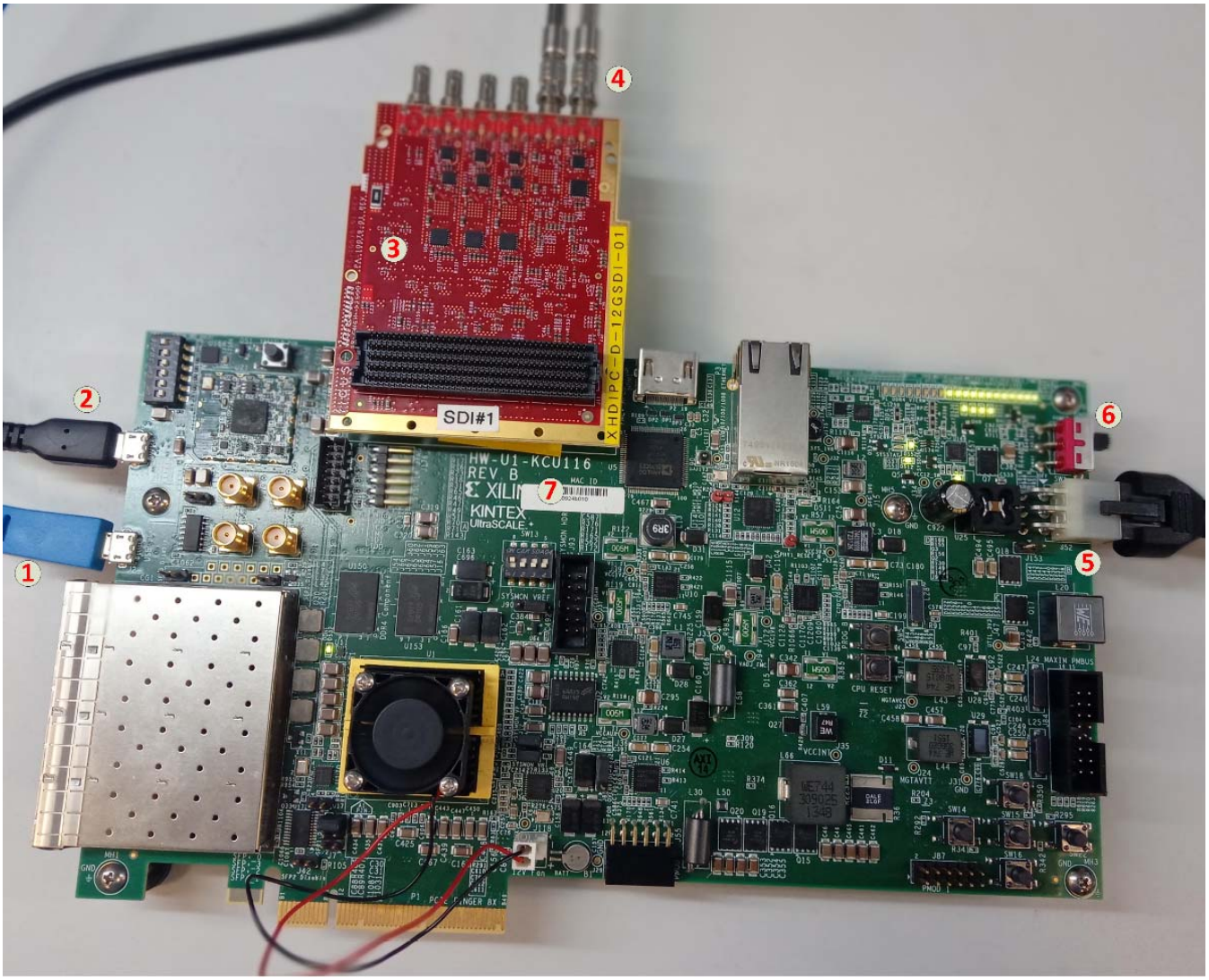

Figure 5-39: **KCU116 Board Setup**

1. Connect a USB cable from the host PC to the USB JTAG port (1). Ensure the appropriate device drivers are installed.
2. Connect a second USB cable from the host PC to the USB UART port (2). Ensure that the USB UART drivers described in Hardware have been installed.
3. Insert Fidus FMC card into FMC slot (3).
4. Connect the HD-BNC to HD-BNC cable (4) in Fidus FMC.
5. Connect the KCU116 board to a power supply slot (5).
6. Switch on the KCU116 board (6).
7. Make sure that the HW-KCU116 board revision (7) is Rev B.

8. Start Tera Term or PuTTY to connect to the COM port interface 0 on the Host PC with 115200 bps, 8 bits, No parity, 1 stop bit, and no flow control as configuration.
9. Verify the FMC_VADJ voltage to 1.8V.

Compiling Software in the Vitis Software Platform

The UHD-SDI application example design generates a .elf file automatically. Follow the steps if you want to open the Vitis software platform project for the UHD-SDI example design from the Vivado Design Suite:

1. In the Vivado Design Suite, click **Tools** > **Launch Vitis**.
2. Select **Exported location** and **workspace** as **`<Proj Dir>/SW/xsdi_app`** and click **OK** to launch and open the Vitis software platform project.

The Vitis software platform project is now open.

Running the Design on the Hardware

The following steps are used to run the BIT and ELF files on the hardware setup:

1. Connect the JTAG cable and USB-UART cable to the board.
2. Go to **`<Component Name>_ex/imports`**.
3. Start the Xilinx Software Debugger (**`XSDB`**) by sourcing xsdb from the build area to the command prompt.
4. Run the following command, **`source xsdb.tcl`**, to program FPGA and to execute the application.
5. To observe the results, open Tera Term or PUTTY, and configure the serial port (Interface 0) to 115200 baud with the default configuration.

Note: Make sure that the UART cable is connected to the board and the PC.

The UART console now displays the SDI stream details on the console.

UART Console Screens

The following figure shows initial UART console output along with menu options.

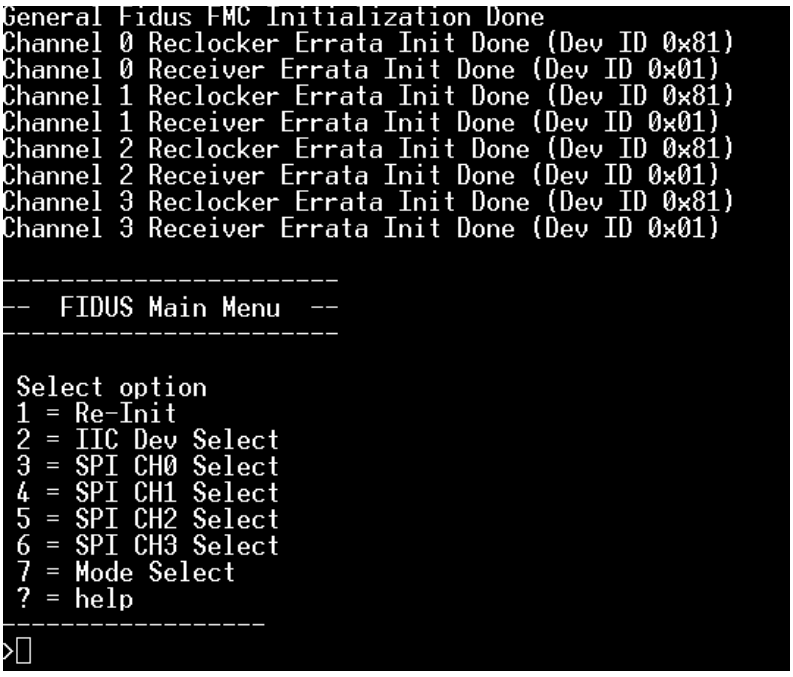

Figure 5-40: **UART Console Selection Menu**

The following figure shows the UART console output that shows available options.

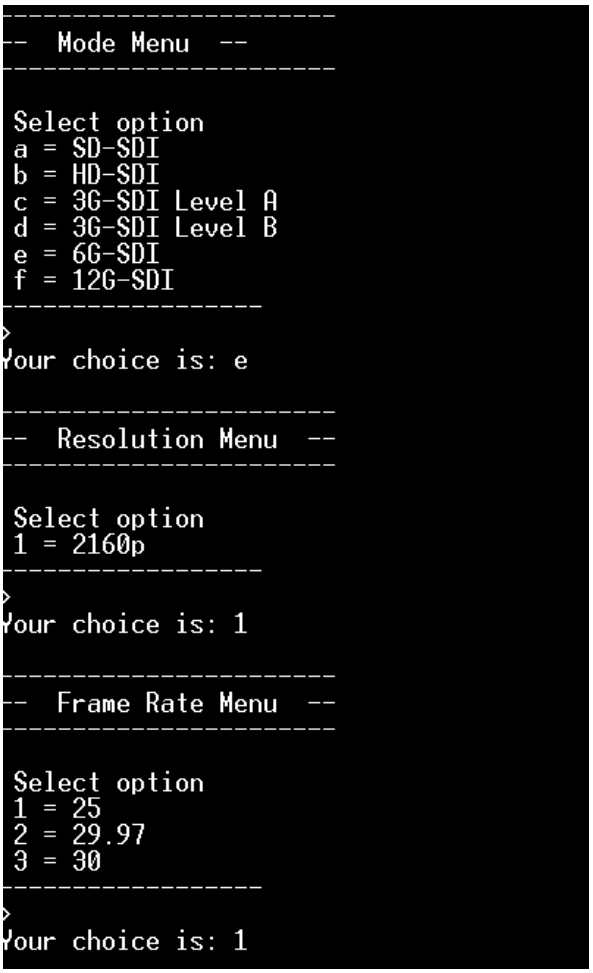

Figure 5-41: **UART Console - Transmit Mode and Video Resolution Selection Menu**

The following figure shows the UART console output that shows test pass indication.

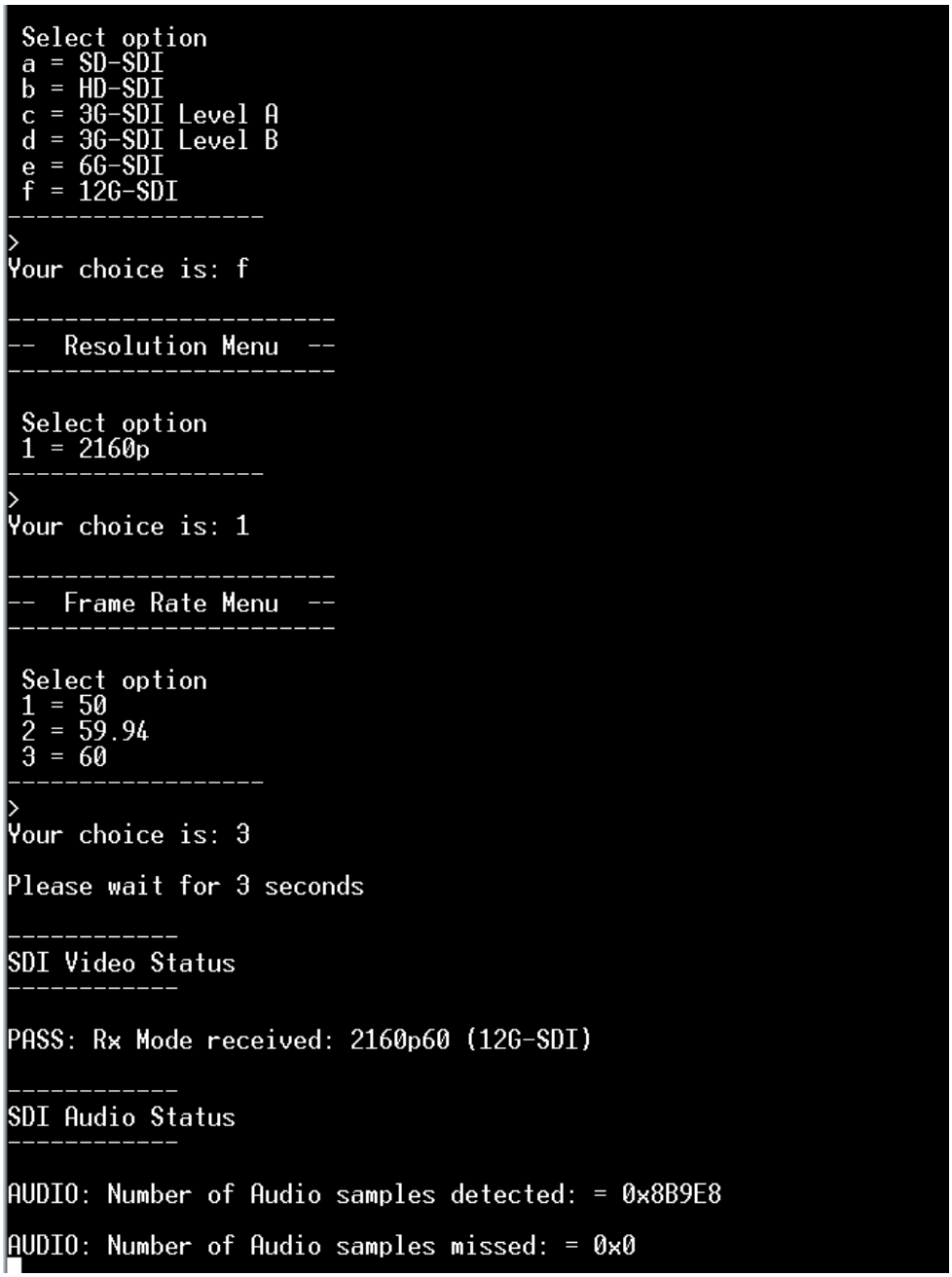

Figure 5-42: **UART Console: Shows Received UHD SDI Mode Matches with Transmitted UHD SDI Mode**

VCK190 SMPTE UHD-SDI Audio-Video Pass-Through Example Design

The UHD-SDI Audio-Video pass-through example design, shown in Figure 5-43, is built using the SMPTE UHD-SDI TX and RX subsystems along with Audio IP core.

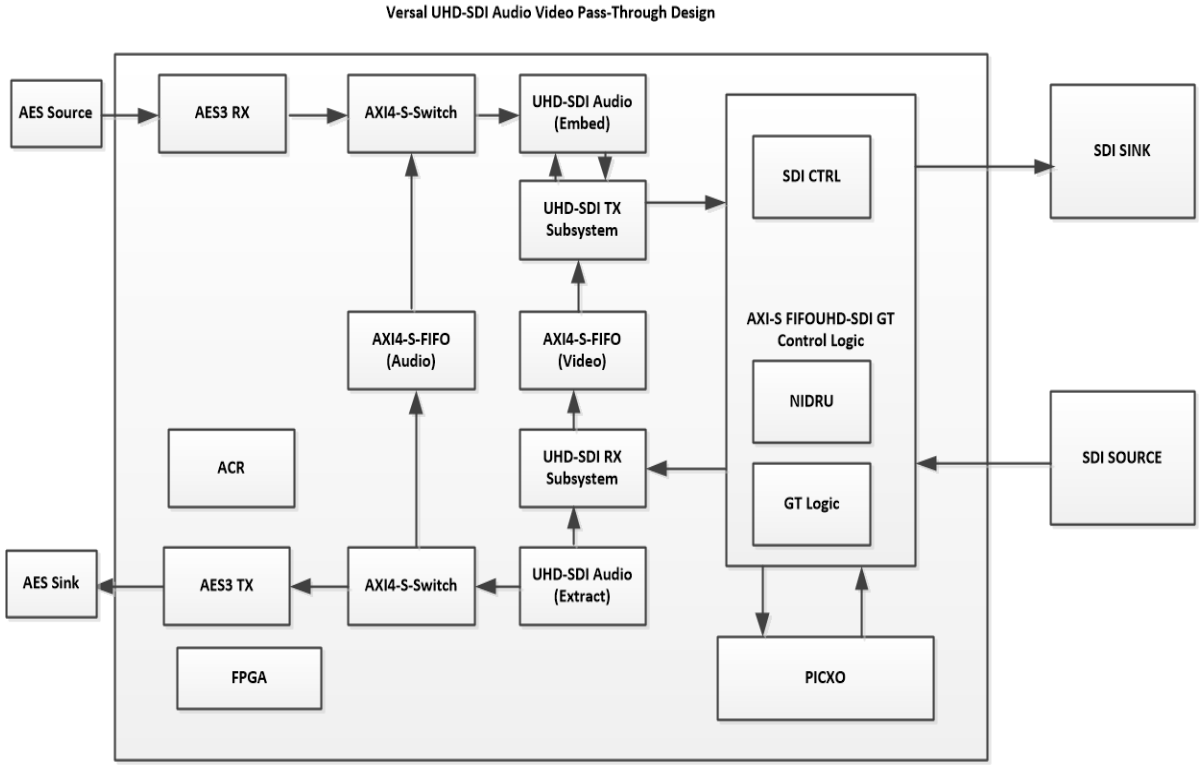

Figure 5-43: **UHD-SDI Audio-Video Pass-through Design**

SDI Video Pass-through

- The SMPTE UHD-SDI RX Subsystem receives and processes video or image data with audio.
- The Xilinx® Versal GTY transceiver (RX) recovers the clock and feeds to the PICXO IP for jitter attenuation.
- The jitter-attenuated clock is used as a reference clock by the GTY transceiver for the TX data path.
- The UHDSDI GT Bridge IP passes data and control information between the GTY transceiver and UHDSDI RX and TX subsystem.
- An AXI4-Stream FIFO is used for synchronization and temporary storage between the SMPTE UHD-SDI RX Subsystem and the SMPTE UHD-SDI TX Subsystem.

- The SMPTE UHD-SDI TX Subsystem transmits SDI data from the AXI4-Stream FIFO after the application programs the SMPTE UHD-SDI TX Subsystem sub-core registers, based on the received SDI stream and ST-352 payload packet data.

SDI Audio Pass-through

In this mode, up to 32 channels of audio can be extracted by the UHD-SDI Audio (Extract) IP. Audio, received from the RX data path, is embedded into the native SDI stream by the UHD-SDI Audio (Embed) IP. An AXI4-Stream FIFO is used for temporary storage between the UHD-SDI Audio (Extract) and the UHD-SDI Audio (Embed) IP.

Versal GT Bridge Controller for SDI

The SMPTE UHD-SDI TX and RX subsystems interacts with the GT Quad via a GT Bridge Controller for SDI. The Bridge IP is programmed in duplex mode in the VCK190 example design. This is because both RX and TX subsystems are required and use the maximum line rate of 11.88 Gb/s. Figure 5-44 below shows the IP settings for the SDI GT Bridge controller:

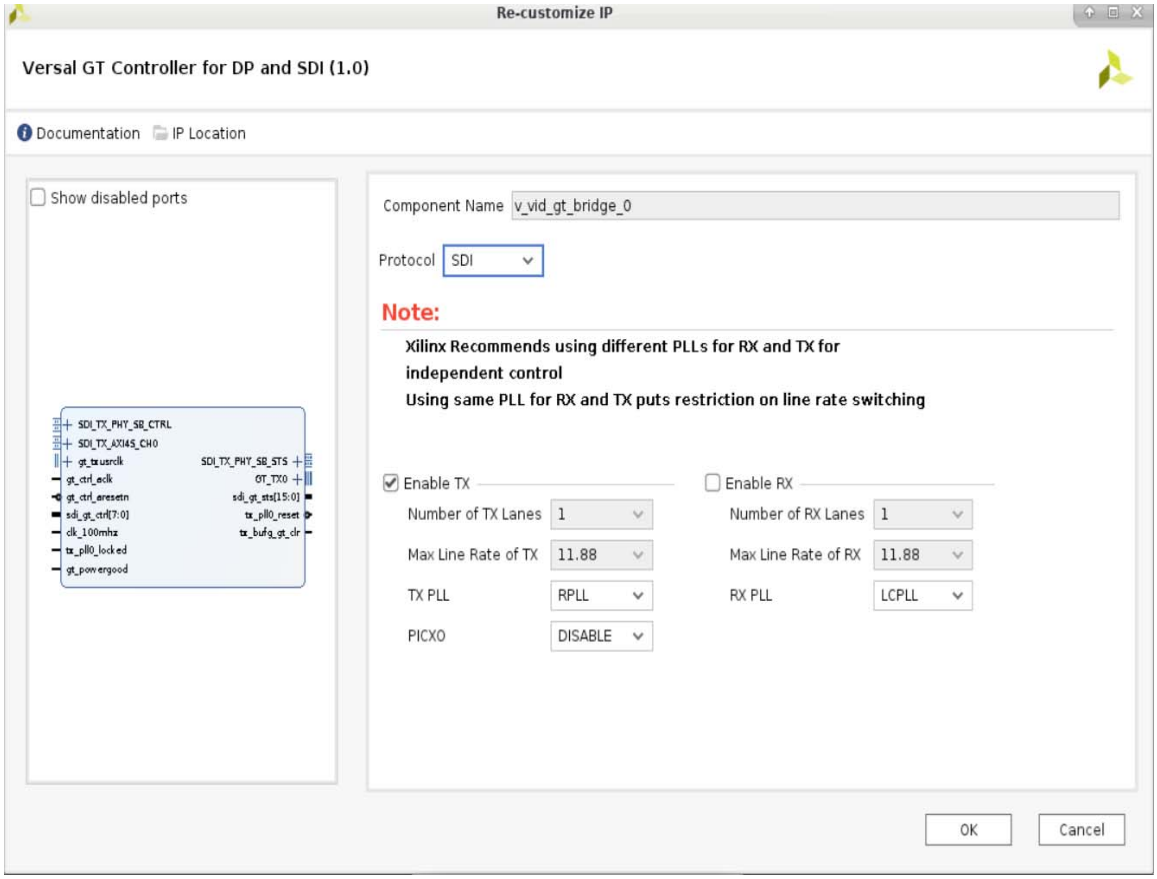

Figure 5-44: **SDI GT Bridge Controller**

Clocking

The UHD-SDI example design uses the v_vid_gt_bridge_v1_0 core to configure Versal GTY transceivers and provide options to select the transceiver reference clocking. LCPLL is allocated for the SMPTE UHD-SDI RX transceiver, and RPLL is allocated for the SMPTE UHD-SDI TX in this pass-through design.

Running the VCK190 Audio-Video Pass-Through Example

Design

1. Open the Vivado Design Suite and create a new project.
2. In the pop-up window, press **Next** five times.
3. Select the VCK190 Board.

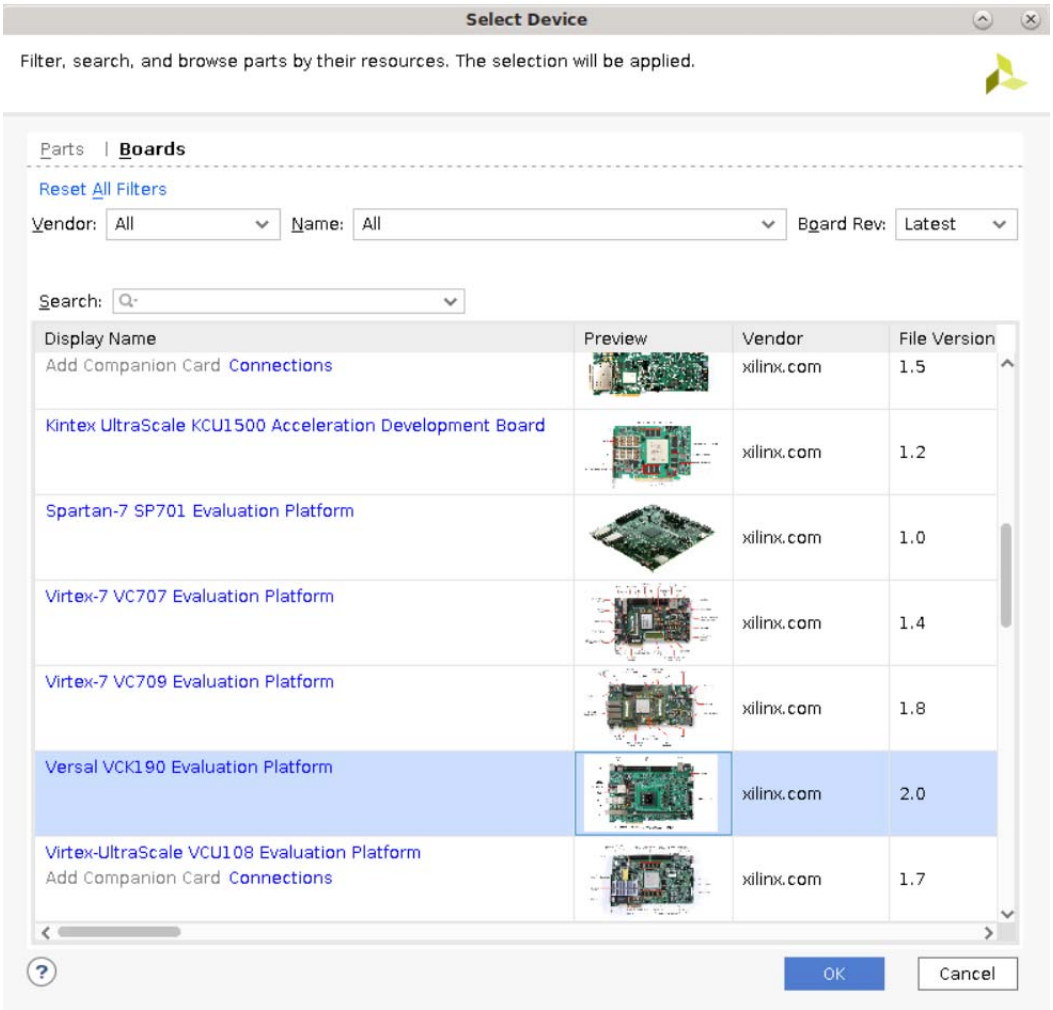

Figure 5-45: **Select the VCK190 Board**

4. Click **Finish**.
5. In the IP Catalog, under Video Connectivity, double-click **SMPTE UHD-SDI RX Subsystem**.

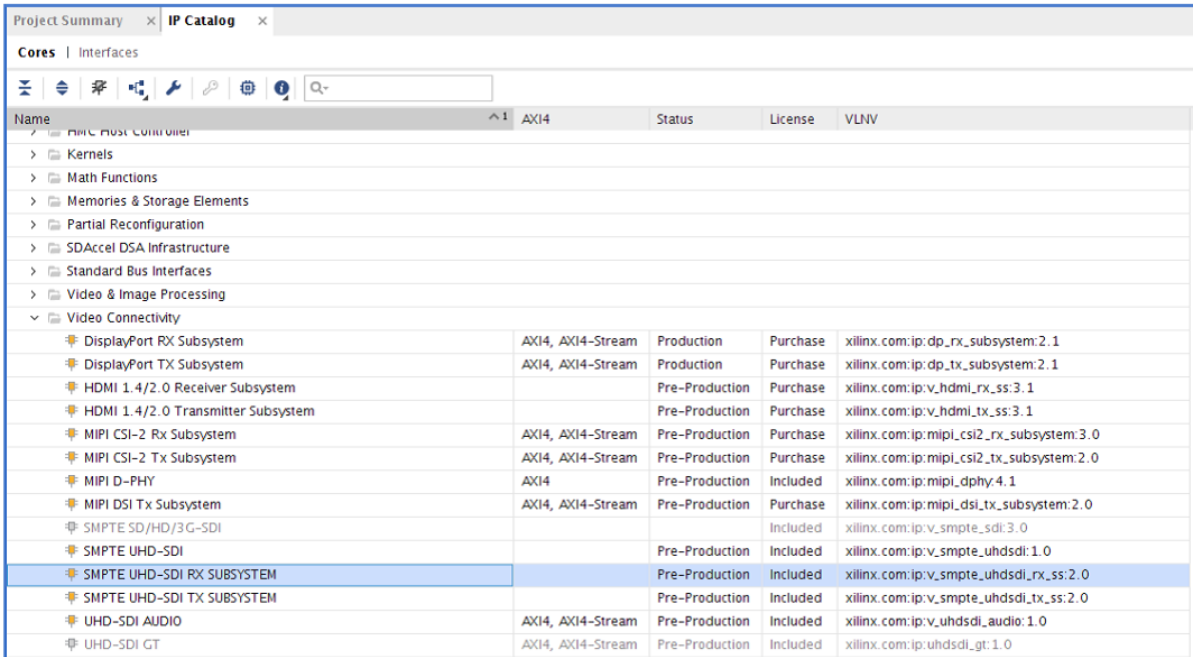

Figure 5-46: **Selecting the SMPTE UHD-SDI RX Subsystem**

Note: For the application example design flow, the IP configuration is based on options selected in the Application Example Design tab. You can rename the IP component name, which is used as the application example design project name.

6. Configure the SMPTE UHD-SDI RX Subsystem Application Example Design tab:
   a. Selecting **VCK190** as the Target Board.
   b. Select **Versal_Audio-Video Pass-Through** as Design Topology and available Board Part name.
   c. Click **OK**.

   The Generate Output Products dialog box appears.

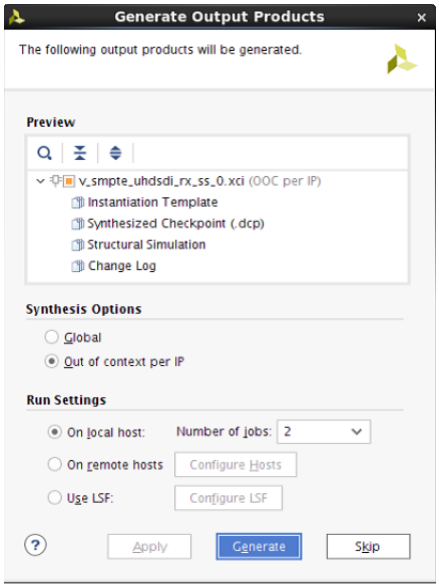

Figure 5-47: **Generate Output Products**

7. Click **Generate**.

   Note: Click **Skip** if you only want to generate the application example design.

8. Under Design source, right-click the **SMPTE UHD-SDI RX Subsystem** component, and then select **Open IP Example Design**.

9. Choose the target project location, and click **OK**. The IP integrator design is then generated and creates the Xilinx® Vitis™ software platform and generates an executable `.elf` file. You can choose to **Run Synthesis**, **Implementation**, or **Generate Device Image**. An overall system IP integrator block diagram of the VCK190-based application example design is displayed, as shown in Figure 5-48.

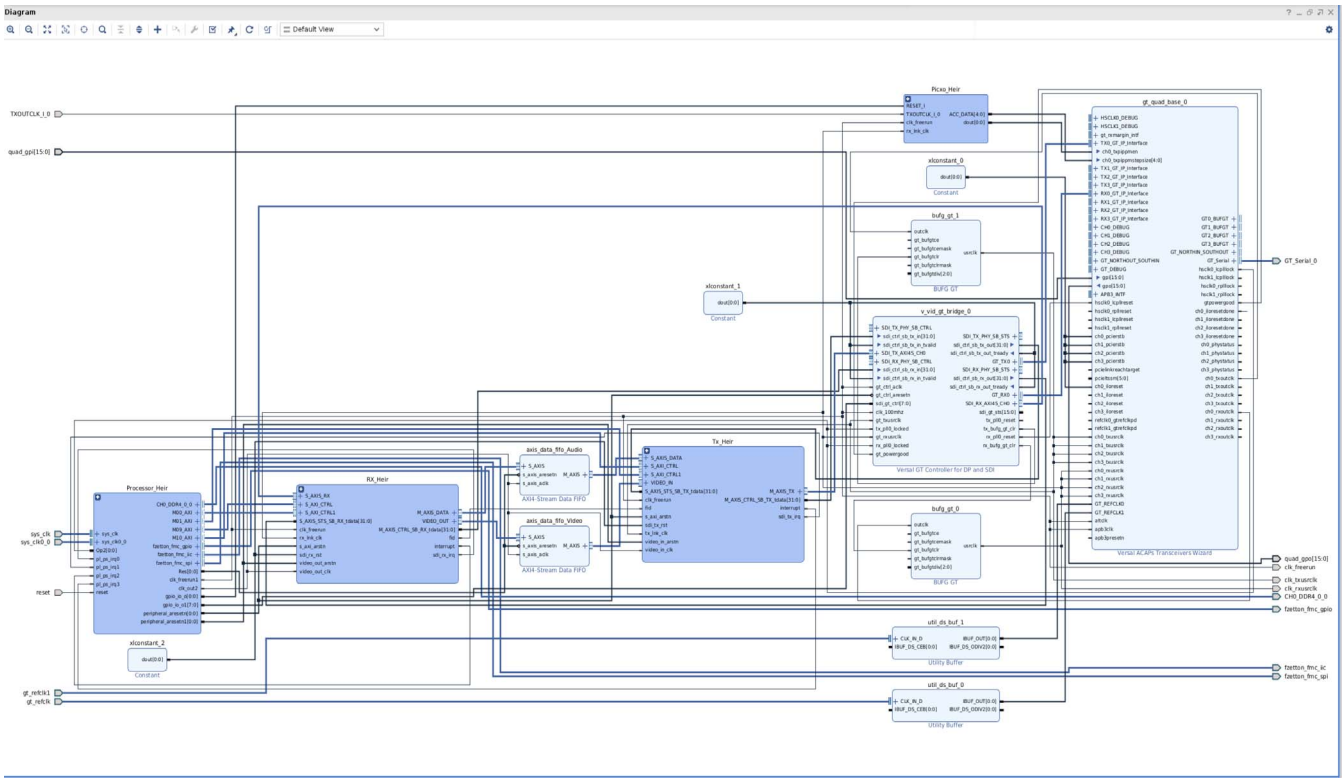

Figure 5-48: **IP Integrator Diagram of the vck190 Audio-Video Pass-Through Example Design**

Requirements

Hardware

The hardware requirements for this reference system are:

- Xilinx Versal VCK190 Evaluation Platform or Xilinx Versal VCK190 Evaluation Platform
- SDI source equipment
- SDI sink equipment
- Two HD-BNC to BNC converter cables
- Fidus TB-FMCH-12GSDI SDI FMC card

Software

This software requirements for this system are:

- Vivado Design Suite 2020.2 or later
- The Vitis software platform 2020.2 or later
- Software terminals (for example, Tera Term, HyperTerminal or PuTTY)

Setup

The reference design runs on the Xilinx Versal Evaluation Board (VCK190) or Xilinx Versal ES1 Evaluation Board (VCK190), using the SDI available on the board, as shown in the example in Figure 5-49.

Note: In the following procedure, the numbers in parentheses correspond to the call out numbers in Figure 5-49.

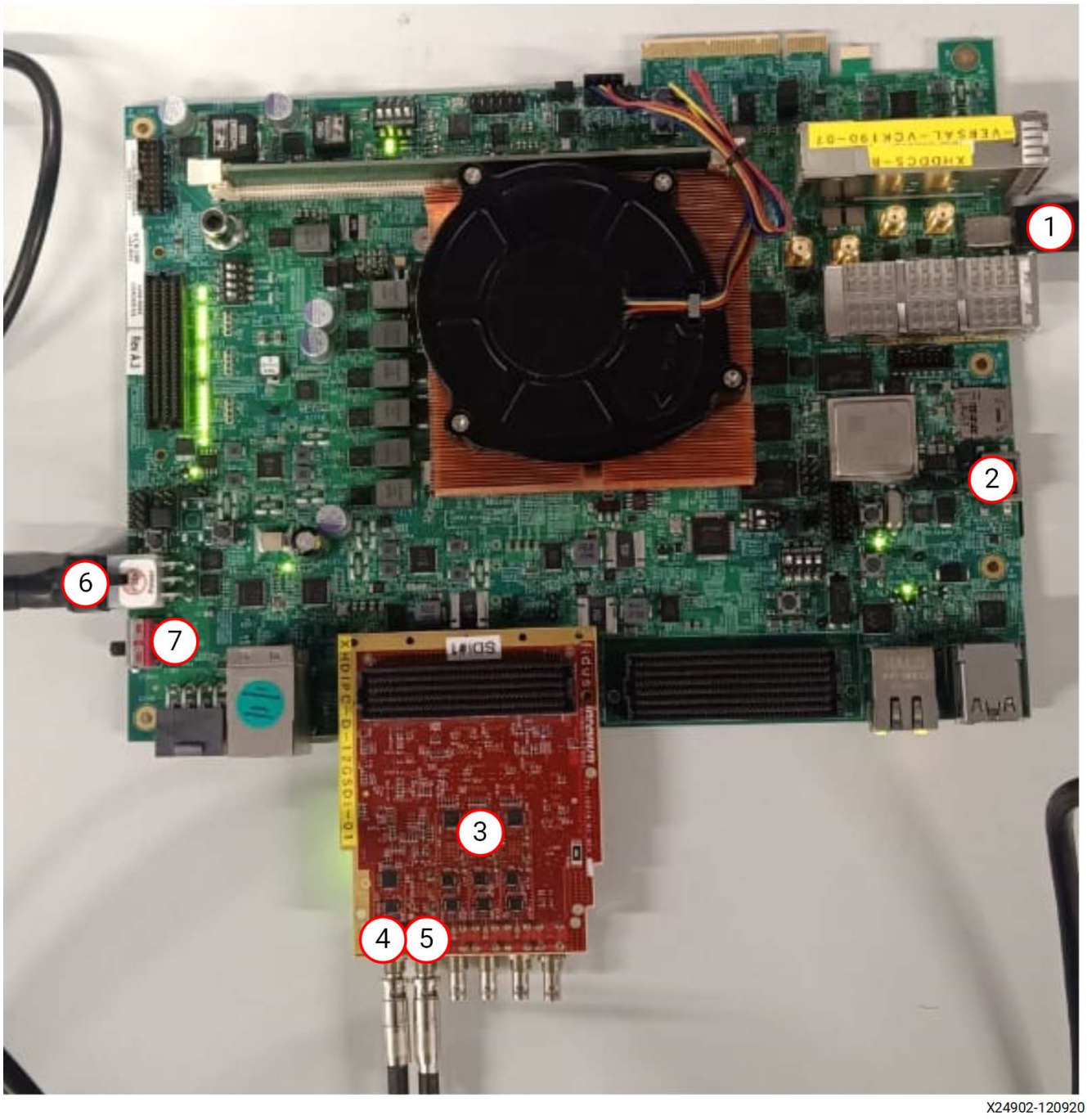

Figure 5-49: **VCK190 Board Setup**

1. Connect a USB cable from the host PC to the USB JTAG port (1). Ensure the appropriate device drivers are installed.
2. Connect a second USB cable from the host PC to the USB UART port (2). Ensure that the USB UART drivers described in Hardware have been installed.
3. Insert a Fidus FMC card into FMC slot (3).

4. Connect the HD-BNC of FMC Card (4) to the SDI source device.
5. Connect the HD-BNC of FMC Card (7) to the SDI sink device.
6. Connect the VCK190 board to a power supply slot (5).
7. Switch on the VCK190 board (6).
8. Make sure that the HW-VCK190 board revision is as selected in GUI interface.
9. Ensure that the SMPTE ST 352/Payload ID is enabled in the SDI Stream connected to the SDI input link of VCK190.
10. Start Tera Term or PuTTY to connect to the COM port interface 0 on the Host PC configured with 115200 bps, 8 bits, No parity, 1 stop bit, and no flow control.

Compiling Software on the Vitis Software Platform

The UHD-SDI application example design generates a `.elf` file automatically. Use the following procedure if you want to open the Vitis software platform project for UHD-SDI example design from the Vivado Design Suite:

1. In the Vivado Design Suite, click **Tools > Launch Vitis**.
1. Select **Exported location and workspace** as **<Proj Dir>/SW/xsdi_app** and click **OK**. This launches the Vitis software platform project and opens the project.

Running the Design on the Hardware

Use the following steps to run the BIT and ELF files on the hardware setup:

1. Connect the JTAG cable and USB-UART cable to the board.
2. Switch on the Board.
3. Open the board user interface GUI for Versal. The Enter Board Information tab opens, as shown in Figure 5-50.

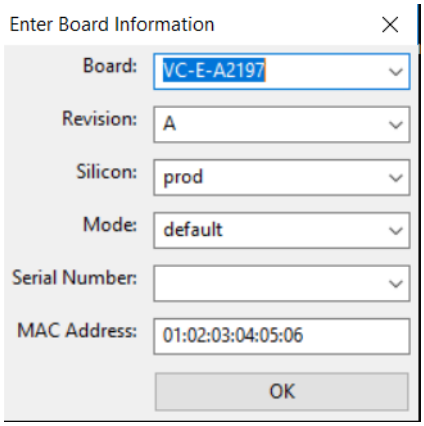

Figure 5-50: **Enter Board Information**

4. Fill in the information as per the board part used, and then click **OK**. This displays a new tab.
5. Click on the **Clock** tab, and then set the user1 FMC1 si570 frequency clock to **148.35** as shown in Figure 5-51 below:

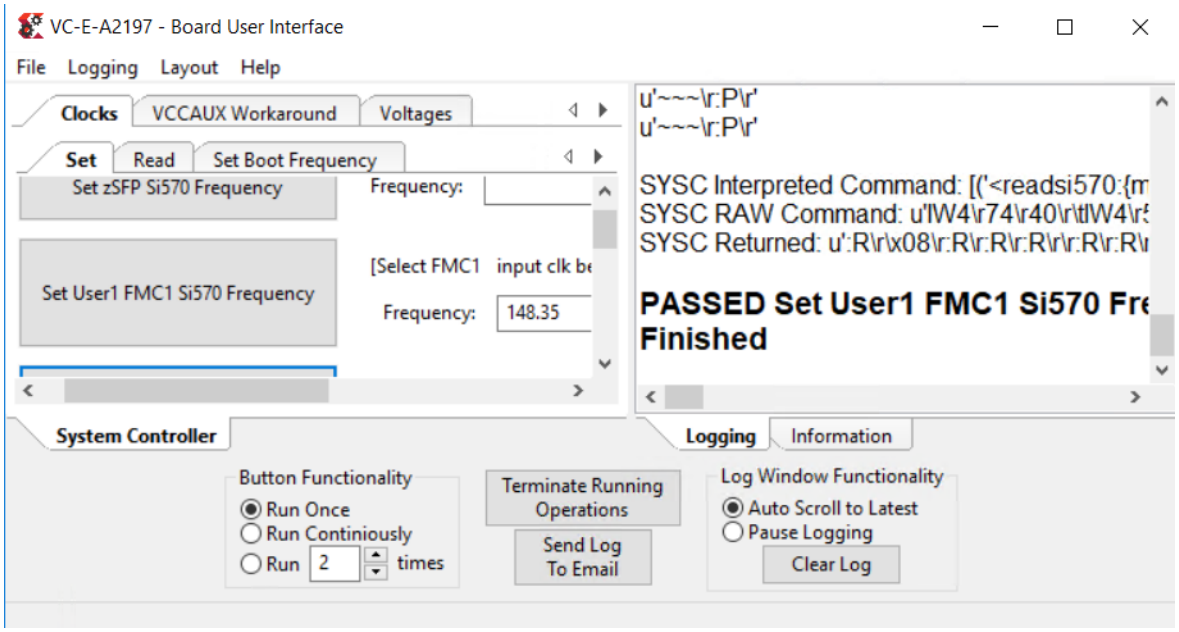

Figure 5-51: **Setting the Clock Frequency**

6. Select the **VCCAUX Workaround** tab, and then select the checkbox for **VCC_RAM (.78v)** as shown in Figure 5-52.

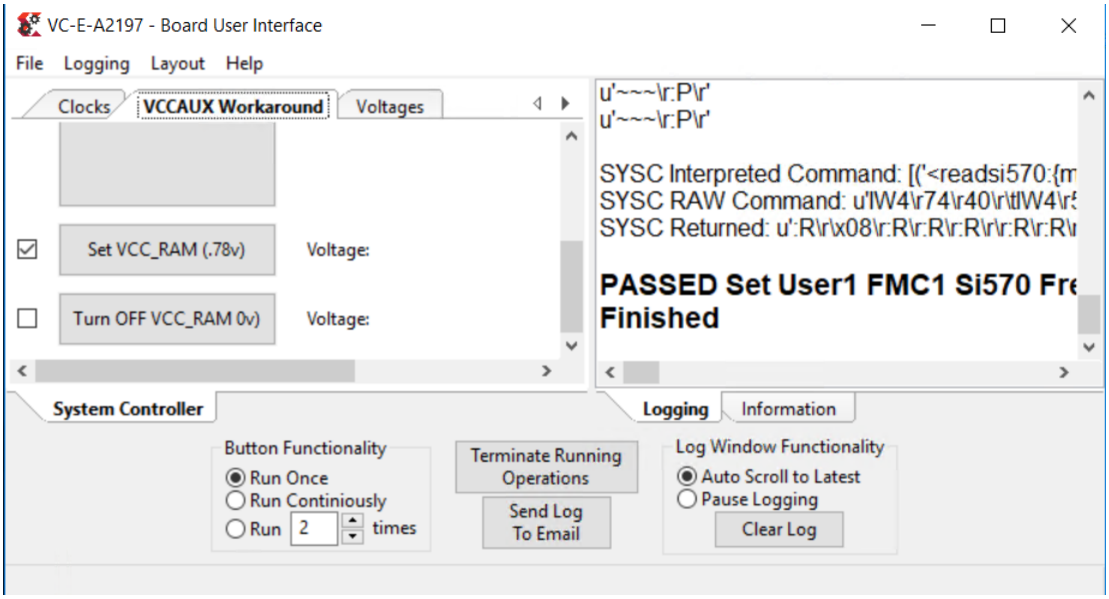

Figure 5-52: **Setting VCC_RAM**

7. Scroll to and select the **FMC** tab.
8. Select the checkbox for **Set Vadj (0.5 to 1.5V)**, and then enter a voltage value in the 0.5-1.5V range, as shown in Figure 5-53.

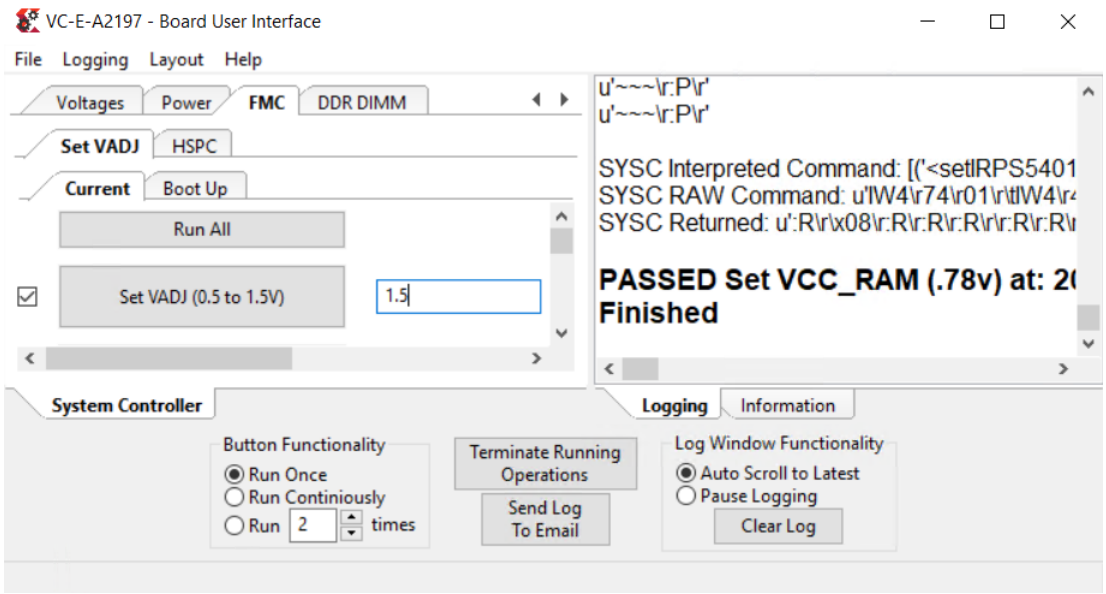

Figure 5-53: **Setting the VADJ Value**

9. Once the settings are passed, close the tabs.
10. Navigate to `<Component Name>_ex/imports`.
11. Start the Xilinx Software Debugger (XSDB) by using the command prompt to source XSDB from the build area.
12. Run the following command to program FPGA and to execute the application:

```
source xsdb.tcl
```

13. To observe the results:
    a. Make sure that the UART cable is connected to the board and the PC.
    b. Open Tera Term or PuTTY and configure the serial port (Interface 0) to 115200 baud with the default configuration. The UART console then displays SDI stream details on the console.

UART Console Screens

Main Menu

Figure 5-54 shows the initial UART console output along with the main menu options:

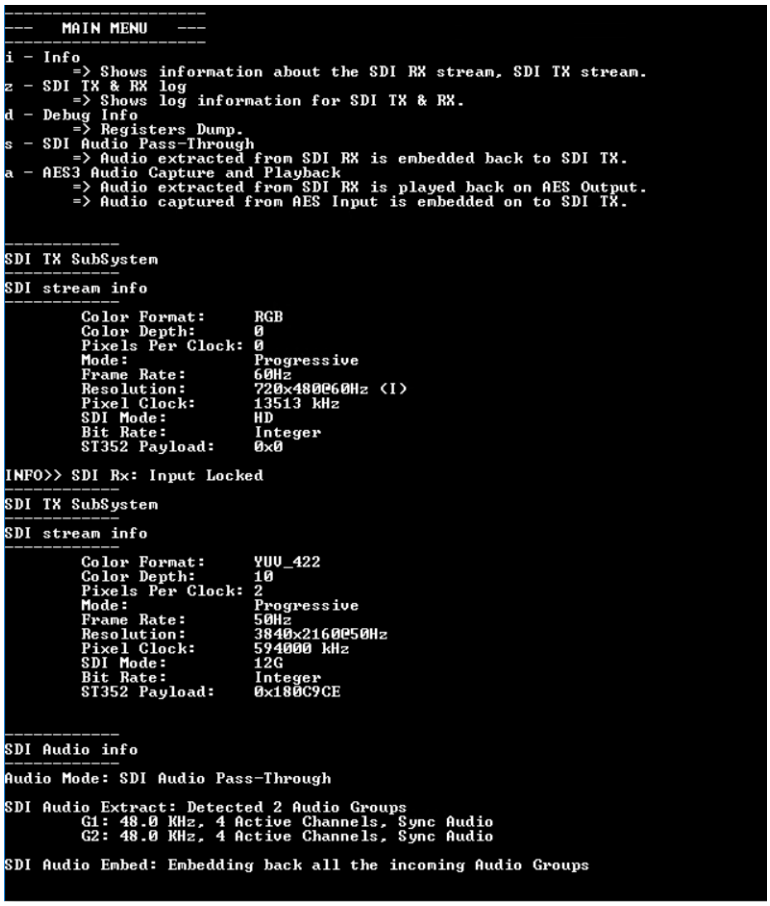

Figure 5-54: **UART Console Output and Menu Options**

Info Option

Press the **i** key to display the SDI TX subsystem information, as shown in Figure 5-55.

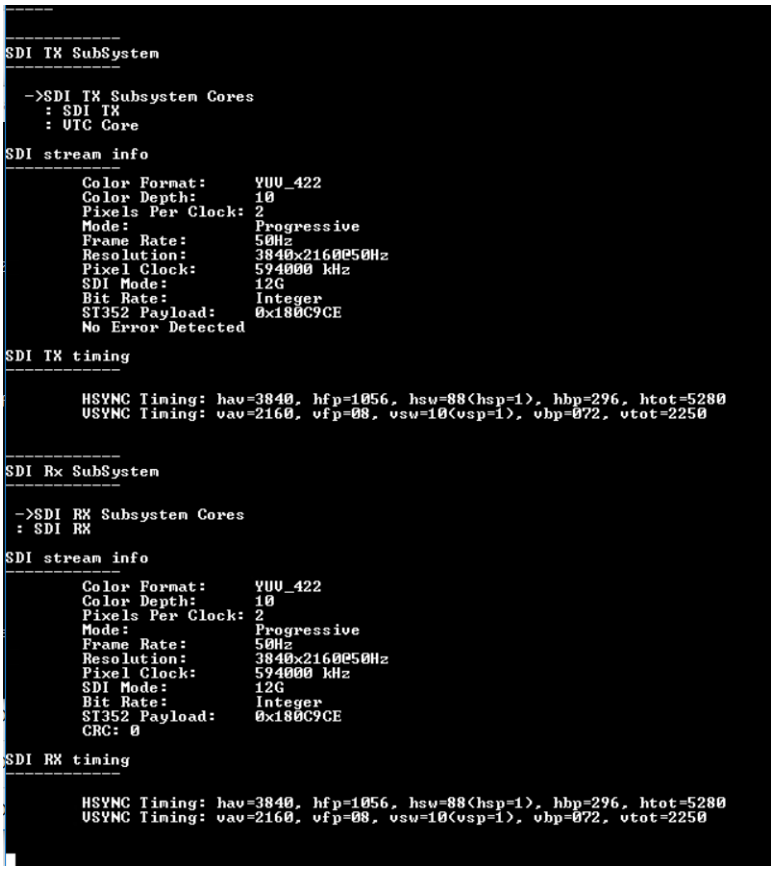

Figure 5-55: **UART Info Option**

SDI TX and RX Log Option

Press the **z** key to display SDI TX and RX log information, as shown in Figure 5-56.

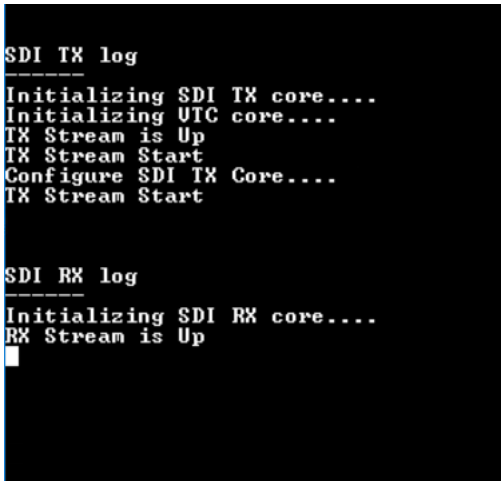

Figure 5-56: **UART SDI TX and RX Log**

Debug Option

Press the **z** key to display SDI TX Subsystem debug information, as shown in Figure 5-57.

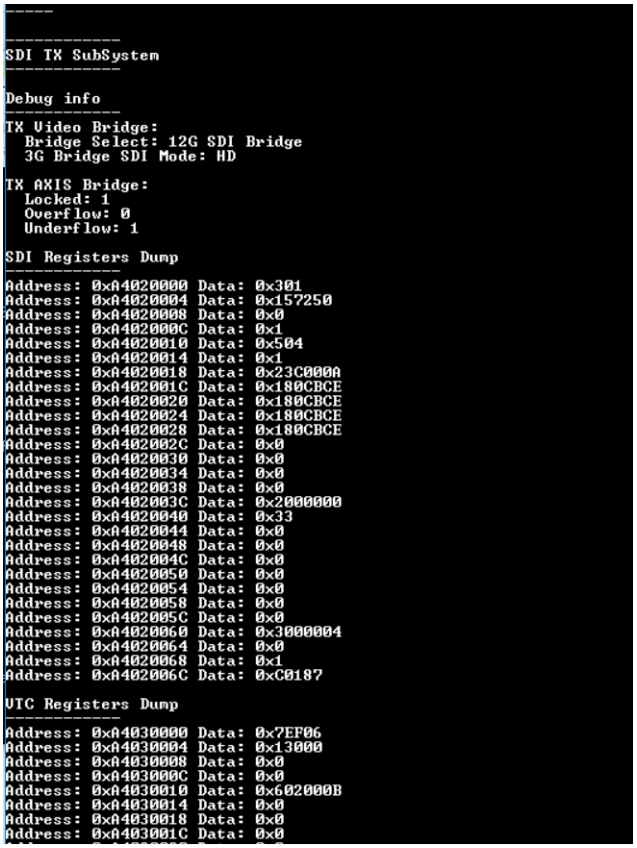

Figure 5-57: **UART Debug Information**

SDI Audio Option

Press the **a** key to display SDI Audio information, as shown in Figure 5-58.

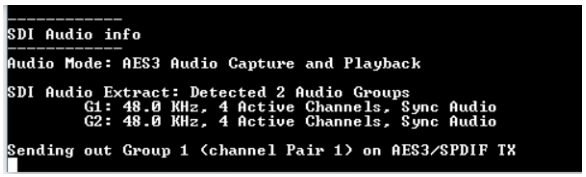

Figure 5-58: **UART SDI Audio Option**

Appendix A

Verification, Compliance, and Interoperability

The SMPTE UHD-SDI RX Subsystem has been verified using both simulation and hardware testing.

A highly parameterizable transaction-based simulation test suite has been used to verify the subsystem. The tests include:

- Different SDI standard.
- Different resolutions with different video timing parameters.
- Recovery from error conditions.
- Register read and write access.

Interoperability

The SMPTE UHD-SDI RX Subsystem has been tested using standard off-the-shelf SDI test equipment with a variety of UHD-SDI devices. It is compliant with the SMPTE SDI standards.

Hardware Validation

The SMPTE UHD-SDI RX Subsystem is tested in hardware for functionality, performance, and reliability using Xilinx® evaluation platforms. The SMPTE UHD-SDI RX Subsystem verification test suites for all possible modules are continuously being updated to increase test coverage across the range of possible parameters for each individual module.

The SMPTE UHD-SDI RX Subsystem has been validated using

- Zynq® UltraScale+™ MPSoC ZCU106 Evaluation Kit
- Kintex® UltraScale+ KCU116 Evaluation Kit
- Versal ACAP VCK190 Evaluation Kit v2.0 and ES v1.0

The SMPTE UHD-SDI RX Subsystem is tested with following devices:

- Phabrix QX 12G as Source and Sync device
- Phabrix R1000 as Source and Sync device
- Phabrix SX as Source and Sync device
- Omnitek Ultra 4K Tool box as Source and Sync device

Table A-1: **Tested Configurations**

Board	PLL Source	Sink Type	Model Number	Sink Firmware	Status
ZCU106 (GTH)	QPLL0/QPLL1	Phabrix	QX	ver 3.0	Pass
		Omnitek	Ultra 4K	ver 4.0.74.0	Pass
		Black Magic	Teranex Mini	NA	Pass
	CPLL	Phabrix	QX	ver 3.0	Pass
		Omnitek	Ultra 4K	ver 4.0.74.0	Pass
		Black Magic	Teranex Mini	NA	Pass
KCU116 (GTY)	QPLL0/QPLL1	Phabrix	QX	ver 3.0	Pass
		Omnitek	Ultra 4K	ver 4.0.74.0	Pass
		Black Magic	Teranex Mini	NA	Pass
	CPLL	Phabrix	QX	ver 3.0	Pass
		Omnitek	Ultra 4K	ver 4.0.74.0	Pass
		Black Magic	Teranex Mini	NA	Pass
VCK190(GTY)	LCPLL	Phabrix	QX	ver 3.0	Pass
	RPLL	Phabrix	QX	ver 3.0	Pass

Video Resolutions

A series of interoperability test scenarios, listed in Table A-2 to Table A-4, are validated for different resolutions. Test equipment is used to drive the SDI mode specific traffic and display the received SDI data. Figure A-1 shows the UHD-SDI validation setup using SDI Test equipment.

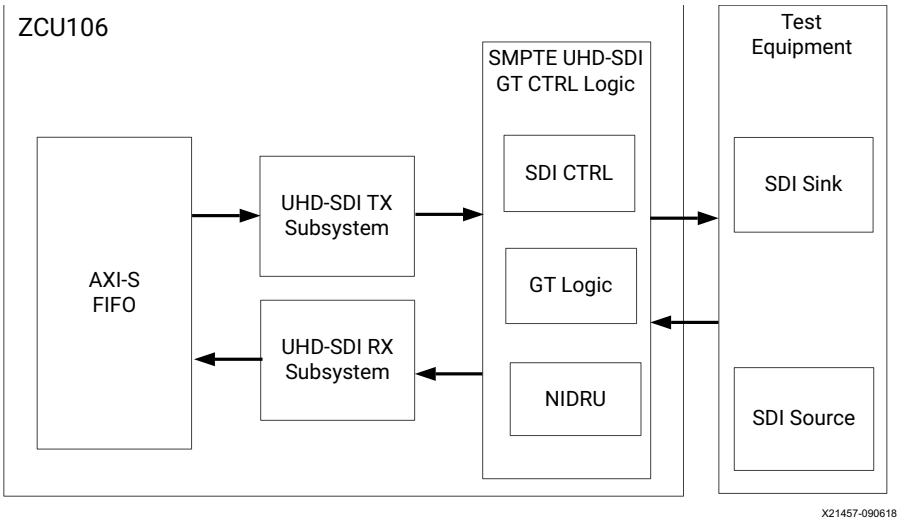

Figure A-1: **SMPTE UHD-SDI Test Equipment Setup**

Table A-2 shows the tested video resolutions using the Phabrix QX 12G as the SDI source and SDI sink.

Table A-2: **Tested Video Resolutions for YCbCr 4:2:2 at 10 Bits/Component**

SDI Mode	Resolution	Horizontal		Vertical		Frame Rate (Hz)
		Total	Active	Total	Active	
HD-SDI	1920x1080i50	2640	1920	1125	1080	50
	1920x1080i59.94	2200	1920	1125	1080	59.94
	1920x1080i60	2200	1920	1125	1080	60
	1280x720p25	3960	1280	750	720	25
	1280x720p29.97	3300	1280	750	720	29.97
	1280x720p30	3300	1280	750	720	30
	1280x720p50	1980	1280	750	720	50
	1280x720p59.94	1650	1280	750	720	59.94
	1280x720p60	1650	1280	750	720	60
	1920x1080p23.98	2750	1920	1125	1080	23.98
	1920x1080p24	2750	1920	1125	1080	24
	1920x1080p25	2640	1920	1125	1080	25
	1920x1080p29.97	2200	1920	1125	1080	29.97
	1920x1080p30	2200	1920	1125	1080	30
3G-SDI Level A	1920x1080p50	2640	1920	1125	1080	50
	1920x1080p59.94	2200	1920	1125	1080	59.94
	1920x1080p60	2200	1920	1125	1080	60
HD-SDI	2048x1080p23.98	2750	2048	1125	1080	23.98
	2048x1080p24	2750	2048	1125	1080	24
	2048x1080p25	2640	2048	1125	1080	25
	2048x1080p29.97	2200	2048	1125	1080	29.97
	2048x1080p30	2200	2048	1125	1080	30
3G-SDI Level A, 3G-SDI Level B	2048x1080p47.95	2750	2048	1125	1080	47.95
	2048x1080p48	2750	2048	1125	1080	48
	2048x1080p50	2640	2048	1125	1080	50
	2048x1080p59.94	2200	2048	1125	1080	59.94
	2048x1080p60	2200	2048	1125	1080	60
6G-SDI	3840x2160p23.98	5500	3840	2250	2160	23.98
	3840x2160p24	5500	3840	2250	2160	24
	3840x2160p25	5280	3840	2250	2160	25
	3840x2160p29.97	4400	3840	2250	2160	29.97
	3840x2160p30	4400	3840	2250	2160	30

Table A-2: **Tested Video Resolutions for YCbCr 4:2:2 at 10 Bits/Component *(Cont'd)***

SDI Mode	Resolution	Horizontal		Vertical		Frame Rate (Hz)
		Total	Active	Total	Active	
12G-SDI	3840x2160p50	5280	3840	2250	2160	50
	3840x2160p59.94	4400	3840	2250	2160	59.94
	3840x2160p60	4400	3840	2250	2160	60
6G-SDI	4096x2160p23.98	5500	4096	2200	2160	23.98
	4096x2160p24	5500	4096	2200	2160	24
	4096x2160p25	5280	4096	2250	2160	25
	4096x2160p29.97	4400	4096	2250	2160	29.97
	4096x2160p30	4400	4096	2250	2160	30
12G-SDI	4096x2160p47.95	5500	4096	2250	2160	47.95
	4096x2160p48	5500	4096	2250	2160	48
	4096x2160p50	5280	4096	2250	2160	50
	4096x2160p59.94	4400	4096	2250	2160	59.94
	4096x2160p60	4400	4096	2250	2160	60

Table A-3 shows the tested video resolutions using the Phabrix R1000/Phabrix SX as the SDI source and SDI sink.

Table A-3: **Tested Video Resolutions for YCbCr 4:2:2 at 10 Bits/Component**

SDI Mode	Resolution	Horizontal		Vertical		Frame Rate (Hz)
		Total	Active	Total	Active	
SD-SDI	625i50	864	720	625	576	50
HD-SDI	1920x1080sF23.98	2750	1920	1125	1080	23.98
	1920x1080sF24	2750	1920	1125	1080	23.98
	1920x1080sF25	2640	1920	1125	1080	23.98
	1920x1080sF29.97	2200	1920	1125	1080	29.97
	1920x1080sF30	2200	1920	1125	1080	30
3G-SDI Level A, 3G-SDI Level B	1920x1080p47.95	2750	1920	1125	1080	47.95
	1920x1080p48	2750	1920	1125	1080	48

Table A-4 shows the tested video resolutions using the Omnitek Ultra 4K Tool box as the SDI source and SDI sink.

Table A-4: **Tested Video Resolutions for YCbCr 4:2:2 at 10 Bits/Component**

SDI Mode	Resolution	Horizontal		Vertical		Frame Rate (Hz)
		Total	Active	Total	Active	
SD-HDI	720x576i50	-	720	-	576	50
HD-SDI	1920x1080i47.95	-	1920	-	1080	47.95
	1920x1080i48	-	1920	-	1080	48
	1920x1080i50	-	1920	-	1080	50
	1920x1080i59.94	-	1920	-	1080	59.94
	1920x1080i60	-	1920	-	1080	60
	2048x1080i47.95	-	2048	-	1080	47.95
	2048x1080i48	-	2048	-	1080	48
	2048x1080i50	-	2048	-	1080	50
	2048x1080i59.94	-	2048	-	1080	59.94
	2048x1080i60	-	2048	-	1080	60
	1280x720p25	-	1280	-	720	25
	1280x720p29.97	-	1280	-	720	29.97
	1280x720p30	-	1280	-	720	30
	1280x720p50	-	1280	-	720	50
	1280x720p59.94	-	1280	-	720	59.94
	1280x720p60	-	1280	-	720	60
	1920x1080p23.98	-	1920	-	1080	23.98
	1920x1080p24	-	1920	-	1080	24
	1920x1080p25	-	1920	-	1080	25
	1920x1080p29.97	-	1920	-	1080	29.97
	1920x1080p30	-	1920	-	1080	30
	1920x1080sF23.98	-	1920	-	1080	23.98
	1920x1080sF24	-	1920	-	1080	24
	1920x1080sF25	-	1920	-	1080	25
	1920x1080sF29.97	-	1920	-	1080	29.97
	1920x1080sF30	-	1920	-	1080	30
	2048x1080p23.98	-	2048	-	1080	23.98
	2048x1080p24	-	2048	-	1080	24
	2048x1080p25	-	2048	-	1080	25
	2048x1080p29.97	-	2048	-	1080	29.97

Table A-4: **Tested Video Resolutions for YCbCr 4:2:2 at 10 Bits/Component *(Cont'd)***

SDI Mode	Resolution	Horizontal		Vertical		Frame Rate (Hz)
		Total	Active	Total	Active	
HD-SDI	2048x1080p30	-	2048	-	1080	30
	2048x1080sF23.98	-	2048	-	1080	23.98
	2048x1080sF24	-	2048	-	1080	24
	2048x1080sF25	-	2048	-	1080	25
	2048x1080sF29.97	-	2048	-	1080	29.97
	2048x1080sF30	-	2048	-	1080	30
3G-A	1920x1080p47.95	-	1920	-	1080	47.95
	1920x1080p48	-	1920	-	1080	48
	1920x1080p50	-	1920	-	1080	50
	1920x1080p59.94	-	1920	-	1080	59.94
	1920x1080p60	-	1920	-	1080	60
	2048x1080p47.95	-	2048	-	1080	47.95
	2048x1080p48	-	2048	-	1080	48
	2048x1080p50	-	2048	-	1080	50
	2048x1080p59.94	-	2048	-	1080	59.94
	2048x1080p60	-	2048	-	1080	60
3G-B DL	1920x1080p47.95	-	1920	-	1080	47.95
	1920x1080p48	-	1920	-	1080	48
	1920x1080p50	-	1920	-	1080	50
	1920x1080p59.94	-	1920	-	1080	59.94
	1920x1080p60	-	1920	-	1080	60
	2048x1080p47.95	-	2048	-	1080	47.95
	2048x1080p48	-	2048	-	1080	48
	2048x1080p50	-	2048	-	1080	50
	2048x1080p59.94	-	2048	-	1080	59.94
	2048x1080p60	-	2048	-	1080	60
3G-B DS	1920x1080i47.95	-	1920	-	1080	47.95
	1920x1080i48	-	1920	-	1080	48
	1920x1080i50	-	1920	-	1080	50
	1920x1080i59.94	-	1920	-	1080	59.94
	1920x1080i60	-	1920	-	1080	60
	2048x1080i47.95	-	2048	-	1080	47.95
	2048x1080i48	-	2048	-	1080	48

Table A-4: **Tested Video Resolutions for YCbCr 4:2:2 at 10 Bits/Component *(Cont'd)***

SDI Mode	Resolution	Horizontal		Vertical		Frame Rate (Hz)
		Total	Active	Total	Active	
3G-B DS	2048x1080i50	-	2048	-	1080	50
	2048x1080i59.94	-	2048	-	1080	59.94
	2048x1080i60	-	2048	-	1080	60
	1920x1080sF23.98	-	1920	-	1080	23.98
	1920x1080sF24	-	1920	-	1080	24
	1920x1080sF25	-	1920	-	1080	25
	1920x1080sF29.97	-	1920	-	1080	29.97
	1920x1080sF30	-	1920	-	1080	30
	2048x1080sF23.98	-	2048	-	1080	23.98
	2048x1080sF24	-	2048	-	1080	24
	2048x1080sF25	-	2048	-	1080	25
	2048x1080sF29.97	-	2048	-	1080	29.97
	2048x1080sF30	-	2048	-	1080	30
6G-SDI	3840x2160p23.98	-	3840	-	2160	23.98
	3840x2160p24	-	3840	-	2160	24
	3840x2160p25	-	3840	-	2160	25
	3840x2160p29.97	-	3840	-	2160	29.97
	3840x2160p30	-	3840	-	2160	30
	4096x2160p23.98	-	4096	-	2160	23.98
	4096x2160p24	-	4096	-	2160	24
	4096x2160p25	-	4096	-	2160	25
	4096x2160p29.97	-	4096	-	2160	29.97
	4096x2160p30	-	4096	-	2160	30
12G-SDI	3840x2160p47.95	-	3840	-	2160	47.95
	3840x2160p48	-	3840	-	2160	48
	3840x2160p50	-	3840	-	2160	50
	3840x2160p59.94	-	3840	-	2160	59.94
	3840x2160p60	-	3840	-	2160	60
	4096x2160p47.95	-	4096	-	2160	47.95
	4096x2160p48	-	4096	-	2160	48
	4096x2160p50	-	4096	-	2160	50
	4096x2160p59.94	-	4096	-	2160	59.94
	4096x2160p60	-	4096	-	2160	60

Table A-5 shows tested video resolutions for YCbCr 4:4:4 at 10 Bits/Component.

Table A-5: **Tested Video Resolutions for YCbCr 4:4:4 at 10 Bits/Component**

SDI Mode	Video Resolution	Horizontal	Vertical	Scan Type/Frame Rate
		Active	Active	
12G	3840x2160	3840	2160	p23.98
12G	3840x2160	3840	2160	p24
12G	3840x2160	3840	2160	p25
12G	3840x2160	3840	2160	p29.97
12G	3840x2160	3840	2160	p30
12G	4096x2160	4096	2160	p23.98
12G	4096x2160	4096	2160	p24
12G	4096x2160	4096	2160	p25
12G	4096x2160	4096	2160	p29.97
12G	4096x2160	4096	2160	p30
3G B-DL	1920x1080	1920	1080	p23.98
3G B-DL	1920x1080	1920	1080	p24
3G B-DL	1920x1080	1920	1080	p25
3G B-DL	1920x1080	1920	1080	p29.97
3G B-DL	1920x1080	1920	1080	p30
3G B-DL	1920x1080	1920	1080	SF23.98
3G B-DL	1920x1080	1920	1080	SF24
3G B-DL	1920x1080	1920	1080	SF25
3G B-DL	1920x1080	1920	1080	SF29.97
3G B-DL	1920x1080	1920	1080	SF30
3G B-DL	1920x1080	1920	1080	i47.95
3G B-DL	1920x1080	1920	1080	i48
3G B-DL	1920x1080	1920	1080	i50
3G B-DL	1920x1080	1920	1080	i59.94
3G B-DL	1920x1080	1920	1080	i60
3G B-DL	2048x1080	2048	1080	p23.98
3G B-DL	2048x1080	2048	1080	p24
3G B-DL	2048x1080	2048	1080	p25
3G B-DL	2048x1080	2048	1080	p29.97
3G B-DL	2048x1080	2048	1080	p30
3G B-DL	2048x1080	2048	1080	SF23.98
3G B-DL	2048x1080	2048	1080	SF24

Table A-5: **Tested Video Resolutions for YCbCr 4:4:4 at 10 Bits/Component *(Cont'd)***

SDI Mode	Video Resolution	Horizontal	Vertical	Scan Type/Frame Rate
		Active	Active	
3G B-DL	2048x1080	2048	1080	SF25
3G B-DL	2048x1080	2048	1080	SF29.97
3G B-DL	2048x1080	2048	1080	SF30
3G B-DL	2048x1080	2048	1080	i47.95
3G B-DL	2048x1080	2048	1080	i48
3G B-DL	2048x1080	2048	1080	i50
3G B-DL	2048x1080	2048	1080	i59.94
3G B-DL	2048x1080	2048	1080	i60
3G A	1280x720	1280	720	p25
3G A	1280x720	1280	720	p29.97
3G A	1280x720	1280	720	p30
3G A	1280x720	1280	720	p50
3G A	1280x720	1280	720	p59.94
3G A	1280x720	1280	720	p60
3G A	1280x720	1280	720	p60
3G A	1920x1080	1920	1080	sF23.98
3G A	1920x1080	1920	1080	sF24
3G A	1920x1080	1920	1080	sF25
3G A	1920x1080	1920	1080	sF29.97
3G A	1920x1080	1920	1080	sF30
3G A	1920x1080	1920	1080	p23.98
3G A	1920x1080	1920	1080	p24
3G A	1920x1080	1920	1080	p25
3G A	1920x1080	1920	1080	p29.97
3G A	1920x1080	1920	1080	p30
3G A	1920x1080	1920	1080	i47.95
3G A	1920x1080	1920	1080	i48
3G A	1920x1080	1920	1080	i50
3G A	1920x1080	1920	1080	i59.94
3G A	1920x1080	1920	1080	i60
3G A	2048x1080	2048	1080	p23.98
3G A	2048x1080	2048	1080	p24
3G A	2048x1080	2048	1080	p25
3G A	2048x1080	2048	1080	p29.97

Table A-5: **Tested Video Resolutions for YCbCr 4:4:4 at 10 Bits/Component *(Cont'd)***

SDI Mode	Video Resolution	Horizontal	Vertical	Scan Type/Frame Rate
		Active	Active	
3G A	2048x1080	2048	1080	SF23.98
3G A	2048x1080	2048	1080	SF24
3G A	2048x1080	2048	1080	SF25
3G A	2048x1080	2048	1080	SF29.97
3G A	2048x1080	2048	1080	SF30
3G A	2048x1080	2048	1080	i49.95
3G A	2048x1080	2048	1080	i48
3G A	2048x1080	2048	1080	i50
3G A	2048x1080	2048	1080	i59.94
3G A	2048x1080	2048	1080	i60

Table A-6: **HFR Video Resolutions Tested in Simulation for 10 Bits/Component for Native SDI mode**

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
6G	YCbCr 422	1920x1080p120	1920	1080	120
	YCbCr 422	1920x1080p119.88	1920	1080	119.88
	YCbCr 422	1920x1080p100	1920	1080	100
	YCbCr 420	1920x1080p120	1920	1080	120
	YCbCr 420	1920x1080p119.88	1920	1080	119.88
	YCbCr 420	1920x1080p100	1920	1080	100
	YCbCr 422	2048x1080p120	2048	1080	120
	YCbCr 422	2048x1080p119.88	2048	1080	119.88
	YCbCr 422	2048x1080p100	2048	1080	100
12G	YCbCr 444	1920x1080p120	1920	1080	120
	YCbCr 444	1920x1080p119.88	1920	1080	119.88
	YCbCr 444	1920x1080p100	1920	1080	100
	YCbCr 444	2048x1080p120	2048	1080	120
	YCbCr 444	2048x1080p119.88	2048	1080	119.88
	YCbCr 444	2048x1080p100	2048	1080	100

Table A-7: **HFR Video Resolutions Tested in Simulation for 10 Bits/Component in AXI- stream interface Sub-system configuration:**

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
6G	YCbCr 422	1920x1080p120	1920	1080	120
	YCbCr 422	1920x1080p119.88	1920	1080	119.88
	YCbCr 422	1920x1080p100	1920	1080	100
	YCbCr 422	2048x1080p120	2048	1080	120
	YCbCr 422	2048x1080p119.88	2048	1080	119.88
	YCbCr 422	2048x1080p100	2048	1080	100
12G	YCbCr 444	1920x1080p120	1920	1080	120
	YCbCr 444	1920x1080p119.88	1920	1080	119.88
	YCbCr 444	1920x1080p100	1920	1080	100
	YCbCr 444	2048x1080p120	2048	1080	120
	YCbCr 444	2048x1080p119.88	2048	1080	119.88
	YCbCr 444	2048x1080p100	2048	1080	100

Table A-8: **Non-HFR Tested Video Resolutions for 12 Bits/Component**

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
3GB	RGB	1920x1080p30	1920	1080	30
	RGB	1920x1080p29.97	1920	1080	29.97
	RGB	1920x1080p25	1920	1080	25
	RGB	1920x1080p24	1920	1080	24
	RGB	1920x1080p23.94	1920	1080	23.94
	RGB	1920x1080psF30	1920	1080	30
	RGB	1920x1080psF29.97	1920	1080	29.97
	RGB	1920x1080psF25	1920	1080	25
	RGB	1920x1080psF24	1920	1080	24
	RGB	1920x1080psF23.94	1920	1080	23.94
	RGB	1920x1080i60	1920	1080	60
	RGB	1920x1080pi59.97	1920	1080	59.97
	RGB	1920x1080i50	1920	1080	50
	YCbCr 444	1920x1080p30	1920	1080	30
	YCbCr 444	1920x1080p29.97	1920	1080	29.97
	YCbCr 444	1920x1080p25	1920	1080	25
	YCbCr 444	1920x1080p25	1920	1080	24
	YCbCr 444	1920x1080p24	1920	1080	23.94
	YCbCr 444	1920x1080psF30	1920	1080	30
	YCbCr 444	1920x1080psF29.97	1920	1080	29.97
	YCbCr 444	1920x1080psF25	1920	1080	25
	YCbCr 444	1920x1080psF24	1920	1080	24
	YCbCr 444	1920x1080p23.94	1920	1080	23.94
	YCbCr 444	1920x1080i60	1920	1080	60
	YCbCr 444	1920x1080i59.94	1920	1080	59.94
	YCbCr 444	1920x1080i50	1920	1080	50
	YCbCr 422	1920x1080p30	1920	1080	30
	YCbCr 422	1920x1080p29.97	1920	1080	29.97
	YCbCr 422	1920x1080p25	1920	1080	25
	YCbCr 422	1920x1080p24	1920	1080	24
	YCbCr 422	1920x1080p23.94	1920	1080	23.94

Table A-8: **Non-HFR Tested Video Resolutions for 12 Bits/Component *(Cont'd)***

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
3GB	YCbCr 422	1920x1080psF30	1920	1080	30
	YCbCr 422	1920x1080psF29.97	1920	1080	29.97
	YCbCr 422	1920x1080psF25	1920	1080	25
	YCbCr 422	1920x1080psF24	1920	1080	24
	YCbCr 422	1920x1080psF23.94	1920	1080	23.94
	YCbCr 422	1920x1080i60	1920	1080	60
	YCbCr 422	1920x1080i59.97	1920	1080	59.97
	YCbCr 422	1920x1080i50	1920	1080	50
3GA	RGB	1920x1080i60	1920	1080	60
	RGB	1920x1080i59.97	1920	1080	59.97
	RGB	1920x1080i50	1920	1080	50
	RGB	1920x1080p30	1920	1080	30
	RGB	1920x1080p29.97	1920	1080	29.97
	RGB	1920x1080p25	1920	1080	25
	RGB	1920x1080p24	1920	1080	24
	RGB	1920x1080p23.94	1920	1080	23.94
	YCbCr 444	1920x1080i60	1920	1080	60
	YCbCr 444	1920x1080i59.94	1920	1080	59.97
	YCbCr 444	1920x1080i50	1920	1080	50
	YCbCr 444	1920x1080p30	1920	1080	30
	YCbCr 444	1920x1080p29.97	1920	1080	29.97
	YCbCr 444	1920x1080p25	1920	1080	25
	YCbCr 444	1920x1080p24	1920	1080	24
	YCbCr 444	1920x1080p23.94	1920	1080	23.94
	YCbCr 422	1920x1080i60	1920	1080	60
	YCbCr 422	1920x1080i59.97	1920	1080	59.97
	YCbCr 422	1920x1080i50	1920	1080	50
	YCbCr 422	1920x1080p30	1920	1080	30
	YCbCr 422	1920x1080p29.97	1920	1080	29.97
	YCbCr 422	1920x1080p25	1920	1080	25
	YCbCr 422	1920x1080p24	1920	1080	24
	YCbCr 422	1920x1080p23.94	1920	1080	23.94
	YCbCr 422	1920x1080psF30	1920	1080	30
	YCbCr 422	1920x1080psF29.97	1920	1080	29.97

Table A-8: **Non-HFR Tested Video Resolutions for 12 Bits/Component *(Cont'd)***

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
3GA	YCbCr 422	1920x1080psF25	1920	1080	25
	YCbCr 422	1920x1080psF24	1920	1080	24
	YCbCr 422	1920x1080psF23.94	1920	1080	23.94
6G	RGB	1920x1080p60	1920	1080	60
	RGB	1920x1080p59.97	1920	1080	59.97
	RGB	1920x1080p50	1920	1080	50
	RGB	2048x1080p60	2048	1080	60
	RGB	2048x1080p59.97	2048	1080	59.97
	RGB	2048x1080p50	2048	1080	50
	RGB	2048x1080p48	2048	1080	48
	RGB	2048x1080p47.95	2048	1080	47.95
	YCbCr 444	1920x1080p60	1920	1080	60
	YCbCr 444	1920x1080p59.97	1920	1080	59.97
	YCbCr 444	1920x1080p50	1920	1080	50
	YCbCr 444	2048x1080p60	2048	1080	60
	YCbCr 444	2048x1080p59.97	2048	1080	59.97
	YCbCr 444	2048x1080p50	2048	1080	50
	YCbCr 444	2048x1080p48	2048	1080	48
	YCbCr 444	2048x1080p47.95	2048	1080	47.95
	YCbCr 422	1920x1080p60	1920	1080	60
	YCbCr 422	1920x1080p59.97	1920	1080	59.97
	YCbCr 422	1920x1080p50	1920	1080	50
	YCbCr 422	2048x1080p60	2048	1080	60
	YCbCr 422	2048x1080p59.97	2048	1080	59.97
	YCbCr 422	2048x1080p50	2048	1080	50
	YCbCr 422	2048x1080p48	2048	1080	48
	YCbCr 422	2048x1080p47.95	2048	1080	47.95
12G	RGB	3840x2160p30	3840	2160	30
	RGB	3840x2160p29.97	3840	2160	29.97
	RGB	3840x2160p25	3840	2160	25
	RGB	3840x2160p24	3840	2160	24
	RGB	3840x2160p23.94	3840	2160	23.94
	RGB	4096x2160p30	4096	2160	30
	RGB	4096x2160p29.97	4096	2160	29.97

Table A-8: **Non-HFR Tested Video Resolutions for 12 Bits/Component *(Cont'd)***

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
12G	RGB	4096x2160p25	4096	2160	25
	RGB	4096x2160p24	4096	2160	24
	RGB	4096x2160p23.94	4096	2160	23.94
	YCbCr 444	3840x2160p30	3840	2160	30
	YCbCr 444	3840x2160p29.97	3840	2160	29.97
	YCbCr 444	3840x2160p25	3840	2160	25
	YCbCr 444	3840x2160p24	3840	2160	24
	YCbCr 444	3840x2160p23.94	3840	2160	23.94
	YCbCr 444	4096x2160p30	4096	2160	30
	YCbCr 444	4096x2160p29.97	4096	2160	29.97
	YCbCr 444	4096x2160p25	4096	2160	25
	YCbCr 444	4096x2160p24	4096	2160	24
	YCbCr 444	4096x2160p23.94	4096	2160	23.94
	YCbCr 422	3840x2160p30	3840	2160	30
	YCbCr 422	3840x2160p29.97	3840	2160	29.97
	YCbCr 422	3840x2160p25	3840	2160	25
	YCbCr 422	3840x2160p24	3840	2160	24
	YCbCr 422	3840x2160p23.94	3840	2160	23.94
	YCbCr 422	4096x2160p30	4096	2160	30
	YCbCr 422	4096x2160p29.97	4096	2160	29.97
	YCbCr 422	4096x2160p25	4096	2160	25
	YCbCr 422	4096x2160p24	4096	2160	24
	YCbCr 422	4096x2160p23.94	4096	2160	23.94
	YCbCr 420	3840x2160p30	3840	2160	30
	YCbCr 420	3840x2160p29.97	3840	2160	29.97
	YCbCr 420	3840x2160p25	3840	2160	25
	YCbCr 420	3840x2160p24	3840	2160	24
	YCbCr 420	3840x2160p23.94	3840	2160	23.94
12G	RGB	1920x1080p120	1920	1080	120
	RGB	1920x1080p119.88	1920	1080	119.88
	RGB	1920x1080p100	1920	1080	100
	YCbCr 444	1920x1080p120	1920	1080	120
	YCbCr 444	1920x1080p119.88	1920	1080	119.88
	YCbCr 444	1920x1080p100	1920	1080	100

Table A-8: **Non-HFR Tested Video Resolutions for 12 Bits/Component *(Cont'd)***

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
12G	RGB	2048x1080p120	2048	1080	120
	RGB	2048x1080p119.88	2048	1080	119.88
	RGB	2048x1080p100	2048	1080	100
	YCbCr 444	2048x1080p120	2048	1080	120
	YCbCr 444	2048x1080p119.88	2048	1080	119.88
	YCbCr 444	2048x1080p100	2048	1080	100
	YCbCr 422	1920x1080p120	1920	1080	120
	YCbCr 422	1920x1080p119.88	1920	1080	119.88
	YCbCr 422	1920x1080p100	1920	1080	100
	YCbCr 420	1920x1080p120	1920	1080	120
	YCbCr 420	1920x1080p119.88	1920	1080	119.88
	YCbCr 420	1920x1080p100	1920	1080	100
	YCbCr 422	2048x1080p120	2048	1080	120
	YCbCr 422	2048x1080p119.88	2048	1080	119.88
	YCbCr 422	2048x1080p100	2048	1080	100
	YCbCr 420	2048x1080p120	2048	1080	120
	YCbCr 420	2048x1080p119.88	2048	1080	119.88
	YCbCr 420	2048x1080p100	2048	1080	100

Table A-9: **HFR Tested Video Resolutions for 12 Bits/Component for Native SDI Mode**

SDI Mode	Color Format	Resolution	Horizontal Active	Vertical Active	Frame Rate (Hz)
12G	RGB	1920x1080p120	1920	1080	120
	RGB	1920x1080p119.88	1920	1080	119.88
	RGB	1920x1080p100	1920	1080	100
	YCbCr 444	1920x1080p120	1920	1080	120
	YCbCr 444	1920x1080p119.88	1920	1080	119.88
	YCbCr 444	1920x1080p100	1920	1080	100
	RGB	2048x1080p120	2048	1080	120
	RGB	2048x1080p119.88	2048	1080	119.88
	RGB	2048x1080p100	2048	1080	100
	YCbCr 444	2048x1080p120	2048	1080	120
	YCbCr 444	2048x1080p119.88	2048	1080	119.88
	YCbCr 444	2048x1080p100	2048	1080	100
	YCbCr 422	1920x1080p120	1920	1080	120
	YCbCr 422	1920x1080p119.88	1920	1080	119.88
	YCbCr 422	1920x1080p100	1920	1080	100
	YCbCr 420	1920x1080p120	1920	1080	120
	YCbCr 420	1920x1080p119.88	1920	1080	119.88
	YCbCr 420	1920x1080p100	1920	1080	100
	YCbCr 422	2048x1080p120	2048	1080	120
	YCbCr 422	2048x1080p119.88	2048	1080	119.88
	YCbCr 422	2048x1080p100	2048	1080	100
	YCbCr 420	2048x1080p120	2048	1080	120
	YCbCr 420	2048x1080p119.88	2048	1080	119.88
	YCbCr 420	2048x1080p100	2048	1080	100

Debugging

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

Finding Help on Xilinx.com

To help in the design and debug process when using the SMPTE UHD-SDI RX, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the SMPTE UHD-SDI RX. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

The Solution Center specific to the SMPTE UHD-SDI RX core is listed below.

- Xilinx Video Solution Center

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product.

Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

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

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

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

Master Answer Record for the SMPTE UHD-SDI RX

AR# 68766

Technical Support

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

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

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

Debug Tools

There are many tools available to address SMPTE UHD-SDI 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 with the logic debug IP cores, including:

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

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 6].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Check that MMCM lock and PLL lock signal(s) are asserted.
- Verify the IO pin planning and XDC constraints.
- Follow the recommended reset sequence.
- Verify all clocks are connected and are with expected frequencies.
- Enable AXI4 Lite based register interface to get core status and control.
- Make sure serial line trace lengths are equal.
- Verify the FMC_VADJ voltage to 1.8V in case of FMC card usage.

Figure B-1 shows the steps to perform a hardware debug.

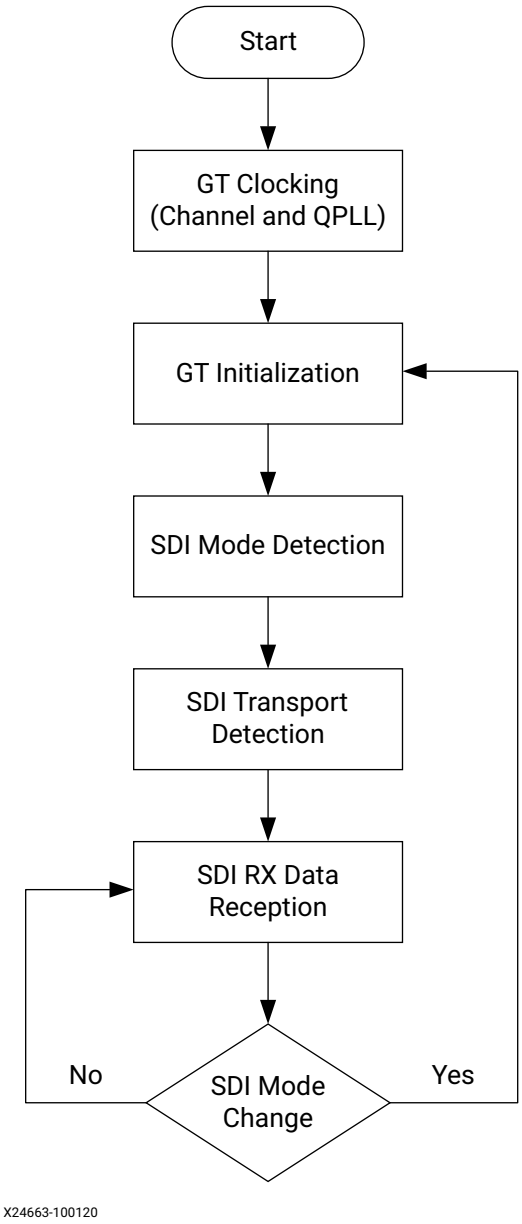

Figure B-1: **Hardware Debugging**

GT Clocking

Note: Refer to the Versal ACAP Lounge support forum at Xilinx Support for details about clock implementation and other tool tweaks.

- Make sure QPLL is getting reset before starting the IP.
- Monitor the QPLL LOCK signal.
- Verify that QPLL input clock frequency is of expected value.

- It is mandatory to reset the QPLL if clock input to QPLL is stopped or unstable.
- See AR# 57738 for debugging GT reference clock issues.
- Make sure to use QPLL default settings from latest GT Wizard IP core based on target device.
- Check the voltage rails on the transceivers. See AR# 57737 for more information.
- Measure **RXOUTCLK** is of expected frequency.
- Make sure **RXOUTCLK** of the transceiver is the clock driving **rx_usrclk**, **RXUSRCLK**, and **RXUSRCLK2**.
- Monitor **RXBUFFSTATUS**[2:0] for overflow and underflow errors.

GT Initialization

- GTRXRESETDONE is asserted High after GT completes initialization.
- Make sure GT is not reset during normal operation.
- See AR# 59435 for more information on debugging GT reset problems.
- Follow the recommended GT reset sequence.

SDI Mode Detection

- The UHD-SDI RX core hunts for TRS symbols and asserts **rx_mode_locked** if the UHD-SDI core detects error-free TRS symbols (that is, a valid XYZ word and with no CRC errors on the received line) for continuous three video lines.
- Look for pulses on the **rx_eav**, **rx_sav** signals at the correct places on each line along with **rx_trs** signal.
- Make sure no CRC errors are reported on **rx_crc_err_ds1** to ds16.
- Make sure supported modes are enabled in the **rx_mode_enable**[5:0] signal.
- Verify that **rx_mode**[2:0] reported by the SDI core matches the incoming SDI mode.
- SDI Mode detection searches the mode in the following fashion if all of the modes are set in r**x_mode_enable**: HD -> 3G -> 6G -> 12GA -> 12GB -> SD -> HD.
- Make sure that **rx_frame_en** signal is asserted High during core operation.

SDI Transport Detection

- The UHD-SDI core reports the video format by asserting **rx_t_locked** signal.
- This functionality takes two video frames to report the transport format
- Check the **rx_t_family[3:0]**, **rx_t_rate[3:0]**, **rx_t_scan** and **rx_t_locked** signals.

- Make sure the incoming video format is supported by the UHD-SDI core.
- Check that the **rx_bit_rate** signal is connected to the core.

UHD-SDI RX Data Reception

- UHD-SDI core receives the data in this stage until it receives erroneous video lines or change in incoming video source.

UHD-SDI Mode Change

- UHD-SDI core deasserts **rx_mode_locked** signal if it receives 15 or more erroneous video lines continuously.
- After **rx_mode_locked** is deasserted, it starts the SDI mode scan.

SDI RX to Video Bridge Debug:

- Make sure correct SDI mode (**rx_mode**[2:0] input) is passed on to SDI RX to Video Bridge subcore.
- Verify first pixel data is data on input native SDI data stream is appearing on native Video interface by monitoring **vid_data**[59:0] data bus along with other control signals such as **vid_active_video**, **vid_hblank** and **vid_vblank**.

Video In to AXI4-Stream Debug:

- Verify first pixel data is data on input native video interface is appearing on AXI4 Video interface by monitoring **m_axis_video_tdata**[63:0] data bus along with control signals such as **m_axis_video_tvalid** and **m_axis_video_tready**.
- Make sure that underflow and overflow outputs are not asserted. If asserted, verify the clock connections and along with expected frequencies.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. See Figure B-2 and Figure B-3. Output **s_axi_arready** asserts when the read address is valid, and output **s_axi_rvalid** asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The **s_axi_aclk** and **aclk** inputs are connected and toggling.
- The interface is not being held in reset, and **s_axi_areset** is an active-Low reset.

- The interface is enabled, and **s_axi_aclken** is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a debug feature capture that the waveform is correct for accessing the AXI4-Lite interface.

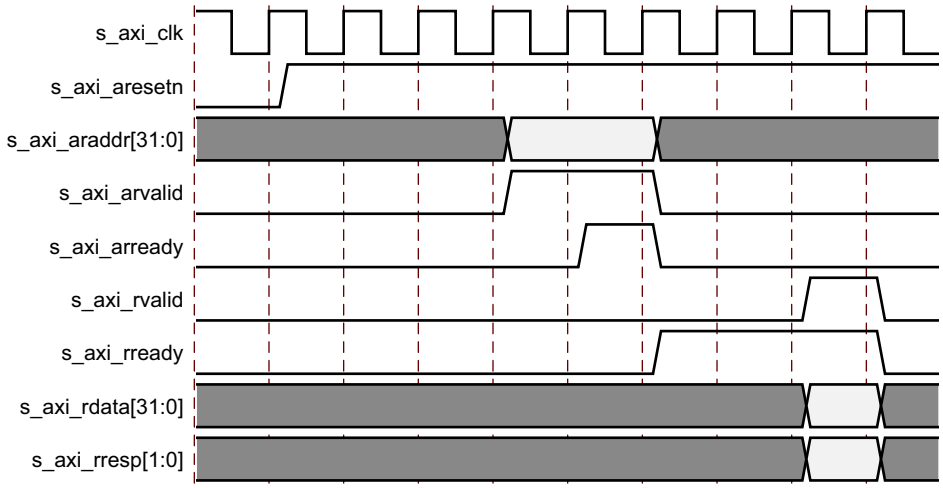

Figure B-2: **Read**

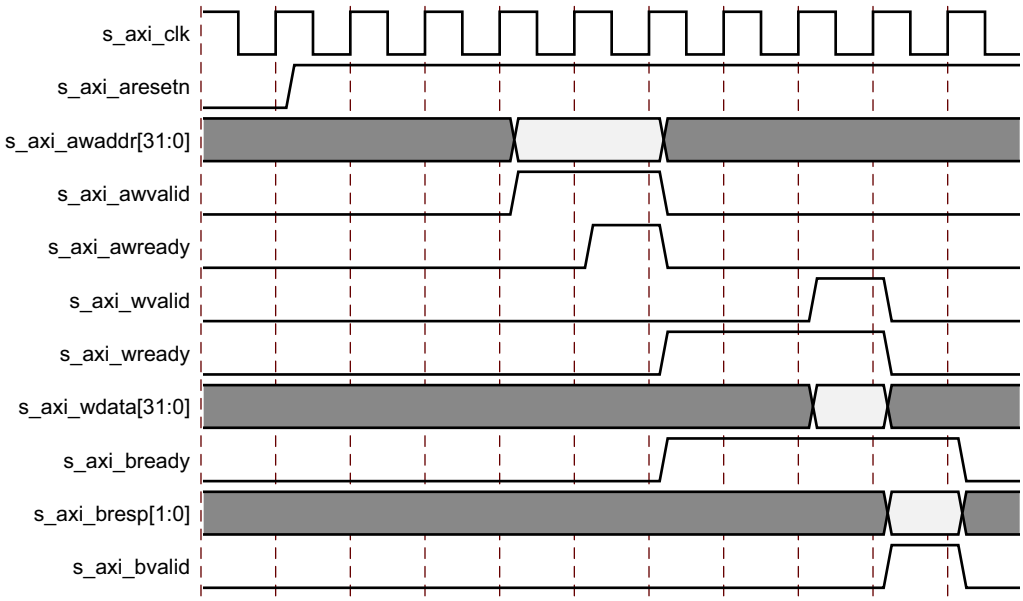

Figure B-3: **Write**

AMD XILINX

Appendix C

Audio Test Pattern Generator IP

This appendix provides information on Audio Test Pattern Generator IP core (audio_tpg_v1_0) that is used in UHD-SDI Subsystems example designs and provides details on register programming. This IP core is hidden in catalogue and can be generated only in IP integrator canvas using the `create_bd_cell` Tcl command.

Feature Summary

- AXI4S compliant
- 16/24 bit data support
- Support for 2,4,6,8...32 channels
- 1K, 2K sine tones and incremental pattern generation
- Checker block for validating incoming incremental pattern
- Generate and consume mode
- Audio sampling can be changed at runtime

Unsupported Features, Limitations

- No 20 bit data support
- Checker supports only incremental pattern
- Same data on is sent over all left channels and all right channels for sine tone outputs.
- Supports only 32K, 44.1K and 48K audio sampling
- 1K and 2K sine tone outputs

Port Descriptions

Table C-1: **Port Descriptions**

Signal Name	Interface	Type	Description
s_axi_aclk	Clock	Input	Input clock for AXI4Lite Interface
s_axi_aresetn	reset	Input	Active-Low reset for AXI4Lite Interface
s_axi_*	s_axi_ctrl		AXI-Lite Interface
aud_mclk	Clock	Input	Input audio clock. Typically a multiple of Fs
m_axis_aud_aclk	Clock	Input	AXIS Audio streaming clock
m_axis_aud_resetn	Reset	Input	Active-Low AXIS audio reset
m_axis_aud_*	Audio AXIS Interface	Master	Master AXIS audio interface
s_axis_aud_*	Audio AXIS Interface	Slave	Slave AXIS audio interface
m_aud_pulse	Signal	Output	Pulse on aud_clk with sample frequency

Register Space

The Audio Pattern Generator (PatGen) IP register space is shown in Table C-2.

Table C-2: **Audio Pattern Generator Register Space**

Reg Offset	Name	Reg Description
0x00	Control Register	Control register
0x04	Fs Multiplier	Register to specify Fs multiplier
0x08	Patgen Configuration	PatGen configuration such as channels, pattern, sampling and data width
0x0C	Silence time register	Specify any silence time to create a beep pattern
0x10-0x24	Channel Status Bits	Specify the 192 bits channel status
0x28-0x3C	User Status bits	Specify the 192 bits user bits
0x40	Validity register	Set the validity bit for samples
0x44	Checker Status	To check the checker result
0x48	Valid sample counter	Counts the number valid samples that were checked
0x4C	Number of samples missed counter	Counts number data mismatches that occurred
0x50	Data received	Incoming data sample register
0x54	Previous data	Previous sampled data register

Note: The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (*_wdata) signal, and is not impacted by the AXI Write Data Strobe (*_wstrb) signal. For a Write, both the AXI Write Address Valid (*_awvalid) and AXI Write Data Valid (*_wvalid) signals should be asserted together.

Core Control Register (0x00)

This register is used to control the APG and change its behavior to either generate audio traffic or put it in by-pass mode. The APG, by default, is in bypass mode. When APG is to be started, by-pass should be disabled.

Table C-3: **Core Control Register**

Name	Default Value	Access	Description
Bypass enable	0x1	RW	[31] This bit is used to enable the bypass mode. In this mode the APG is bypassed and whatever that comes on input AXIS is forwarded to output AXIS interface. This bit has a higher priority than bit [0].
Reserved			[30:2]
Start Checker	0x0	RW	[1] This bit enables the checker. Checker keeps checking the AXIS data received from the S_AXIS ports.
Enable APG	0x0	RW	[0] Set this bit to start the APG or put in generate mode. This should be enabled after all other config registers have been written.

Note: All of the following registers, except 0x40, can be written only when the 'Enable APG' bit of Core Control Register is 0. All of the following registers have to be programmed first before starting the APG.

Fs Multiplier Register (0x04)

Program this register to specify the multiplier value of the aud_clk.

Table C-4: **Fs Multiplier Register**

Name	Default Value	Access	Description
RSVD			[31:16]
Multi Value	0x180	RW	[15:0] Specify the multiplier value of the aud_clk. For example, for Fs = 48 kHz, the supplied aud_clk is 18.432 MHz, then this register should be programmed with a value of 384. i.e. 48 kHz * 384 = 18.432 MHz

APG Config Register (0x08)

This registers provides capability to enable/disable the core.

Table C-5: **APG Config Register**

Name	Default Value	Access	Description
Channel count	0x2	RW	[31:24] Set number of audio channels. Valid values are 2, 4, 6 and 8.
Data width	0x4	RW	[23:16] Specify the width of audio data samples. Valid values are: 001 – 16 bit audio samples 010 – 20 bit audio samples 100 – 24 bit audio samples
Sampling Frequency	0x2	RW	[15:8] Specify the sampling rate of the audio samples. Valid values are: 000 – 32 kHz 001 – 44.1 kHz 010 – 48 kHz
Pattern	0x0	R/W	[7:0] Select from internal patterns. Valid values are: 000 – 1 kHz sine wave 001 – 2 kHz sine wave 010 – 1 kHz sine wave on left, 2 kHz sine wave on right channel 111 – Incremental pattern

APG Silence Register (0x0C)

This register is used to insert a silence time. This can be programmed to create a beep pattern of either 1 kHz or 2 kHz sine wave.

Table C-6: **APG Silence Register**

Name	Default Value	Access	Description
Reserved	0		[31:16]
Silence interval	0	RW	[15:0] Specify a value to insert periodic silence. The silence time is twice the value that is programmed here. Silence time = 2*value*aud_clk Period A value of 0 means no silence.

Channel Status Register (0x10-0x24)

Specify the 192 bits of Channels Status information using these 6 registers. Each register is to be programmed with 32 bits of data in order of LSB to MSB.

Table C-7: **Channel Status Register**

Name	Default Value	Access	Description
Channel Status	0	RW	[31:0] Specify the 32-bit information.

User Info Register (0x28-0x3C)

Specify the 192 bits of User information using these 6 registers. Each register is to be programmed with 32 bits of data in order of LSB to MSB.

Table C-8: **User Info Register**

Name	Default Value	Access	Description
User info	0	RW	[31:0] Specify the 32-bit information.

Validity Control (0x40)

This bit allows the user to set the validity bit in the Audio sample.

Table C-9: **Validity Control**

Name	Default Value	Access	Description
Reserved	0		[31:1]
Validity	0x0	R/W	[0] This bit can be used to set the validity bit inside the audio sample frame. Receivers typically ignore the samples, which have the validity bit set to '1'.

Checker Status (0x44)

This register allows the user to check the status of the checker function

Table C-10: **Checker Status**

Name	Default Value	Access	Description
Checker started	0x0	R	[31] 1 – when at least one valid sample was detected
Reserved	0x0		[30:8]
Data mismatch channel	0x0	R	[7:5] Channel ID where the first data mismatch occurred
ID error	0x0	R	[4] Error due to wrong ID pattern
Parity	0x0	R	[3] Error in parity bit
Preamble	0x0	R	[2] Error in preamble bits
Data mismatch error	0x0	R	[1] Error in incremental pattern
Checker status (Channel0_status)	0x0	R	[0] When start checker is enabled, the checker keeps checking for an incremental pattern. If there is a mismatch, this bit goes High.

Valid Sample Counter (0x48)

Table C-11: **Valid Sample Counter**

Name	Default Value	Access	Description
User info	0	RW	[31:0] Specify the 32 bit information.

Number of Samples Missed Counter (0x4C)

Table C-12: **Number of Samples Missed Counter**

Name	Default Value	Access	Description
No_of_samples_missed	0x0	R	[31:0] Counter value indicating number of missed samples in case of an error.

Data Received (0x50)

This register gives the information of the incoming data sample when the first data mismatch error occurred.

Table C-13: **Data Received**

Name	Default Value	Access	Description
Reserved			[31:24]
Incoming data	0x0	R	[23:0] Data received on s_aud_tdata when the error occurred.

Previous Data (0x54)

This register stores the previous sampled data available when the data mismatch error occurred.

Table C-14: **Previous Data**

Name	Default Value	Access	Description
Reserved			[31:24]
Sampled data	0x0	R	[23:0] Previous sampled data when error occurred. Current data should be sampled data + 1

A data mismatch error occurs when incoming data is not equal to the previously sampled data + 1.

Clocking

There are three possible clock inputs available:

- **s_axi_aclk**: AXI4-Lite control interface clock
- aud_aclk: Reference audio MCLK. This is typically a multiple of Fs (for example, Fs*384).
- **m_axis_aud_aclk**: AXIS streaming interface clock

Resets

The **s_axi_aresetn** resets the register interface and all puts all the registers in their default state.

Programming Sequence

The APG can be put in generate-consume mode using the following programming sequence:

1. Setup the APG configuration registers for no of channels, data width, sampling, and pattern.

   Note: These values cannot be updated after the APG is started.

2. Program the Multiplier value. Program the channel status and user bits.

   Note: The multiplier value should match with the aud_clk frequency

3. Program the Control registers to put the APG in generate mode.

   Note: In this mode, the APG outputs the audio samples and at the same time consumes the audio on AXIS slave interface by asserting ready.

To change the APG to bypass mode, simply program the Control register bit [31] to '1'. All other registers are ignored when APG is set in the bypass mode. Irrespective of the APG mode (generate or bypass), the checker can be enabled by programming the Control Register bit [1] to 1.

Note: In bypass mode, the APG connects the Master and Slave AXIS interfaces. In generate-consume mode, the APG continuously generates the samples at a programmed rate. A small buffer is available which can overflow if the samples are not consumed.

AMD XILINX

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

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

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

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

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

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

References

These documents provide supplemental material useful with this product guide:

1. *Vivado® Design Suite User Guide: Designing IP Subsystems Using IP integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *ISE® to Vivado Design Suite Migration Guide* (UG911)
6. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
7. *Vivado Design Suite User Guide - Implementation* (UG904)
8. AXI4-Stream Video IP and System Design (UG934)
9. *SMPTE UHD-SDI Product Guide* (PG205)
10. *Video In to AXI4-Stream LogiCORE IP Product Guide* (PG043)
11. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
12. *UltraScale Architecture GTY Transceivers User Guide (*UG578*)*
13. *UltraScale FPGAs Transceivers Wizard* (PG182)
14. *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)
15. *SMPTE UHD-SDI TX Subsystem Product Guide* (PG289)
16. *Vivado Design Suite: AXI Reference Guide* (UG1037)
17. *UHD-SDI Audio Core IP Product Guide* (PG309)
18. *UHD-SDI GT IP Product Guide* (PG380)
19. *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics* (DS957)
20. *Versal ACAP GTY Transceivers Architecture Manual* (AM002, v1.1)
21. *AXI SmartConnect LogiCORE IP Product Guide* (PG247)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/26/2022	2.0	• Added parameters to select HFR and YCbCr444 support to reduce resource count.
06/30/2021	2.0	• Added 12G/6G HFR resolutions for 10 bit for AXI Interface subsystem configuration. • Updated unsupported feature section • Update TS_det register configuration for HFR • Updated Figure 5-44 • Updated Table A-6 • Added Table A-7
01/11/2021	2.0	• Added HLG HDR feature • Added Versal ACAP Block Automation in UHD-SDI RX Subsystem section • Added VCK190 SMPTE UHD-SDI Audio-Video Pass-Through Example Design section
10/05/2020	2.0	• Updated Features for 12-bit support and HFR support added in design • Updated Unsupported Features • Updated Reference clock selection in Clocking section for integer and fractional frame rates • Added Table A-6 for tested HFR resolutions in simulation Video Resolutions • Added Table A-8 for tested 12-bit resolutions in simulation Video Resolutions
11/08/2019	2.0	• Added Table A-1, Table A-5, Table C-9, and Table C-10 • Updated Table 2-8, Table 2-14, Table 5-1, Table C-4, Table C-5, Table C-6, and Table C-13 • Updated Features section • Updated Unsupported Features section. • Updated Requirements section • Updated SDK instances to Vitis software platform • Updated Vivado menu commands
07/30/2019	2.0	• Updated Figure C-2 and Figure C-4 • Updated Component Name section • Updated Table 2-8, Table C-2, Table C-3, and Table C-13 • Added Figure 5-32

Date	Version	Revision
12/05/2018	2.0	• Updated UHD-SDI Audio Extract Use Case section • Updated ZCU106 SMPTE UHD-SDI Audio-Video Pass-Through Example Design section • Updated Table 5-2 and HD-SDI TXOUTCLK frequencies • Added UHD-SDI GT Ports section • Added RX_ST352_DATA_DS<n> registers for C stream
04/04/2018	2.0	• Added ZCU106 UHD-SDI Audio-Video Pass-Through Example Design • Added KCU116 UHD-SDI Audio-Video Loopback Example Design • Added Appendix C for UHD-SDI IP • Added Appendix D for Audio Test Pattern Generator IP
10/04/2017	1.0	Initial Xilinx release.

Please Read: Important Legal Notices