Zynq UltraScale+ MPSoC ZCU106 Video Codec Unit Targeted Reference Design

User Guide

UG1250 (v2020.1) June 3, 2020

XILINX

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/03/2020	2020.1	Added multi stream(4x) support for all HDMI LLP2 designs. Added single stream audio support for HDMI LLP2 NV12 design. Added PICXO and fractional frame rate support for SDI designs. Added XAVC profile support. Added new XDMA host driver support for PCIe design.
10/31/2019	2019.2	Added single-channel stream-based SCD and HDMI interlace video support to the multistream audio design. Added multistream and DCI 4k resolution support to the PL DDR HDMI design. Added a new LLP2 design for ultra-low latency streaming support. Added support for PCIe-based file encoding and decoding.
05/29/2019	2019.1	Updated hardware and software tools for Vivado Design Suite 2019.1 and Petalinux-2019.1. This release has all the designs supported in 2018.3 and the following new designs: • SCD feature in multi-stream design • Multi-stream audio support • PCIe based file transcoding • PLDDR in HDMI pipeline • SDI-RX and SDI-TX designs with 4:2:2 10 bit support • SDI RX/TX design with audio support
12/05/2018	2018.3	Updated for hardware and software tools for Vivado Design Suite 2018.3. Updated for HDMI video display, HDMI video capture and HDMI display with audio, 10G HDMI video capture and HDMI display, 10G HDMI video capture and HDMI display with Vitis tool support, and SDI video display designs. Updated with complete VCU TRD design details and with design components for the audio and streaming feature. Added 1080p30 multi-stream support.
07/27/2018	2018.2	Updated for hardware and software tools for Vivado Design Suite 2018.2. Updated Figure 1-3, Figure 3-2, Figure 3-3, Figure 3-9, Figure 3-11, Figure 5-1, and Figure 5-6.
06/29/2018	2018.1	Updated for hardware and software tools for Vivado Design Suite 2018.1.
02/15/2018	2017.4	Updated for hardware and software tools for Vivado Design Suite 2017.4. Updated Figure 3-16 and Figure 3-17. Removed *GStreamer Interface Library Description*. **Limited release**.
12/20/2017	2017.3	Updated for Vivado Design Suite 2017.3. Video support is upgraded from 4KP30 to 4KP60. Added multi-stream encode/decode support, pipelined MIPI video input, and the HDMI TX video display pipeline. Updated *Exported APIs*. **Limited release**.
12/01/2017	2017.2	Initial Xilinx release. **Limited release.**

Table of Contents

Appendix B: Additional Resources and Legal Notices

Introduction

About this TRD

This document describes the features and functions of the Zynq® UltraScale+™ MPSoC Video Codec Unit (VCU) targeted reference design (TRD). The VCU TRD is an embedded video encoding/decoding application partitioned between the SoC processing system (PS), VCU, and programmable logic (PL) for optimal performance. The design demonstrates the capabilities and performance throughput of the VCU embedded macro block available in Zynq UltraScale+ MPSoC devices.

The TRD serves as a platform to tune the performance parameters of the VCU to arrive at optimal configurations for encoder and decoder blocks.

The TRD demonstrates the following hard block features in the PS and PL:

- VCU hard block capable of performing up to 4K (3840 x 2160/4096 x 2160)
- Simultaneous encoding and decoding of single and multiple streams
- PS DisplayPort controller for 4K (3840 x 2160) @ 30 Hz
- PL-based HDMI-TX/SDI-TX for 4K (3840 x 2160/4096 x 2160) @ 60 Hz
- GPU used for rendering a graphical user interface (GUI)
- Extensible platform uses:
  - GStreamer v1.16.1 pipeline architecture to construct a multimedia pipeline [Ref 1]
  - Standard Linux software frameworks
  - OpenMAX™ v1.1.2 based client interface for the VCU
  - Modular and hierarchical architecture (enables partner modules)
  - Configurable IP Subsystems
- System software configuration:
  - Linux symmetric multi-processing (SMP) on the application processing unit (APU)

This user guide describes the architecture of the reference design and provides a functional description of its components. It is organized as follows:

- Chapter 1, Introduction (this chapter) provides a high-level overview of the Zynq UltraScale+ MPSoC architecture, the reference design architecture, and a summary of key features.
- Chapter 2, Targeted Reference Design Details gives an overview of the design modules and design components that make up this reference design.
- Chapter 3, APU Software Platform describes the APU software platform covering the middleware and operating system layers of the Linux software stack and the Linux GStreamer application running on the APU.
- Chapter 4, System Considerations describes system architecture considerations including boot flow, system address map, video buffer formats, and performance analysis.
- Chapter 5, Hardware Platform describes the hardware platform of the design including key PS and PL peripherals.
- Appendix A, Input Configuration File lists additional resources and references.

Zynq UltraScale+ MPSoC Overview

The Zynq device is a heterogeneous, multi-processing SoC built on the 16-nm FinFET technology. Figure 1-1 shows a high-level block diagram of the device architecture and key building blocks inside the processing system (PS) and the programmable logic (PL).

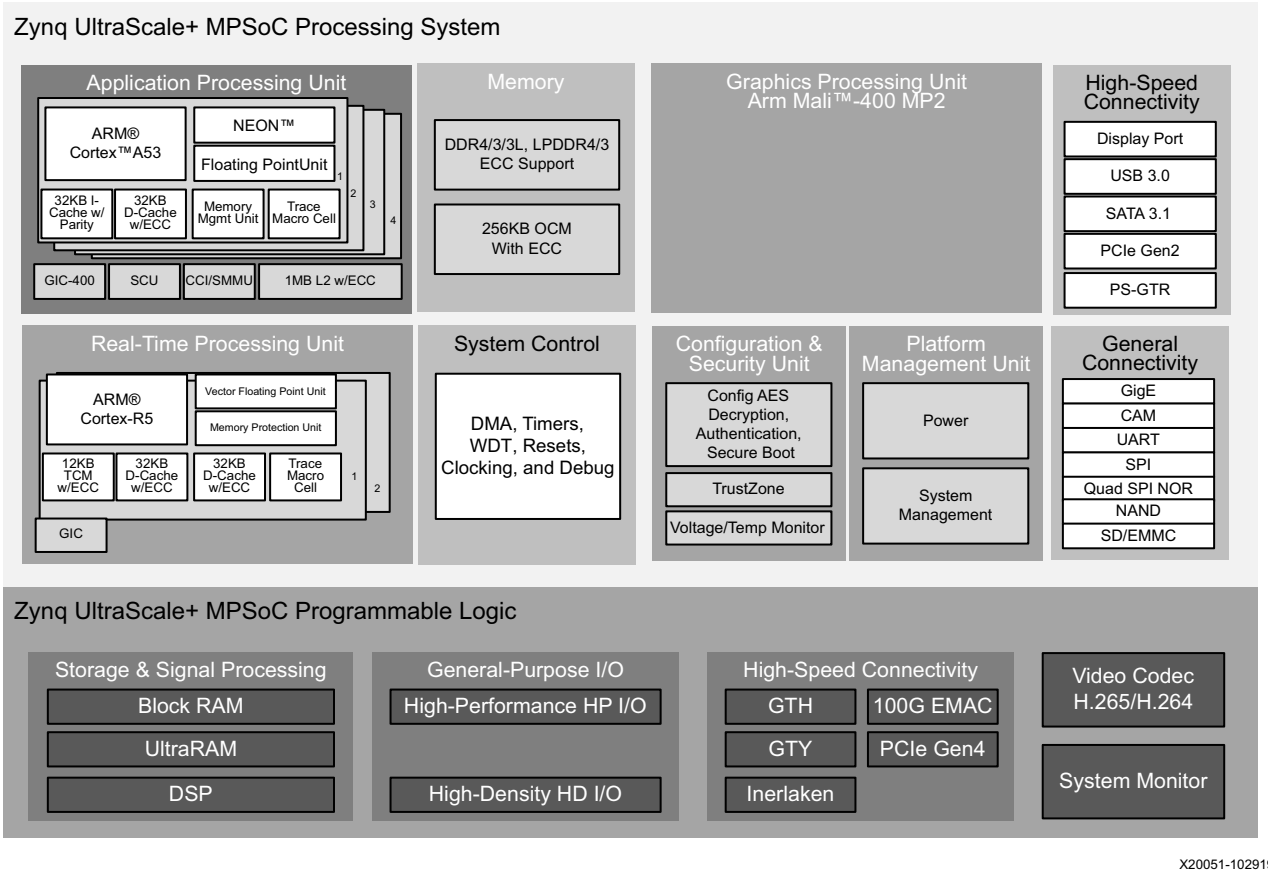

Figure 1-1: **Zynq UltraScale+ MPSoC Block Diagram**

The MPSoC key features include:

- Application processing unit (APU) with a 64-bit quad-core Arm® Cortex™-A53 processor
- Real-time processing unit (RPU) with a 32-bit dual-core Arm Cortex-R5 processor
- Multimedia blocks
  - Graphics processing unit (GPU) Arm Mali-400MP2
  - Video codec (encoder/decoder) unit up to 4K (3840 x 2160) 60 frames per second (FPS)

    - DisplayPort controller interface up to 4K (3840 x 2160) 30 FPS
- High-speed peripherals
    - PCIe root complex and Endpoint (Gen1 or Gen2 x1, x2, and x4 lanes)
    - USB 3.0/2.0 with host, device and on-the-go (OTG) modes
    - SATA 3.1 host
- Low-speed peripherals
    - Gigabit Ethernet, controller area network (CAN), universal asynchronous receiver-transmitter (UART), Serial Peripheral Interface (SPI), Quad SPI, NAND flash memory, Secure Digital embedded Multimedia Card (SD/eMMC), inter IC (I2C), and general purpose I/O (GPIO)
- Platform management unit (PMU)
- Configuration security unit (CSU)
- 6-port DDR controller with error correction code (ECC), supporting x32 and x64 DDR4/3/3L and LPDDR4/3

Reference Design Overview

The MPSoC has a heterogeneous processor architecture. The TRD makes use of multiple processing units available inside the PS using this software configuration:

The APU consists of quad Arm Cortex-A53 cores configured to run in SMP Linux mode. The main task of the application is to configure and control the video pipelines using a Qt v5.9.4 based graphical user application. See Figure 1-2.

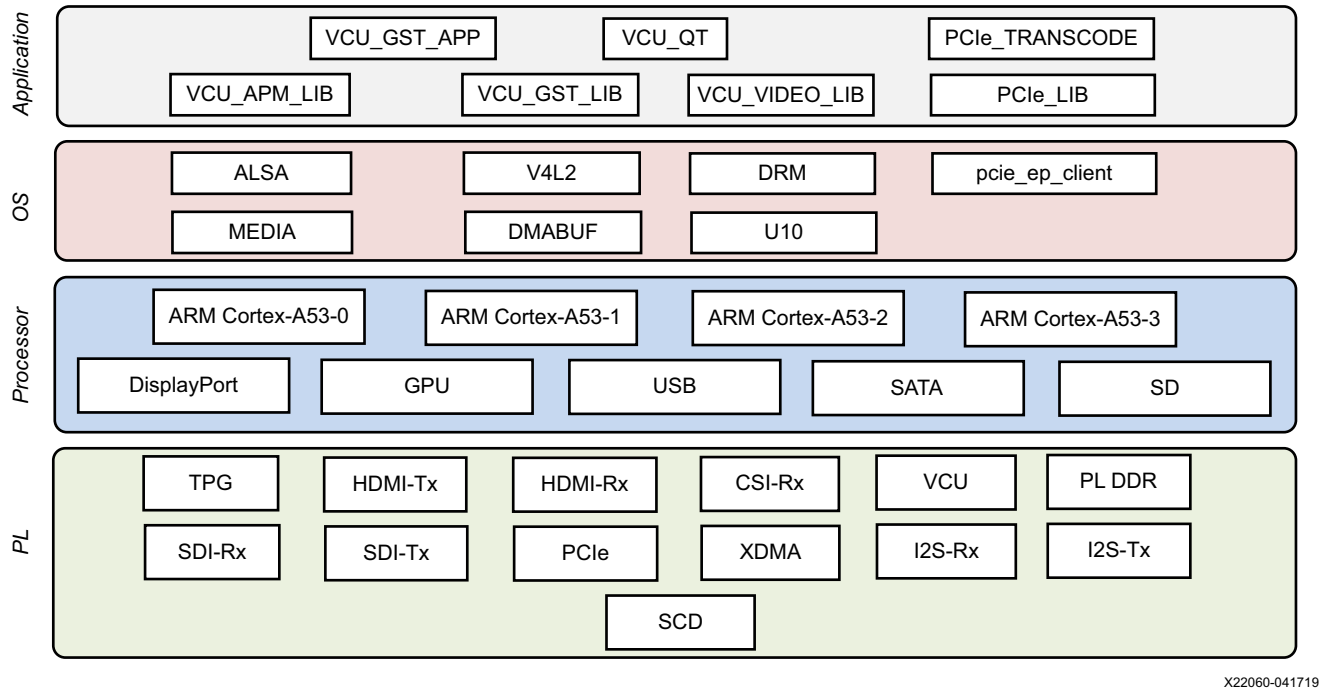

Figure 1-2: **Key Reference Design Components**

Figure 1-2 shows the software state after the boot process has completed and the individual applications have been started on the target processing units. The TRD does not use virtualization and therefore does not run a hypervisor on the APU.

The APU application controls the following video data paths implemented in the PS and PL (see Figure 1-3, page 11):

- Capture pipeline capturing video frames into DDR memory from a high definition media interface (HDMI source connected through the PL, an image sensor on an FMC daughter card connected via MIPI CSI-2 RX Subsystem through the PL, serial digital interface (SDI) source connected through the PL, and a Test Pattern Generator (TPG) implemented inside the PL. Additionally, video can be sourced from a SATA drive, USB 3.0 device, or an SD card, which is also used as a boot device.
- Processing (memory-to-memory) pipeline includes VCU encode/decode. Video frames are read from DDR memory, processed by the VCU, and written back to memory.
- Display pipeline reading video frames from memory and sending them to a monitor via the DisplayPort TX Controller inside the PS, SDI Transmitter Subsystem through the PL or the HDMI Transmitter Subsystem through the PL. The DisplayPort TX Controller supports two layers—one for video, the other for graphics and the SDI Transmitter Subsystem with mixer IP support up to four layers and HDMI Transmitter Subsystem with mixer IP supports up to eight such layers. The graphics layer is rendered by the GPU.
- Audio Capture pipeline to capture audio frames from HDMI-RX, SDI-RX and I2S-RX interfaces.

- Audio Renderer pipeline to playback the audio frames through HDMI-TX, SDI-TX, DP, and I2S-TX interfaces.
- Transcode, encode, or decode pipeline to transfer the file from the HOST machine to the client board (zcu106) through PCIe XDMA bridge interface in the PL. The file is passed to the VCU encoder and decoder block for transcoding, encoding, or decoding. The transcoded, encoded, or decoded file is written back to HOST machine using the PCIe XDMA bridge interface read channel.
- LLP2 pipeline to stream-out and stream-in live captured video at ultra-low latencies using Sync IP.

The TRD consists of 14 designs which are highlighted in four colors as shown in Figure 1-3.

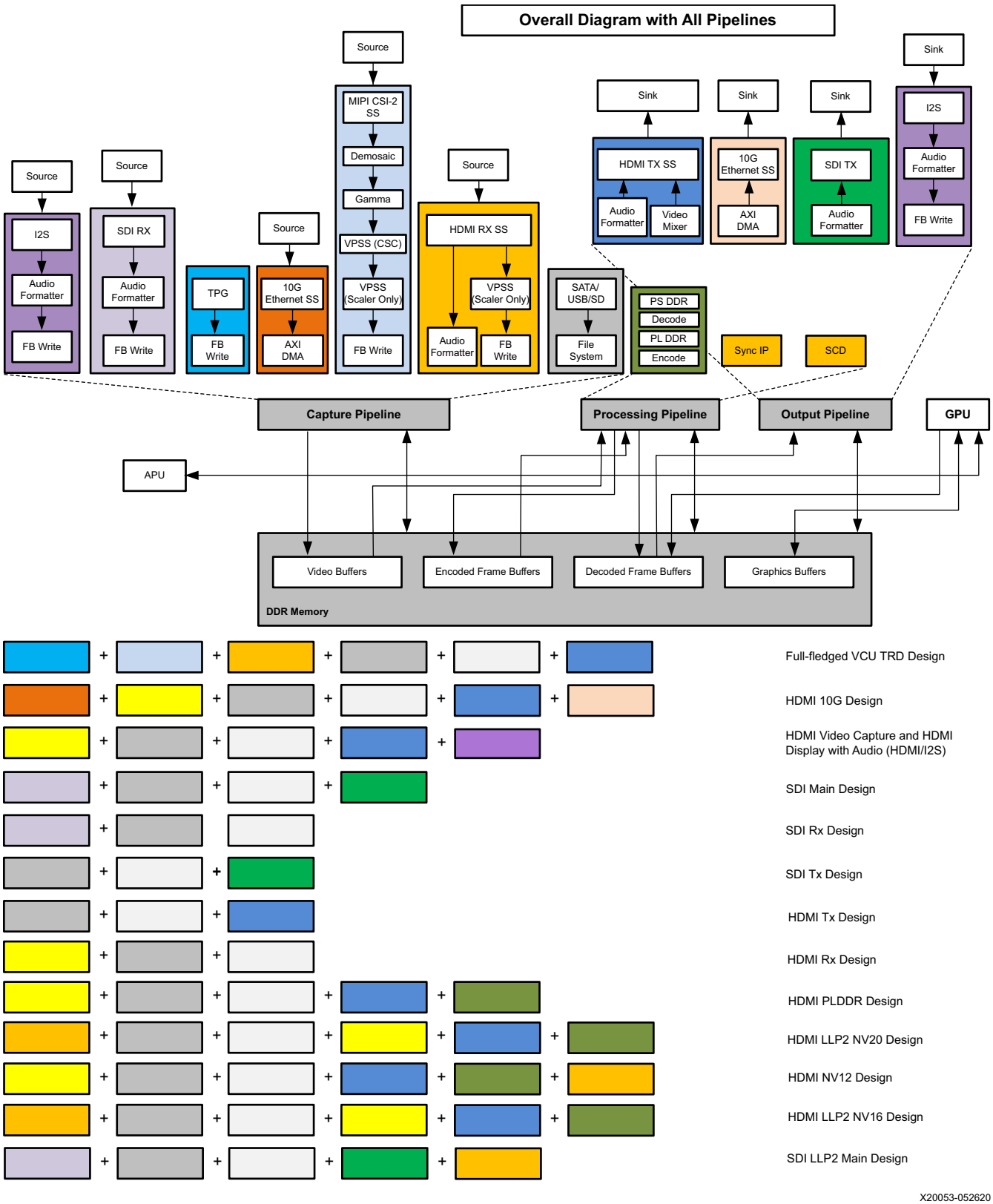

Figure 1-3: **VCU TRD Block Diagram**

Note: In Figure 1-3, except for all of the VCU Audio designs, HDMI pipelines in all other designs exclude Audio Formatter IP and thus do not have audio. For XV20 designs PLDDR is included in the processing pipeline

The remaining blocks are common to all designs. See Chapter 2, Targeted Reference Design Details for more details.

The reference design targets the ZCU106 evaluation board. The board has an onboard HDMI transmitter and receiver connector, SDI transmitter and receiver connector, and a DisplayPort connector interface. The evaluation board provides the HDMI reference clock, data recovery unit (DRU) clock, and the reference clock for the design. The PS_REF_CLK is sourced from another dedicated clock generator present on the evaluation board. Figure 1-4 shows the block diagram of the TRD along with the board components.

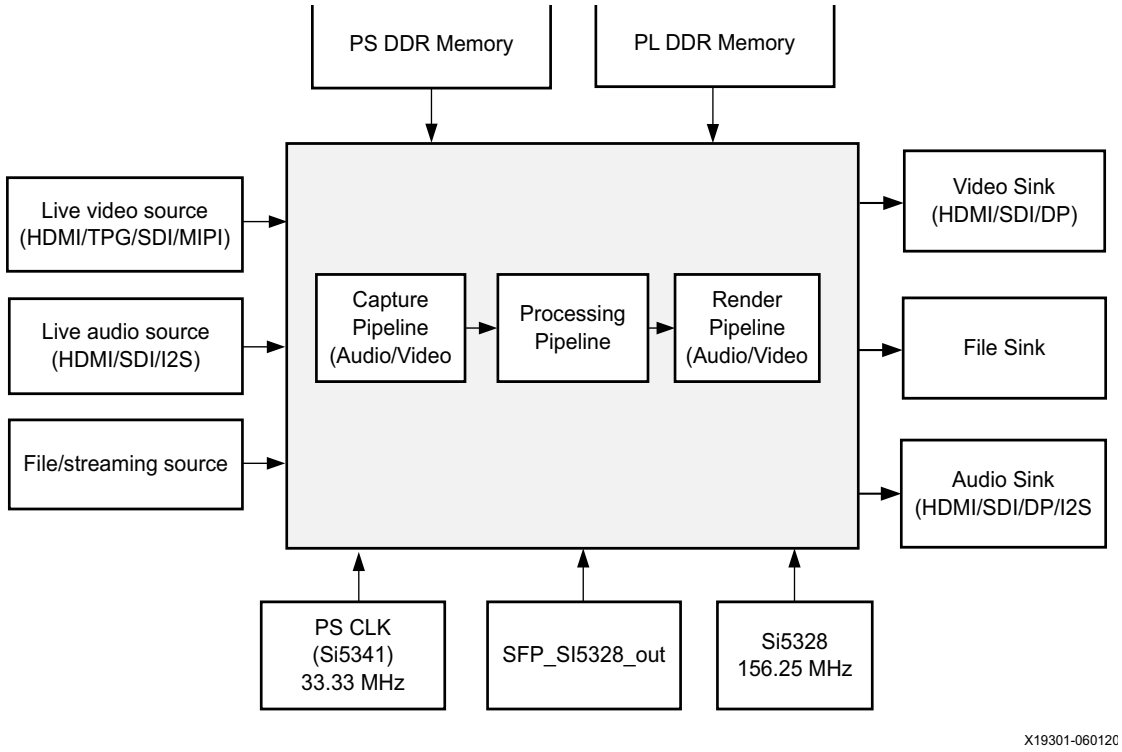

Figure 1-4: **High-Level Block Diagram of ZCU106 Device Architecture**

Key Features

Target platforms and extensions:

- ZCU106 evaluation board (see *ZCU106 Evaluation Board User Guide* (UG1244)) [Ref 2]
- Optional: Leopard Imaging LI-IMX274MIPI-FMC image sensor daughter card [Ref 3]
- SDI Receiver - Blackmagic Design Teranex Mini HDMI to 12G converter
- SDI Transmitter - Blackmagic Design Teranex Mini 12G to HDMI converter

Xilinx tools:

- Vivado® Design Suite 2020.1
- PetaLinux tools 2020.1

Hardware interfaces and IP:

- GPU
- Video inputs
  - TPG
  - HDMI RX
  - MIPI CSI-2 RX
  - File source (SD card, SATA and USB 3.0 drives)
  - SDI RX
  - Stream In
- Video outputs
  - DisplayPort TX controller
  - HDMI TX
  - SDI TX
  - PICXO IP
- Audio Inputs
  - HDMI RX
  - SDI RX
  - I2S RX
- Audio Outputs:
  - HDMI TX
  - SDI TX
  - I2S TX
  - DP
- Video compression/decompression
  - VCU hard block
- SYNC IP
- Auxiliary peripherals

- SD
- I2C
- GPIO
- 1G/10G Ethernet
- UART
- USB 2.0/USB 3.0
- AXI Performance Monitor (APM)
- PCIe
- Digilent PMOD audio card [I2S2]
- 3.5mm auxiliary cables
- Speakers

Software components:

- Operating systems
  - APU: SMP Linux
- Linux frameworks/libraries
  - Video: Video4Linux (V4L2), Media controller
  - Audio: libalsa
  - Display: Direct Rendering Manager/Kernel Mode Setting (DRM/KMS), X-Server (X.Org)
  - Graphics: Qt5, OpenGL ES2
- User application:
  - APU: GStreamer-based command line application, QT GUI application

Supported video formats:

- Input resolution
  - DCI 4Kp60 (4096 x 2160)
  - 4Kp30 (3840 x 2160)
  - 1080p60 (1920 x 1080)
  - 1080p30 (1920 x 1080)
- Output resolution
  - DCI 4Kp60 (4096 x 2160) — HDMI/SDI
  - 4Kp30 (3840 x 2160) — HDMI and DisplayPort

    - Native 1080p60 on both DisplayPort and HDMI
- Pixel formats
    - NV12
    - NV16
    - XV15
    - XV20

Targeted Reference Design Details

Design Modules

The VCU TRD consists of 14 design modules (DMs). A short summary of each design follows.

PL HDMI Video Capture

This module enables capture from the HDMI RX subsystem implemented in the programmable logic (PL) into a file or to stream-out video. The video captured from the HDMI RX subsystem is encoded and stored in SD cards or USB/SATA drives. The module can stream-out encoded data through an Ethernet interface.

PL HDMI Video Display

This module enables video display to HDMI TX implemented in the PL. The video stored in the SD card or USB/SATA drive is decoded and displayed on HDMI TX. The module can stream-in encoded data through an Ethernet interface and decode and display it on HDMI TX.

Multi-Stream Audio Design

This module enables capture of audio data from I2S RX /HDMI RX and Video data from the HDMI RX/MIPI RX subsystem. This module enables support for Scene Change Detection IP (SCD IP). Stream-based SCD is supported in the HDMI pipeline in this design. The audio/video data can be played through HDMI TX in the PL and recorded in SD cards or USB/SATA drives. This module can stream-in/out the audio/video data through an Ethernet interface. This design supports the following streams:

- Stream 1
  - Input Source: Video and audio are captured from HDMI RX
  - Output Sink: Video and audio are played on HDMI TX
- Stream 2
  - Input Source: Video is captured from the MIPI RX subsystem and audio is captured from the I2S RX subsystem

- Output Sink: Video is played on HDMI TX and audio is played on I2S TX

PL 10G HDMI Video Capture and HDMI Display

This module enables capture of video from an HDMI RX subsystem implemented in the PL. The video can be displayed through HDMI TX through the PL, and recorded in SD cards or USB/SATA drives. The module can stream-in or stream-out encoded data through the 10G Ethernet interface.

Note: PL 10G Ethernet is supported only in 10G HDMI Video Capture and HDMI Display and HDMI Video Capture and HDMI Display with Vitis tool support designs. All other designs support PS 1G Ethernet.

PL SDI Video Display

This module enables the video display to the SDI TX Subsystem implemented in the PL. The video stored in SD cards or USB/SATA drives is decoded and displayed via SDI TX. This module can stream-in encoded data through an Ethernet interface and decode and display it on SDI TX.

PL SDI Video Capture

This module enables capture of video from an SDI RX subsystem implemented in the PL into a file or to stream-out video. The video captured from the SDI RX subsystem is encoded and stored in SD cards or USB/SATA drives. The module can stream-out encoded data through an Ethernet interface.

Note: This module supports XV20 (YUV 4:2:2 10-bit) format only.

Full-fledged VCU TRD

This module enables video capture from an HDMI source, an image sensor connected through CSI-2 RX, or a Test Pattern Generator (TPG) implemented in the PL. This module also enables support for Scene Change Detection IP (SCD IP). SCD is supported in memory-based mode. The video can be displayed via DP TX through the processing system (PS) using HDMI TX through the PL, and can be recorded in SD cards or USB/SATA drives. The module can stream-in or stream-out encoded data through an Ethernet interface.

PL DDR HDMI Video Capture and HDMI Display

This module enables capture of video from an HDMI RX subsystem implemented in the PL. The video can be displayed through HDMI TX through the PL and recorded in SD cards or USB/SATA drives. The module can stream-in or stream-out encoded data through an Ethernet interface. This module supports multi-stream for XV20 pixel format. It also supports DCI 4K (4096 x 2160) resolution at 60 FPS.

This is the new design approach proposed to use PL_DDR for decoding and PS_DDR for encoding so that DDR bandwidth would be enough to support high bandwidth VCU applications requiring simultaneous encoder and decoder operations and transcoding at 4K @60FPS. This approach makes most effective use of limited AXI4 read/write issuance capability in minimizing latency for the decoder. DMA buffer sharing requirements determine how capture, display, and intermediate processing stages should be mapped to the PS or PL DDR.

LLP2 PS DDR NV12 HDMI Video Capture and HDMI Display

This module enables capture of video and audio data from an HDMI Rx subsystem implemented in the PL. The video and audio data can be displayed through the HDMI Tx subsystem implemented in the PL. The module can stream-out and stream-in live captured video and audio through an Ethernet interface at ultra-low latencies using Sync IP. This module supports multi-stream video for NV12 pixel format. This module also supports single-stream audio.

The VCU encoder and decoder operate in slice mode. An input frame is divided into multiple slices (8 or 16) horizontally. The encoder generates a slice_done interrupt at every end of the slice. Generated NAL unit data can be passed to a downstream element immediately without waiting for the frame_done interrupt. The VCU decoder also starts processing data as soon as one slice of data is ready in its circular buffer instead of waiting for complete frame data. The Sync IP does an AXI transaction-level tracking so that the producer and consumer can be synchronized at the granularity of AXI transactions instead of granularity at the video buffer level. Sync IP is responsible for synchronizing buffers between Capture DMA and VCU encoder as both work on same buffer.

The capture element (FB write DMA) writes video buffers in raster-scan order. SyncIP monitors the buffer level while the capture element is writing into DRAM and allows the encoder to read input buffer data if the requested data is already written by DMA, otherwise it blocks the encoder until DMA completes its writes. On the decoder side, the VCU decoder writes decoded video buffer data into DRAM in block-raster scan order and displays reads data in raster-scan order. To avoid display under-run problems, software ensures a phase difference of "~frame_period/2", so that decoder is ahead compare to display.

LLP2 PL DDR NV16 HDMI Video Capture and HDMI Display

This module enables capture of video from an HDMI Rx subsystem implemented in the PL. The video can be displayed through the HDMI Tx subsystem implemented in the PL. The module can stream-out and stream-in live captured video frames through an Ethernet interface at ultra-low latencies using Sync IP. This module supports multi-stream for NV16 pixel format. In this design PL_DDR is used for decoding and PS_DDR for encoding so that DDR bandwidth would be enough to support high bandwidth VCU applications requiring simultaneous encoder and decoder operations and transcoding at 4k @60 FPS.

The VCU encoder and decoder operate in slice mode. An input frame is divided into multiple slices (8 or 16) horizontally. The encoder generates a slice_done interrupt at every end of the slice. Generated NAL unit data can be passed to a downstream element immediately without waiting for the frame_done interrupt. The VCU decoder also starts processing data as soon as one slice of data is ready in its circular buffer instead of waiting for complete frame data. The Sync IP does an AXI transaction-level tracking so that the producer and consumer can be synchronized at the granularity of AXI transactions instead of granularity at the video buffer level. Sync IP is responsible for synchronizing buffers between Capture DMA and VCU encoder as both work on same buffer.

The capture element (FB write DMA) writes video buffers in raster-scan order. SyncIP monitors the buffer level while the capture element is writing into DRAM and allows the encoder to read input buffer data if the requested data is already written by DMA, otherwise it blocks the encoder until DMA completes its writes. On the decoder side, the VCU decoder writes decoded video buffer data into DRAM in block-raster scan order and displays reads data in raster-scan order. To avoid display under-run problems, software ensures a phase difference of "~frame_period/2", so that decoder is ahead compare to display.

LLP2 PL DDR XV20 HDMI Video Capture and HDMI Display

This module enables capture of video from an HDMI Rx subsystem implemented in the PL. The video can be displayed through the HDMI Tx subsystem implemented in the PL. The module can stream-out and stream-in live captured video frames through an Ethernet interface at ultra-low latencies using Sync IP. This module supports multi-stream for XV20 pixel format. In this design PL_DDR is used for decoding and PS_DDR for encoding so that DDR bandwidth would be enough to support high bandwidth VCU applications requiring simultaneous encoder and decoder operations and transcoding at 4k @60 FPS.

The VCU encoder and decoder operate in slice mode. An input frame is divided into multiple slices (8 or 16) horizontally. The encoder generates a slice_done interrupt at every end of the slice. Generated NAL unit data can be passed to a downstream element immediately without waiting for the frame_done interrupt. The VCU decoder also starts processing data as soon as one slice of data is ready in its circular buffer instead of waiting for complete frame data. The Sync IP does an AXI transaction-level tracking so that the producer and consumer can be synchronized at the granularity of AXI transactions instead of granularity at the video buffer level. Sync IP is responsible for synchronizing buffers between Capture DMA and VCU encoder as both work on same buffer.

The capture element (FB write DMA) writes video buffers in raster-scan order. SyncIP monitors the buffer level while the capture element is writing into DRAM and allows the encoder to read input buffer data if the requested data is already written by DMA, otherwise it blocks the encoder until DMA completes its writes. On the decoder side, the VCU decoder writes decoded video buffer data into DRAM in block-raster scan order and displays reads data in raster-scan order. To avoid display under-run problems, software

ensures a phase difference of "~frame_period/2", so that decoder is ahead compare to display.

PL DDR SDI Video Capture and SDI Display

This module enables capture of video from an SDI RX Subsystem implemented in the PL. The video can be displayed through SDI TX through the PL and recorded in SD cards or USB/SATA drives. The module can stream-in or stream-out encoded data through an Ethernet interface. SDI with dynamic frame rate support is enabled.

Note: This module supports single-stream for XV20 (YUV 4:2:2 10-bit) pixel format only. This design also supports interlace mode.

This is the new design approach proposed to use PL_DDR for decoding and PS_DDR for encoding so that DDR bandwidth would be enough to support high-bandwidth VCU applications requiring simultaneous encoder and decoder operations and transcoding at a maximum resolution of 4K @60FPS. This approach makes the most effective use of limited AXI4 read/write issuance capability in minimizing latency for the decoder. DMA buffer sharing requirements determine how capture, display, and intermediate processing stages should be mapped to the PS or PL DDR. PICXO IP is used for synchronization of Tx_out_clk and Rx_out_clk from the GT. SDI supports fractional and integer frame rates.

LLP2 PL DDR SDI Video Capture and SDI Display

This module enables capture of video from an SDI RX subsystem implemented in the PL. The video can be displayed through SDI TX through the PL and recorded in SD cards or USB/SATA drives. The module can stream-in or stream-out encoded data through an Ethernet interface. This module supports single-stream for XV20(YUV 4:2:2 10 bit) pixel format. In this design PL_DDR is used for decoding and PS_DDR for encoding so that DDR bandwidth would be enough to support high bandwidth VCU applications requiring simultaneous encoder and decoder operations and transcoding at 4k @60 FPS

The VCU encoder and decoder operate in slice mode. An input frame is divided into multiple slices (8 or 16) horizontally. The encoder generates a slice_done interrupt at every end of the slice. Generated NAL unit data can be passed to a downstream element immediately without waiting for the frame_done interrupt. The VCU decoder also starts processing data as soon as one slice of data is ready in its circular buffer instead of waiting for complete frame data. The hardware Sync (syncIP) element is responsible synchronizing buffers between the capture DMA and the VCU encoder, and between the VCU decoder and the display element. PICXO IP is used for synchronization of Tx_out_clk and Rx_out_clk from the GT. SDI supports fractional and integer frame rates.

The capture element (FB write DMA) writes video buffers in raster-scan order. SyncIP monitors the buffer level while the capture element is writing into DRAM and allows the encoder to read input buffer data if the requested data is already written by DMA, otherwise it blocks the encoder until DMA completes its writes. On the decoder side, the VCU decoder writes decoded video buffer data into DRAM in block-raster scan order, and

displays reads data in raster-scan order. To avoid display under-run problems, software ensures a phase difference of "~frame_period/2", so that the decoder is ahead compared to the display.

VCU TRD PCIe

This module is used for transcoding MP4 files, encoding the RAW file and decoding the encoded files from the host machine to the client board (zcu106) through the PCIe XDMA bridge interface in the PL. The input file is passed to the VCU encoder and decoder block for the transcode, encode and decode use case. The output file of the transcode, encode and decode use case is written back to the host machine using the PCIe XDMA bridge interface read channel.

The Zynq UltraScale+ MPSoC VCU TRD wiki for 2020.1 provides additional content including:

- Prerequisites for building and running the reference designs.
- Instructions for running the pre-built SD card image on the evaluation board.
- Detailed step-by-step design and tool flow tutorials for each design module.

Design Components

The reference design zip files can be downloaded from the Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit Documentation website. The file contains the following components grouped by APU or PL.

APU

- vcu_apm_lib: Library that provides the interface to query read and write throughput of the VCU encoder/decoder.
- vcu_gst_lib: Interface library that manages the video/audio-video capture, processing, and display pipelines using the GStreamer, V4L2, Advanced Linux Sound Architecture (ALSA) [Ref 6], and DRM frameworks.
- petalinux_bsp: PetaLinux board support package (BSP) to build a pre-configured SMP Linux image for the APU. The BSP includes the following components:
  - First stage boot loader (FSBL)
  - Arm trusted firmware (ATF)
  - U-Boot
  - Linux kernel
  - Device tree

    - PMU firmware
    - Root file system (rootfs).

- vcu_qt: Application that uses the vcu_gst_lib, vcu_apm_lib, and vcu_video_lib libraries and provides a GUI to control and visualize various parameters of this design. The GUI is supported only on DP.
- vcu_video_lib: Library that configures various video pipelines in the design
- vcu_gst_app: Command line application that uses the vcu_gst_lib, vcu_apm_lib, and vcu_video_lib libraries. It allows you to configure and run the capture, display, record, stream in, and stream out pipelines through the command line.
- pcie_transcode: Command-line application that uses the pcie_lib library. It allows you to transcode the MP4 file into ts, encode the RAW file into ts, or decode the encoded file.
- pcie_lib: This library provides abstract APIs for pcie_transcode applications that interact with PCIe user space configuration.
- host_package: The host package installs the PCIe XDMA driver on the host machine. It identifies the PCIe endpoint ZCU106 Board connected to the host machine. This package has the application for sending files from the host machine along with the encoder parameters for transcoding, encoding, or decoding the file on the ZCU106 PCIe endpoint, and writes back the transcoded, encoded, or decoded file to the host machine.

PL

- Vivado: Vivado® IP integrator design that integrates the capture, processing (encode/decode), and display pipeline.

APU Software Platform

Introduction

This chapter describes the application processing unit (APU) Linux software platform, which is further subdivided into a middleware layer, an operating system (OS) layer, and an application stack (see Figure 3-1). The two layers are examined in conjunction because they interact closely for most Linux subsystems. These layers are further grouped by vertical domains which reflect the organization of this chapter:

- Video
- Audio
- Display
- Graphics
- Accelerator
- PCIe

Software Architecture

Figure 3-1 shows the APU Linux software platform.

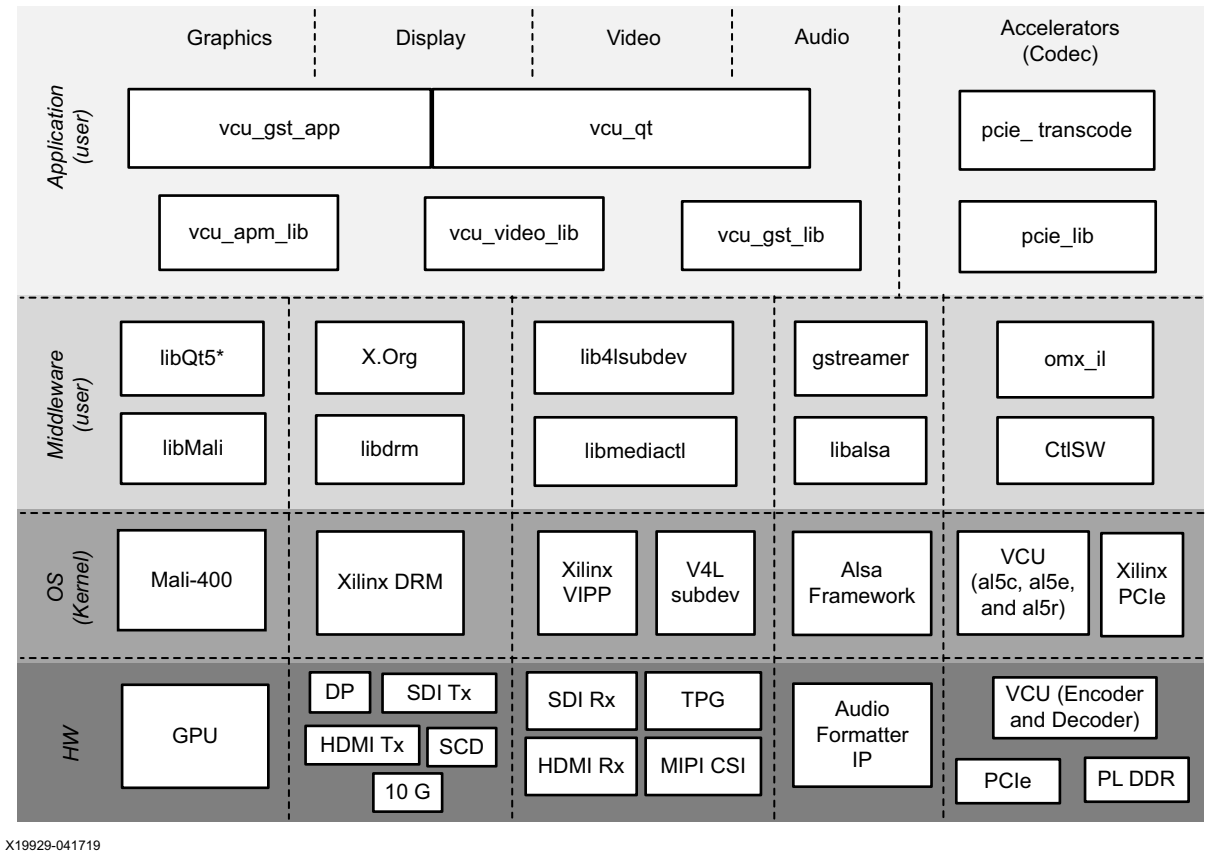

Figure 3-1: **APU Linux Software Platform**

The middleware layer is a horizontal layer implemented in the user-space. It provides the following functionality:

- Interfaces with the application layer
- Provides access to kernel frameworks

The OS layer is a horizontal layer implemented in the kernel-space. It provides the following functionality:

- Provides a stable, well-defined API to user-space
- Includes device drivers and kernel frameworks (subsystems)
- Accesses the hardware

Video

To model and control video capture pipelines such as the ones used in this TRD on Linux systems, multiple kernel frameworks and APIs must work in concert. For simplicity, the overall solution is referred to as Video4Linux (V4L2), although the framework only provides part of the required functionality. Individual components are discussed in the following sections.

Driver Architecture

Figure 3-2 shows the VL42 driver stack (a generic V4L2 driver model of a video pipeline). The video pipeline driver loads the necessary subdevice drivers and registers the device nodes it needs, based on the video pipeline configuration specified in the device tree. The framework exposes the following device node types to user space to control certain aspects of the pipeline:

- Media device node: `/dev/media*`
- Video device node: `/dev/video*`
- V4L subdevice node: `/dev/v4l-subdev*`

Note: The * means [0 . . .n], e.g., `/dev/media1`, `/dev/media2`, and so on.

These steps describe the data flow within software:

1. The V4L2 source driver allocates frame buffer for the capture device.
2. The V4L2 framework imports/exports the DMA_BUF file descriptor (FD) to the next GStreamer element.
3. The encoder reads the source buffer from the capture device, encodes it, and writes the encoded bitstream to a bitstream buffer. The encoded bitstream does not use DMA_BUF framework for sharing the buffer.
4. The decoder allocates a decoded frame buffer, reads the bitstream buffer, and writes the decoded frame buffer into memory.
5. The decoder shares the decoded frame buffer using the DMA_BUF framework with the DRM display device.

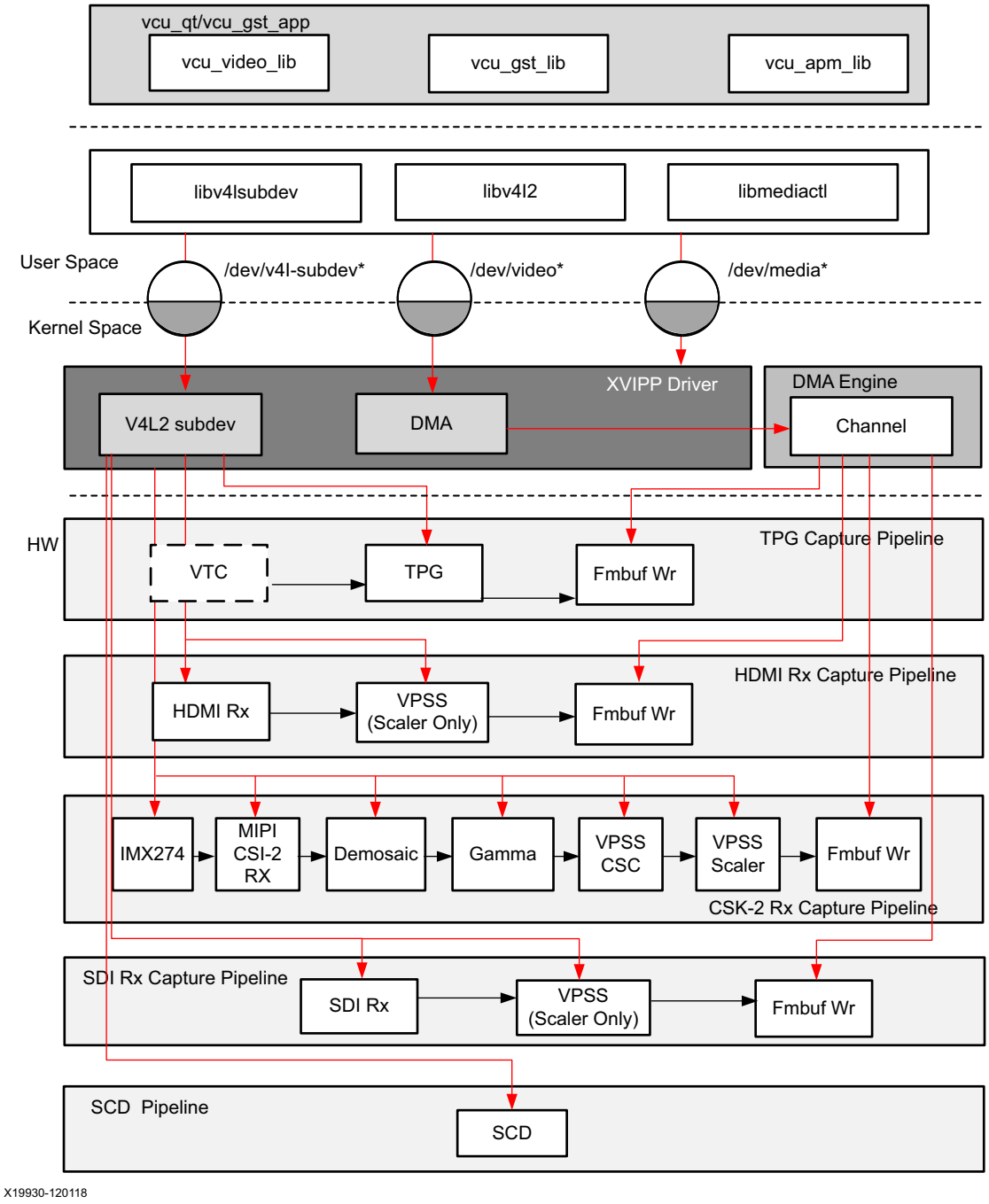

Figure 3-2: **VL42 Driver Stack**

Media Framework

The main goal of the media framework is to discover the device topology of a video pipeline and to configure it at run time. To achieve this, pipelines are modeled as an oriented graph of building blocks called *entities* connected through pads.

A pad is a connection endpoint through which an entity can interact with other entities. Data produced by an entity flows from the entity's output to one or more entity inputs. A link is a point-to-point oriented connection between two pads, either on the same entity or on different entities. Data flows from a source pad to a sink pad.

An entity is a basic media hardware building block. It can correspond to a large variety of blocks such as physical hardware devices (e.g., image sensors), logical hardware devices (e.g., soft IP cores inside the PL), DMA channels, or physical connectors. Physical or logical devices are modeled as subdevice nodes and DMA channels as video nodes.

A media device node is created that allows the user space application to configure the video pipeline and its subdevices through the libmediactl and libv4l2subdev libraries. The media controller API provides this functionality:

- Enumerates entities, pads, and links
- Configures pads
  - Sets media bus format
  - Sets dimensions (width/height)
- Configures links
  - Enable/disable
  - Validates formats

Figure 3-3 shows the media graph for the SDI-RX, TPG, HDMI RX, and CSI RX video capture pipelines as generated by the media-ctl utility. The TPG subdevice is shown in white with its corresponding control interface address and subdevice node in the center. The numbers on the edges are pads and the solid arrows represent active links. The grey boxes are video nodes that correspond to Frame Buffer Write channels, in this case write channels (outputs).

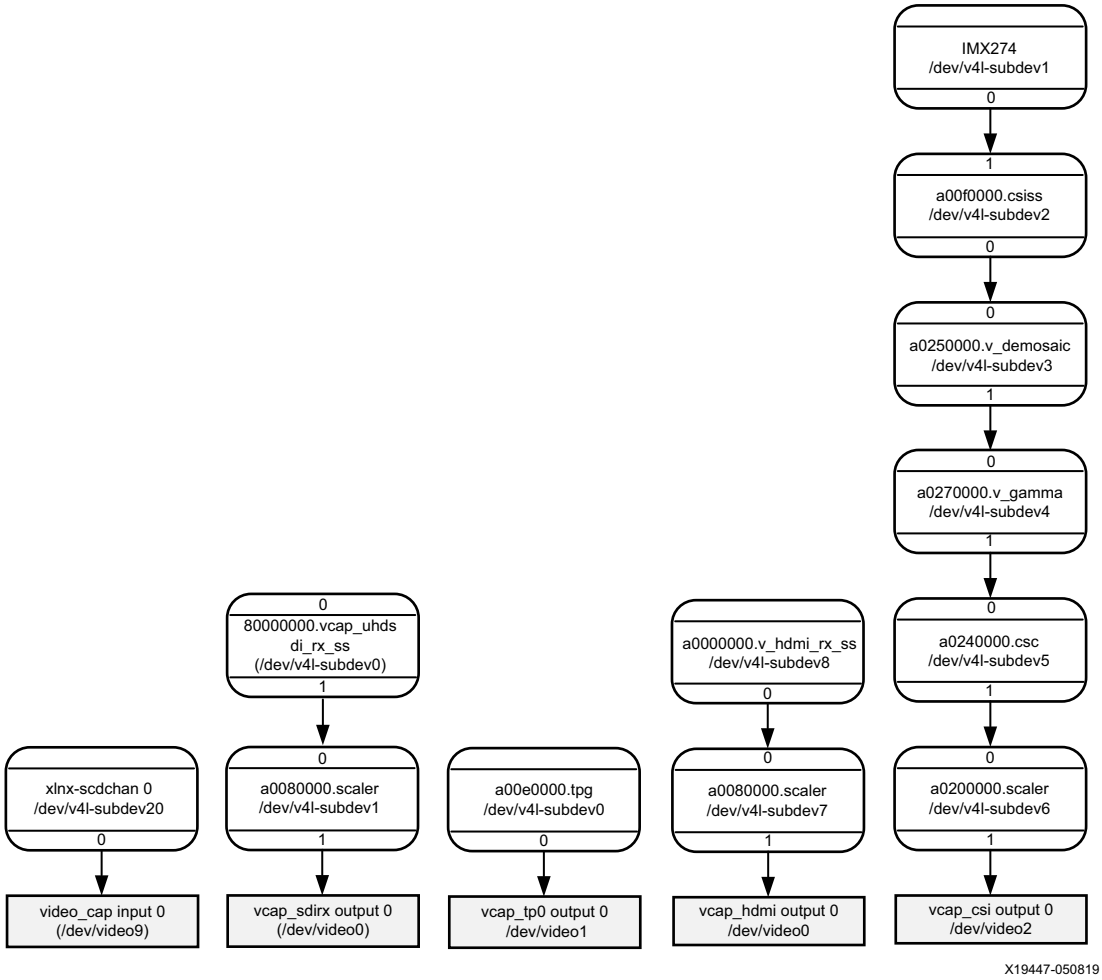

Figure 3-3: **Video Capture Media Pipelines from Left: SDI, TPG, HDMI RX, and CSI RX**

Graphics

Qt is a full development framework with tools designed to streamline the creation of applications and user interfaces for desktop, embedded, and mobile platforms. Qt uses standard C++ with extensions including signals and slots that simplify handling of events. This helps in the development of both the GUI and server applications which receive their own set of event information and should process them accordingly.

Display

Linux kernel and user-space frameworks for display and graphics are intertwined and the software stack can be quite complex with many layers and different standards and APIs. On the kernel side, the display and graphics portions are split with each having their own APIs. However, both are commonly referred to as a single framework, namely DRM/KMS. This split is advantageous, especially for SoCs that often have dedicated hardware blocks for display and graphics. The display pipeline driver responsible for interfacing with the display

uses the kernel mode setting (KMS) API and the GPU responsible for drawing objects into memory uses the direct rendering manager (DRM) API. Both APIs are accessed from user-space through a single device node.

Accelerator

The Video Codec Unit (VCU) core supports multi-standard video encoding and decoding of H.264 and H.265 standards. A software stack on the CPU controls various functions of Encoder and Decoder blocks.

The VCU software stack consists of a custom kernel module and a custom user space library known as Control Software (CtrlSW). The OpenMAX™ (OMX) integration layer (IL) is integrated on top of CtrlSW, and the GStreamer framework is used to integrate the OMX IL component and other multimedia elements (see the OpenMAX website [Ref 5]).

OpenMAX (Open Media Acceleration) is a cross-platform API that provides a comprehensive streaming media codec and application portability by enabling accelerated multimedia components.

GStreamer is the cross-platform/open source multimedia framework. Its core function is to provide a framework for plug-ins, data flow, and media type handling and negotiation. It also provides an API to write applications using the various plug-ins.

You can develop your application at all three levels: CtrlSW, OMX IL, and GStreamer (Figure 3-4).

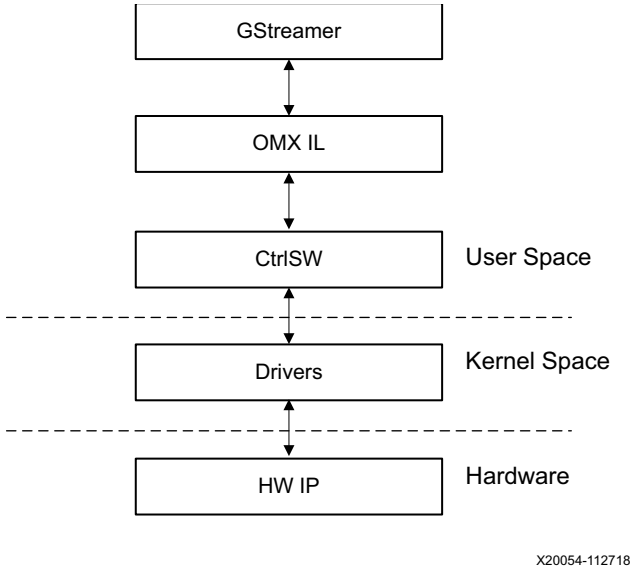

Figure 3-4: **Acceleration Layers**

Audio

Advanced Linux Sound Architecture (ALSA) arranges hardware audio devices and their components into a hierarchy of cards, devices, and subdevices. It reflects the capabilities of our hardware as seen by ALSA.

ALSA cards correspond one-to-one to hardware sound cards. A card can be denoted by its ID or by a numerical index starting at zero.

ALSA hardware access happens at the device level. The devices of each card are enumerated starting from zero.

In this TRD design, sound cards are created for the HDMI-RX capture pipeline, the HDMI-TX playback pipeline, and the I2S RX and SDI RX Capture and I2S TX and SDI TX playback pipelines. See Figure 3-5.

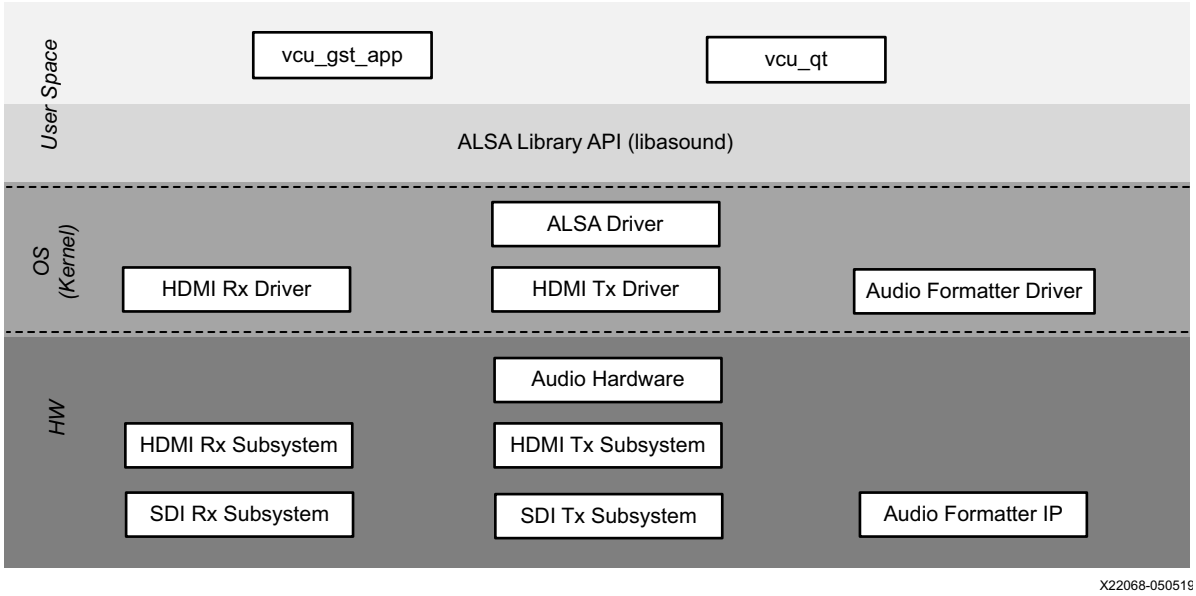

Figure 3-5: **Audio Design**

For this TRD, the supported parameters are:

- Sampling rate: 48 kHz
- Sample width: 24 bits per sample
- Sample encoding: Little endian
- Number of channels: 2
- Supported format: S24_32LE

PCIe

PCIe Software

The Xilinx PCI Express DMA (XDMA) IP provides high-performance scatter gather (SG) direct memory access (DMA) via the Endpoint block for PCI Express. Using this IP and the associated drivers and software enable you to generate high-throughput PCIe memory transactions between a host PC and a Xilinx FPGA (see Figure 3-6).

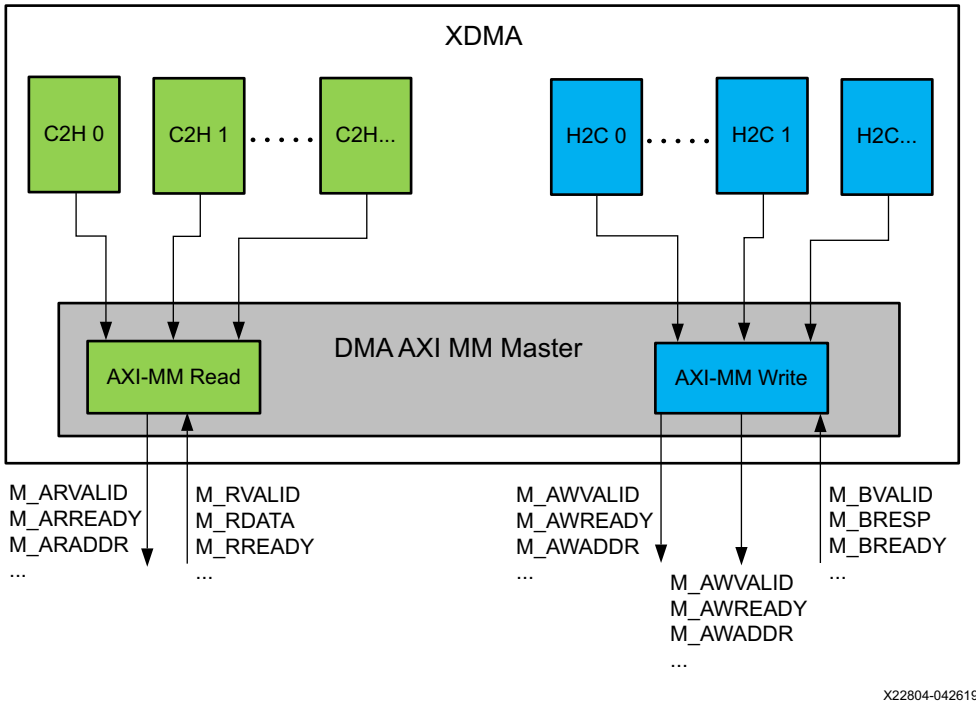

Figure 3-6: **PCI Express DMA (XDMA) IP**

DMA Driver

The purpose of a DMA driver that sits in the host CPU is to prepare for peripheral DMA transfers, because only the operating system (OS) has full control over the memory system, the file system, and the user space processes.

Initially the peripheral device's DMA engine is programmed with the source and destination addresses of the memory ranges to copy. In a read case, the PCIe Endpoint block driver running on the client allocates the destination buffer in the client DDR and passes that address to the host DMA application through userspace registers in the design. When the destination buffer is ready, the DMA application running on the host starts the DMA engine by programming the destination address in the DMA registers.

The device is then signaled to begin the DMA transfer, and when the transfer is finished, the device usually provides interrupts to inform the CPU about completed transfers. For each

interrupt, an interrupt handler, previously installed by the driver, is called and the finished transfer can be acknowledged accordingly by the OS. After the interrupt is raised, the host side DMA application writes the read_transfer_done bit in the user space registers. Based on the read_transfer_done, the pcie_endpoint_client driver running on the client copies the buffer to the user space to perform the next operation.

XDMA Host Linux Driver

The XDMA driver consists of these user accessible devices:

- xdma0_control (to access XDMA registers)
- xdma0_user (to access the AXI-Lite Master interface)
- xdma0_bypass (to access the DMA-Bypass interface)
- xdma0_h2c_0, xdma0_c2h_0 (to access each channel)

The vcu_trd_pcie design XDMA is configured in Memory mode and has one read and write channel. For additional information on the XDMA drivers, refer to Xilinx Answer 71435.

XDMA Host Application

The XDMA host application transfers files from host to client in chunks of buffers using DMA memory-based transfers. The application receives transcoded, encoded, or decoded buffers from the client by using PCIe DMA transfers and creating a new transcoded, encoded, or decoded file.

PCIe Endpoint Client Driver

The PCIe endpoint client driver creates buffers for write/read data from user space and passes that buffer address to the XDMA host application to trigger transfers between host and client. The driver provides IOCTL calls for acquiring the transferring file length from the host to start and stop the data transfer using user space registers in the XDMA.

Software Stack

The APU Linux multimedia software stack is divided into an application layer and a platform layer. The application layer is purely implemented in the Linux user-space whereas the platform layer contains middleware (user-space libraries) and operating system (OS) components (kernel-space drivers). Figure 3-7 shows a simplified version of the Linux software stack. This chapter focuses on the application layer implemented in the user-space.

A user application based on GStreamer demonstrates the features of the TRD. Figure 3-7 shows the software stack present in the TRD. Table 3-1 describes the software components.

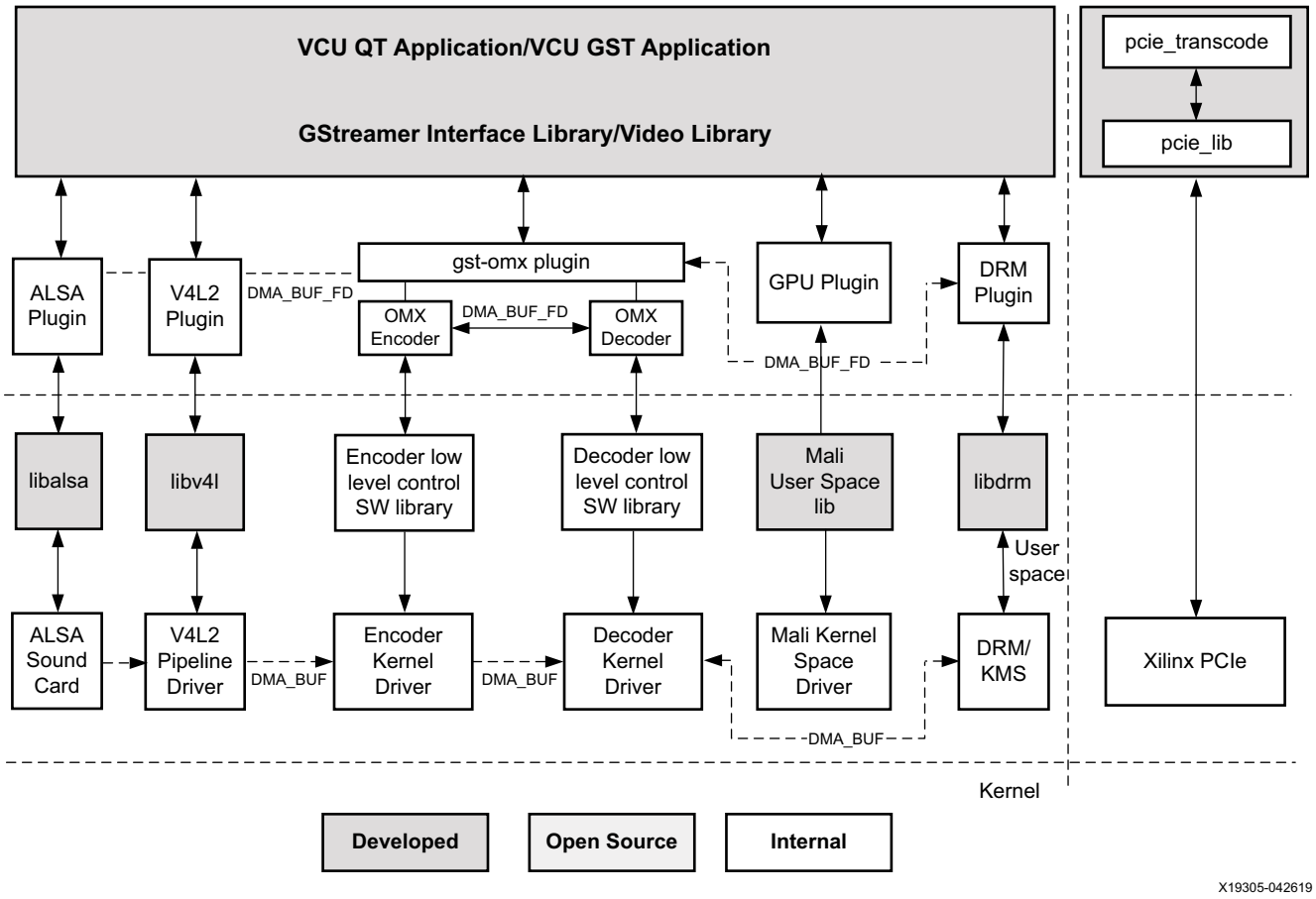

Figure 3-7: **TRD Software Stack**

Table 3-1: **Software Stack Components**

Component	Description
Kernel drivers	This layer contains the kernel drivers for HDMI, Test Pattern Generator (TPG), IMX274 sensor driver, MIPI CSI-2 RX Subsystem, Xilinx Video Demosaic, Xilinx Video Gamma LUT, VPSS Color Space Converter (CSC), Xilinx Video Processing Subsystem (VPSS Only configuration, 2X configuration), HDMI TX Subsystem, HDMI RX Subsystem, Xilinx Video Pipeline (XVIPP), Mixer, VCU, Xilinx PL sound card, Xilinx Audio Formatter, DisplayPort controller, Sync IP, and the Mali GPU.
User space libraries	User space libraries include the media and v4l2 lib for the video pipeline, GStreamer libraries, lib_decode libraries for VCU, libdrm for the DRM device, libalsa, and Mali user-space libraries for the GPU.
OpenMAX v1.1.2	The OpenMAX integration layer (IL) components for encoder and decoder provides an abstraction for VCU to a user space media framework like GStreamer (a complete, cross-platform solution to play, record, convert, and stream audio and video) [Ref 5]. It implements a standard application programming interface (API) for the user space media framework.

Table 3-1: **Software Stack Components *(Cont'd)***

Component	Description
GStreamer framework	GStreamer is the cross-platform/open source multimedia framework, and provides the infrastructure to integrate multiple multimedia components and create pipelines. Various GStreamer plug-ins are used for input, filter, and display components.

The TRD application (vcu_qt) is a multi-threaded Linux application with the following main tasks:

- Displays unprocessed video from one or more sources.
- Applies a processing function (encode/decode).
- Provides a GUI for user input.
- Interfaces with lower level layers in the stack to control video pipeline parameters and video data flow.

The application consists of multiple components that have been specifically developed for the VCU TRD (see Figure 3-8). These interfaces are explained in more detail in subsequent sections:

- GUI application (vcu_qt)
- GStreamer interface library (vcu_gst_lib)
- Video library (vcu_video_lib)
- AXI Performance Monitor (APM) library (vcu_apm_lib)
- GStreamer command line application (vcu_gst_app)
- PCIe command line app (pcie_transcode)
- PCIe library (pcie_lib)

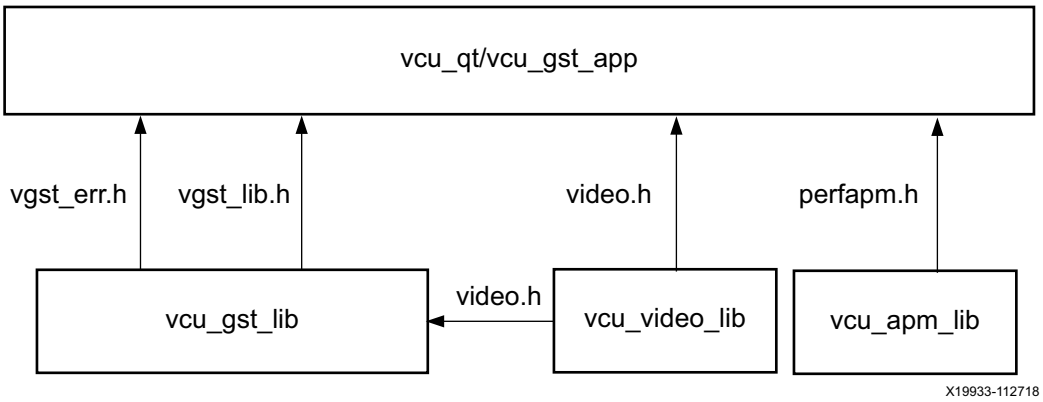

Figure 3-8: **Video Application Interfaces**

GUI Application (vcu_qt)

The vcu_qt application is a multi-threaded Linux application that uses the Qt graphics toolkit to render a GUI. The GUI provides control knobs for user input and a display area to show the captured video stream. The GUI shown in Figure 3-9 contains the following control elements displayed on top of the video output area:

- Control bar (top)
- Video info panel (top-right)
- System performance panels (bottom)

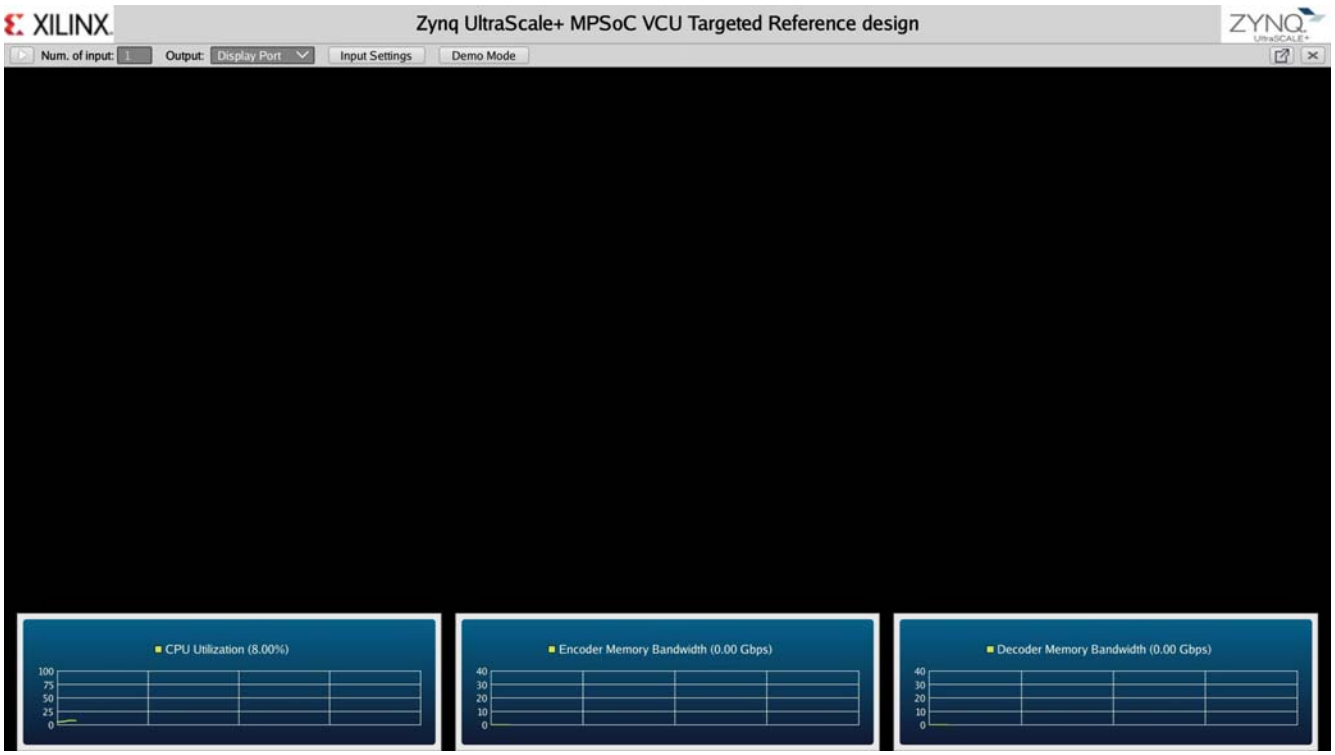

Figure 3-9: **Control Elements Displayed on Top of the Video Output Area**

Number of Inputs

This determines the number of active video sources. In the current version of the TRD, a maximum of four sources are supported (the default value is one). See Figure 3-10 for input settings.

Figure 3-10: **Input Settings**

Input Options

The following 4K/1080p video sources (3840 x 2160) are available:

- HDMI: Implemented in the PL
- File: TS/MP4/MKV streams reside in the SD card or USB/SATA drives
- Test Pattern Generator (TPG): Implemented in the PL
- CSI: Implemented in the PL (option: MIPI: MIPI CSI, model LI-IMX274 MIPI-FMC v1.1)
- SDI: Implemented in the PL
- Stream In: Stream from network or Internet

Codec

- **Enc**—This option selects encoder in the pipeline.
- **Enc-Dec**—This option selects encode and decode in the pipeline.
- **Pass-through**—This option selects displaying the raw video source.

SCD

- **Enable**—To enable SCD
- **Disable**—To disable SCD

Preset

There are six predefined presets. If you edit any control options, preset the mode switches to **Custom**. See Table 3-2.

Table 3-2: **Predefined Preset Descriptions**

Preset	Description
AVC Low (1)	Encoder type = H264, bitrate = 10 Mb/s
AVC Medium (1)	Encoder type = H264, bitrate = 30 Mb/s
AVC High (1)	Encoder type = H264, bitrate = 60 Mb/s
HEVC Low (1)	Encoder type = H265, bitrate = 10 Mb/s
HEVC Medium (1)	Encoder type = H265, bitrate = 30 Mb/s
HEVC High (1)	Encoder type = H265, bitrate = 60 Mb/s
Custom	User-specific options

Notes:

1. The following settings are common for these options: Profile = **High** for H264 and **Main** for H265, Rate control = **CBR**, Filler data = **true**, QP =**auto**, L2 cache = **true**, Latency mode = **Normal**, Low bandwidth = **false**, GoP (Group of Pictures) mode = **Basic**, B-frame = **0**, Slice = **8,** and GoP length = **60** (see Figure 3-11).

Output Options

This option allows you to select the sink for the pipeline. Supported output sink types are:

- DisplayPort
- Record
- Stream Out

For the **DisplayPort** output option, either the **enc-dec Codec** or **Pass-through** option can be selected.

For the **Record** and **Stream Out** output options, only **Encode** can be selected in **Codec**.

Demo Mode

By clicking on this button, the button text state changes to stop and you can play all pipelines (TPG, MIPI, HDMI) with raw and preset configurations.

Every ten seconds, playback preset changes and plays in a loop until you click the **stop** button.

If no source is connected, an error popup displays.

If any error returns in any playback, the demo skips and continues to play other pipelines.

With HDMI AUDIO design, to enable audio in a Demo mode, go to **Input Settings**. Select **Input1: HDMI > Settings > Audio Settings > Enable Audio > true**, and click **Demo Mode**.

Settings

You can control the Encoder, Record, and Stream Out configuration from the GUI (see Figure 3-11 and Table 3-3). Settings options are enabled when the pipeline is in the stop state.

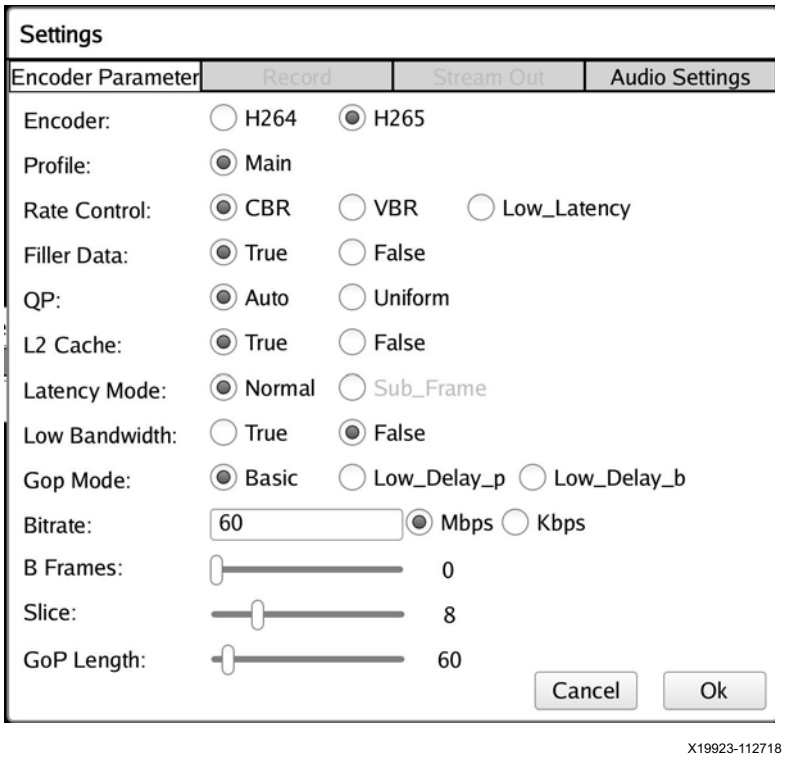

Figure 3-11: **Encoder Parameter Panel**

Table 3-3: **Encoder Parameter Panel Settings**

Encoder Parameter	Setting
Encoder	This can be either H264 or H265.
Profile	The standard defines a sets of capabilities, which are referred to as profiles, targeting specific classes of applications. These are declared as a profile code (profile_idc) and a set of constraints applied in the encoder. This allows a decoder to recognize the requirements to decode that specific stream. H264 supports Baseline, Main, and High profile. In H265, only the Main profile is supported.
QP	Quantization in an encoder is controlled by a quantization parameter. It specifies how to generate the QP per coding unit (CU). Two modes are supported: • Uniform: All CUs of the slice use the same QP • Auto: The QP is chosen according to the CU content using a pre-programmed lookup table.

Table 3-3: **Encoder Parameter Panel Settings *(Cont'd)***

Encoder Parameter	Setting
Rate Control	Selects the way the bit rate is controlled: CBR: Use *constant bit rate* control. VBR: Use *variable bit rate* control. LOW_LATENCY: Use variable bit rate for low latency application.
Bitrate	Encoding bitrate. In digital multimedia, bitrate often refers to the number of bits used per unit of playback time to represent a continuous medium such as audio or video after source coding (data compression).
B-frame	Short for bidirectional frame or bidirectional predictive frame, a video compression method used by the MPEG standard. The setting ranges from 0 to 4.
Slice	Number of slices produced for each frame. Each slice contains one or more complete macroblock/CTU row(s). Slices are distributed over the frame as regularly as possible. If slice size is also defined, more slices can be produced to fit the slice size requirement. Range: • 4-22 4K resolution with HEVC codec • 4-32 4K resolution with AVC codec • 4-32 1080p resolution with HEVC codec • 4-32 1080p resolution with AVC codec
GoP Length	In video coding, a group of pictures, or GoP structure, specifies the order in which intra- and inter-frames are arranged. And GoP Length is a length between two intra-frames. The GoP is a collection of successive pictures within a coded video stream. Each coded video stream consists of successive GoPs from which the visible frames are generated. Its range is from 1– 1000. The GoP length must be a multiple of B-Frames+1.
Filler Data	Filler data network abstraction layer (NAL) units for CBR rate control. It can be enabled or disabled. Applies to CBR mode only.
L2 Cache	Enable or disable L2cache buffer in the encoding process.
Latency Mode	Encoder latency mode. It can be normal or sub_frame mode.
Low Bandwidth	If enabled, decreases the vertical search range used for P-frame motion estimation to reduce the bandwidth.
GoP Mode	Group of Pictures mode. It can be Basic, low_delay_p, or low_delay_b.

Record

The Record panel allows you to configure recording parameters. See Figure 3-12 and Table 3-4.

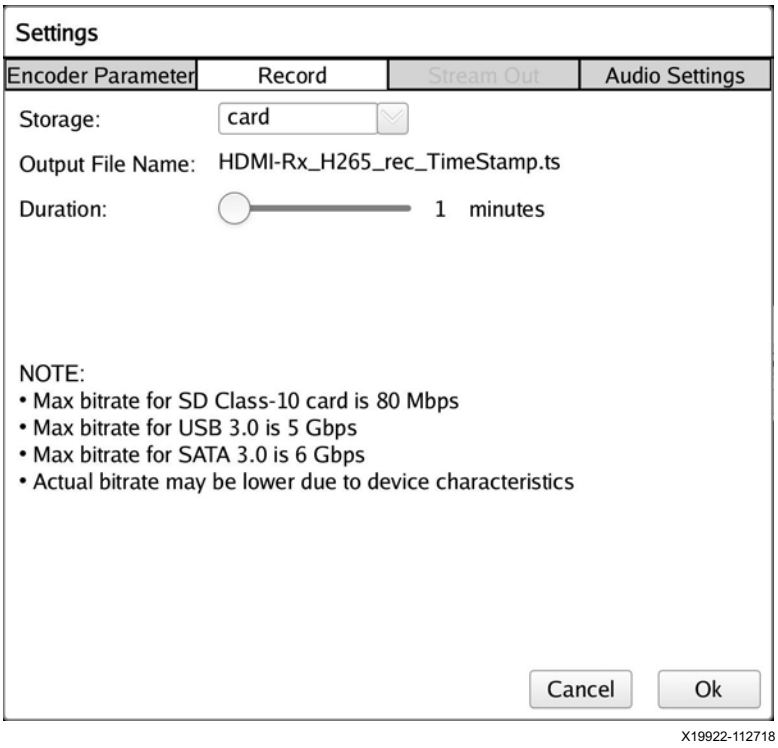

Figure 3-12: **Record Panel**

Table 3-4: **Record Panel Settings**

Parameter	Setting
Storage	This option specifies the storage device for the recorded file. The list is dynamically populated based on mounted storage devices. Supported storage devices include SD cards and USB/SATA drives. ***Note:*** Because of speed and storage constraints, using SATA or USB 3.0 with an ext4-formatted storage device is recommended for recording.
Output File Name	Name of the output file. A recorded file is saved as *source*_H26x_rec_<*timestamp*>ts where *source* can be *HDMI*, *TPG*, or *MIPI* and *codec* can be H264/H265.
Duration	This option specifies the recording time duration. It ranges from 1–3 minutes.

Stream Out

The Stream Out panel allows you to configure streaming parameters. See Figure 3-13 and Table 3-5.

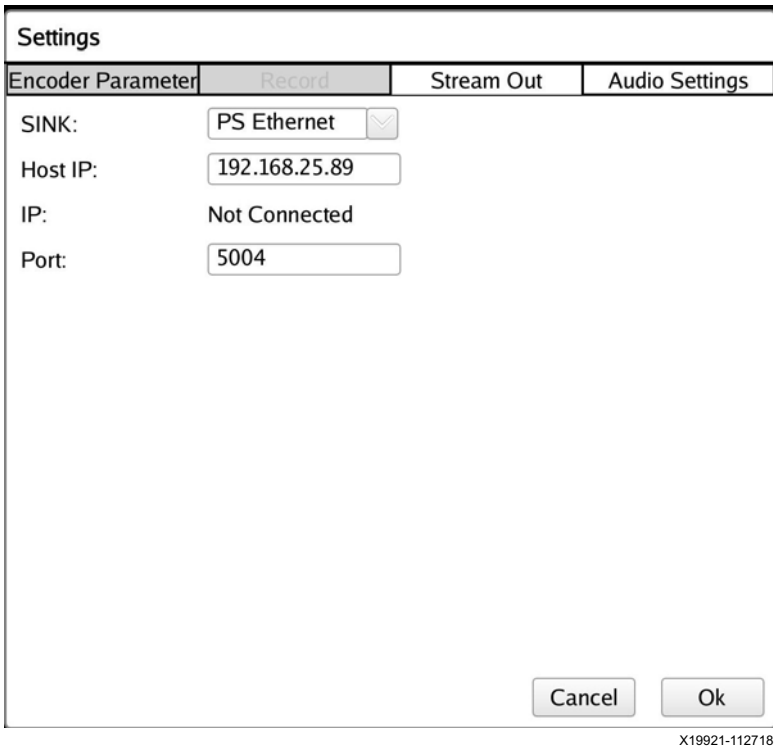

Figure 3-13: **Stream Out Panel**

Table 3-5: **Stream Out Panel Settings**

Parameter	Setting
SINK	Provides the sink option for the stream out case. It is set to **PS Ethernet**.
Host IP	Provides the option to enter the Host IP address.
IP	Shows the IP address of the board if the Ethernet link is up. If no Ethernet link is connected, it shows **Not Connected.**
Port	Port number of the Ethernet link. The default is **5004**.

Note: IDR is not user configurable. In the encoder code, the idr value = gop-length.

Audio Setting

The Audio Settings panel is shown in Figure 3-14 and described in Table 3-6.

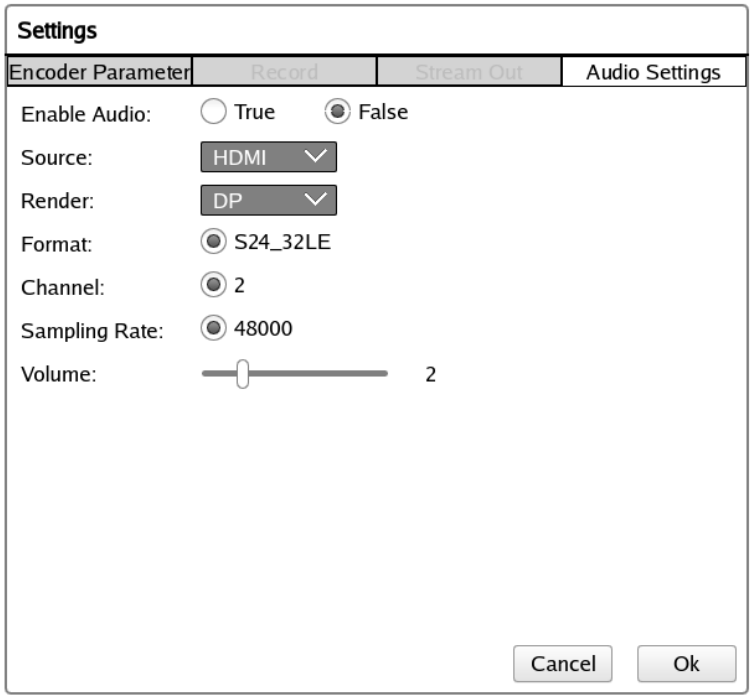

Figure 3-14: **Audio Setting Panel**

Table 3-6: **Audio Settings**

Parameter	Setting
Enable Audio	Enable or disable audio in pipeline.
Format	Audio format. Currently S24_32LE format is supported.
Channel	Number of audio channels. Currently two channels are supported.
Sampling Rate	Audio sampling rate. Currently 48000 is supported.
Volume	Audio volume. Ranges from 0 to 10, default value is 2.
Source	Available sources are HDMI and I2S
Renderer	Available renderers are DP and I2S

Monitor Options

The GUI monitors CPU utilization and bandwidth utilization for encoder and decoder AXI ports. See Figure 3-15 through Figure 3-17.

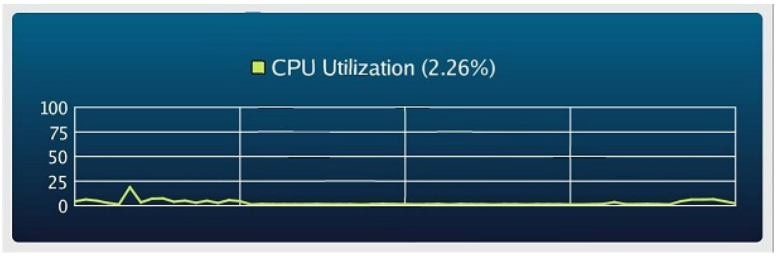

Figure 3-15: **CPU Utilization Plot**

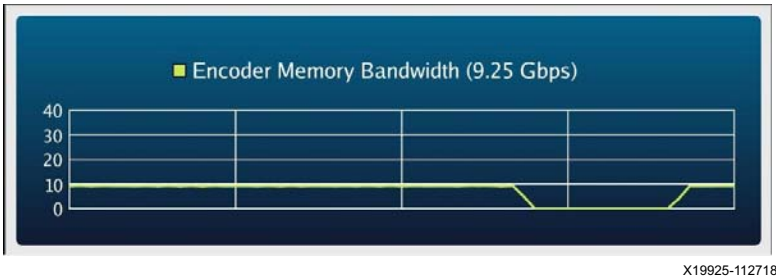

Figure 3-16: **Encoder Bandwidth Utilization Plot**

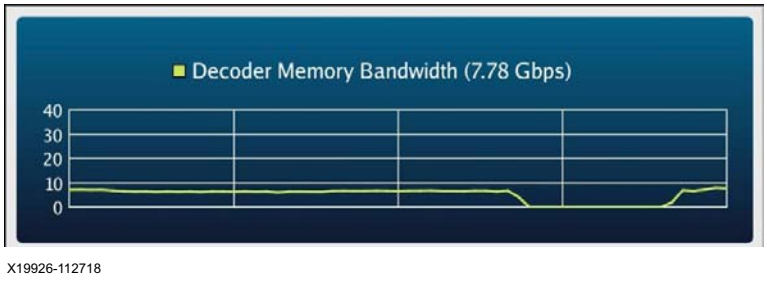

Figure 3-17: **Decoder Bandwidth Utilization Plot**

GStreamer Application (vcu_gst_app)

The vcu_gst_app is a command line multi-threaded Linux application that uses the vcu_gst_lib interface similar to vcu_qt. The difference is to manually feed the input configuration and run the pipeline each time, whereas with vcu_qt, the application has to launch only once.

The command line application requires an input configuration file (`input.cfg`) to be provided in plain text. Refer to Appendix A, Input Configuration File for the file format and description.

GStreamer Interface Library (vcu_gst_lib)

The VCU GStreamer interface configures various video pipelines in the design and controls the data flow through these pipelines. It implements these features:

- Display configuration
- VCU configuration
- Video pipeline control
- Audio pipeline control
- Video buffer management

The VCU GStreamer interface library exports interfaces that:

- set video pipeline parameters such as resolution, format, and source type (v4l2src, filesrc)
- set encoder parameters
- start and stop the pipeline
- calculate FPS
- perform error handling
- calculate bit rate for file/stream-in playback
- poll for an end of stream (EOS) event

Description

GStreamer is a library for constructing graphs of media-handling components. The applications it supports range from simple playback and audio/video streaming to complex audio (mixing) and video processing.

GStreamer uses a plug-in architecture which makes the most of GStreamer functionality implemented as shared libraries. The GStreamer base functionality contains functions for registering and loading plug-ins and for providing the fundamentals of all classes in the form of base classes. Plug-in libraries get dynamically loaded to support a wide spectrum of codecs, container formats, and input/output drivers.

Table 3-7 describes the plug-ins used in the GStreamer interface library.

Table 3-7: **GStreamer Plug-ins**

Plug-in	Description
v4l2src	v4l2src can be used to capture video from V4L2 devices like Xilinx HDMI-RX and TPG. Example pipeline: `gst-launch-1.0 v4l2src ! kmssink` This pipeline shows the video captured from a `/dev/video0` and rendered on a display unit.
kmssink	The kmssink is a simple video sink that renders raw video frames directly in a plane of a DRM device. Example pipeline: `gst-launch-1.0 v4l2src ! "video/x-raw, format=NV12, width=3840, height=2160" ! kmssink`
omxh26xdec	Decoder omxh26xdec is a hardware-accelerated video decoder that decodes encoded video frames. Example pipeline: `gst-launch-1.0 filesrc location=/media/card/abc.mp4 ! qtdemux ! h26xparse ! omxh26xdec ! kmssink` This pipeline shows an `.mp4` multiplexed file where the encoded format is h26x encoded video. ***Note:*** Use omxh264dec for H264 decoding and omxh265dec for H265 decoding.
omxh26xenc	Encoder omxh26xenc is a hardware-accelerated video encoder that encodes raw video frames. Example pipeline: `gst-launch-1.0 v4l2src ! omxh26xenc ! filesink location=out.h26x` This pipeline shows the video captured from a V4L2 device that delivers raw data. The data is encoded to the h26x encoded video type and stored to a file. ***Note:*** Use omxh264enc for H264 encoding and omxh265enc for H265 encoding.
alsasrc	The alsasrc plug-in can be used to capture audio from audio devices like Xilinx HDMI-RX. Example pipeline: `gst-launch-1.0 alsasrc device=hw:1,1 ! queue ! audioconvert ! audioresample ! audio/x-raw, rate=48000, channels=2, format=S24_32LE ! alsasink device="hw:1,0"` This pipeline shows the audio captured from an ALSA source and plays on an ALSA sink.
alsasink	The alsasink is a simple audio sink that plays raw audio frames. Example pipeline: `gst-launch-1.0 alsasrc device=hw:1,1 ! queue ! audioconvert ! audioresample ! audio/x-raw, rate=48000, channels=2, format=S24_32LE ! alsasink device="hw:1,0"` This pipeline shows the audio captured from the ALSA source and plays on an ALSA sink.

Table 3-7: **GStreamer Plug-ins *(Cont'd)***

Plug-in	Description
faad	Decoder faad is an audio decoder that decodes encoded audio frames. Example pipeline: `gst-launch-1.0 filesrc location=out.ts ! tsdemux ! aacparse ! faad ! audioconvert ! audioresample ! audio/x-raw,rate=48000,channnels=2, format=S24_32LE ! alsasink device="hw:1,0"` This pipeline shows a `.ts` multiplexed file where the encoded format is aac encoded audio. The data is decoded and played on an ALSA sink device.
faac	Encoder faac is an audio encoder that encodes raw audio frames. Example pipeline: `gst-launch-1.0 alsasrc device=hw:1,1 num-buffers=500 ! audio/x-raw, format=S24_32LE, rate=48000 ,channels=2 ! queue ! audioconvert ! audioresample ! faac ! aacparse ! mpegtsmux ! filesink location=out.ts` This pipeline shows the audio captured from an ALSA device that delivers raw data. The data is encoded to aac format and stored to a file.
xilinxscd	xilinxscd is hardware-accelerated IP that enables detection of scene change in a video stream. This plugin generates upstream events whenever there is scene change in an incoming video stream so the encoder can insert an Intra frame to improve video quality. Example pipeline: `gst-launch-1.0 -v v4l2src ! video/x-raw, width=3840, height=2160, format=NV12, framerate=60/1 ! xilinxscd io-mode=5 ! omxh26xenc ! filesink location=/run/out.h26x` This pipeline shows the video captured from a V4L2 device that delivers raw data. This raw data is passed through the xilinxscd plugin which analyzes the stream in runtime and provides an event to the encoder whether any scene change is detected in a video stream or not. The encoder uses this information to insert an I-frame in an encoded bit-stream. Use omxh264enc for H264 encoding and omxh265enc for H265 encoding.
appsrc	The appsrc element can be used by applications to insert data into a GStreamer pipeline. Unlike most GStreamer elements, appsrc provides external API functions.
appsink	appsink is a sink plugin that supports many different methods, enabling the application to manage the GStreamer data in a pipeline. Unlike most GStreamer elements, appsink provides external API functions.

VCU Encoder/Decoder Features

The VCU encoder/decoder supports the following major features:

- Region of Interest Encoding: Region of Interest Encoding tags regions in a video frame to be encoded with user supplied quality (high, medium, low, and don't-care) relative to the picture background (untagged region).
- Scene Change Detection: SCD IP generates the SCD event which is passed along with the buffer to the encoder, where the encoder makes decisions to insert an I-frame instead of a P-frame or a B-frame. Inserting an I-frame at the place where a scene is changed would retain the quality of the video.
- Interlaced Video: VCU supports encoding and decoding of H265 interlaced video.

- DCI-4k Encode/Decode: VCU is capable of encoding or decoding at 4096x2160p60, provided the VCU Core-clk frequency is set to 712 MHz.
- Low-Latency (LLP1): The frame is divided into multiple slices; the VCU encoder output and decoder input are processed in slice mode. The VCU Encoder input and Decoder output still works in frame mode.
- Xilinx Low-Latency (LLP2): The frame is divided into multiple slices; the VCU encoder output and decoder input are processed in slice mode. sThe producer (Capture DMA) and the consumer (VCU Encoder) works on the same buffer by having synchronization IP in place. There is no sync IP in between decoder and display. The decoder will signal the display component when half of input frame is ready.
- XAVC: The VCU encoder can produce xAVC Intra or xAVC Long GOP compliant bitstreams. XAVC is mainly used for video record use-cases, as h264parse does not support XAVC parsing. Serial and streaming use-cases might not be valid as the parser does not support these.

Note: For more information on VCU Encoder and Decoder features, Refer to the *H.264/H.265 Video Codec Unit LogiCORE IP Product Guide* (PG252) [Ref 22].

Multi-Stream

When the number of inputs is more than one in the command line application, it is a multi-stream use case. In multi-stream use cases, multiple HDMIs are the same replica of a single source. Table 3-8 provides information about the multi-stream features supported in various designs.

Table 3-8: **Multi-Stream Features**

Design	Input Source	Format	Resolution
Full-fledged VCU TRD	HDMI-RX MIPI TPG	NV12	4kp60 2-4kp30 4-1080p60 8-1080p30
PL DDR HDMI Video Capture and HDMI Display	HDMI-RX	XV20	4kp60 2-4kp30 4-1080p60
LLP2 PS DDR NV12 HDMI Audio Video Capture and HDMI Display	HDMI-RX	NV12	4kp60 2-4kp30 4-1080p60 for encoder 2-1080p60 for decoder
LLP2 PL DDR NV16 HDMI Video Capture and HDMI Display	HDMI-RX	NV16	4kp60 2-4kp30 1080p60 for encoder 4-1080p60 for decoder

Table 3-8: ***(Cont'd)*Multi-Stream Features**

Design	Input Source	Format	Resolution
LLP2 PL DDR XV20 HDMI Video Capture and HDMI Display	HDMI-RX	XV20	4kp60 2-4kp30 4-1080p60 for encoder 2-1080p60 for decoder
Multi-Stream Audio Design	HDMI-RX MIPI	NV12	4kp60 2-4kp30 2-1080p60

Note: When using Low Latency mode (LLP1/LLP2), the encoder and decoder are limited by the number of internal cores. The encoder has a maximum of four streams and the decoder has a maximum of two streams.

The command line application supports multi-streaming, multi-recording, or multi-display. Figure 3-18 shows a use case of the vcu_gst_app running three HDMI and one MIPI in multi-stream in 1080p60 resolution. For 4-1080p60 input, the source type can be TPG, MIPI, or HDMI. For multi-streaming or multi-recording, the source type can be TPG, HDMI, or MIPI.

Figure 3-19 shows a use case of the vcu_gst_app running seven HDMI and one MIPI in multi-stream in 1080p30 resolution. For 8-1080p30 input, the source type can be MIPI or HDMI. Here only half of each stream is displayed to showcase eight different streams on a single screen.

X20153-112718

Figure 3-18: **Multi-Stream—3 HDMI and 1 MIPI Input Sources @ 1080p60**

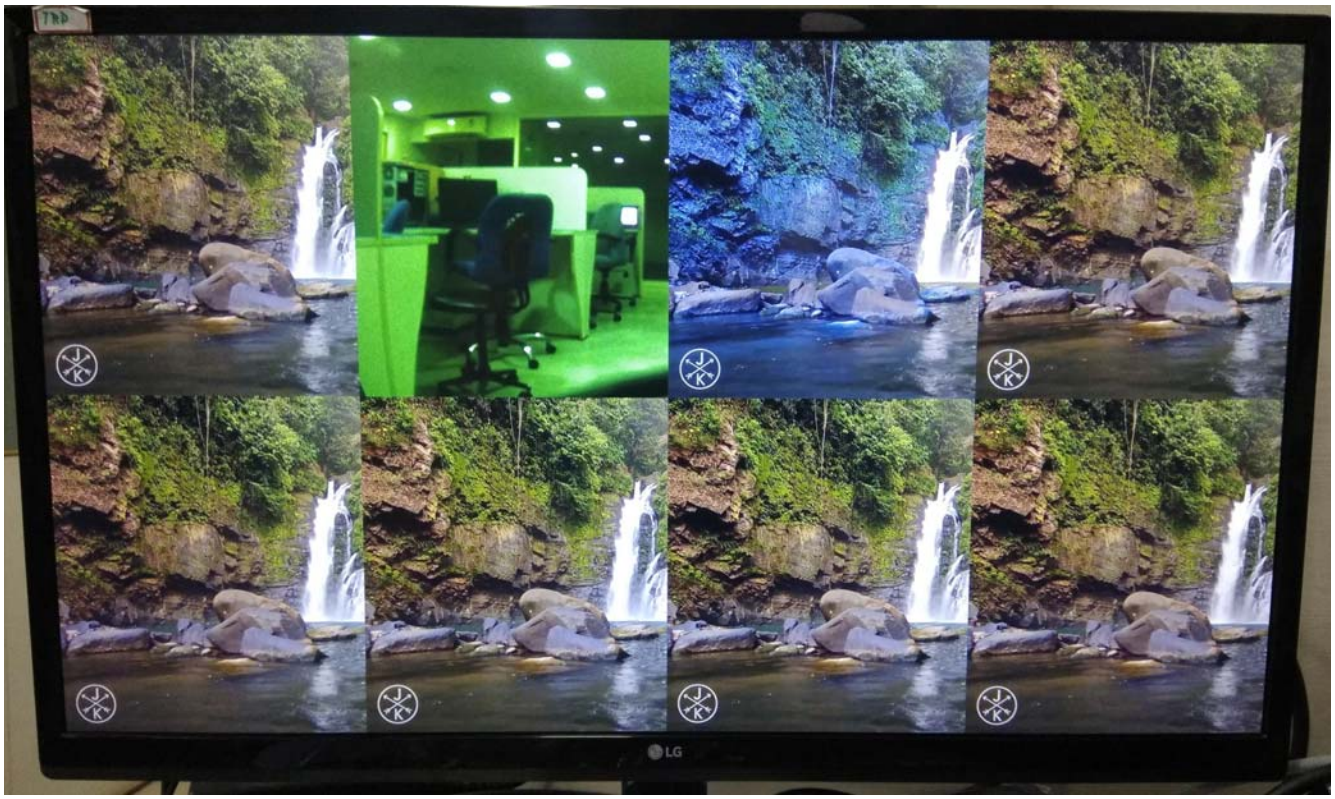

Figure 3-19: **Multi-Stream—7 HDMI and 1 MIPI Input Sources @ 1080p30**

Video Buffer Management

In the case of a raw/processed pipeline, the video capture device (v4l2src), video processing accelerator (VCU element), and kmssink plugin use DMABUF framework for sharing buffers between peer elements (see Figure 3-20).

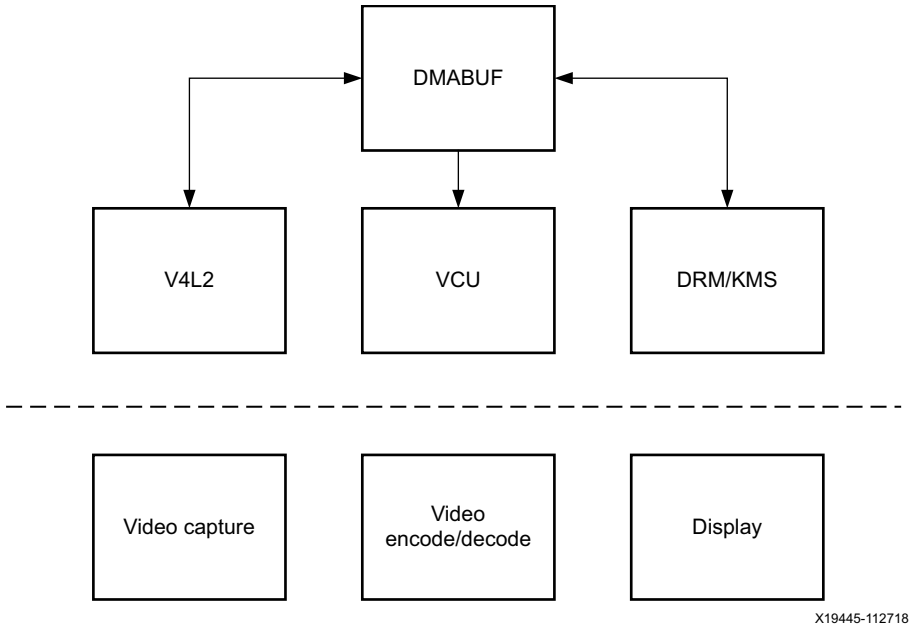

Figure 3-20: **Buffer Sharing**

The following steps are performed in DMA buffer sharing.

In the capture-encode side:

1. The V4L2 capture device (the client driver) allocates buffer.
2. The v4l2src plug-in exports/imports the DMA buffer to the gst-omx plug-in.
3. The gst-omx plug-in passes the file descriptor to the encoder driver.
4. The encoder driver uses the DMA_BUF framework and reads the kernel buffer for encoding.

In the playback side:

1. The decoder driver allocates DMA buffer.
2. The gst-omx plug-in exports the file descriptor (FD) to the kmssink plug-in.
3. The kmssink plug-in passes the file descriptor to the DisplayPort controller driver.
4. The DisplayPort driver uses the kernel DMA_BUF framework to know the decoder buffer location.
5. The DisplayPort DMA reads the decoded buffer without copying the buffer in kernel memory.

AXI Performance Monitor (APM) Library (vcu_apm_lib)

This library provides an interface to the vcu_qt application for reading VCU encoder/decoder memory throughput performance numbers.

The programming model:

1. Calls perf_monitor_init() on startup.
2. Periodically calls perf_monitor_get_rd_wr_cnt() for each VCU APM. This API returns the number of read+write transactions happening on the AXI Performance Monitor port in bytes.
3. Calls perf_monitor_deinit() on exit.

Video Library (vcu_video_lib)

The vcu_video_lib library configures various video pipelines in the design and implements:

- query display configurations
- media pipeline configuration for video capture

The vcu_video_lib library exports and imports the following interfaces:

- TPG video source controls (to vcu_gst_lib library)
- CSI video source controls (to vcu_gst_lib library)
- interfaces from various middleware layers (V4L2, Media Controller, DRM)

Query Display Configurations

The libdrm library is used to validate if the resolution if supported by the monitor and to query the native resolution of the monitor. The graphics plane is configured by the Qt EGLFS backend outside of this library. The pixel format for each of the two planes is configured statically in the device tree.

Media Pipeline Configuration

The video capture pipeline present in this design is a TPG/HDMI/MIPI/SDI/SCD Input. It implements a media controller interface that allows you to configure the media pipeline and its sub-devices. The libmediactl and libv4l2subdev libraries provide the following functionality:

- Enumerate entities, pads, and links
- Configure sub-devices
  - Set media bus format
  - Set dimensions (width/height)

The video_lib library sets the media bus format and video resolution on each sub-device source and sink pad for the entire media pipeline. The formats between pads that are connected through links need to match.

System Considerations

This chapter describes the boot process and address mapping.

Boot Process

The reference design uses a non-secure boot flow and SD boot mode. The sequence diagram in Figure 4-1 shows the exact steps and order in which the individual boot components are loaded and executed.

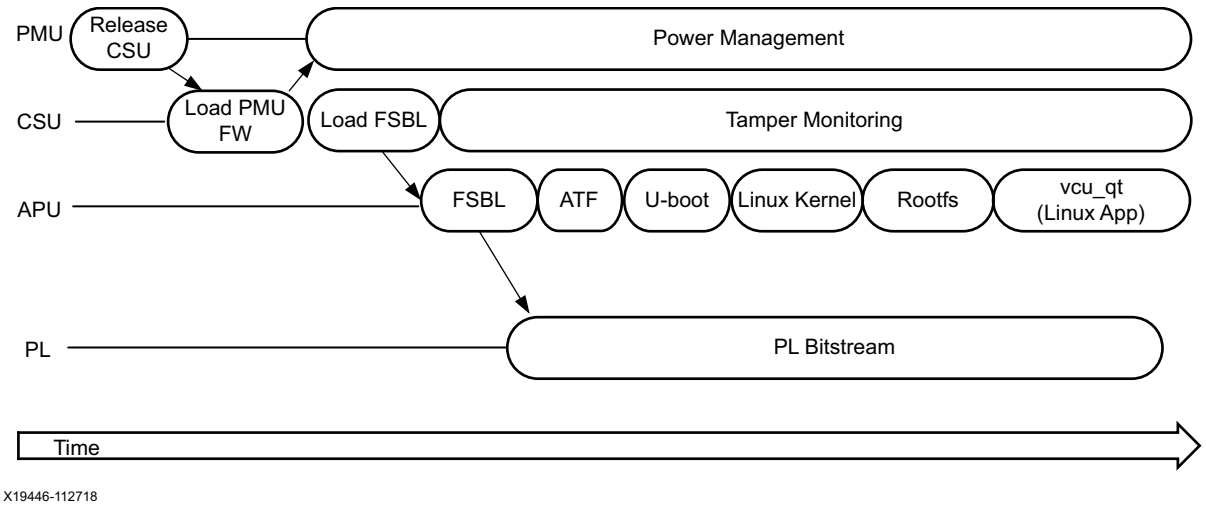

Figure 4-1: **Boot Flow Sequence**

The platform management unit (PMU) is responsible for handling primary pre-boot tasks and is the first unit to wake up after power-on reset (POR). After the initial boot process, the PMU continues to run and is responsible for handling various clocks and resets of the system as well as system power management. In the pre-configuration stage, the PMU executes the PMU ROM and releases the reset of the configuration security unit (CSU). It then enters the PMU server mode where it monitors power.

The CSU handles the configuration stages and executes the boot ROM as soon as it comes out of reset. The boot ROM determines the boot mode by reading the boot mode register, it initializes the on-chip memory (OCM), and reads the boot header. The CSU loads the PMU firmware into the PMU RAM and signals to the PMU to execute the firmware, which

provides advanced management features instead of the PMU ROM. It then loads the first stage boot loader (FSBL) into OCM and switches into tamper monitoring mode.

In this design, the FSBL is executed on APU-0. It initializes the PS and configures the PL and APU based on the boot image header information. The following steps are performed:

1. The PL is configured with a bitstream and the PL reset is deasserted.
2. The Arm trusted firmware (ATF) is loaded into OCM and executed on APU-0.
3. The second stage boot loader U-Boot is loaded into DDR to be executed by APU-0.

Note: At this point, RPU-1 is still held in reset because no executable has been loaded thus far.

For more information on the boot process, see chapters *Programming View of Zynq UltraScale+ MPSoC Devices* and *System Boot and Configuration* in *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137) [Ref 7], and chapter *Boot and Configuration* in *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 8].

Global Address Map

For more information on system addresses, see chapter 8 in *Zynq UltraScale+ MPSoC Technical Reference Manual* (UG1085) [Ref 8].

Memory

The DMA instances in the PL use a 36-bit address space so they can access the DDR Low and DDR High address regions for receiving and transmitting video buffers to be shared with the APU application. Table 4-1 lists the APU software components used in this design and where they are stored or executed from in memory.

Table 4-1: **Software Executables and Their Memory Regions**

Component	Processing Unit	Memory
FSBL	APU-0	OCM
Arm trusted firmware (ATF)	APU-0	OCM
U-boot	APU-0	DDR
Linux kernel/device tree/rootfs	APU (SMP)	DDR
vcu_qt application (Linux)	APU (SMP)	DDR

Video Buffer Format

The TRD uses two layers (or planes) for DisplayPort TX and up to eight layers for the HDMI TX Subsystem. These layers get alpha-blended inside the display subsystem, which sends a single video stream to the DisplayPort controller or HDMI Transmitter Subsystem. The bottom layer is used for video frames and the top layer is used for graphics. The graphics layer consists of the GUI and is rendered by the GPU. It overlays certain areas of the video

frame with GUI control elements while other parts of the frame are transparent. A mechanism called *pixel alpha* is used to control the opacity of each pixel in the graphics plane.

The pixel format used for the graphics plane is called ARGB8888 or AR24. It is a packed format that uses 32 bits to store the data value of one pixel (32 bits per pixel or BPP), 8 bits per component (BPC) —also called color depth or bit depth. The individual components are: alpha value (A), red color (R), green color (G), blue color (B). The alpha component describes the opacity of the pixel: An alpha value of 0% means the pixel is fully transparent (invisible); an alpha value of 100% means the pixel is fully opaque.

The pixel formats used for the video plane are NV12, NV16, XV15 and XV20. These are two-plane versions of the YUV 4:2:0 and YUV 4:2:2 format, respectively. The three components are separated into two sub-images or planes.

In NV12 and XV15 formats, chroma planes are sub-sampled in both the horizontal and vertical dimensions by a factor of 2. That is to say, for a 2x2 square of pixels, there are 4 Y samples but only 1 U sample and 1 V sample. Bit-depth for each sample is 8-bit for NV12 and 10-bit for XV15. The Y plane is first in memory. A combined CbCr plane immediately follows the Y plane in memory.

In NV16 and XV20 formats, chroma planes are sub-sampled only in the horizontal dimension by a factor of 2. Thus, there is the same amount of lines in chroma planes as in the luma plane. For a 2x2 group of pixels, there are 4 Y samples and 2 U and 2 V samples each. Bit-depth for each sample is 8-bit for NV16 and 10-bit for XV20. The Y plane is first in memory. A combined CbCr plane immediately follows the Y plane in memory. The CbCr plane is the same width and height, in bytes, as the Y plane.

Aside from the pixel format, a video buffer is further described by a number of other parameters (see Figure 4-2). For this design, the relevant parameters are width, height, and stride as the PS display pipeline does not allow for setting an x or y offset.

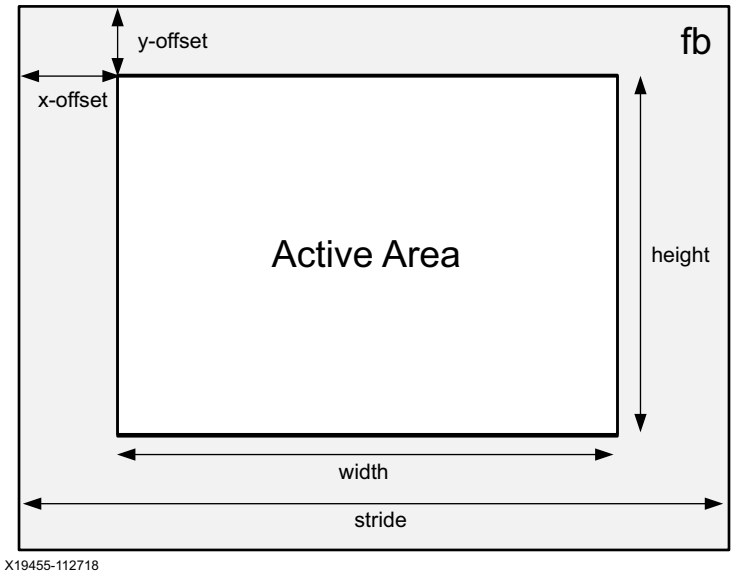

Figure 4-2: **Video Buffer Area**

The active area is the part of the video buffer that is visible on the screen. The active area is defined by the height and width parameters, also called the video dimensions. Those are typically expressed in number of pixels because the bits per pixel depend on the pixel format as explained above.

The stride or pitch is the number of bytes from the first pixel of a line to the first pixel of the next line of video. In the simplest case, the stride equals the width multiplied by the bits per pixel, converted to bytes. For example, AR24 requires 32 BPP which is four bytes per pixel. A video buffer with an active area of 1920 x 1080 pixels therefore has a stride of 4 x 1920 = 7,680 bytes. Some DMA engines require the stride to be a power of two to optimize memory accesses. In this design, the stride always equals the width in bytes.

Hardware Platform

Introduction

This chapter describes the targeted reference design (TRD) hardware architecture. Figure 5-1 shows a block diagram of the design components inside the PS and PL on the ZCU106 base board and the LI-IMX274MIPI-FMC image sensor daughter card.

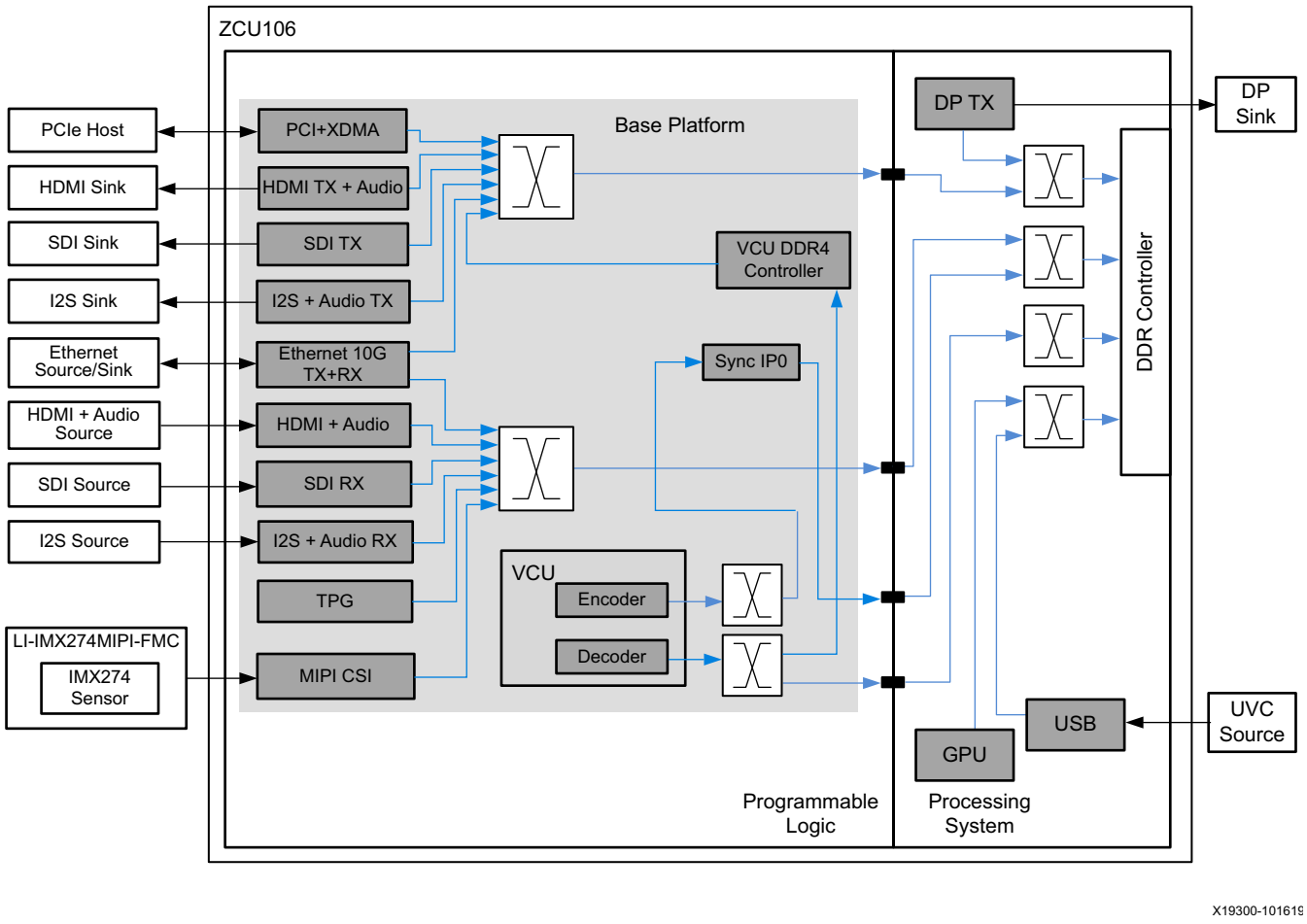

Figure 5-1: **Hardware Block Diagram**

At a high level, the design consists of these three types of video pipelines:

- Capture/Input Pipelines
- Processing Pipelines
- Display/Output Pipelines

Capture/Input Pipelines

- The HDMI RX capture pipeline (in the PL) consists of the HDMI RX Subsystem IP, Video Processing Subsystem IP enabled for VPSS and color space conversion functionality, and the Frame Buffer Write IP that converts the packed video data to a semi-planar format and writes the data into memory.
- The Test Pattern Generator (TPG) capture pipeline (in the PL) consists of the TPG sourcing the live video input that goes to a Frame Buffer Write IP.
- The MIPI CSI-2 RX capture pipeline (FMC + PL) consists of an IMX274 sensor, MIPI CSI-2 Receiver Subsystem (CSI RX), the AXI4-Stream subset converter, Demosaic IP, Gamma LUT IP, Video Processing Subsystem IP enabled for VPSS and color space conversion functionality, and the Frame Buffer Write IP.
- The Ethernet 10G input pipeline (in the PL) consists of 10G/25G Ethernet Subsystem IP that receives video data over Ethernet and AXI DMA IP that writes it to memory.
- The SDI RX capture pipeline (in the PL) consists of the SDI RX Subsystem and Video Processing Subsystem IP enabled for VPSS and color space conversion functionality and the Frame Buffer Write IP.
- The audio input/capture pipeline (in the PL) consists of Audio Formatter IP that receives audio input from the HDMI RX Subsystem IP and writes the data to memory.

Processing Pipelines

- The Video Codec Unit (VCU) processing pipeline (in the PL) consists of the VCU IP which contains the VCU primitive with four 128-bit memory-mapped AXI4 interfaces. The outputs of these interfaces are multiplexed for each of the encoder and decoder ports.
- Sync IP is responsible for synchronizing buffers between Capture DMA and the VCU encoder and is used in LLP2 designs which demand ultra low latency.
- The accelerator processing pipeline (in the PL) consists of a dummy accelerator that has one 128-bit memory-mapped AXI4 interface coming out, which is multiplexed with encoder/decoder ports of the VCU.

  Note: The accelerator processing pipeline is only supported for versions up to, and including 2019.1.

Display/Output Pipelines

- The HDMI TX display pipeline (in the PL) is controlled by the Video Mixer, which fetches both graphics (rendered by GPU in the graphics layer) and the video layer from memory and sends the data to the HDMI TX Subsystem. The HDMI TX Subsystem processes data and sends it out to an external display device.
- The DP TX display pipeline (in the PS) consists of the PS DisplayPort controller. DisplayPort direct memory access (DPDMA) fetches both graphics and the video layer from memory. The DisplayPort controller processes data and sends it out to external display devices using the DisplayPort Standard.
- The SDI TX display pipeline (in the PL) is controlled by the Video Mixer, which fetches the video layer from memory and sends to the SDI TX Subsystem. The SDI TX Subsystem processes data and sends it out to an external display device.
- The USB universal video class (UVC) capture pipeline (in the PS) consists of the USB Controller. It takes recorded video files and writes the data into DDR memory.
- The Ethernet 10G output pipeline (in the PL) consists of AXI DMA IP that reads data from memory followed by the 10G/25G Ethernet Subsystem IP that transmits data to Ethernet.
- The audio output pipeline (in the PL) consists of Audio Formatter IP that reads audio data from memory and sends it out to the HDMI TX Subsystem IP, which sends it to the output device.
- The design uses the PCIe® Endpoint block with high-performance XDMA for data transfers between the host system memory and the Endpoint. In the card-to-host direction, the XDMA block moves data from the Endpoint PS DDR to the host memory through PCIe.

The block diagram highlights these two partitions of the design:

- The hardware Base Platform consists of all the capture and display pipelines and VCU processing pipelines.

Clocking

This section describes the clocking mechanism used in the TRD. The primary clock is sourced from si570_user sources that provide a 300 MHz reference clock to the PL. A mixed-mode clock manager (MMCM) block in the PL uses the si570 clock as a primary input clock and generates the reference clock for the VCU PLL and video pixel clock. The VCU PLL generates the core clock and MCU clock based on the input reference. PL_CLK0 from the processing system is also used as the AXI4-Lite clock.

The USER_MGT_SI570_CLOCK is used as source for the DRU_CLK/SDI GT reference clock. Figure 5-2 shows the clocking mechanism used for the TRD. The 125 MHz mig clock is used

as the PL DDR ref clock. The VCU_DDR4 soft IP generates the 250 MHz phy_clk required for processing the data.

Note: The audio design uses pl_clk2 as the Video Pixel clock (instead of MMCM output) for both TX and RX pipelines. The Ethernet 10G design uses the SPF_SI5328_OUT clock from the board as the DRU clock, because USER_MGT_SI570_CLOCK is used by the Ethernet Subsystem as the GT reference clock.

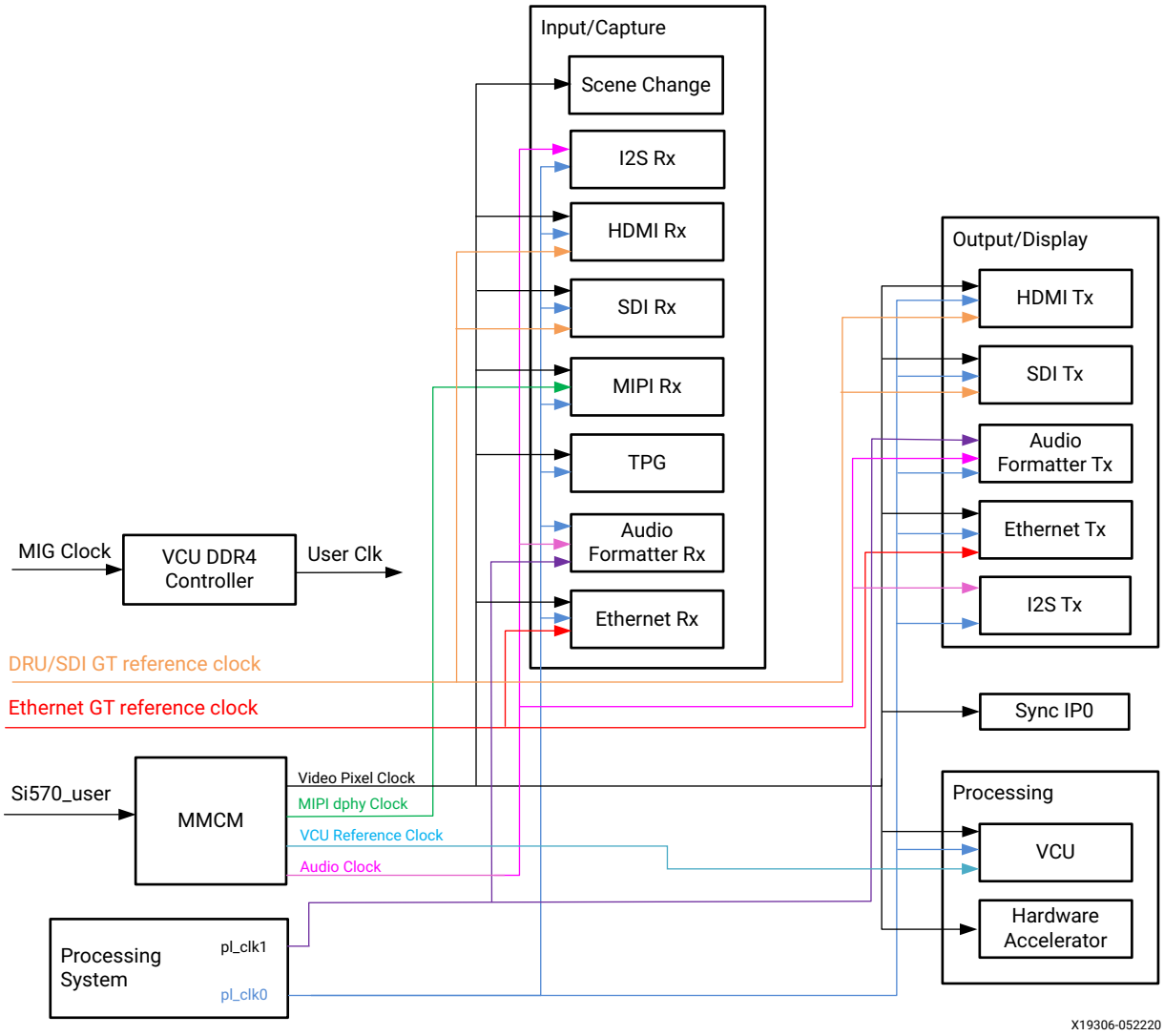

Figure 5-2: **Clocking Mechanism for the TRD**

Reset

A synchronous reset mechanism is used in the TRD. PL_RESET0 is used as a master reset signal. Interconnect and peripheral reset signals are generated using proc_sys_rst IP in the PL.

Video Pipelines

A *live-capture/file-src* element receives frames from an external source or produces video frames internally. The captured video frames are written into memory.

A *processing* element reads video frames from memory, does certain processing, and then writes the processed frames back into memory.

A *display* element reads video frames from memory and sends the frames to a sink. In cases where sink is displayed, this pipeline is also referred to as display pipeline.

TPG Capture Pipeline

The TPG capture pipeline is shown in Figure 5-3.

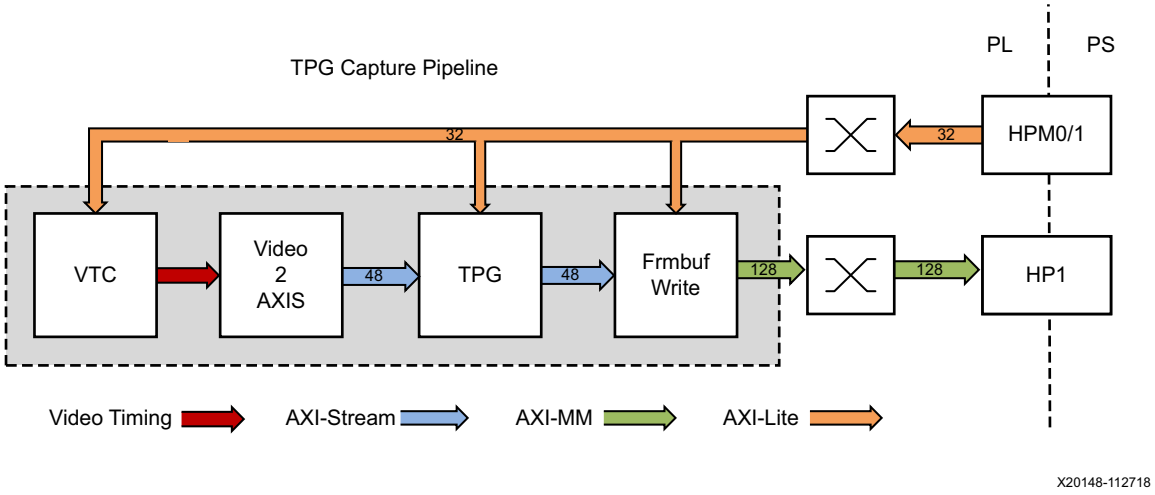

Figure 5-3: **TPG Video Capture Pipeline**

This pipeline consists of three main components, each of them controlled by the APU via an AXI4-Lite based register interface:

- The Video Timing Controller (VTC) generates video timing signals including horizontal and vertical sync and blanking signals. The timing signals are converted to AXI4-Stream using the video-to-AXI4-Stream bridge with the data bus tied off. The video timing over AXI4-Stream bus is connected to the input interface of the TPG, thus making the TPG behave like a timing-accurate video source with a set frame rate as opposed to using the free-running mode.
- The Video Test Pattern Generator (TPG) can be configured to generate various test patterns including color bars, zone plates, moving ramps, moving boxes and more. The color space format is configurable and set to YUV 4:2:0 in this design. For more

information, see the *Video Test Pattern Generator LogiCORE IP Product Guide* (PG103) [Ref 9].

- The Video Frame Buffer Write IP provides high-bandwidth direct memory access between memory and AXI4-Stream video type target peripherals, which support the AXI4-Stream Video protocol. In this pipeline, the IP takes AXI4-Stream input data from the TPG and converts it to memory-mapped AXI4 format. The output is connected to the HP1 high performance PS/PL interface via an AXI interconnect. For each video frame transfer, an interrupt is generated. A GPIO is used to reset the core between resolution changes. For more information refer to the *Video Frame Buffer Read and Video Frame Buffer Write LogiCORE IP Product Guide* (PG278) [Ref 10].

HDMI RX Capture Pipeline

The HDMI receiver capture pipeline is shown in Figure 5-4.

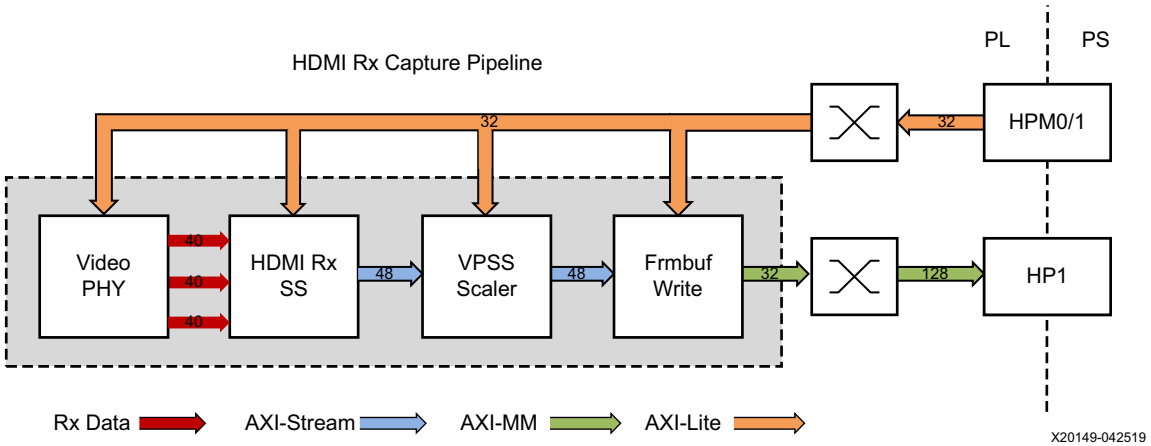

Figure 5-4: **HDMI Video Capture Pipeline**

This pipeline consists of four main components, each of them controlled by the APU via an AXI4-Lite base register interface:

- The Video PHY Controller (VPHY) enables plug-and-play connectivity with Video Transmit or Receive Subsystems. The interface between the media access control (MAC) and physical (PHY) layers are standardized to enable ease of use in accessing shared gigabit-transceiver (GT) resources. The data recovery unit (DRU) is used to support lower line rates for the HDMI protocol. An AXI4-Lite register interface is provided to enable dynamic accesses of transceiver controls/status. For more information refer to the *Video PHY Controller LogiCORE IP Product Guide* (PG230) [Ref 11].
- The HDMI Receiver Subsystem (HDMI RX) interfaces with PHY layers and provides HDMI decoding functionality. The subsystem is a hierarchical IP that bundles a collection of HDMI RX-related IP subcores and outputs them as a single IP. The subsystem receives the captured TMDS data from the video PHY layer. It then extracts the video stream from the HDMI stream and in this design converts it to an

AXI4-Stream output interface. For more information, see the *HDMI 1.4/2.0 Receiver Subsystem Product Guide* (PG236) [Ref 12].

- The Video Processing Subsystem (VPSS) is a collection of video processing IP subcores. In this design, the VPSS uses the Video Scaler only configuration which provides scaling, color space conversion, and chroma resampling functionality. The VPSS takes AXI4-Stream input data from the HDMI RX Subsystem and depending on the input format and resolution, converts and scales it to the desired output format and resolution again using AXI4-Stream. A GPIO is used to reset the subsystem between resolution changes. For more information, see the *Video Processing Subsystem Product Guide* (PG231) [Ref 13].
- The Video Frame Buffer Write IP uses the same configuration as the one in the TPG capture pipeline. It takes AXI4-Stream input data from the VPSS and converts it to memory-mapped AXI4 format. The output is connected to the HP1 high performance PS/PL interface via an AXI interconnect. For each video frame transfer, an interrupt is generated. A GPIO is used to reset the IP between resolution changes.

Similar to the TPG pipeline, the HDMI RX, VPSS Video Scaler, and Frame Buffer Write IPs are configured to transport two pixels per clock (ppc), enabling up to 2160p60 performance. In the DCI4K pipeline, the resolution is 4096 x 2160.

MIPI CSI-2 RX Capture Pipeline

The MIPI CSI-2 receiver capture pipeline is shown in Figure 5-5.

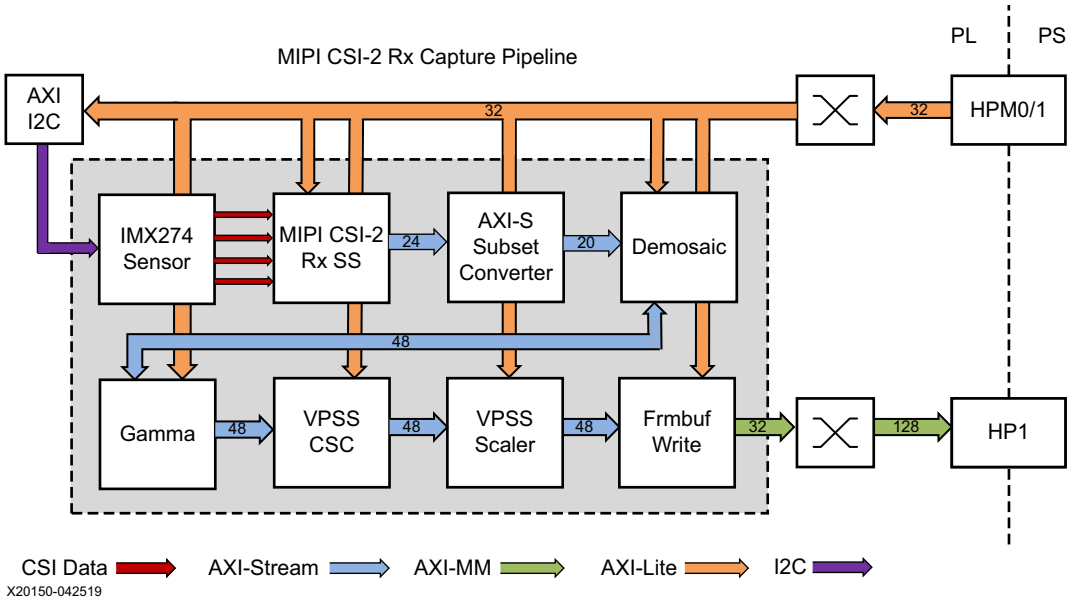

Figure 5-5: **CSI Video Capture Pipeline**

This pipeline consists of eight components, six of which are controlled by the APU via an AXI4-Lite based register interface, one is controlled by the APU via an I2C register interface, and one is configured statically:

- The Sony IMX274 is a 1/2.5 inch CMOS digital image sensor with an active imaging pixel array of 3864H x2196V. The image sensor is controlled via an I2C interface using an AXI I2C Controller in the PL. It is mounted on a FMC daughter card and has a MIPI output interface that is connected to the MIPI CSI-2 RX Subsystem inside the PL. For more information, see the LI-IMX274MIPI-FMC data sheet [Ref 3].
- The MIPI CSI-2 Receiver Subsystem (CSI RX) includes a MIPI D-PHY core that connects four data lanes and one clock lane to the sensor on the FMC card. It implements a CSI-2 receive interface according to the MIPI CSI-2 standard v1.1. The subsystem captures images from the IMX274 sensor in RAW10 format and outputs AXI4-Stream video data. For more information, see the *MIPI CSI-2 Receiver Subsystem Product Guide* (PG232) [Ref 14].
- The AXI subset converter is a statically configured IP core that converts the raw 10-bit (RAW10) AXI4-Stream input data to raw 8-bit (RAW8) AXI4-Stream output data by truncating the two least significant bits (LSB) of each data word.
- The Demosaic IP core reconstructs sub-sampled color data for images captured by a Bayer color filter array image sensor. The color filter array overlaid on the silicon substrate enables CMOS image sensors to measure local light intensities that correspond to different wavelengths. However, the sensor measures the intensity of only one principal color at any location (pixel). The Demosaic IP receives the RAW8 AXI4-Stream input data and interpolates the missing color components for every pixel to generate a 24-bit, 8bpc RGB output image transported via AXI4-Stream. A GPIO is used to reset the IP between resolution changes.
- The Gamma LUT IP core is implemented using a look-up table (LUT) structure that is programmed to implement a gamma correction curve transform on the input image data. A programmable number of gamma tables enable having separate gamma tables for all color channels, in this case red, green, and blue. The Gamma IP takes AXI4-Stream input data and produces AXI4-Stream output data, both in 24-bit RGB format. A GPIO is used to reset the IP between resolution changes.
- The Video Processing Subsystem (VPSS) is a collection of video processing IP subcores. This instance is uses the Color Space Converter (CSC) configuration to perform color correction tasks including contrast, brightness, and red/green/blue gain control. The CSC takes AXI4-Stream input data and produces AXI4-Stream output data, both in 24-bit RGB format. A GPIO is used to reset the subsystem between resolution changes. For more information, see the *Video Processing Subsystem Product Guide* (PG231) [Ref 13].
- The Video Processing Subsystem (VPSS) is a collection of video processing IP subcores. This instance uses the VPSS only configuration, which provides scaling, color space conversion, and chroma resampling functionality. The VPSS takes AXI4-Stream input

data in 24-bit RGB format and converts it to a 16-bit, 8bpc YUV 4:2:0 output format using AXI4-Stream. A GPIO is used to reset the subsystem between resolution changes.

- The Video Frame Buffer Write IP uses the same configuration as the one in the TPG and HDMI RX capture pipelines. It takes YUV 4:2:0 sub-sampled AXI4-Stream input data and converts it to memory-mapped AXI4 format which is written to memory as 16-bit packed YUYV. The memory-mapped AXI interface is connected to the HP1 high performance PS/PL port via an AXI interconnect. For each video frame transfer, an interrupt is generated. A GPIO is used to reset the IP between resolution changes.

Similar to the TPG and HDMI RX capture pipelines, all the IPs in this pipeline are configured to transport 2ppc, enabling up to 2160p60 performance.

SDI RX Capture Pipeline

The SDI RX capture pipeline is shown in Figure 5-6.

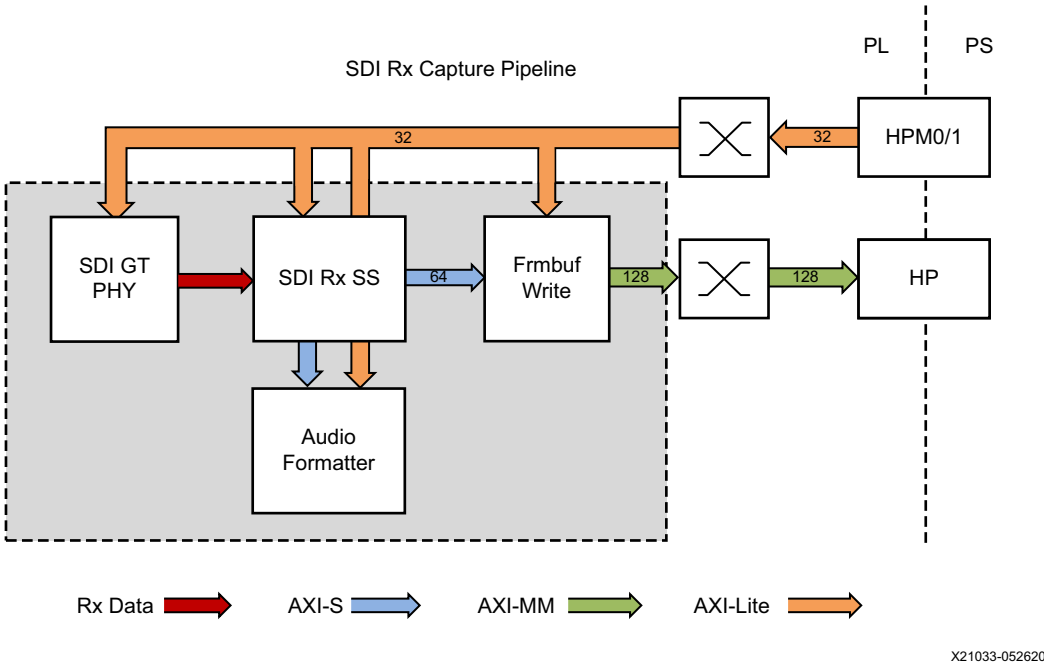

Figure 5-6: **SDI RX Capture Pipeline**

The serial digital interface (SDI) Receiver Subsystem implements an SDI receive interface in accordance with the SDI family of standards. The subsystem receives video from a native SDI interface and generates AXI4-Stream video. The SMPTE UHD-SDI receiver core receives multiplexed native SDI data streams and generates non-multiplexed 10-bit SDI data streams in YUV422 format.

The Video Frame Buffer Write IP is used as the Frame Grabber logic, which is designed to allow efficient and high bandwidth access between AXI4-Streaming Video In interfaces to

the Zynq® UltraScale+ MPSoC PS DDR memory. The Video Frame Buffer IP can write a variety of video formats to the Zynq UltraScale+ MPSoC PS DDR memory.

DP TX Display Pipeline

The DP TX display pipeline (see Figure 5-7) is configured to read video frames from memory via two separate channels: one for video, the other for graphics. The video and graphics layers are alpha-blended to create a single output video stream that is sent to the monitor via the DisplayPort controller. This design does not use the audio feature of the DisplayPort controller, therefore it is not discussed in this user guide. The major components used in this design, as shown in the figure, are:

- DisplayPort DMA (DPDMA)
- Audio/Video (A/V) buffer manager
- Video blender
- DisplayPort controller (DP TX)
- PS-GTR gigabit transceivers

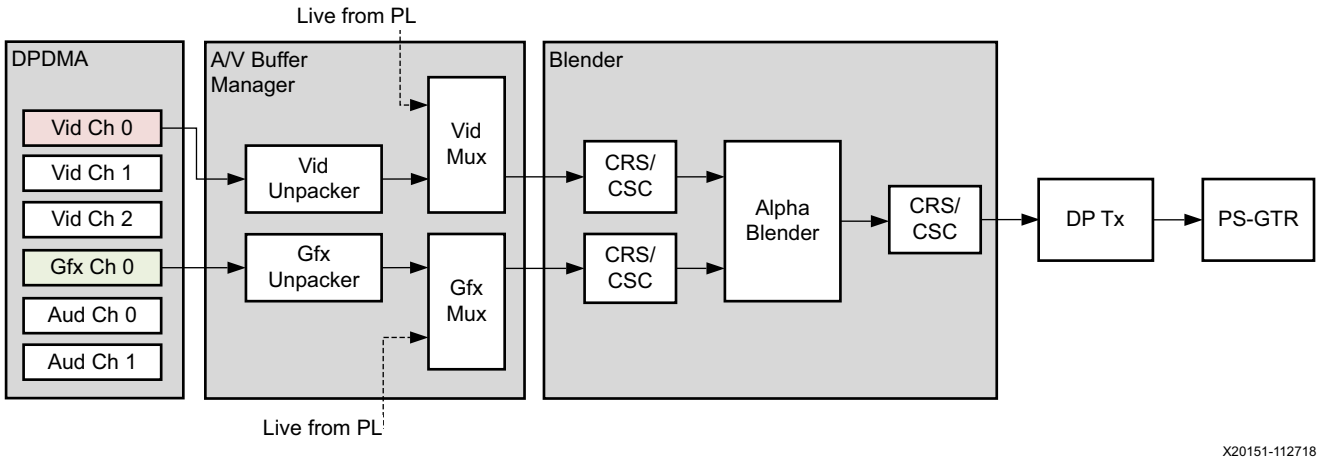

Figure 5-7: **Display Pipeline Showing DPDMA, A/V Buffer Manager, Video Blender, and DP Transmitter**

The DPDMA is a 6-channel DMA engine that fetches data from memory and forwards it to the A/V buffer manager. The video layer can consist of up to three channels, depending on the chosen pixel format, whereas the graphics layer is always a single channel. The used pixel formats are described in Video Buffer Format. The remaining two channels are used for audio.

The A/V buffer manager can receive data either from the DPDMA (non-live mode) or from the PL (live mode) or a combination of the two. In this design, only non-live mode is used for both video and graphics. The three video channels feed into a video pixel unpacker and the graphics channel into a graphics pixel unpacker. Because the data is not timed in non-live mode, video timing is locally generated using the internal Video Timing Controller.

A stream selector forwards the selected video and graphics streams to the dual-stream video blender.

The video blender unit consists of input color space converters (CSC) and chroma re-samplers (CRS), one pair per stream, a dual-stream alpha blender, and one output color space converter and chroma re-sampler. The two streams must have the same dimensions and color format before entering the blender. The alpha blender can be configured for global alpha (single alpha value for the entire stream) or per pixel alpha. A single output stream is sent to the DisplayPort controller.

The DisplayPort controller supports the DisplayPort v1.2a protocol. It does not support multi-stream transport or other optional features. The DisplayPort controller is responsible for managing the link and physical layer functionality. The controller packs video data into transfer units and sends them over the main link. In addition to the main link, the controller has an auxiliary channel, which is used for source/sink communication.

Four high-speed gigabit transceivers (PS-GTRs) are implemented in the serial input output unit (SIOU) and shared between the following controllers: PCIe, USB 3.0, DP, SATA, and SGMII Ethernet. The DP controller supports up to two lanes at a maximum line rate of 5.4 Gb/s. The link rate and lane count are configurable based on bandwidth requirements.

For more information on the DisplayPort controller and the PS-GTR interface, see Chapter 29 *PS-GTR Transceivers* and Chapter 33 *DisplayPort Controller* in *Zynq UltraScale+ Device Technical Reference Manual* (UG1085) [Ref 8].

HDMI TX Display Pipeline

The HDMI TX display pipeline is shown in Figure 5-8.

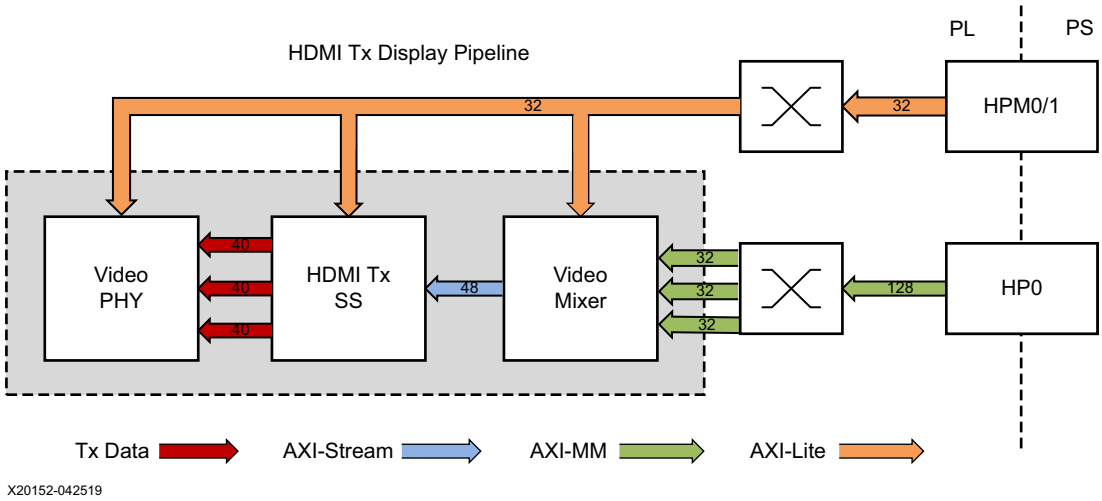

Figure 5-8: **HDMI TX Display Pipeline**

This pipeline consists of three main components, each of them controlled by the APU via an AXI4-Lite base register interface:

- The Video Mixer IP core is configured to support blending of up to two video layers and one graphics layer into one single output video stream. The three layers are configured to be memory-mapped AXI4 interfaces connected to the HP0 high performance PS/PL interface via an AXI interconnect; the main AXI4-Stream layer is unused. The two video layers are configured for 16-bit YUYV, the graphics layer is configured for 32-bit ARGB, (see Video Buffer Format for details). A built-in color space converter and chroma resampler convert the input formats to a 24-bit RGB output format. Pixel-alpha blending is used to blend the graphics layer with the underlying video layers. The AXI4-Stream output interface is a 48-bit bus that transports 2 ppc for up to 2160p60 performance. It is connected to the HDMI TX Subsystem input interface. A GPIO is used to reset the subsystem between resolution changes. For more information refer to the *Video Mixer LogiCORE IP Product Guide* (PG243) [Ref 15].
- The HDMI Transmitter Subsystem (HDMI TX) interfaces with PHY layers and provides HDMI encoding functionality. The subsystem is a hierarchical IP that bundles a collection of HDMI TX-related IP sub-cores and outputs them as a single IP. The subsystem generates an HDMI stream from the incoming AXI4-Stream video data and sends the generated TMDS data to the video PHY layer. For more information refer to the *HDMI 1.4/2.0 Transmitter Subsystem Product Guide* (PG235) [Ref 14].
- The Video PHY Controller is shared between the HDMI RX and HDMI TX pipelines. Refer to HDMI RX Capture Pipeline for more information on the VPHY and its configuration.

SDI TX Display Pipeline

The SDI TX display pipeline is shown in Figure 5-9.

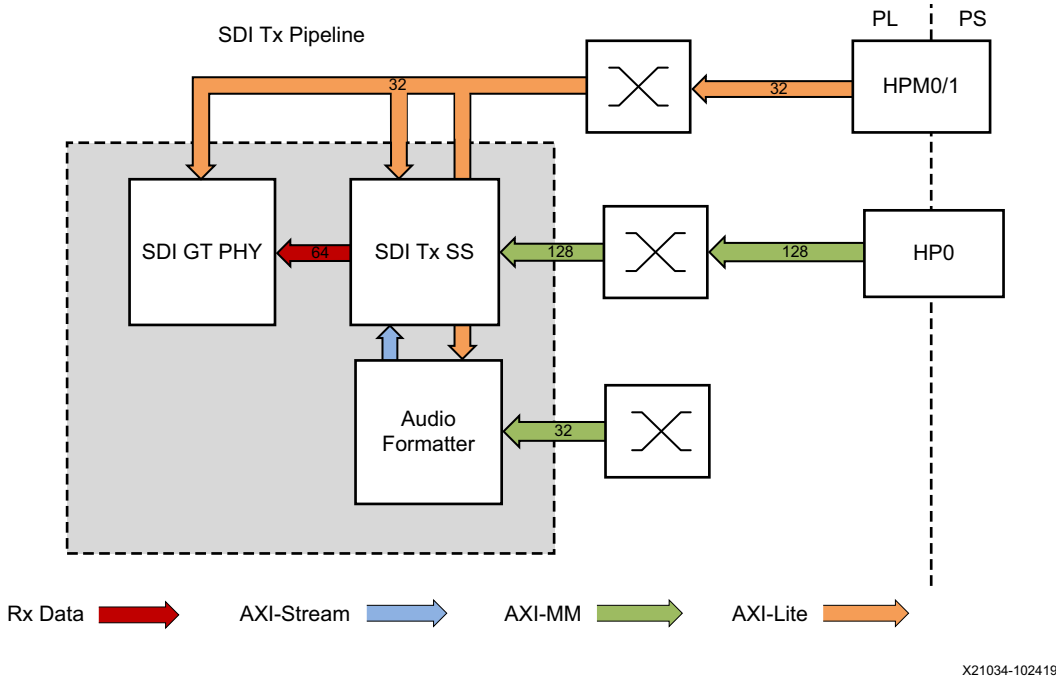

Figure 5-9: **SDI TX Display Pipeline**

The SMPTE UHD-SDI Transmitter Subsystem accepts AXI4 Video streams and outputs native SDI streams by using Xilinx transceivers as the physical layer.

The Video Mixer enables you to mix video layers and allows mixing up to four streaming or memory layers. Each layer can be up to 4K resolution and can perform color space conversion. The TRD design uses memory layer 1 to fetch video data.

Ethernet 10G Input/Capture Pipeline

The Ethernet 10G input/capture pipeline is shown in Figure 5-10.

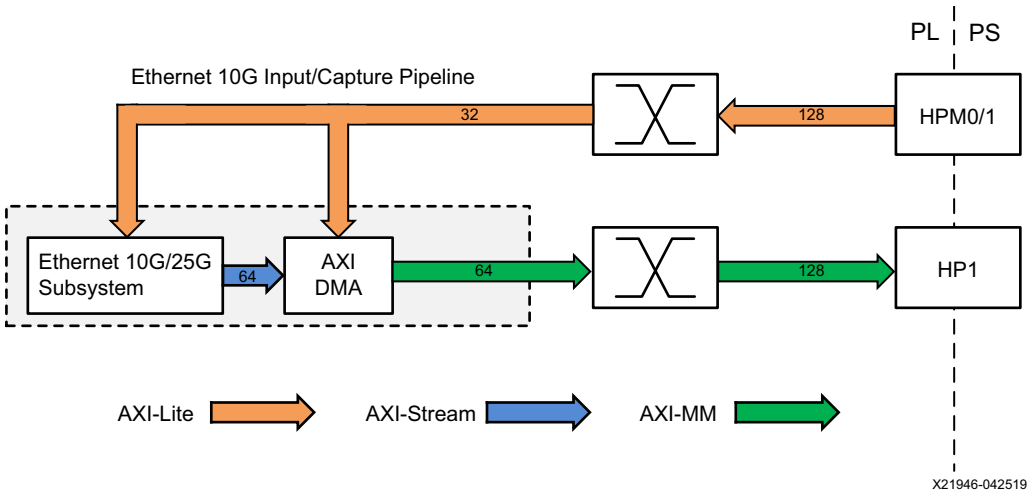

Figure 5-10: **Ethernet 10G Input/Capture Pipeline**

This pipeline consists of two components, each of them controlled by the APU through an AXI4-Lite base register interface:

- The 10G/25G high speed Ethernet Subsystem implements the 25G Ethernet MAC with a physical coding sublayer (PCS) as specified by the 25G Ethernet Consortium. The 156.25 MHz reference clock to the transceiver is provided by the Si570 programmable oscillator available on the ZCU106 board. For more information, see *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210) [Ref 16].
- The AXI DMA with enabled scatter gather (SG) mode provides high-bandwidth direct memory access between memory and the Ethernet 10G Subsystem via AXI interconnect. For more information, see *AXI DMA LogiCORE IP Product Guide* (PG021) [Ref 17].

Ethernet 10G Output Pipeline

The Ethernet 10G output pipeline is shown in Figure 5-11.

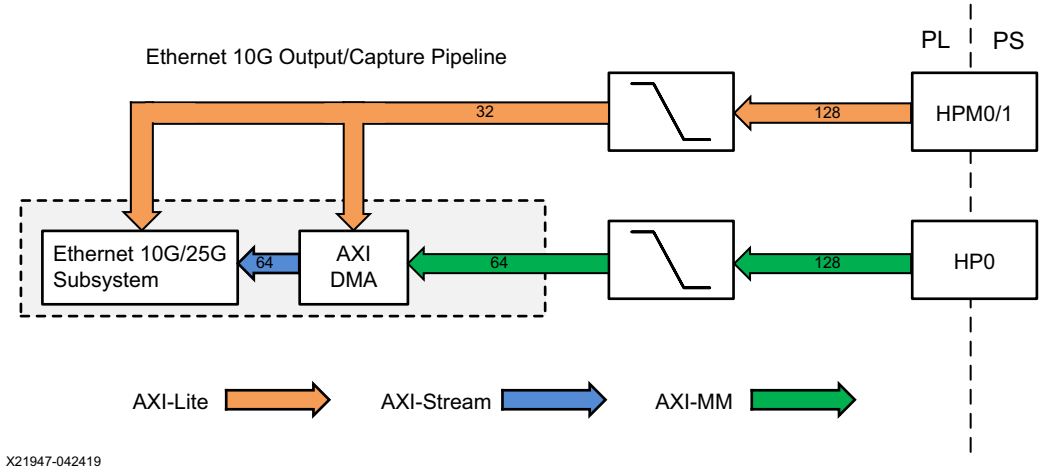

Figure 5-11: **Ethernet 10G Output Pipeline**

This pipeline consists of two main components—the 10G/25G high speed Ethernet Subsystem and AXI DMA, each shared with the Ethernet 10G input/capture pipeline. Refer to Ethernet 10G Input/Capture Pipeline for more information and for the configuration of each component.

HDMI Audio RX Pipeline

The HDMI audio RX pipeline is shown in Figure 5-12.

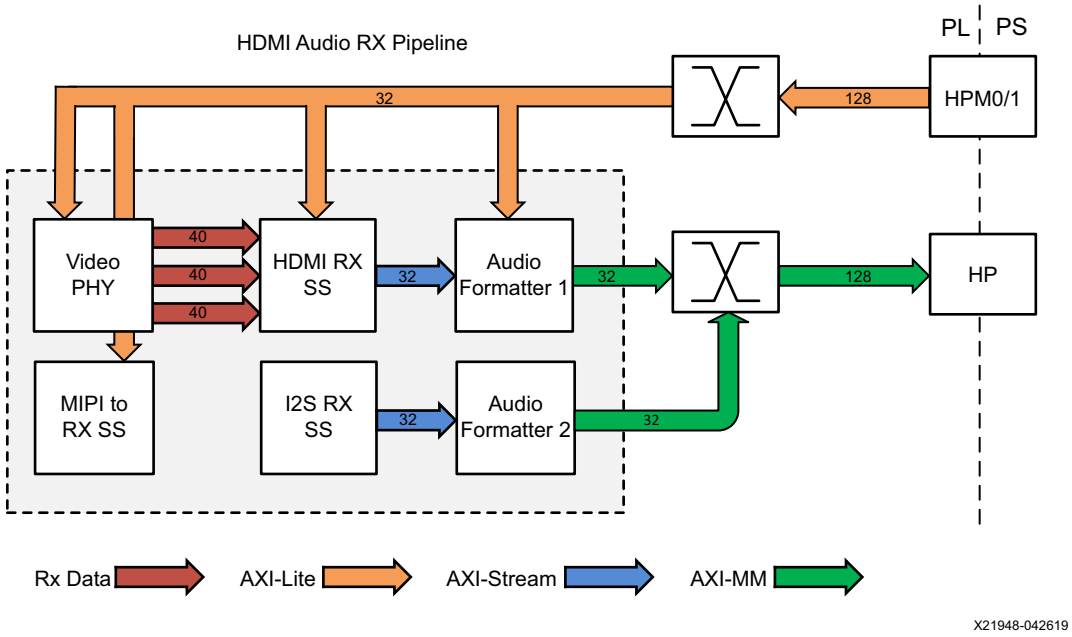

Figure 5-12: **HDMI Audio RX Pipeline**

This pipeline consists of two components, each of them controlled by the APU through an AXI4-Lite base register interface:

- The Video PHY Controller is shared with the HDMI RX and HDMI TX pipelines. Refer to HDMI RX Capture Pipeline for more information on the VPHY and its configuration.
- The HDMI RX Subsystem is shared with the HDMI RX pipeline. Refer to HDMI RX Capture Pipeline for more information on the VPHY and its configuration.
- The Audio Formatter provides high-bandwidth direct memory access between memory and AXI4-Stream target peripherals. Initialization, status, and management registers are accessed through an AXI4-Lite slave interface. It is configured with both read and write interface enabled for a maximum of two audio channels and interleaved memory packing mode with memory data format configured as AES to PCM.

Note: The Audio Engineering Society (AES) standard was developed for the exchange of digital audio signals between professional audio devices.

HDMI Audio TX Pipeline

This pipeline consists of three main components—Video PHY Controller, HDMI RX Subsystem, and Audio Formatter, each shared with the audio input/capture pipeline. Refer to the following sections for more information and for the configuration of each component:

- Video PHY Controller (see HDMI RX Capture Pipeline)
- HDMI RX Subsystem (see HDMI RX Capture Pipeline)
- Audio Formatter (see HDMI Audio RX Pipeline)

Note: HDMI RX Subsystem IP is available from Xilinx. HDMI 1.4/2.0 Receiver Subsystem v3.1 is the current version as of this printing.

PCIe Capture Pipeline

The PCIe capture pipeline is shown in Figure 5-13.

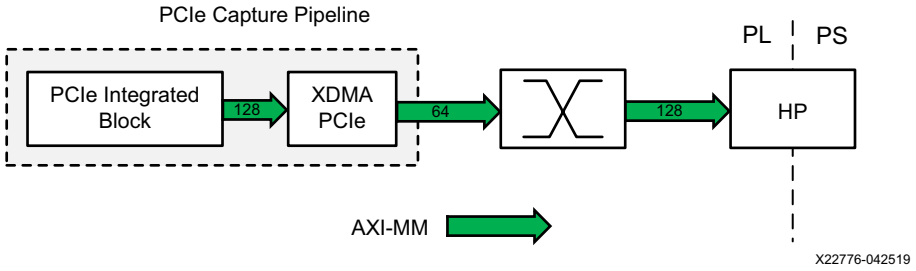

Figure 5-13: **PCIe Capture Pipeline**

The design uses the PCIe Endpoint block with high-performance XDMA for data transfers between the host system memory and the Endpoint. In the host-to-card direction, the XDMA block moves data from the host memory to the End point Memory through PCIe.

SCD Design Pipeline

The SCD design pipeline is shown in Figure 5-14.

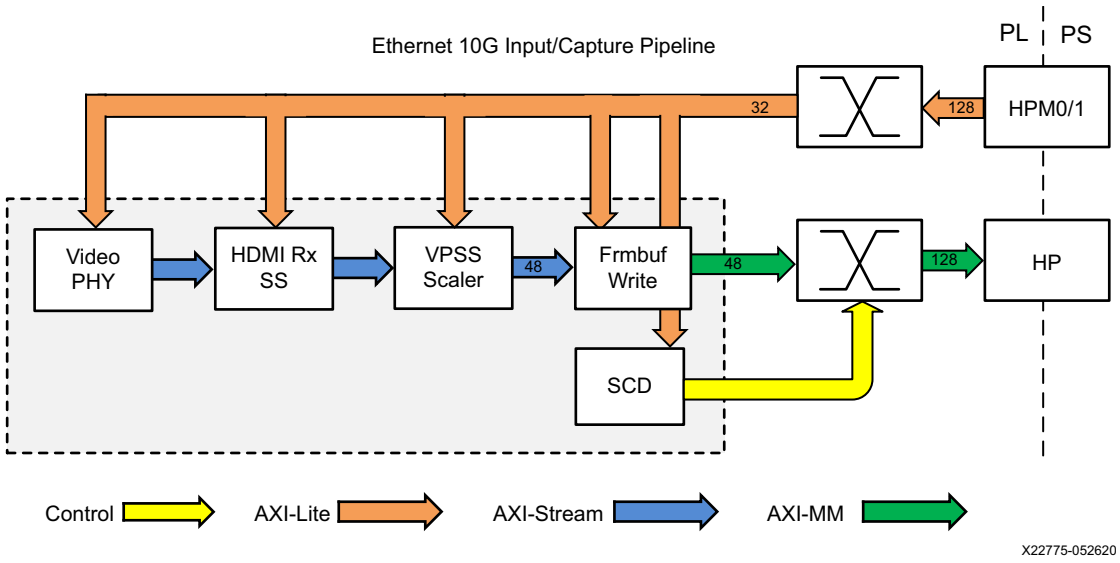

Figure 5-14: **SCD Pipeline**

Video Scene Change is used with the Zynq UltraScale+ VCU subsystem to identify when to update the reference frame for better performance while encoding streams. This is done using the Video Scene Change detection IP interrupt flag. It sends fewer frames that help in reducing the compressed stream size thereby saves bandwidth.

The Video Scene Change detection on the IP core can read up to eight video streams in memory mode and one video stream in stream mode. In memory mode, input is read from the memory mapped AXI4 interface. In stream mode, input is read from the AXI4-Stream interface and output stream is same as received input stream. For more information refer to the *Video Scene Change Detection LogiCORE IP Product Guide* (PG322) [Ref 23].

I2S Audio Pipeline

The I2S Transmitter and Receiver cores are soft Xilinx IP cores, which make easy to implement inter-IC-sound (I2S) interfaces used to connect audio devices for transmitting and receiving PCM audio. The I2S Transmitter and I2S Receiver cores provide an easy way to interface the I2S based audio DAC/ADC. These IPs require minimal register programming and support any audio sampling rates. For more information refer to the *I2S Transmitter and I2S Receiver LogiCORE IP Product Guide* (PG308) [Ref 24].

VCU DDR4 Controller Pipeline

The VCU DDR4 controller pipeline is shown in Figure 5-15.

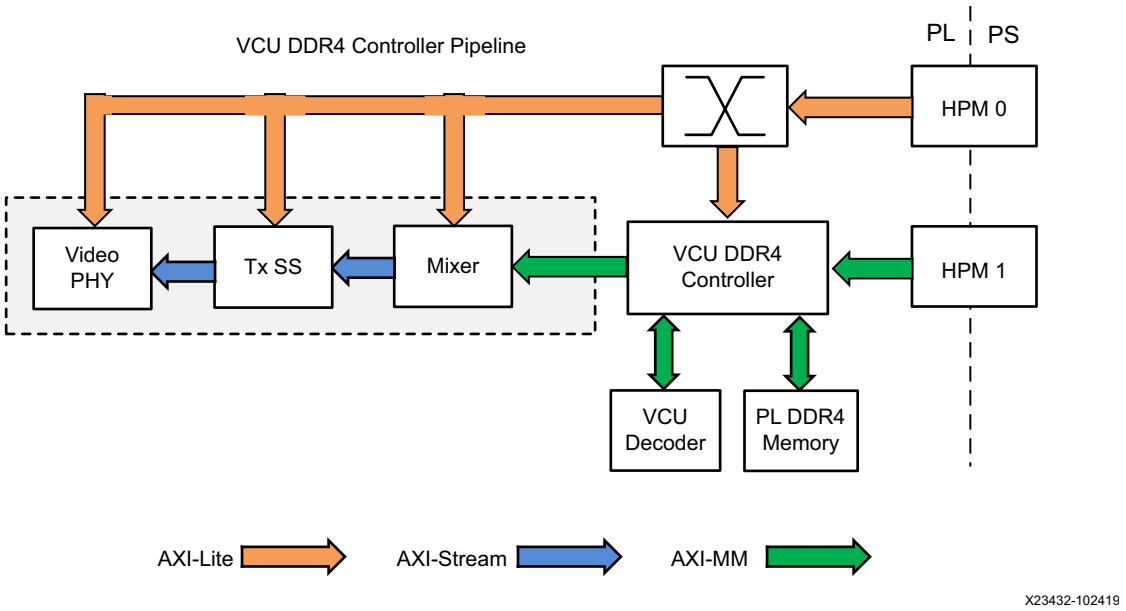

Figure 5-15: **PL_DDR Pipeline**

The Zynq UltraScale+ MPSoC VCU DDR4 Controller is an application-specific DDR controller that is only supported for use with the Zynq UltraScale+ MPSoC VCU (H.264/H.265 Video Codec unit). It is required to support the YUV 10-bit formats and the DCI4K resolution 10-bit formats.

Low Latency Pipeline

The low latency pipeline is shown in Figure 5-16.

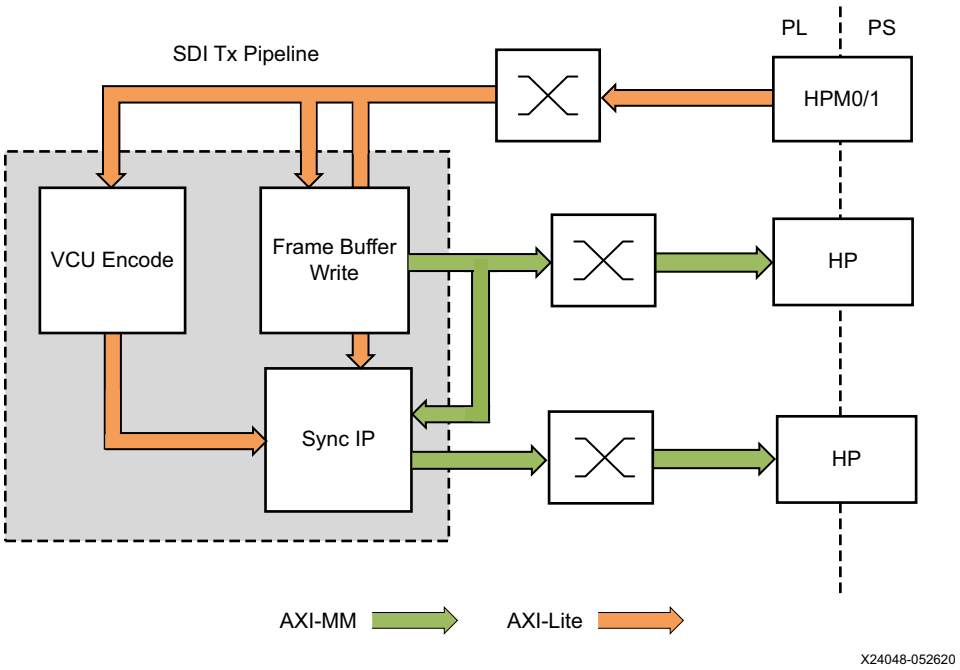

Figure 5-16: **Low Latency Pipeline**

It is possible to reduce VCU processing latency from one frame to one-frame/num-slices. The Sync IP element is responsible for synchronizing buffers between the Capture DMA and the VCU encoder, and the VCU Decoder and Display element. The Sync IP does AXI transaction-level tracking so that the producer and consumer can be synchronized at the granularity of AXI transactions instead of granularity at the Video Buffer level. The Low Latency Pipeline has these features:

- Sync IP can track up to four producer transactions simultaneously (four channels)
- Each channel can track up to three buffer sets
- Each buffer set has a luma buffer and chroma buffer
- Each consumer port can hold 256 AXI transactions, without back-pressure to the consumer
- Encoder mode Sync IP supports four simultaneous channels for tracking and control
- Decoder mode Sync IP supports two simultaneous channels for tracking and control

Address Map

Table 5-1 shows the address map for various IP blocks used in PL for the VCU TRD full-fledged design.

Table 5-1: **Address Map for IP Blocks of the VCU TRD Full-fledged Design**

IP Core	Base Address	Offset
AXI Interrupt Controller	0x00A0052000	4K
HDMI I2C Controller	0x00A0050000	4K
MIPI CSI-2 Receiver Subsystem	0x00A00F0000	64K
Sensor I2C Controller	0x00A0051000	4K
Sensor Demosaic	0x00A0250000	64K
HDMI Frame Buffer Read	0x00A0040000	64K
HDMI Frame Buffer Write 0	0x00A0010000	64K
TPG Frame Buffer Write	0x00A00C0000	64K
CSI Frame Buffer Write	0x00A0260000	64K
HDMI Frame Buffer Write 1	0x00A02B0000	64K
HDMI Frame Buffer Write 2	0x00A02C0000	64K
HDMI Frame Buffer Write 3	0x00A0280000	64K
HDMI Frame Buffer Write 4	0x00A0290000	64K
HDMI Frame Buffer Write 5	0x00A02A0000	64K
HDMI Frame Buffer Write 6	0x00A02D0000	64K
Gamma LUT	0x00A0270000	64K
HDMI 1.4/2.0 Receiver Subsystem v2.0	0x00A0000000	64K
HDMI 1.4/2.0 Transmitter Subsystem v2.0	0x00A0020000	128K
Video Mixer	0x00A0070000	64K
Video Processing Subsystem (VPSS)	0x00A0080000	256K
Video Processing Subsystem (VPSS-CSC)	0x00A0240000	64K
Video Processing Subsystem (VPSS-Scaler)	0x00A0200000	256K
Video Timing Controller	0x00A00D0000	64K
Video Test Pattern Generator (TPG)	0x00A00E0000	64K
H.264/H.265 Video Codec Unit (VCU)	0x00A0100000	1M
Video PHY Controller	0x00A0060000	64K
Video Scene Change	0x00A02E0000	64K

Table 5-2 shows the address map of various IP blocks used in the PL of an audio design.

Table 5-2: **Address Map for Audio Design IP Blocks**

IP Core	Base Address	Offset
Audio Clock Recovery	0x00_A029_0000	64K
Audio Formatter1	0x00_A005_2000	4K
Audio Formatter2	0x00_A005_1000	4K
AXI GPIO	0x00_A005_3000	4K
AXI Interrupt Controller	0x00_A005_5000	4K
HDMI ACR Control	0x00_A005_6000	4K
S_AXI	0x00_A005_0000	4K
I2S receiver	0x00_A00C_0000	64K
I2S transmitter	0x00_A00D_0000	64K
MIPI CSI-2 Receiver Subsystem	0x00_A00F_0000	64K
Sensor I2C Controller	0x00_A005_4000	4K
Sensor Demosaic	0x00_A025_0000	64K
HDMI Frame Buffer Read	0x00_A004_0000	64K
HDMI Frame Buffer Write	0x00_A001_0000	64K
MIPI Frame Buffer Write	0x00_A026_0000	64K
Gamma LUT	0x00_A027_0000	64K
HDMI Receiver subsystem	0x00_A000_0000	64K
HDMI Transmitter subsystem	0x00_A002_0000	128K
Video Mixer	0x00_A007_0000	64K
Video Processing Subsystem (VPSS)	0x00_A008_0000	256K
Video Processing Subsystem (VPSS-CSC)	0x00_A024_0000	64K
Video Processing Subsystem (VPSS-Scaler)	0x00_A020_0000	256K
Scene Change	0x00_A028_0000	64K
H.264/H.265 Video Codec Unit (VCU)	0x00_A010_0000	1M
Video PHY Controller	0x00_A006_0000	64K

Table 5-3 shows the address map of various IP blocks used in the PL of an Ethernet 10G design.

Table 5-3: **Address Map for Ethernet 10G Design IP Blocks**

IP Core	Base Address	Offset
AXI DMA	0x00_8000_0000	4K
HDMI I2C Controller	0x00_A005_0000	4K
HDMI Frame Buffer Read	0x00_A004_0000	64K
HDMI Frame Buffer Write 0	0x00_A001_0000	64K
HDMI Frame Buffer Write 1	0x00_A02B_0000	64K

Table 5-3: **Address Map for Ethernet 10G Design IP Blocks *(Cont'd)***

IP Core	Base Address	Offset
HDMI Frame Buffer Write 1	0x00_A02C_0000	64K
HDMI 1.4/2.0 Receiver Subsystem v2.0	0x00_A000_0000	64K
HDMI 1.4/2.0 Transmitter Subsystem v2.0	0x00_A002_0000	128K
Video Mixer	0x00_A007_0000	64K
Video Processing Subsystem (VPSS)	0x00_A008_0000	256K
H.264/H.265 Video Codec Unit (VCU)	0x00_A010_0000	1M
Video PHY Controller	0x00_A006_0000	64K
Ethernet 10G/25G Subsystem	0x00_8000_1000	4K

Table 5-4 shows the address map of various IP blocks used in the PL of an SDI design.

Table 5-4: **Address Map for PLDDR SDI Design IP Blocks**

IP Core	Base Address	Offset
Audio Formatter	0x00_A000_0000	4K
AXI GPIO	0x00_A006_0000	4K
GPIO Resets	0x00_A006_3000	4K
GPIO Registers	0x00_A006_1000	4K
GPIO Registers	0x00_A006_2000	4K
SDI TX Frame buffer read	0x00_B001_0000	64K
Video frame buffer read	0x00_A00C_0000	64K
SDI RX Frame buffer write	0x00_B000_0000	64K
Video frame buffer write	0x00_A00D_0000	64K
SDI Receiver Subsystem	0x00_A003_0000	64K
SDI Transmitter Subsystem	0x00_A004_0000	128K
SDI audio embed	0x00_A001_0000	64K
SDI audio extract	0x00_A002_0000	64K
H.264/H.265 Video Codec Unit (VCU)	0x00_A010_0000	1M
VCU DDR4 Controller	0x48_0000_0000	2G

Table 5-5 shows the address map of various IP blocks used in the PL of an HDMI DDR design.

Table 5-5: **Address Map for HDMI PLDDR Design IP Blocks**

IP Core	Base Address	Offset
HDMI frame buffer write 0	0x00_A001_0000	64K
HDMI frame buffer write 1	0x00_A008_0000	64K
HDMI frame buffer write 2	0x00_A009_0000	64K

Table 5-5: **Address Map for HDMI PLDDR Design IP Blocks *(Cont'd)***

IP Core	Base Address	Offset
HDMI frame buffer write 3	0x00_A00A_0000	64K
HDMI Receiver subsystem	0x00_A000_0000	64K
Video Processing Subsystem (VPSS)	0x00_A004_0000	256K
HDMI frame buffer read	0x00_A00B_0000	64K
HDMI Transmitter subsystem	0x00_A002_0000	128K
Video Mixer	0x00_A00C_0000	64K
AXI IIC Bus Interface	0x00_A00D_0000	4K
Video frame buffer read	0x00_A011_0000	64K
Video frame buffer write	0x00_A012_0000	64K
H.264/H.265 Video Codec Unit (VCU)	0x00_A020_0000	1M
VCU DDR4 Controller	0x48_0000_0000	2G
Video PHY Controller	0x00_A013_0000	64K

Table 5-6 shows the address map of various IP blocks used in the PL of an HDMI DDR design.

Table 5-6: **Address Map for LLP2 Audio NV12 Blocks**

IP Core	Base Address	Offset
Audio Clock Recovery	0x00_A029_0000	64K
Audio Formatter1	0x00_A005_2000	4K
Audio Formatter2	0x00_A005_1000	4K
AXI GPIO	0x00_A005_3000	4K
Audio Clock Wizard	0x00_A00E_0000	64K
AXI Interrupt Controller	0x00_A005_5000	4K
HDMI ACR Control	0x00_A005_6000	4K
HDMI Frame Buffer Write	0x00_A001_0000	64K
HDMI Frame Buffer Write	0x00_A028_0000	64K
HDMI Frame Buffer Write	0x00_A02A_0000	64K
HDMI Frame Buffer Write	0x00_A02B_0000	64K
HDMI Receiver subsystem	0x00_A000_0000	64K
Video Processing Subsystem (VPSS)	0x00_A008_0000	256K
HDMI Frame Buffer Read	0x00_A004_0000	64K
HDMI Transmitter subsystem	0x00_A002_0000	128K
Video Mixer	0x00_A007_0000	64K
Sync IP	0x00_A02C_0000	64K
Video Processing Subsystem (VPSS-CSC)	0x00_A024_0000	64K

Table 5-6: **Address Map for LLP2 Audio NV12 Blocks *(Cont'd)***

IP Core	Base Address	Offset
Video Processing Subsystem (VPSS-Scaler)	0x00_A020_0000	256K
HDMI_ctrl_iic	0x00_A005_0000	4K
H.264/H.265 Video Codec Unit (VCU)	0x00_A010_0000	1M
Video PHY Controller	0x00_A006_0000	64K

Interrupt Map

Table 5-7 shows interrupt ID mapping for the VCU TRD full-fledged design.

Table 5-7: **Interrupt ID Map for Full-fledged Design**

IP Core	Interrupt ID
HDMI CTL IIC	94
HDMI Frame Buffer Read	89
HDMI Frame Buffer Write_0	90
HDMI Frame Buffer Write_1	108
HDMI Frame Buffer Write_2	109
HDMI RX	91
HDMI TX	93
Interrupt Controller	107
MIPI Frame Buffer Write	105
MIPI RX SS	104
Sensor IIC	106
TPG Frame Buffer Write	97
VCU	96
Video mixer	95
Video Physical controller	92
HDMI CTL IIC	94

Note: AXI Interrupt Controller used to accommodate all the interrupts as total number of PL-PS interrupts exceeds 16 in the design.

Table 5-8 shows interrupt ID mapping for VCU audio design.

Table 5-8: **Interrupt ID Map for VCU Audio Design**

IP Core	Interrupt ID
HDMI I2C Controller	94
HDMI 1.4/2.0 Transmitter Subsystem v2.0	93
Video PHY Controller	92
HDMI Frame Buffer Read	89
HDMI Frame Buffer Write	90
HDMI 1.4/2.0 Receiver Subsystem v2.0	91
Audio Formatter MM2S 1	104
Audio Formatter S2MM 1	105
Audio Formatter MM2S 2	106
Audio Formatter S2MM 2	107
I2S transmitter	108
I2S receiver	109
Interrupt Controller	110
VCU Host Interrupt	95
Video Mixer	96

Table 5-9 shows interrupt ID mapping for an Ethernet 10G design.

Table 5-9: **Interrupt ID Map for Ethernet 10G Design**

IP Core	Interrupt ID
HDMI I2C Controller	94
HDMI 1.4/2.0 Transmitter Subsystem v2.0	93
Video Mixer	95
HDMI Frame Buffer Read	89
HDMI Frame Buffer Write 0	90
HDMI 1.4/2.0 Receiver Subsystem v2.0	91
Video PHY Controller	92
VCU	96
DMA S2MM	108
DMA MM2S	111
HDMI Frame Buffer Write 1	109
HDMI Frame Buffer Write 2	110

Table 5-10 shows interrupt ID mapping for an SDI design.

Table 5-10: **Interrupt ID Map for PLDDR SDI Design**

IP Core	Interrupt ID
Audio Formatter mm2s	104
Audio Formatter s2mm	105
Frame Buffer Read	91
Frame Buffer Read	106
Frame Buffer Write	92
Frame Buffer Write	107
SDI Audio Extract	93
SDI RX	89
SDI TX	90
VCU	94

Table 5-11 shows interrupt ID mapping for an SDI design.

Table 5-11: **Interrupt ID Map for LLP2 XV20/NV16 Designs**

IP Core	Interrupt ID
HDMI Frame Buffer Write 0	90
HDMI Frame Buffer Write 1	109
HDMI Frame Buffer Write 2	110
HDMI Frame Buffer Write 3	111
HDMI Rx	91
HDMI Tx	93
HDMI Frame Buffer Read	89
Video Mixer	95
HDMI CTRL IIC	94
SYNC IP	107
VCU	96
Video PHY Controller	92

Table 5-12 shows interrupt ID mapping for an SDI design.

Table 5-12: **Interrupt ID Map for LLP2 Audio NV12 Designs**

IP Core	Interrupt ID
VCU	95
Audio Formatter1 MM2S	104
Audio Formatter1 S2MM	105

Table 5-12: **Interrupt ID Map for LLP2 Audio NV12 Designs *(Cont'd)***

IP Core	Interrupt ID
Audio Formatter2 MM2S	106
Audio Formatter2 S2MM	107
I2S Receiver	109
I2S Transmitter	108
Interrupt Controller	110
Frame Buffer Write 0	90
HDMI Rx	91
HDMI Tx	93
Video Mixer	96
HDMI CTRL IIC	94
Video PHY Controller	92

Table 5-13 shows interrupt ID mapping for an SDI design.

Table 5-13: **Interrupt ID Map for Pixco LLP2 SDI Designs**

IP Core	Interrupt ID
Sync IP	108
Audio Formatter M2SS	104
Audio Formatter MM2S	105
UHDSDI Audio Extract	93
SDI Frame Buffer Write	92
UHDSDI Receiver	89
SDI Frame Buffer Read	91
UHDSDI Transmitter	90
Frame Buffer Read	106
Frame Buffer Write	107
VCU	94

Input Configuration File

This list describes the file format of the input configuration file (`input.cfg`).

Descriptions

- Common Configuration: Starting point of a common configuration
- Num Of Input: Provides the number of inputs. Ranges from 1 to 8
- Output: Selects the video display interface
  - Options: HDMI, SDI, or DP
- Out Type:
  - Options: display, record and stream
- Display Rate: Pipeline frame rate
  - Options: 30 or 60
- Exit: Tells the application that common configuration is finished
- Input Configuration: Starting point of input configuration
- Input Num: Starting n^{th} input configuration
  - Options: 1–8
- Input Type: Input source type
  - Options: TPG, HDMI, HDMI_2, HDMI_3, HDMI_4, HDMI_5, HDMI_6, HDMI_7, MIPI, File, SDI, Stream
- Accelerator Flag: Enables/disables the Vitis™ tool accelerator. For this release, the accelerator works as a bypass filter.
  - Options: True, False
- Enable SCD Flag: Enables/disables the SCD plugin before encoding
- Options: True, False
- Uri: File path or Network URL. Applicable for file playback and stream-in pipeline only. Supported file formats for playback are ts, mp4, and mkv.

    - Options:
        - `file:///media/usb/abc.mp4` (for file path)
        - udp://192.168.25.89:5004/ (for network streaming)
        - Here `192.168.25.89` is the IP address and `5004` is the port number
- Raw: Tells the pipeline to run the raw or processed pipeline
- Options: True, False
    - Width: Width of the live source
    - Options: 4096, 3840, 1920
- Height: Height of the live source
    - Options: 2160, 1080
- Accelerator Flag: Enables/disables the Vitis tool accelerator. For this release, the accelerator functions as a bypass filter.
    - Options: True, False

    Note: This design module is only available for versions up to, and including 2019.1.
- Format: Format of input data
    - Options: NV12, NV16, XV15, XV20
- Enable LLP2: Enable or disable LLP2 pipeline
    - Options: True, False
- Exit: Tells the application when input configuration is finished
- Encoder Configuration: Starting point of encoder configuration
- Encoder Num: Starting n^{th} encoder configuration
    - Options: 1–8
- Encoder Name: Name of encoder
    - Options: AVC, HEVC
- Profile: Name of profile. The default filter is *high* for AVC and *main* for HEVC.
    - Options: Baseline, main, or high for AVC. Main for HEVC
- Rate Control: Rate control options
    - Options: CBR, VBR, and low latency
- Filler Data: Filler data NAL units for CBR rate control
    - Options: True, False
- QP: The QP control mode is used by the VCU encoder

    - Options: Uniform or Auto
- L2 Cache: Enable or disable the L2 Cache buffer in the encoding process
    - Options: True, False
- Latency Mode: Encoder latency mode
    - Options: normal, sub_frame
- Low Bandwidth: If enabled, decreases the vertical search range used for P-frame motion estimation to reduce the bandwidth
    - Options: True, False
- GoP Mode: Group of Pictures mode
    - Options: Basic, low_delay_p, low_delay_b
- Bitrate: Target bit rate in Kbps
    - Options: 1–60000
- B frames: Number of B frames between two consecutive P frames
    - Options: 0–4
- Slice: Number of slices produced for each frame. Each slice contains one or more complete macroblock/coding tree unit (CTU) row(s). Slices are distributed over the frame as regularly as possible. If slice size is defined as well, more slices can be produced to fit the slice size requirement. The default slice value is 8.
    - Options:
        - 4–22 4Kp resolution with HEVC
        - 4–32 4Kp resolution with AVC
        - 4–32 1080p resolution with HEVC
        - 4–32 1080p resolution with AVC
- GoP Length: Distance between two consecutive I frames
    - Options: 1-1000
- GDR Mode: Gradual decoder refresh mode
    - Options: Horizontal, Vertical, Disabled

    Note: GDR mode is currently supported with LLP1 and LLP2 low-delay-p use-cases only.
- Entropy Mode: Entropy mode for H.264 (AVC) encoding process
    - Options: CAVLC, CABAC, Default
- Max Picture Size: Max Picture Size for frame when CBR/VBR rate-control is used to limit instantaneous peak in the bit-stream.
    - Options: TRUE or FALSE

- Format: Format of input data
  - Options: NV12, NV16, XV15, and XV20
- Preset:
  - Options: HEVC_HIGH, HEVC_MEDIUM, HEVC_LOW, AVC_HIGH, AVC_MEDIUM, AVC_LOW, Custom
- Exit: Tells the application encoder that encoder configuration is finished
- Record Configuration: Starting point of a record configuration
- Record Num: Starting n^{th} record configuration
- Options: 1–8
- Out File Name: Record file path
  - Options: `/media, /usb, /abc.ts`
- Duration: Duration in minutes
  - Options: 1–3
- Exit: Tells the application that record configuration is finished
- Streaming Configuration: Starting point of streaming configuration
- Streaming Num: Starting n^{th} streaming configuration
  - Options: 1–8
- Host IP: The host to send the packets to

  Options: 192.168.25.89
- Port: The port to send the packets to
  - Options: 1024–65534
- Exit: Tells the application that streaming configuration is finished
- Audio Configuration: Starting point of audio configuration
- Audio Enable: Enable or disable audio in the pipeline
  - Options: True, False
- Audio Format: Format of the audio
  - Options: S24_32LE

Sampling Rate: Sets audio sampling rate

- Options: 48000

Num of Channel: Number of audio channels

  - Options: 2
- Volume: Sets the volume level
  - Options: 0.0-10.0
- Source: Required audio source
  - Options: HDMI, SDI and I2S
- Renderer: Required audio sink

  Options: HDMI, SDI I2S and DP
- Exit: Indicates to the application that audio configuration is finished
- Trace Configuration: Starting point of trace configuration
- FPS Info: Displays fps info on the console
  - Options: True, False
- APM Info: Displays the apm counter number on the console
  - Options: True, False
- Pipeline Info: Displays pipeline info on the console
  - Options: True, False
- Exit: Tells the application that trace configuration is finished

Appendix B

Additional Resources and Legal Notices

Xilinx Resources

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

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.

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® integrated design environment (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

The most up-to-date information for this design is available on these websites:

Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit

Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit Documentation

Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit Master Answer Record (AR 69344)

H.264/H.265 Video Codec Unit IP Core website

Zynq UltraScale+ MPSoC VCU TRD wiki for 2020.1

These documents and sites provide supplemental material:

1. *GStreamer open source media framework* (gstreamer.freedesktop.org/)
2. *ZCU106 Evaluation Board User Guide* (UG1244)
3. Leopard Imaging Inc. website
4. Xilinx Software Development Kit (XSDK)
5. OpenMAX website
6. Advanced Linux Sound Architecture (ALSA) project homepage
7. *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137)
8. *Zynq UltraScale+ Device Technical Reference Manual* (UG1085)
9. *Video Test Pattern Generator LogiCORE IP Product Guide* (PG103)
10. *Video Frame Buffer Read and Video Frame Buffer Write LogiCORE IP Product Guide* (PG278)
11. *Video PHY Controller LogiCORE IP Product Guide* (PG230)
12. *HDMI 1.4/2.0 Receiver Subsystem Product Guide* (PG236)
13. *Video Processing Subsystem Product Guide* (PG231)
14. *MIPI CSI-2 Receiver Subsystem LogiCORE IP Product Guide* (PG232)
15. *Video Mixer LogiCORE IP Product Guide* (PG243)
16. *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210)
17. *AXI DMA LogiCORE IP Product Guide* (PG021)
18. *HDMI 1.4/2.0 Transmitter Subsystem Product Guide* (PG235)
19. Intel Platform Management Field Replaceable Unit (FRU) Information Storage Definition

20. Open Asymmetric Multi Processing (OpenAMP) framework repository (github.com/OpenAMP/open-amp)
21. *Zynq UltraScale+ MPSoC Data Sheet: Overview* (DS891)
22. *H.264/H.265 Video Codec Unit LogiCORE IP Product Guide* (PG252)
23. *Video Scene Change Detection LogiCORE IP Product Guide* (PG322)
24. *I2S Transmitter and I2S Receiver LogiCORE IP Product Guide* (PG308)
25. *SMPTE UHD-SDI Transmitter Subsystem v2.0 LogiCORE IP Product Guide* (PG289)
26. *SMPTE UHD-SDI Receiver Subsystem v2.0 LogiCORE IP Product Guide* (PG290)
27. *UHD SDI Audio v1.0 LogiCORE IP Product Guide* (PG309)
28. *Audio Formatter v1.0 LogiCORE IP Product Guide* (PG330)
29. *DMA/Bridge Subsystem for PCI Express Product Guide* (PG195)

Please Read: Important Legal Notices