Video Processing Subsystem v2.4

Product Guide

Vivado Design Suite

PG231 (v2.4) February 21, 2024

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

Table of Contents

Introduction

The Video Processing Subsystem is a collection of video processing IP subcores, bundled together in hardware and software, abstracting the video processing pipe. It provides the end-user with an out of the box ready to use video processing core, without having to learn about the underlying complexities. The Video Processing Subsystem enables streamlined integration of various processing blocks including (but not limited to) scaling, deinterlacing, color space conversion and correction, chroma resampling, and frame rate conversion.

Features

- One, two, four, and eight pixel-wide AXI4-Stream video interface
  - If deinterlacer is enabled in the data pipeline, then the overall processing subsystem gives one sample per clock equivalent performance
- Video resolution support up to 8k at 60 fps. If deinterlacer is enabled in the data pipeline, the maximum video resolution that can be supported is 1080p at 60 fps.
- Runtime color space support for RGB, YUV 4:4:4, YUV 4:2:2, YUV 4:2:0
- 8, 10, 12, and 16 bits per component support
- Deinterlacing: supports 32-bit and 64-bit memory address
- Scaling
- Color space conversion and correction
- Chroma resampling between YUV 4:4:4, YUV 4:2:2, YUV 4:2:0
- Frame rate conversion using dropped/repeated frames

IP Facts

AMD LogiCORE™ IP Facts Table	
Subsystem Specifics	
Supported Device Family[1]	AMD UltraScale+™ Families, AMD UltraScale™ Architecture, AMD Zynq™ 7000, AMD 7 series, AMD Versal™ Adaptive SoC
Supported User Interfaces	AXI4-Lite, AXI4-Stream, AXI4
Resources	See Table 1
Provided with Subsystem	
Design Files	Encrypted HLS C
Example Design	Verilog
Test Bench	Not Provided
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver[2]	Standalone, Linux
Tested Design Flows[3]	
Design Entry	AMD Vivado™ Design Suite
Simulation	For supported simulators, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).
Synthesis	Vivado Design Suite
Support	
Release Notes and Known Issues	Master Answer Record: 65449
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Support web page	

Notes:

1. For a complete list of supported devices, see the AMD Vivado™ IP catalog.
2. Standalone driver details can be found in Vitis directory (<install_directory>/Vitis/<release>/data/embeddedsw/doc/xilinx_drivers.htm). Linux: Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of third-party tools, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).

Overview

Introduction

The Video Processing Subsystem enables streamlined integration of various processing blocks including (but not limited to) scaling, deinterlacing, color space conversion and correction, chroma resampling, and frame rate conversion.

Navigating Content by Design Process

AMD Adaptive Computing documentation is organized around a set of standard design processes to help you find relevant content for your current development task. You can access the AMD Versal™ adaptive SoC design processes on the Design Hubs page. You can also use the Design Flow Assistant to better understand the design flows and find content that is specific to your intended design needs.

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

Feature Summary

The Video Processing Subsystem has the following features:

- One, two, four, and eight pixel-wide video interface
  - If deinterlacer is enabled in the data pipeline, then the overall processing subsystem gives 1 sample per clock equivalent performance
- Runtime color space support for RGB, YUV 4:4:4, YUV 4:2:2, YUV 4:2:0
- 8, 10, 12, and 16 bits per component support
- Deinterlacing: supports 32-bit and 64-bit memory address
- Scaling
- Color space conversion and correction
- Chroma resampling between YUV 4:4:4, YUV 4:2:2, YUV 4:2:0
- Frame rate conversion
- Supports resolutions up to 8192 x 4320

The Video Processing Subsystem is a hierarchical IP that bundles a collection of video processing IP subcores and outputs them as a single IP. The Video Processing Subsystem has design time configurability in terms of performance and quality. You can configure the subsystem IP through one single graphical user interface. A preview of this GUI is shown in Figure 18.

All video processing IP subcores are developed using AMD Vitis™ HLS.

Applications

- Color space (RGB/YUV) and format (YUV 4:4:4/4:2:2/4:2:0) conversion.
- Scale up and down up to 8k/4k at 60 Hz.
- Zoom mode, wherein a user-defined window, the input stream is scaled to panel resolution.
- Picture-In-Picture mode wherein the input stream is scaled down to a user-defined window size and displayed at the user defined co-ordinates on the panel.
  - Ability to paint the PIP background to a defined color.
- Interlaced to progressive conversion.
- Frame rate conversion:
  - Drop frames if input rate > output rate.

- Repeat frames if output rate < input rate.

Licensing and Ordering

This AMD LogiCORE™ IP module is provided at no additional cost with the AMD Vivado™ Design Suite under the terms of the End User License.

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

Chapter 3

Product Specification

Standards

The Video Processing Subsystem core is compliant with the AXI4-Stream Video Protocol and AXI4-Lite interconnect standards. Refer to the Video IP: AXI Feature Adoption section of the *Vivado Design Suite: AXI Reference Guide* (UG1037) for additional information.

Performance

Maximum Frequencies

The following are typical clock frequencies for the target devices:

- AMD Virtex™ 7 devices with –2 speed grade, AMD Virtex™ UltraScale™, and AMD Virtex UltraScale+ devices with –2 speed grade or higher: 300 MHz
- AMD Kintex™ 7 devices with –2 speed grade, AMD Kintex™ UltraScale+™, and AMD Kintex UltraScale+ devices with –2 speed grade or higher: 300 MHz
- AMD UltraScale+ devices with -1 speed grade or higher: 300 MHz
- AMD Artix™ 7 devices with –2 speed grade or higher: 150 MHz
- AMD Zynq™ 7000 SoC devices with –2 and –1 speed grade or higher: 200 MHz
- AMD Versal™ adaptive SoC with -1 speed grade or higher: 300 MHz

The maximum achievable clock frequency can vary. The maximum achievable clock frequency and all resource counts can be affected by other tool options, additional logic in the device, using a different version of AMD tools, and other factors.

Latency

The latency of the Video Processing Subsystem depends on the configuration of the core. Generally, the latency is on the order of several lines. For example, a vertical scaler with a 6-tap polyphase filter operating on RGB video introduces four video lines of delay.

In the Full Fledged configuration with the use of a DMA, the latency is one full frame time plus several lines because the video DMA engine is used in the data flow and programmed to read one frame buffer behind the write frame buffer location.

In interlaced video, one of field additional delay is incurred by the deinterlacer algorithm.

Scaler Only Mode

In Scaler Only mode, at the start of the first frame, the Scaler will copy the coefficients from the AXI4-Lite interface into a local RAM. This will take about 500 cycles. After this, the scaler should be able to start filling its internal line buffer. Scaler starts generating output when the line buffer is filled about half full. If the supports 4:2:0, the buffer for chroma resampling will also need to be filled. As a safe measure, Scaler uses the number of vertical taps for the number of lines of latency. After every line, the Scaler will have to do some bookkeeping, and therefore for about 20 cycles the Scaler will not be able to accept data. You can reduce the initial latency by reducing the number of vertical taps.

Resource Use

The following table shows the representative resource utilization for several configurations of this IP core. In the following table, each row describes a test case. The columns are divided into test parameters and results. The test parameters describe the core configuration. Any configuration parameters that are not listed have their default values.

The default AMD Vivado Design Suite settings were used. You might be able to optimize on these numbers using different settings. Because surrounding circuitry affects placement and timing, no guarantee can be given that these figures are repeatable in a larger design.

All configurations (except where noted), are configured for four samples per clock, 10-bit data width, and a frame size of 8192 x 4320.

Table 1: **Video Processing Subsystem Resource Utilization**

Configuration	LUTs	FFs	DSPs	36k BRAMs	18k BRAMs
Full Fledged	47846	66309	249	80	29
Full Fledged • Deinterlacing disabled DMA excluded	33968	46474	244	45	14
Scaler Only • RGB/YUV444/422/420 support color space conversion enabled	11510	15176	176	22	10
Scaler Only • RGB/YUV 444 support only	9198	11939	176	8	2

Table 1: **Video Processing Subsystem Resource Utilization** *(cont'd)*

Configuration	LUTs	FFs	DSPs	36k BRAMs	18k BRAMs
Deinterlacing Only	6691	8169	5	35	3
Color Space Conversion Only • RGB/YUV 444 support only demo window disabled	3349	3972	36	0	0
420-422 Chroma Resampling Only	1470	2261	16	12	7
422-444 Chroma Resampling Only	2073	2527	32	0	0

Notes:

1. These numbers were generated using Vivado Design Suite 2022.1 with Speed file PRODUCTION 1.29 08-03-2020 on Device xczu7evffvc1156-2-e.

Port Descriptions

The following figure shows the Video Processing Subsystem diagram in its Full Fledged configuration. The IP has four AXI interfaces:

- AXI4-Stream streaming video input (`s_axis`)
- AXI4-Stream streaming video output (`m_axis`)
- AXI-MM memory mapped interface (`m_axi_mm`)
- AXI-Lite control interface (`s_axi_ctrl`).

Figure 1: **Full Fledged Video Processing Subsystem**

The AXI Streaming, AXI Memory-Mapped, and AXI Lite interfaces can be run at their own clock rate, therefore, three separate clock interfaces are provided named `aclk_axis`, `aclk_axi_mm`, and `aclk_ctrl`, respectively. The `aresetn_ctrl` signal is the reset signal of the IP, and `aresetn_io_axis` is an outgoing signal that can be used to hold IPs in reset when the Video Processing Subsystem is not ready to consume data on the streaming input. Finally, `deint_field_id` signal indicates field polarity in case of interlaced operation.

AXI4-Stream Video

The Video Processing Subsystem has AXI4-Stream video input and output interfaces named `s_axis` and `m_axis`, respectively. These interfaces follow the interface specification as defined in the Video IP chapter of the *Vivado Design Suite: AXI Reference Guide* (UG1037). The video AXI4-Stream interface can be single, dual, quad, or octa pixels per clock and can support 8, 10, 12, or 16 bits per component. For example, the pixel mapping per color format and bus signals for 10 bits per component are shown in Table 2 through Table 6.

Table 2: **Dual Pixels per Clock, 10 Bits per Component Mapping for RGB**

63:60	59:50	49:40	39:30	29:20	19:10	9:0
zero padding	R1	B1	G1	R0	B0	G0

Table 3: **Dual Pixels per Clock, 10 Bits per Component Mapping for YUV 4:4:4**

63:60	59:50	49:40	39:30	29:20	19:10	9:0
zero padding	V1	U1	Y1	V0	U0	Y0

Table 4: **Dual Pixels per Clock, 10 Bits per Component Mapping for YUV 4:2:2**

63:60	59:50	49:40	39:30	29:20	19:10	9:0
zero padding	zero padding	zero padding	V0	Y1	U0	Y0

Table 5: **Dual Pixels per Clock, 10 Bits per Component Mapping for YUV 4:2:0, for Even Lines**

63:60	59:50	49:40	39:30	29:20	19:10	9:0
zero padding	zero padding	zero padding	V0	Y1	U0	Y0

Table 6: **Dual Pixels per Clock, 10 Bits per Component Mapping for YUV 4:2:0, for Odd Lines**

63:60	59:50	49:40	39:30	29:20	19:10	9:0
zero padding	zero padding	zero padding	zero padding	Y1	zero padding	Y0

This IP always generates three video components even if the video format is set to be YUV 4:2:0 or YUV 4:2:2 at runtime. The unused components can be set to zero. All video streaming interfaces follow the interface specification as defined in the AXI4-Stream Video IP and System Design Guide (UG934)

The following table shows the interface signals for input and output AXI4-Stream video streaming interfaces.

Table 7: **AXI4 Streaming Interface Signals**

Name	Direction	Width	Description
s_axis_tdata	In	floor(((3 x bits_per_component x pixels_per_clock) + 7) / 8) x 8	Input Data
s_axis_tready	Out	1	Input Ready
s_axis_tvalid	In	1	Input Valid
s_axis_tid	In	1	Input data stream identifier
s_axis_tdest	In	1	Input data routing identifier
s_axis_tkeep	In	(s_axis_video_tdata width)/8	Input byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream
s_axis_tlast	In	1	Input End of Line
s_axis_tstrb	In	(s_axis_video_tdata width)/8	Input byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte
s_axis_tuser	In	1	Input Start of frame
m_axis_tdata	Out	floor(((3 x bits_per_component x pixels_per_clock) + 7) / 8) x 8	Output Data
m_axis_tdest	Out	1	Output data routing identifier
m_axis_tid	Out	1	Output data stream identifier
m_axis_tkeep	Out	(m_axis_video_tdata width)/8	Output byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream
m_axis_tlast	Out	1	Output End of Line
m_axis_tready	In	1	Output Ready
m_axis_tstrb	Out	(m_axis_video_tdata width)/8	Output byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte
m_axis_tuser	Out	1	Output Start of frame
m_axis_tvalid	Out	1	Output Valid

Both video streaming interfaces run at the video stream clock speed `aclk_axis`.

AXI-MM Memory-Mapped Interface

The video DMA read and write ports and the deinterlacer read and write ports are concentrated by an AXI4 cross-bar interconnect such that there is only one AXI4 interface on the boundary of the subsystem. The AXI4 interface runs on the `aclk_axi_mm` clock domain. The signals follow the specification as defined in the *Vivado Design Suite: AXI Reference Guide* (UG1037).

The AXI4 interface is only present with the Full Fledged functionality and Deinterlacer Only configurations. The other configurations do not require access to eternal memory.

For full fledged mode, when the Built-in DMA is enabled, the MM interface data-width can be either of 128, 256 or 512 bits. Data-width is automatically configured based on the PPC of the design.

AXI-Lite Control Interface

The following table shows the AXI4-Lite control interface signals. This interface runs at the `aclk_ctrl` clock. Control of the video processing pipe is only supported through the Video Processing Subsystem driver. Data-width is automatically configured based on the PPC of the design.

***Note*:** Control of the video processing pipe is only supported through the video processing subsystem driver. The register map is provided for debug purposes only.

***Note*:** AXI write and read data_width selection is compatible to the part selected.

For example : KU+ can support only up to 64 bits and ZU+/Versal supports up to 128 bits.

Table 8: **AXI Lite Control Interface**

Name	Direction	Width	Description
s_axi_ctrl_awaddr	In	20	Write address
s_axi_ctrl_awprot	In	3	Write address protection
s_axi_ctrl_awvalid	In	1	Write address valid
s_axi_ctrl_awready	Out	1	Write address ready
s_axi_ctrl_wdata	In	32	Write data
s_axi_ctrl_wstrb	In	4	Write data strobe
s_axi_ctrl_wvalid	In	1	Write data valid
s_axi_ctrl_wready	Out	1	Write data ready
s_axi_ctrl_bresp	Out	2	Write response
s_axi_ctrl_bvalid	Out	1	Write response valid
s_axi_ctrl_bready	In	1	Write response ready
s_axi_ctrl_araddr	In	20	Read address

Table 8: **AXI Lite Control Interface** *(cont'd)*

Name	Direction	Width	Description
s_axi_ctrl_arprot	In	3	Read address protection
s_axi_ctrl_arvalid	In	1	Read address valid
s_axi_ctrl_aready	Out	1	Read address ready
s_axi_ctrl_rdata	Out	32	Read data
s_axi_ctrl_rresp	Out	2	Read data response
s_axi_ctrl_rvalid	Out	1	Read data valid
s_axi_ctrl_rready	In	1	Read data ready

Clocks and Resets

The following table provides an overview of the clocks and resets. See section Clocking for more information.

Table 9: **Clocks and Resets**

Name	Direction	Width	Description
Clocks			
aclk_axis	In	1	Clock at which AXI4-Stream input and output are running.
aclk_ctrl	In	1	AXI4-Lite clock for CPU control interface.
aclk_axi_mm	In	1	Clock at which AXI4 interface is running.
Resets			
aresetn_ctrl	In	1	Reset, associated with `aclk_ctrl` (active-Low). The `aresetn_ctrl` signal resets the entire IP including the data path and AXI4-Lite registers.
aresetn_io_axis	Out	1	Used to hold upstream logic in reset while the Video Processing Subsystem is not ready to consume data on streaming input (active-Low).

Field Polarity

The `deint_field_id` signal indicates the polarity of the incoming field when the input video is interlaced. This signal is only used by the deinterlacer with interlaced data. This signal is ignored for progressive video inputs. If needed, the user can propagate the Input field ID to the down stream modules.

Table 10: **Field Polarity**

Name	Direction	Width	Description
deint_field_id	In	1	Field polarity, odd is low, high is even

Register Space

The following sections describe the register mapping of the VPSS subsystem for different modes.

***Note*:** Control of the video processing pipe is only supported through the Video Processing Subsystem driver. The register map is provided for debug purposes only.

Scaler Only Mode Registers

The scaler only mode has specific registers that allow you to dynamically control the operation of the core. Table 11 provides a detailed description of all the registers that apply globally to the IP.

The scaler only configuration consists of the following register configurable IPs:

- Vertical scaler
- Horizontal scaler
- GPIO

The AXI interconnect bundles the AXI4-Lite interfaces of the scalers and GPIO into in AXI4-Lite interface at the subsystem boundary. The GPIO block allows for providing a soft reset function for the scaler subsystem.

Vertical Scaler

The following table provides the register map of vertical scaler registers in the video processing subsystem.

Table 11: **Vertical Scaler Registers**

Register	Description
0x000	Control signals • bit 0 - ap_start (Read/Write/COH) • bit 1 - ap_done (Read/COR) • bit 2 - ap_idle (Read) • bit 3 - ap_ready (Read) • bit 7 - auto_restart (Read/Write) • Others - reserved

Table 11: **Vertical Scaler Registers** *(cont'd)*

Register	Description
0x004	Global Interrupt Enable Register • bit 0 - Global Interrupt Enable (Read/Write) • Others - reserved
0x008	IP Interrupt Enable Register (Read/Write) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x00c	IP Interrupt Status Register (Read/TOW) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x010	Input Height • bit 15~0 - HwReg_HeightIn[15:0] (Read/Write) • Others - reserved
0x014	Reserved
0x018	Width • bit 15~0 - HwReg_HeightOut[15:0] (Read/Write) • Others - reserved
0x01c	Reserved
0x020	Out Height • bit 15~0 - HwReg_HeightOut[15:0] (Read/Write) • Others - reserved
0x024	Reserved
0x028	Line Rate • bit 31~0 - HwReg_LineRate[31:0] (Read/Write)
0x02c	Reserved
0x030	Color Mode • bit 7~0 - HwReg_ColorMode[7:0] (Read/Write) • Others - reserved
0x034	Reserved
0x800	Vertical Filter Coefficients Address (64 * NR TAPS * 16b) Word n • bit [15: 0] - HwReg_vfltCoeff[2n] • bit [31:16] - HwReg_vfltCoeff[2n+1]

Horizontal Scaler

The following table provides the register map of horizontal scaler registers in the video processing subsystem.

Table 12: **Horizontal Scaler Registers**

Register	Description
0x000	Control signals • bit 0 - ap_start (Read/Write/COH) • bit 1 - ap_done (Read/COR) • bit 2 - ap_idle (Read) • bit 3 - ap_ready (Read) • bit 7 - auto_restart (Read/Write) • Others - reserved
0x004	Global Interrupt Enable Register • bit 0 - Global Interrupt Enable (Read/Write) • Others - reserved
0x008	IP Interrupt Enable Register (Read/Write) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x00c	IP Interrupt Status Register (Read/TOW) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x010	Height • bit 15~0 - HwReg_HeightIn[15:0] (Read/Write) • Others - reserved
0x014	Reserved
0x018	Input Width • bit 15~0 - HwReg_Width[15:0] (Read/Write) • Others - reserved
0x01c	Reserved
0x020	Output Width • bit 15~0 - HwReg_HeightOut[15:0] (Read/Write) • Others - reserved
0x024	Reserved
0x028	Color Mode • bit 7~0 - HwReg_ColorMode[7:0] (Read/Write) • Others - reserved
0x02c	Reserved
0x030	Pixel Rate • bit 31~0 - HwReg_PixelRate[7:0] (Read/Write)
0x034	Reserved
0x038	Output Color Mode • bit 7~0 - HwReg_ColorModeOut[7:0] (Read/Write) • Others - reserved
0x0400	~

Table 12: **Horizontal Scaler Registers** *(cont'd)*

Register	Description
0x07ff	Horizontal Filter Coefficients Address (384 * 16b) Word n : • bit [15: 0] - HwReg_hfltCoeff[2n] • bit [31:16] - HwReg_hfltCoeff[2n+1]
0x4000	Phase Address (8192 * 9b) Word n : • bit [8: 0] - HwReg_phasesH_V[2n] • bit [24:16] - HwReg_phasesH_V[2n+1] • Others - reserved

GPIO

The VPSS in the scaler only mode has two resets. One of them can be driven from an external source which resets the complete VPSS. The other reset is internal to the VPSS subsystem in the scaler only mode and can be controlled by programming the GPIO. The internal reset controls the resetting of vscaler, hscaler, stream FIFOs, and the outgoing `aresetn_io_axis` pin.

For more information, see *AXI GPIO LogiCORE IP Product Guide* (PG144).

Color Space Conversion Only

The following table provides the register map of Color space conversion registers in video processing subsystem.

Table 13: **Color Space Conversion Only Registers**

Register	Description
0x000	Control signals • bit 0 - ap_start (Read/Write/COH) • bit 1 - ap_done (Read/COR) • bit 2 - ap_idle (Read) • bit 3 - ap_ready (Read) • bit 7 - auto_restart (Read/Write) • Others - reserved
0x004	Global Interrupt Enable Register • bit 0 - Global Interrupt Enable (Read/Write) • Others - reserved
0x008	IP Interrupt Enable Register (Read/Write) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved

Table 13: **Color Space Conversion Only Registers** *(cont'd)*

Register	Description
0x00c	IP Interrupt Status Register (Read/TOW) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x010	Input Video Format • bit 7~0 - HwReg_InVideoFormat[7:0] (Read/Write) • Others - reserved
0x014	Reserved
0x018	Output Video Format • bit 7~0 - HwReg_OutVideoFormat[7:0] (Read/Write) • Others - reserved
0x01c	Reserved
0x020	Width • bit 15~0 - HwReg_width[15:0] (Read/Write) • Others - reserved
0x024	Reserved
0x028	Height • bit 15~0 - HwReg_height[15:0] (Read/Write) • Others - reserved
0x02C	Reserved
0x030	Column Start • bit 15~0 - HwReg_ColStart[15:0] (Read/Write) • Others - reserved
0x034	Reserved
0x038	Column End • bit 15~0 - HwReg_ColEnd[15:0] (Read/Write) • Others - reserved
0x03c	Reserved
0x040	Row Start • bit 15~0 - HwReg_RowStart[15:0] (Read/Write) • Others - reserved
0x044	Reserved
0x048	Row End • bit 15~0 - HwReg_RowEnd[15:0] (Read/Write) • Others - reserved
0x04c	Reserved
0x050	Coefficient K11 • bit 15~0 - HwReg_K11[15:0] (Read/Write) • Others - reserved
0x054	Reserved

Table 13: **Color Space Conversion Only Registers** *(cont'd)*

Register	Description
0x058	Coefficient K12 • bit 15~0 - HwReg_K12[15:0] (Read/Write) • Others - reserved
0x05c	Reserved
0x060	Coefficient K13 • bit 15~0 - HwReg_K13[15:0] (Read/Write) • Others - reserved
0x064	Reserved
0x068	Coefficient K21 • bit 15~0 - HwReg_K21[15:0] (Read/Write) • Others - reserved
0x06c	Reserved
0x070	Coefficient K22 • bit 15~0 - HwReg_K22[15:0] (Read/Write) • Others - reserved
0x074	Reserved
0x078	Coefficient K23 • bit 15~0 - HwReg_K23[15:0] (Read/Write) • Others - reserved
0x07c	Reserved
0x080	Coefficient K31 • bit 15~0 - HwReg_K31[15:0] (Read/Write) • Others - reserved
0x084	Reserved
0x088	Coefficient K32 • bit 15~0 - HwReg_K32[15:0] (Read/Write) • Others - reserved
0x08c	Reserved
0x090	Coefficient K33 • bit 15~0 - HwReg_K33[15:0] (Read/Write) • Others - reserved
0x094	Reserved
0x098	Coefficient R Offset • bit 11~0 - HwReg_ROffset_V[11:0] (Read/Write) • Others - reserved
0x09c	Reserved
0x0a0	Coefficient G Offset • bit 11~0 - HwReg_GOffset_V[11:0] (Read/Write) • Others - reserved
0x0a4	Reserved

Table 13: **Color Space Conversion Only Registers** *(cont'd)*

Register	Description
0x0a8	Coefficient B Offset • bit 11~0 - HwReg_BOffset_V[11:0] (Read/Write) • Others - reserved
0x0ac	Reserved
0x0b0	Clamp Minimum • bit 9~0 - HwReg_ClampMin_V[9:0] (Read/Write) • Others - reserved
0x0b4	Reserved
0x0b8	Clamp Maximum • bit 9~0 - HwReg_ClipMax_V[9:0] (Read/Write) • Others - reserved
0x0bc	Reserved
0x0c0	Coefficient K11_2 • bit 15~0 - HwReg_K11_2[15:0] (Read/Write) • Others - reserved
0x0c4	Reserved
0x0c8	Coefficient K12_2 • bit 15~0 - HwReg_K12_2[15:0] (Read/Write) • Others - reserved
0x0cc	Reserved
0x0d0	Coefficient K13_2 • bit 15~0 - HwReg_K13_2[15:0] (Read/Write) • Others - reserved
0x0d4	Reserved
0x0d8	Coefficient K21_2 • bit 15~0 - HwReg_K21_2[15:0] (Read/Write) • Others - reserved
0x0dc	Reserved
0x0e0	Coefficient K22_2 • bit 15~0 - HwReg_K22_2[15:0] (Read/Write) • Others - reserved
0x0e4	Reserved
0x0e8	Coefficient K23_2 • bit 15~0 - HwReg_K23_2[15:0] (Read/Write) • Others - reserved
0x0ec	Reserved
0x0f0	Coefficient K31_2 • bit 15~0 - HwReg_K31_2[15:0] (Read/Write) • Others - reserved
0x0f4	Reserved

Table 13: **Color Space Conversion Only Registers** *(cont'd)*

Register	Description
0x0f8	Coefficient K32_2 • bit 15~0 - HwReg_K32_2[15:0] (Read/Write) • Others - reserved
0x0fc	Reserved
0x100	Coefficient K33_2 • bit 15~0 - HwReg_K33_2[15:0] (Read/Write) • Others - reserved
0x104	Reserved
0x108	R Offset 2 • bit 11~0 - HwReg_ROffset_2_V[11:0] (Read/Write) • Others - reserved
0x10c	Reserved
0x110	G Offset 2 • bit 11~0 - HwReg_GOffset_2_V[11:0] (Read/Write) • Others - reserved
0x114	Reserved
0x118	B Offset 2 • bit 11~0 - HwReg_BOffset_2_V[11:0] (Read/Write) • Others - reserved
0x11c	Reserved
0x120	Clamp Minimum 2 • bit 9~0 - HwReg_ClampMin_2_V[9:0] (Read/Write) • Others - reserved
0x124	Reserved
0x128	Clamp Maximum 2 • bit 9~0 - HwReg_ClipMax_2_V[9:0] (Read/Write) • Others - reserved
0x12c	Reserved

422-444 Chroma Resampling Only

The following table provides the register map of 422-444 chroma resampling only registers in video processing subsystem.

Table 14: **422-444 Chroma Resampling Only Registers**

Register	Description
0x000	Control signals • bit 0 - ap_start (Read/Write/COH) • bit 1 - ap_done (Read/COR) • bit 2 - ap_idle (Read) • bit 3 - ap_ready (Read) • bit 7 - auto_restart (Read/Write) • Others - reserved
0x004	Global Interrupt Enable Register • bit 0 - Global Interrupt Enable (Read/Write) • Others - reserved
0x008	IP Interrupt Enable Register (Read/Write) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x00c	IP Interrupt Status Register (Read/TOW) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x010	Width • bit 15~0 - HwReg_width[15:0] (Read/Write) • Others - reserved
0x014	Reserved
0x018	Height • bit 15~0 - HwReg_height[15:0] (Read/Write) • Others - reserved
0x01c	Reserved
0x020	Input Video Format • bit 7~0 - HwReg_input_video_format[7:0] (Read/Write) • Others - reserved
0x024	Reserved
0x028	Output Video Format • bit 7~0 - HwReg_output_video_format[7:0] (Read/Write) • Others - reserved
0x02c	Reserved
0x030	Coefficients 0 • bit 15~0 - HwReg_coefs_0_0[15:0] (Read/Write) • Others - reserved
0x034	Reserved
0x038-0x0C8	Coefficients 1 • bit 15~0 - HwReg_coefs_<P>_<T>[15:0] (Read/ Write) • Others - reserved

Table 14: **422-444 Chroma Resampling Only Registers** *(cont'd)*

Register	Description
0xcc	Reserved

Notes:

1. SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake.

420-422 Chroma Resampling Only

The following table provides the register map of 420-422 chroma resampling only registers in video processing subsystem.

Table 15: **420-422 Chroma Resampling Only Registers**

Register	Description
0x000	Control signals • bit 0 - ap_start (Read/Write/COH) • bit 1 - ap_done (Read/COR) • bit 2 - ap_idle (Read) • bit 3 - ap_ready (Read) • bit 7 - auto_restart (Read/Write) • Others - reserved
0x004	Global Interrupt Enable Register • bit 0 - Global Interrupt Enable (Read/Write) • Others - reserved
0x008	IP Interrupt Enable Register (Read/Write) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x0c	IP Interrupt Status Register (Read/TOW) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x010	Width • bit 15~0 - HwReg_width[15:0] (Read/Write) • Others - reserved
0x014	Reserved
0x018	Height • bit 15~0 - HwReg_height[15:0] (Read/Write) • Others - reserved
0x01c	Reserved
0x020	Input Video Format • bit 15~0 - HwReg_input_video_format[15:0] (Read/Write) • Others - reserved

Table 15: **420-422 Chroma Resampling Only Registers** *(cont'd)*

Register	Description
0x024	Reserved
0x028	Output Video Format • bit 15~0 - HwReg_output_video_format[15:0] (Read/ Write) • Others - reserved
0x02c	Reserved
0x030	Coefficients 0 • bit 15~0 - HwReg_coefs_0_0[15:0] (Read/Write) • Others - reserved
0x034	Reserved
0x038-0x0C8	Coefficients 1 • bit 15~0 - HwReg_coefs_<P>_<T>[15:0] (Read/ Write) • Others - reserved
0xcc	Reserved

Notes:

1. SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake.

Deinterlace Only

The following table provides the register map of deinterlacer registers in video processing subsystem.

Table 16: **Deinterlacer Registers**

Register	Description
0x000	Control signals • bit 0 - ap_start (Read/Write/COH) • bit 1 - ap_done (Read/COR) • bit 2 - ap_idle (Read) • bit 3 - ap_ready (Read) • bit 7 - auto_restart (Read/Write) • Others - reserved
0x004	Global Interrupt Enable Register • bit 0 - Global Interrupt Enable (Read/Write) • Others - reserved
0x008	IP Interrupt Enable Register (Read/Write) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved

Table 16: **Deinterlacer Registers** *(cont'd)*

Register	Description
0x00c	IP Interrupt Status Register (Read/TOW) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x010	Width • bit 15~0 - HwReg_width[15:0] (Read/Write) • Others - reserved
0x014	Reserved
0x018	Height • bit 15~0 - HwReg_height[15:0] (Read/Write) • Others - reserved
0x01c	Reserved
0x20	Read Frame Buffer • bit 31~0 - read_fb[31:0] (Read/Write)
0x024	Read Frame Buffer • bit 31~0 - read_fb[63:32] (Read/Write)
0x028	Reserved
0x030	Color Format • bit 7~0 - colorFormat[7:0] (Read/Write) • Others - reserved
0x034	Reserved
0x38	Algorithm • bit 7~0 - algo[7:0] (Read/Write) ◦ 0x00 - median algorithm ◦ 0x01 - bob algorithm ◦ 0x02 - weave algorithm ◦ 0x03 - vertical temporal linear interpolation algorithm ◦ 0x05 - reserved ◦ 0x06 - pass through • Others - reserved
0x03c	Reserved
0x40	Invert Field ID • bit 0 - invert_field_id[0] (Read/Write) • Others - reserved
0x044	Reserved
0x050	Write Frame Buffer • bit 31~0 - write_fb[31:0] (Read/Write)
0x054	Write Frame Buffer • bit 31~0 - write_fb[63:32:0] (Read/Write)

Table 16: **Deinterlacer Registers** *(cont'd)*

Register	Description
0x058	Reserved

Notes:

- SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake.

Full Fledged

Full fledged mode uses all the register maps of deinterlacer only, scaler only, 420-422 chroma resampling, and 422-444 chroma resampling only registers mentioned above. Letterbox is also included in full fledge mode. In this Full Fledged mode there are two AXI GPIO IPs in the VPSS. One AXI GPIO, that is, `reset_sel_axis` is configured with width 2. This resets the subcores like `v_csc`, `v_vcresampler`, `v_hcresampler`, `v_letterbox`, `v_hscaler`, and `v_vscaler` through the upper bit of its 2-bit width output and the lower bit of its 2-bit output is connected to the VPSS output signal, that is, `arestn_io_axis[0:0]`. The second AXI GPIO, that is, `reset_sel_axi_mm` is configured with width 1 to reset the `v_deinterlacer` and VDMA IPs.

For more information, see *AXI GPIO LogiCORE IP Product Guide* (PG144).

Letterbox

The following table provides the register map of letterbox registers in the Video Processing Subsystem.

Table 17: **Letterbox Registers**

Register	Description
0x000	Control signals • bit 0 - ap_start (Read/Write/COH) • bit 1 - ap_done (Read/COR) • bit 2 - ap_idle (Read) • bit 3 - ap_ready (Read) • bit 7 - auto_restart (Read/Write) • Others - reserved
0x004	Global Interrupt Enable Register • bit 0 - Global Interrupt Enable (Read/Write) • Others - reserved
0x008	IP Interrupt Enable Register (Read/Write) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved

Table 17: **Letterbox Registers** *(cont'd)*

Register	Description
0x00c	IP Interrupt Status Register (Read/TOW) • bit 0 - Channel 0 (ap_done) • bit 1 - Channel 1 (ap_ready) • Others - reserved
0x010	Width • bit 15~0 - HwReg_width[15:0] (Read/Write) • Others - reserved
0x014	Reserved
0x018	Height • bit 15~0 - HwReg_height[15:0] (Read/Write) • Others - reserved
0x01c	Reserved
0x020	Video Format • bit 15~0 - HwReg_video_format[15:0] (Read/Write) • Others - reserved
0x024	Reserved
0x028	Column Start • bit 15~0 - HwReg_col_start[15:0] (Read/Write)
0x02c	Reserved
0x030	Column End • bit 15~0 - HwReg_col_end[15:0] (Read/Write) • Others - reserved
0x034	Reserved
0x038	Row Start • bit 15~0 - HwReg_row_start[15:0] (Read/Write) • Others - reserved
0x03c	Reserved
0x040	Row End • bit 15~0 - HwReg_row_end[15:0] (Read/Write) • Others - reserved
0x044	Reserved
0x048	Y or R • bit 15~0 - HwReg_Y_R_value[15:0] (Read/Write) • Others - reserved
0x04c	Reserved
0x050	Cb or G • bit 15~0 - HwReg_Cb_G_value[15:0] (Read/Write) • Others - reserved
0x054	Reserved

Table 17: **Letterbox Registers** *(cont'd)*

Register	Description
0x058	Cr or B • bit 15~0 - HwReg_Cr_B_value[15:0] (Read/Write) • Others - reserved
0x05c	Reserved

Notes:

1. SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake.

***Note*:**

1. The Control Register controls the operation of the core. Bit[0] of the Control register, ap_start, kicks off the core from software. Writing 1 to this bit, starts the core to generate a video frame. Bit[1] of the Control register, ap_done, indicates when the IP has completed all operations in the current transaction. A logic 1 on this signal indicates that the IP has completed all operations in this transaction. Bit[2] of the Control register, ap_idle, signal indicates if the IP is operating or idle (no operation). The idle state is indicated by logic 1. This signal is asserted low once the IP starts operating. This signal is asserted high when the IP completes operation and no further operations are performed. Bit[3] of the Control register, ap_ready, signal indicates when the IP is ready for new inputs. It is set to logic 1 when the IP is ready to accept new inputs, indicating that all input reads for this transaction are completed. If the IP has no operations in the pipeline, new reads are not performed until the next transaction starts. This signal is used to make a decision on when to apply new values to the input ports and whether to start a new transaction. Bit[7] of the Control register, auto_restart, can be set to enable the auto-restart mode. Thereafter, the IP restarts automatically at the end of each transaction.
2. Control Register (0x0000), Global Interrupt Enable Register (0x0004), IP Interrupt Enable Register (0x0008), and IP Interrupt Status Register (0x000C) are explained in section S_AXILITE Control Register Map of *Vitis High-Level Synthesis User Guide* (UG1399). These registers definitions might have some additional bits; however, in the current IP, we are accessing only the bits mentioned in Table 11. Therefore, only these bits need to be considered while accessing the Control Register, Global Interrupt Enable Register, IP Interrupt Enable Register, and IP Interrupt Status Register.

Designing with the Core

This section includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

The Video Processing Subsystem is a collection of individual subcore IPs packaged as a hierarchical IP and configured through one single graphical user interface (GUI). The Subsystem can perform the following functions: deinterlacing, scaling, frame rate conversion, color space conversion and correction, and chroma resampling.

Appendix C: Application Software Development describes how to integrate the associated Video Processing Subsystem API into a software application.

Deinterlacing

The Video Deinterlacer converts live incoming interlaced video streams into progressive video streams. Interlaced images might have temporal motion between the two fields that comprise an interlaced frame. The conversion to a progressive format recombines these two fields into one single progressive scan frame. The combining of interlaced video streams results in unsightly motion artifacts in the progressive output image. For this reason, the Video Deinterlacer can be configured to use three field buffers and produce progressive frames based on a combination of spatial and temporal processing.

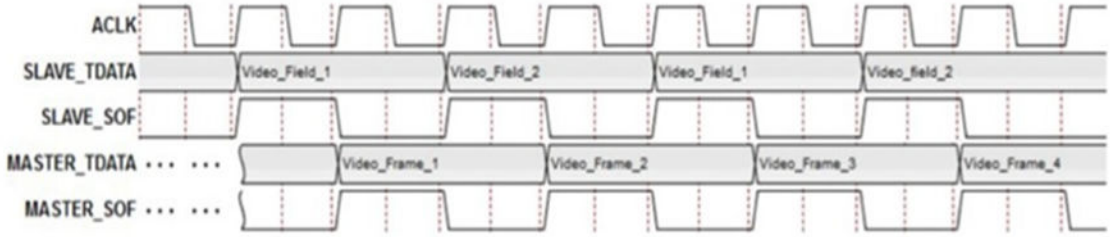
***Note*:** Deinterlacer algorithms do not change the frame rate. For Example, if the input is 1080i60 then the output will be 1080p60.

Start of Frame Signals - m_axis_video_tuser, s_axis_video_tuser

The Start-Of-Frame (SOF) signal, physically transmitted over the AXI4-Stream TUSER signal, marks the first pixel of first incoming video field at slave/input side and first pixel of every outgoing video frame at the master/output side. Every incoming interlaced Video Frame represented by two video fields, which are odd line and even line video field. The SOF pulse is 1 valid transaction wide, and must coincide with the first pixel of the frame. Refer to the following figure.

SOF serves as a frame synchronization signal, which allows downstream cores to reinitialize, and detect the first pixel of a frame/first field. The SOF signal can be asserted an arbitrary number of ACLK cycles before the first pixel value is presented on TDATA, as long as a TVALID is not asserted.

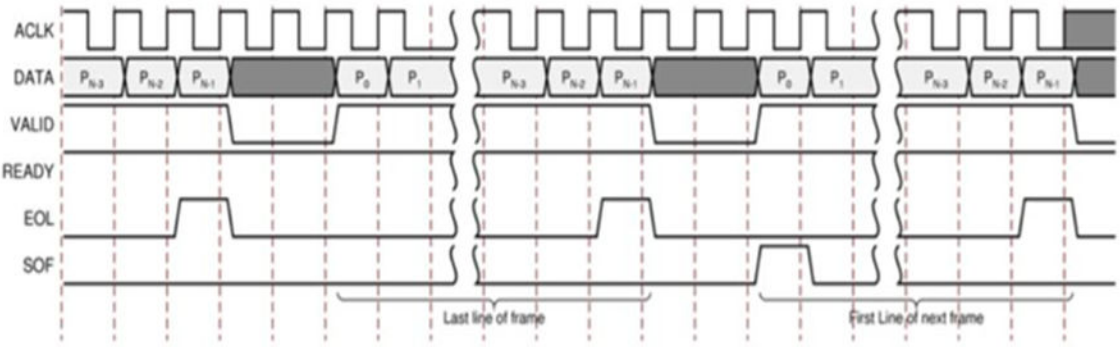
Figure 2: **Example of SOF Handshake, Start of a New Frame**



End of Line Signals - m_axis_video_tlast, s_axis_video_tlast

The End-Of-Line signal, physically transmitted over the AXI4-Stream TLAST signal, marks the last pixel of a line. The EOL pulse is 1 valid transaction wide, and must coincide with the last pixel of a scan-line as shown in the following figure.

Figure 3: **Use of EOL**



The following deinterlacing algorithms are supported:

- **Bob (Line Doubling):** The Bob algorithm takes the lines of each interlaced field (consisting of only even or odd lines of a progressive frame) and duplicates them, filling the entire frame. Line doubling prevents "combing" artifacts but causes a noticeable reduction in video quality. This is noticeable mostly on stationary or slowly moving objects because they seem to bob up and down. This algorithm does not require external frame buffers.

- **Weave:** Weaving is done by assembling 2 consecutive fields together. This performs well when the image hasn't changed between fields, but any move will cause "combing" artifacts - when the pixels in one frame do not align with the pixels in the next frame, forming a jagged edge. Weave algorithm is a motion adaptive method and requires external frame buffers.
- **Vertical Temporal Linear Interpolation (VTLin):** VTLin generates intermediate lines by linear interpolation of its 2 neighboring lines from the current field and 3 reference lines from the previous field. The generated lines are added to lines of the current field, forming the entire frame. VTLin performs better than Bob or Weaving for moving objects, but interpolation results in slight image blurring. VTLin is a motion adaptive algorithm and requires an external frame buffer.
- **Vertical Temporal Median (VTMed):** VTMed makes the generated lines by taking the median of its 2 neighboring lines from the current field and the reference line from the previous field. The generated lines are added to lines of the current field, forming the entire frame. VTMed is a motion adaptive algorithm.
- **Median:** The Median algorithm takes the median of the result from VTLin and VTMed in 2 consecutive fields and the reference line in the next field as the generated lines. The output frame is consisted of generated lines and reference lines of current field. Median is a motion adaptive method and requires an external frame buffer.
- **Bilinear Interpolation:** This algorithm is done by performing bilinear interpolation on the current interlaced field. It fills generated lines by taking the average of 2 neighboring reference lines. The Bilinear Interpolation algorithm is not motion adaptive. It works well for stationary frames but unsatisfactorily for fast moving objects.

Enabling and Disabling Bypass Mode

Deinterlacer IP supports six modes/algorithms including the bypass mode. Every mode/algorithm is designated a number in the algorithm register, that is 0x0038 offset. So, the value 0x6 represents the Bypass mode for the Deinterlacer IP. Bypass mode is used to pass-through high bandwidth progressive resolutions. For all progressive streams, set the interceder to Bypass mode.

To enable the bypass mode, 0x6 has to be written to the algorithm register (0x0038). This will pass the input of the IP to the output without any change in the data and enable the IP to the function in bypass mode.

To disable the bypass mode, any value other than 0x6 has to be written into the algorithm register (0x0038).

Features

The Video Deinterlacer is a low-cost basic deinterlacer. The following list is a summary of the supported features:

- Support for six deinterlacing algorithms. The Bob and Bilinear Interpolation algorithms do not require field buffers. Weave, Vertical Temporal Linear Interpolation, Vertical Temporal Median, and Median algorithms all require field buffers.
- Support for RGB, YUV 4:4:4, YUV 4:2:2, and YUV 4:2:0
- Supports bypass mode

  ***Note*:** In the bypass mode the incoming video streams are not converted in to progressive video streams, that is, the conversion is disabled.
- 8, 10, 12, or 16 bits per component.

Video Format

The video format values are listed in the following table:

Table 18: **Video Format Values**

Video Format	Values
RGB	0
4:4:4	1
4:2:2	2
4:2:0	3

Frame Size

The number of columns must be a multiple of the samples per clock. If the video format is 4:2:2 or 4:2:0, then the number of columns must be even. If the video format is 4:2:0, then the number of rows must be even.

Scaling

Video scaling is the process of converting an input color image of dimensions `Xin` pixels by `Yin` lines to an output color image of dimensions X`out` pixels by Y`out` lines. The IP converts a specified rectangular area of an input digital video image from the original sampling grid to a desired target sampling grid.

The Scaler only mode runs in stream mode, where memory is not required. Any scaling ratio should work in the streaming mode as long as you ensure that the stream has properly dimensioned the performance of the scaler, by choosing pixels per clock and frequency. This is related to the maximum of the input and output resolution.

For example, in case of downscaling choose the maximum input resolution, and in case of upscaling choose the maximum output resolution. Resolution times frame rate determines the pixel throughput. You need to make sure that the scaler is configured to support this pixel throughput plus a margin of at least 10%. That is, when you want to upscale 1080p60 to 4K60, the required throughput is = 4320 * 2160 * 60 (maximum of i/p and o/p resolution is 4k60) = 530 MHz + 10% margin =~ 600 MHz. So, Scaler needs to be configured for 300 MHz input streaming clock and pixels per clock value as 2.

The input image must be provided in raster scan format (left to right and top to bottom). The valid outputs are also given in this order.

Video scaling is a form of 2-D filter operation which can be approximated with the equation shown in the following equation.

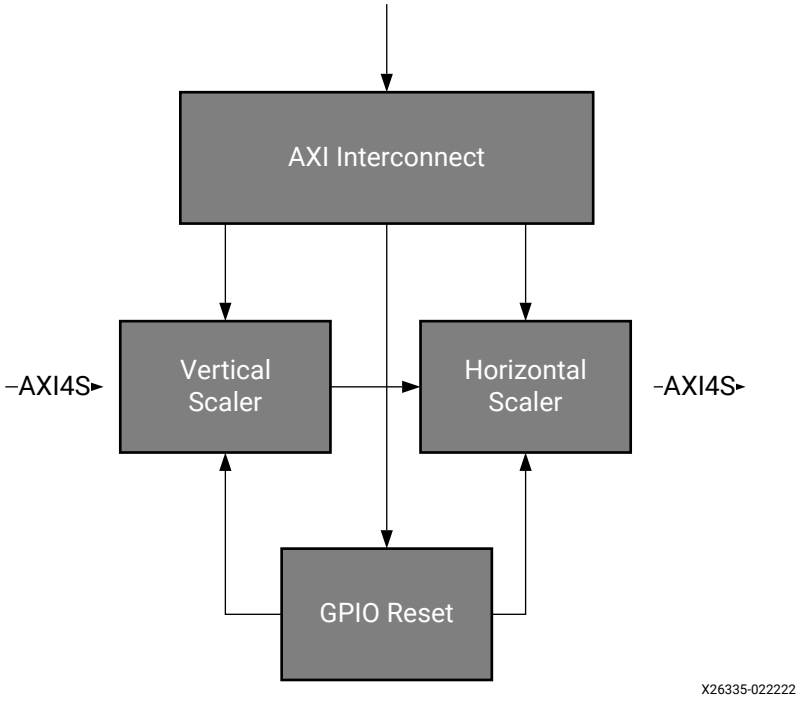
Equation 1: **A**

$$Pix_{out}(x,\ y) = \sum_{HTaps\ -\ 1}^{i\ =\ 0} \sum_{VTaps\ -\ 1}^{j\ =\ 0} Pix_{in}\left[x\ -\ (HTaps\ /2\ )\ +\ i,\ \ y\ -\ (VTaps\ /2\ )\ +\ j\right] \times coef(i,\ j)$$

In this equation, x and y are discrete locations on a common sampling grid; Pix_{out} (x, y) is an output pixel that is being generated at location (x, y); Pix_{in} (x, y) is an input pixel being used as part of the input scaler aperture; Coef (i, j) is an array of coefficients that depend upon the application; and HTaps and VTaps are the number of horizontal and vertical taps in the filter, respectively.

The coefficients in this equation represent weights applied to the set of input samples chosen to contribute to one output pixel, according to the scaling ratio. Scaler only mode requires both external and internal gpio reset to be asserted, when there is a change in input resolution.

Figure 4: **Scaler Only Mode**



Features

The Scaler comes in three different quality levels each at different levels of resource usage:

- Bilinear scaling is the cheapest implementation of the Scaler that uses bilinear interpolation to calculate pixels. Bilinear interpolation produces a greater number of interpolation artifacts (such as aliasing, blurring, and edge halos) than more computationally demanding techniques such as bicubic interpolation.
- Bicubic scaling is a more demanding compared to bilinear scaling, and produces smoother pictures with less artifacts. Compared to bilinear interpolation, which only takes 2 x 2 pixels into account, bicubic interpolation considers a 4 x 4 pixel area.
- The polyphase concept is explained in Polyphase Scaling. The picture quality (and resource usage) of a polyphase scaler depends largely on the number of filter taps used and number of filter phases used. Polyphase scaling offers the highest quality but also has the highest resource utilization.

The following list is a summary of the supported features:

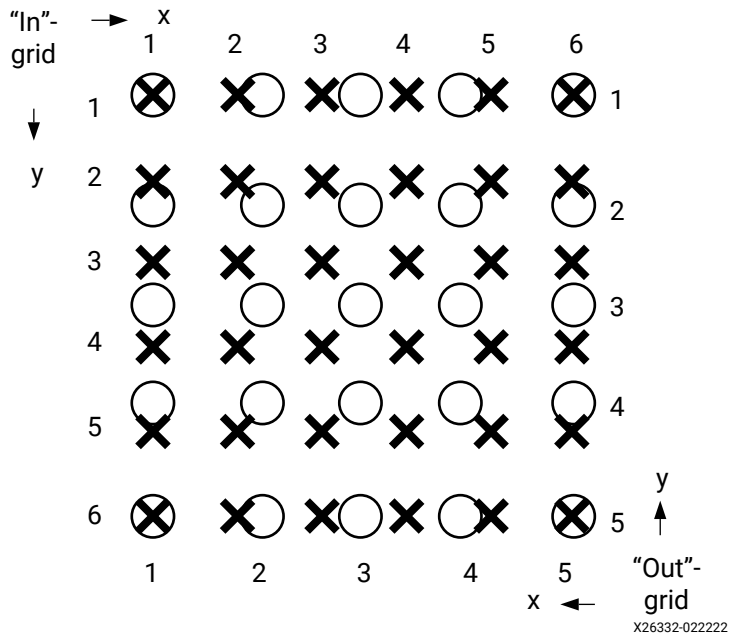
- Polyphase, bicubic, or bilinear scaling modes
- 6, 8, 10, or 12-tap 64 phase polyphase scaling
- Support for and conversion between RGB, YUV 4:4:4, YUV 4:2:2, and YUV 4:2:0
- 8, 10, 12, or 16 bits per video component

Polyphase Scaling

For scaling, the input and output sampling grids are assumed to be different. To express a discrete output pixel in terms of input pixels, it is necessary to know or estimate the location of the output pixel relative to the closest input pixels when superimposing the output sampling grid upon the input sampling grid for the equivalent 2-D space. With this knowledge, the algorithm approximates the output pixel value by using a filter with coefficients weighted accordingly. Filter taps are consecutive data-points drawn from the input image.

As an example, the following figure shows a desired 5x5 output grid ("O") superimposed upon an original 6x6 input grid ("X"), occupying common space. In this case, estimating for output position (x, y) = (1, 1), shows the input and output pixels to be co-located. You can weigh the coefficients to reflect no bias in either direction, and can even select a unity coefficient set. Output location (2, 2) is offset from the input grid in both vertical and horizontal dimensions. Coefficients can be chosen to reflect this, most likely showing some bias towards input pixel (2, 2), etc. Filter characteristics can be built into the filter coefficients by appropriately applying anti-aliasing low-pass filters.

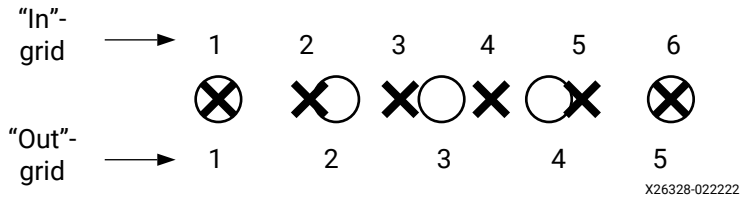
Figure 5: **5x5 Output Grid (“O”) Super-imposed over 6x6 Input Grid (“X”)**



The space between two consecutive input pixels in each dimension is conceptually partitioned into a number of bins or phases. The location of any arbitrary output pixel always falls into one of these bins, thus defining the phase of coefficients used. The filter architecture should be able to accept any of the different phases of coefficients, changing phase on a sample-by-sample basis.

A single dimension is shown in the following figure. As illustrated in this figure, the five output pixels shown from left to right could have the phases 0, 1, 2, 3, 0.

Figure 6: **Super-imposed Grids for 1 Dimension**



The examples in Figure 5 and Figure 6 show a conversion where the ratio Xin/Xout = Yin/Yout = 5/4. This ratio is known as the scaling factor, or SF. The horizontal and vertical Scaling Factors can be different. A typical example is drawn from the broadcast industry, where some footage can be shot using 720p (1280 x 720), but the cable operator needs to deliver it as per the broadcast standard 1080p (1920 x 1080). The SF becomes 2/3 in both H and V dimensions.

Typically, when `Xin > Xout`, this conversion is known as horizontal down-scaling (SF > 1). When `Xin < Xout`, it is known as horizontal up-scaling (SF < 1).

The set of coefficients constitute filter banks in a polyphase filter whose frequency response is determined by the amount of scaling applied to the input samples. The phases of the filter represent subfilters for the set of samples in the final scaled result.

The number of coefficients and their values are dependent upon the required low-pass, anti-alias response of the scaling filter; for example, smaller scaling ratios require lower passbands and more coefficients. Filter design programs based on the Lanczos algorithm are suitable for coefficient generation. Moreover, MATLAB® product fdatool/fvtool can be used to provide a wider filter design toolset.

A general guideline is to use 4 taps per number of scaling ratio for scaling down to get good quality. The following are some recommendations for how many taps to use:

Table 19: **Recommended Taps to Use**

Scaling Ratio	Number of Taps
Upscale	6 taps
Down scale to 1.5	6 taps
Down scale > 1.5 <= 2.5	8 taps
Down scale > 2.5 <= 3.5	10 taps
Down scale > 3.5	12 taps

A direct implementation of Equation 1: A suggests that a filter with VTaps x HTaps multiply operations per output are required. However, the AMD Video Scaler supports only separable filters, which completes an approximation of the 2-D operation using two 1-D stages in sequence - a vertical filter (V-filter) stage and a horizontal filter (H-filter) stage. The intermediate results of the first stage are fed sequentially to the second stage.

The vertical filter stage filters only in the vertical domain, for each incrementing horizontal raster scan position x, creating an intermediate result described as VPix as shown in the following equation.

Equation 2: **B**

$$VPix_{int}[x,y] = \sum_{VTaps-1}^{i=0} Pix_{in}[x, y-(VTaps/2)+i] \times Vcoef[i]$$

The output result of the vertical component of the scaler filter is input into the horizontal filter with the appropriate rounding applied. The separation means this can be reduced to the shown VTaps and HTaps multiply operations, saving FPGA resources as shown in the following equation.

Equation 3: **C**

$$Pix_{out}(x,\ y) = \sum_{HTaps\ -\ 1}^{i=0} VPix_{int}\left[x\ -\ (HTaps\ /2\ )\ +i,\ y\right] \times Hcoef[i]$$

***Note*:** The differences between the Bilinear, Bicubic, and Polyphase architectures are not only marked by a difference in coefficients but with the implementation of optimized architectures for Bilinear and Bicubic scaling.

Phases Control Structure

This is a proprietary control structure that the IP needs to function properly. Reference C code to initialize this phase control structure is given in the v_hscaler driver, that is, API CalculatePhases (InstancePtr, WidthIn, WidthOut, PixelRate) in xv_hscaler_l2.c.

Color Mode

The color mode details are shown in the following table.

Table 20: **Video Format Values**

Video Format	Values
RGB	0
4:4:4	1
4:2:2	2
4:2:0	3

4:2:0 can only be support if this is enabled in the IP. If the vertical scaler gets 4:2:0 in, it will upsample this to 4:2:2, which will then go into the horizontal scaler.

Frame Size

The number of columns must be a multiple of the samples per clock. If the video format is 4:2:2 or 4:2:0, then the number of columns must be even. If the video format is 4:2:0, then the number of rows must be even.

Line Rate

Line rate is calculated as follows:

```
line_rate = (height_in * STEP_PRECISION) / height_out;
```

where, STEP_PRECISION = 2^16 = 65,536

Filter Coefficients

Filter coefficients are layed out in memory as a two dimensional array of [PHASE][TAP]. Each filter coefficient is 16 bit, and is represented as S4.11, that is signed fixed point with four integer bits and 11 fractional bits.

The driver needs to provide built-in tables for 6, 8, 10, or 12 tap filters.

AXI Video Direct Memory Access Engine

Many video applications require frame buffers to handle frame rate changes or changes to the image dimensions (scaling or cropping). The Video DMA engine, which uses the AMD AXI Video Direct Memory Access IP, is designed to allow for efficient high-bandwidth access between AXI4-Stream video interface and AXI4-MM interface. See the *AXI Video Direct Memory Access LogiCORE IP Product Guide* (PG020).

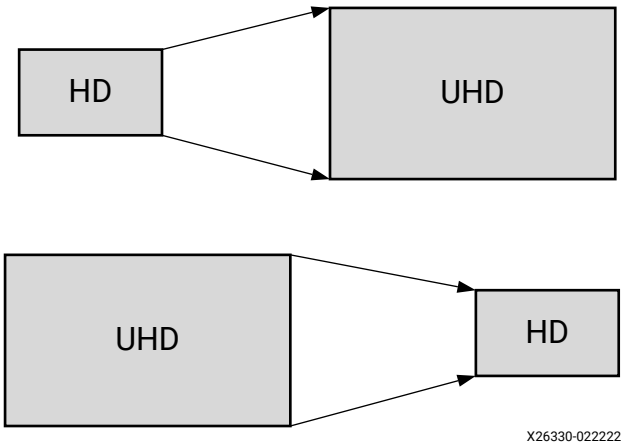
Features

The Video DMA engine within the Full Fledged configuration of the Video Processing Subsystem IP is used to enable frame rate conversion and also more advanced scaling use cases like crop and zoom and scale to picture in picture.

- **Frame Rate Conversion:** Frame rate conversion is implemented by dropping or repeating frames. The DMA engine keeps track of four frame buffers that are being written to in a cyclic fashion. The read portion of the video data flow remains exactly 1 frame behind the write pointer. In case the incoming frame rate is higher than the outgoing frame rate, the write pointer advances more frequently than the read pointer, meaning that frames are skipped. Similarly, the read pointer is always one frame behind the write pointer even in case the outgoing frame rate is higher than the incoming frame rate. In this case, frames are repeated.

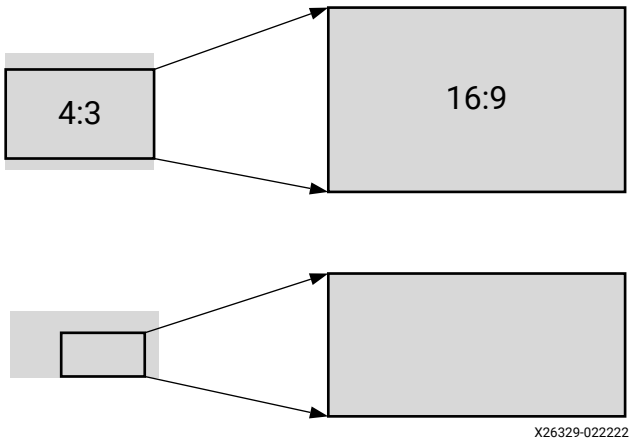
- **Crop and Zoom:** Another use of the video DMA engine is to enable advanced scaling use cases like crop and zoom, and scale to picture in picture. Without the use of memory, only basic scaling can be performed from one resolution to another resolution, as shown in the following figure.

Figure 7: **Basic Scaling**



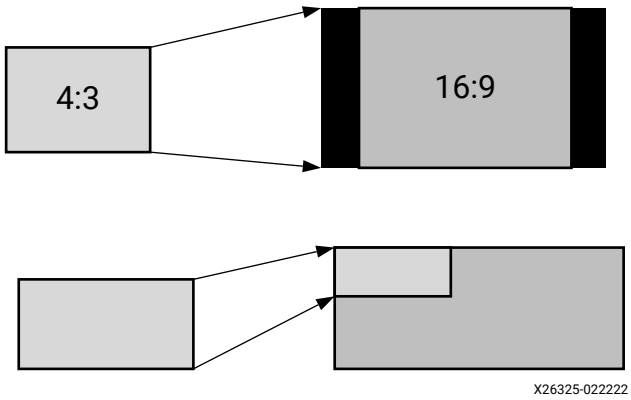
With the video DMA engine in the video path, it is possible to enable a crop and zoom feature as shown in the following figure by cropping out of memory.

Figure 8: **Crop and Zoom**



- **Picture in Picture:** Alternatively, it is also possible to enable a picture in picture feature as shown in the following figure.

Figure 9: **Picture in Picture**



***Note*:** There is no data re-alignment engine implemented, therefore, memory access is aligned to the granularity of the bus. This is automatically taken care of by the driver.

Memory Requirement

DDR memory is used to store video frame buffers for full configuration mode. The subsystem uses five frame buffers for progressive input and three field buffers for interlaced input. You can calculate the amount of memory required by the subsystem using the following equations:

5 * MAX_WIDTHp * MAX_HEIGHTp * NUM_VIDEO_COMPONENTS * BytesPerComp

and

3 * MAX_WIDTHi * MAX_HEIGHTi * NUM_VIDEO_COMPONENTS * BytesPerComp

- **BytesPerComp:**
  - 1 Byte for 8 bit data
  - 2 Byte for 10/12/16 bit data

Memory Bandwidth

The Full Fledged configuration has the following memory bandwidth requirements:

The deinterlacer writes 1 field and reads 2 fields. For example, a resolution of 1080i of 8 bit RGB data at 60 Hz:

write 1 field = 1920 columns * 540 rows * 24 bits * 60 fps = 178 MBytes/second

read 2 fields = 1920 columns * (2*540) rows * 24 bits * 60 fps = 356 MBytes/second

The VDMA writes 1 frame and reads 1 frame. For example, a 4K resolution of 8 bit RGB data at 60 Hz:

write 1 frame = 3840 columns * 2160 rows * 24 bits * 60 fps = 1424 MBytes/second

read 1 frame = 3840 columns * 2160 rows * 24 bits * 60 fps = 1424 MBytes/second

Color Space Conversion and Correction

There are many variations that cause difficulties in accurately reproducing color in imaging systems. These can include:

- Spectral characteristics of the optics (lens, filters)
- Lighting source variations like daylight, fluorescent, or tungsten
- Characteristics of the color filters of the sensor

The Color Space Converter/Correction function provides a method for correcting the image data for these variations. This fundamental block operates on either YUV or RGB data.

As an example, following one of the three color channels through an imaging system from the original light source to the processed image helps understand the functionality of this core.

The blue color channel is a combination of the blue photons from the scene, multiplied by the relative response of the blue filter, multiplied by the relative response of the silicon to blue photons. However, the filter and silicon responses might be quite different from the response of the human eye, so blue to the sensor is quite different from blue to a human being.

This difference can be corrected and made to more closely match the blue that is acceptable to human vision. The Color Space Converter/Correction function multiplies the pixel values by some coefficient to strengthen or weaken it, creating an effective gain. At the same time a mixture of green or red can be added to the blue channel. To express this processing mathematically, the new blue (Bc) is related to the old blue (B), red (R), and green (G) according to:

Equation 4: **D**

$$Bc = K1 \times R \times K2 \times G + K3 \times B$$

where K1, K2, and K3 are the weights for each of the mix of red, green, and blue to the new blue.

Extending this concept, a standard 3 x 3 matrix multiplication can be applied to each of the color channels in parallel simultaneously. This is a matrix operation where the weights define a color-correction matrix. In typical applications, color-correction also contains offset compensation to ensure black [0,0,0] levels are achieved. The following matrix operations can also be used for color correction:

Equation 5: **E**

$$\begin{bmatrix} R_C \\ G_C \\ B_C \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

Equation 6: **F**

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} + \begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

***Note*:** The K coefficients are presented in S3.12 fixed point format (1 sign bit, 3 integer bits, 12 fractional bits). The 16-bit signed integer values (2's compliment) are equivalent to real numbers in the [-8 .. 8] range. The offset values O have a width of the data width plus 1. It is a signed integer with a range of $[-2^{Data\_Width}, 2^{Data\_Width-1}]$.

- **Matrix Operation:** The 3x3 Matrix (and offsets) can be used to perform color space conversion between RGB and YUV 4:4:4:

Equation 7: **G**

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} U \\ Y \\ V \end{bmatrix} + \begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

Equation 8: **H**

$$\begin{bmatrix} U \\ Y \\ V \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix}$$

  As shown in the matrix operation, the input pixels are transformed to a set of corrected output pixels. This can be a very useful function configured as a static application; however, the programmability of the coefficients and offset values allows this function to adapt to changing lighting conditions based on a separate control loop.

- **Clipping and Clamping:** The Clip and Clamp values have the same width as the data width. They are unsigned integers with a range of $[0, 2^{Data\_Width-1}]$.

Features

The Color Space Conversion and Correction function offers a 3 x 3 matrix multiplication for a variety of color correction applications. The coefficient matrix is fully programmable and includes offset compensation, and clipping and clamping of the output is also definable.

The following list is a summary of the supported features:

- User programmable matrix coefficients
- Support for and conversion between RGB, YUV 4:4:4, YUV 4:2:2, and YUV 4:2:0
- 8, 10, 12, or 16 bits per component
- Driver API to set coefficients for converting RGB to YUV, or vice-versa
- Driver API to set/get brightness, contrast, saturation, and gain
- **Color Space Conversion:** The primary purpose of this function is to provide color space conversion between the RGB and YUV domains. The fully programmable 3 x 3 matrix with offsets and clipping and clamping allow the support of multiple video standards.
- **Color Correction:** This function provides support for additional color correction within a user-defined window in the video frame. You can define a second coefficient matrix to be applied only within a demo window. You also program the size and position of the demo window.
- **Filter Coefficients and Offsets:** The coefficients are presented in 16.12 fixed point format. The 16-bit signed integer values (2's compliment) are equivalent to real numbers in the [-8 .. 8] range. The offset value has a width of the data width plus 1. It is a signed integer with a range as shown in the following equation

Equation 9: **I**

$$-2^{Data\_Width},\ 2^{Data\_Width} - 1$$

Matrix computation outputs are rounded to DATA_WIDTH bits by adding half an output LSB prior to truncation. Output values greater than the Clip value are replaced with the Clip value. Output values smaller than the Clamp value are replaced with the Clamp value. The Clip and Clamp values have the same width as the data width. They are unsigned integers with a range of [0 .. $2^{Data\_Width-1}$].

Color Mode

The color mode details are shown in the following table:

Table 21: **Video Format Values**

Video Format	Values
RGB	0
4:4:4	1
4:2:2	2
4:2:0	3

IP supports converting from any format (RGB, 4:4:4, 4:2:2, 4:2:0) to any format (RGB, 4:4:4, 4:2:2, 4:2:0). This assumes that the appropriate formats are enabled in the hardware.

Frame Size

The number of columns must be a multiple of the samples per clock. If the video format is 4:2:2 or 4:2:0, then the number of columns must be even. If the video format is 4:2:0, then the number of rows must be even.

Window

The window function is not supported in the Linux driver. If the Window feature is not enabled in the hardware, then the following registers will not exist:

- HwReg.ColStart
- HwReg.ColEnd
- HwReg.RowStart
- HwReg.RowEnd
- HwReg.K11_2
- HwReg.K12_2
- HwReg.K13_2
- HwReg.K21_2
- HwReg.K22_2
- HwReg.K23_2
- HwReg.K31_2
- HwReg.K32_2
- HwReg.K33_2
- HwReg.ROffset_2
- HwReg.GOffset_2
- HwReg.BOffset_2
- HwReg.ClampMin_2
- HwReg.ClipMax_2

Chroma Resampling

The human eye is not as receptive to chrominance (color) detail as luminance (brightness) detail. Using color-space conversion, it is possible to convert RGB into the YUV color space, where Y is Luminance information, and U and V are derived color difference signals. At normal viewing distances, there is no perceptible loss incurred by sampling the color difference signals (U and V) at a lower rate to provide a simple and effective video compression to reduce storage and transmission costs

The Chroma Resampler function converts between chroma sub-sampling formats of 4:4:4, 4:2:2, and 4:2:0. There are a total of six conversions available for the three supported sub-sampling formats. Conversion is achieved using a FIR filter approach. Some conversions require filtering in only the horizontal dimension, only the vertical dimension, or both. Interpolation operations are implemented using a two-phase polyphase FIR filter. Decimation operations are implemented using a low-pass two-phase polyphase FIR filter to suppress chroma aliasing.

Features

The Chroma Resampler function converts between different chroma sub-sampling formats. The supported formats are 4:4:4, 4:2:2, and 4:2:0. There are three different options for interpolating and decimating the video samples:

- Define a configurable filter with programmable coefficients for high-performance applications.
- Use the predefined static filter with power-of -two coefficients for low-footprint applications.
- Replicate or drop pixels for minimal footprint.

The following list is a summary of the supported features:

- User programmable filter coefficients
- 4, 6, 8, or 10 tap filter
- Support for RGB, YUV 4:4:4, and YUV 4:2:2, and YUV 4:2:0
- 8, 10, 12, or 16 bits per component
- Runtime configurable pass through mode if no chroma resampling is needed

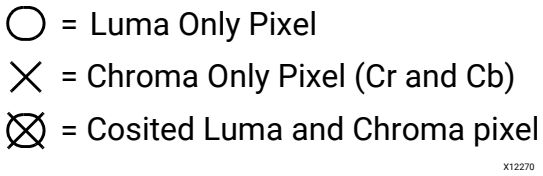
Sub-sampled Video Formats

The sub-sampling scheme is commonly expressed as a three part ratio J:a:b (for example, 4:2:2), that describes the number of luminance and chrominance samples in a conceptual region that is J pixels wide, and 2 pixels high. The parts are (in their respective order):

- J: Horizontal sampling reference (width of the conceptual region). This is usually 4.
- a: Number of chrominance samples (V, U) in the first row of J pixels.
- b: Number of (additional) chrominance samples (V, U) in the second row of J pixels.

To illustrate the most common sub-sampling schemes, The following figure introduces a graphical notation of sampling grid pixels.

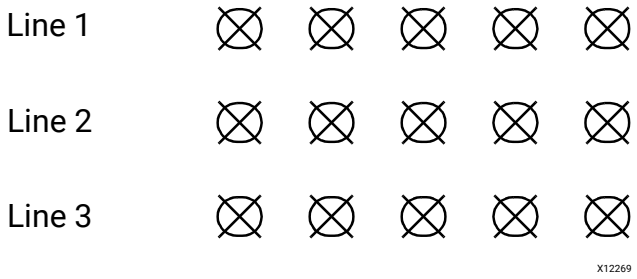
Figure 10: **Luma, Chroma Notation**



4:4:4

Similar to RGB, the 4:4:4 format is used for image capture and display purposes. U and V channels are sampled at the same rate as luminance. Hence, all pixel locations have luma and chroma data co-sited, as shown in the following figure.

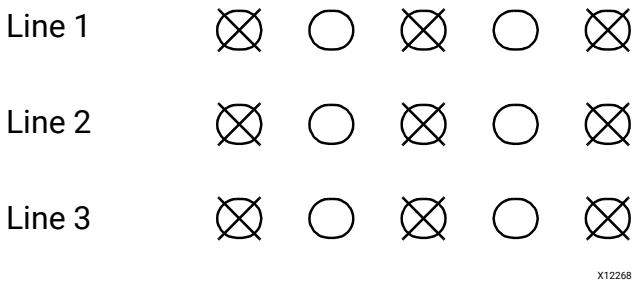
Figure 11: **YUV 4:4:4 Format**



4:2:2

This format contains horizontally sub-sampled chroma. For every two luma samples, there is an associated pair of U and V samples. The sub-sampled chroma locations are co-sited with alternate luma samples as shown in the following figure.

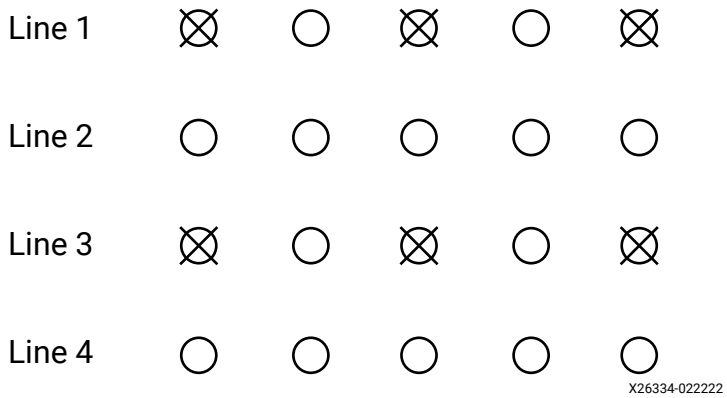
Figure 12: **YUV 4:2:2 Format**



4:2:0

4:2:0 encoding contains horizontally and vertically sub-sampled chroma. Horizontal and vertical chroma positions are co-sited with alternate luma samples on alternate scanlines. The sampling positions are shown in the following figure.

Figure 13: **YUV 4:2:0 Format**



- **Implementation:** Between the three supported sub-sampling formats (4:4:4, 4:2:2, 4:2:0), there are six conversions available. Conversion is achieved using a FIR filter approach. Some require filtering in only the horizontal dimension or in only the vertical dimension, and in some cases in both the horizontal and the vertical dimensions. These are detailed in the following table along with default filter information.

  Table 22: **Chroma Resampling Configuration**

Converter	Filter Configuration
4:4:4 to 4:2:2	Horizontal anti-aliasing
4:4:4 to 4:2:0	Separable 2-D anti-aliasing
4:2:2 to 4:4:4	Horizontal Interpolation
4:2:2 to 4:2:0	Vertical anti-aliasing
4:2:0 to 4:4:4	Separable 2-D Interpolation
4:2:0 to 4:2:2	Vertical Interpolation

  Three implementation options are offered for each conversion operation:

  - DSP48 based filter with programmable coefficients and programmable number of taps. 2D filters must be separable. Coefficients are in the range [-8, 8), represented in 16-bit signed, fixed-point format with four integer bits and 12 fractional bits.
  - The predefined fixed coefficient, non-programmable filter with power of two coefficients (using only shifts and additions for filtering therefore no DSP48s are used). Default coefficients implement linear interpolation for the interpolation and anti-aliasing low pass filters.

- The simplest, lowest footprint solution is to simply drop (decimation) or replicate (interpolation) samples. For down sampling, some samples are passed directly to the output, but others are dropped entirely as appropriate. For up converters, replication of the previous input sample occurs.

- **Convert 4:2:2 to 4:4:4:** This conversion is a 1:2 horizontal interpolation operation, implemented using a two-phase polyphase FIR filter. One of the two output pixels is co-sited with one of the input sample. The ideal output is achieved simply by replicating this input sample.

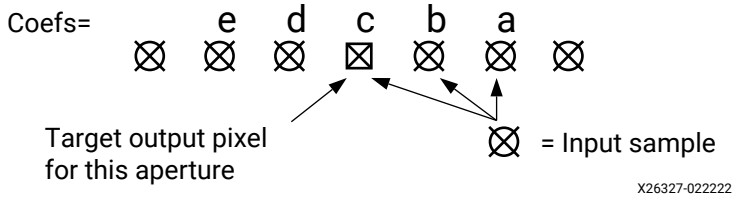
  To evaluate output pixel $o_{x,\,y,}$ the FIR filter convolves COEFk_HPHASEp_x, where k is the coefficient index, $i_{x,y}$ are pixels from the input image, p_x is the interpolation phase (0 or 1, depending on x) and $[\,]^M{}_m$ represents rounding with clipping at M, and clamping at m. DW is the Data Width or number of bits per video component. N_{taps} is the number of filter taps.

  Equation 10: **J**

$$o_{x,\,y} = \left[\sum_{k=0}^{N_{taps}-1} i_{x-k,\;y}\, COEFk\_HPHASEp_X\right]_0^{2^{DW}-1}$$

  In phase 1, COEF00_HPHASE1 is the coefficient applied to the most recent input sample in the filter aperture. The following figure illustrates coefficient use for a four tap filter example, with simplified nomenclature a= COEF00_HPHASE1, b= COEF01_HPHASE1, c=COEF02_HPHASE1, and d= COEF03_HPHASE1.

  Figure 14: **4:2:2 to 4:4:4 Coefficient Configuration**



  The predefined filters replicate the input sample for Phase 0. The Phase 1 filter is [0.5 0.5].

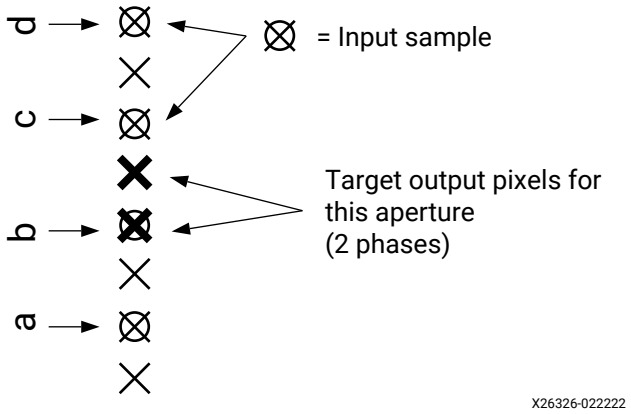
- **Convert 4:4:4 to 4:2:2:** This conversion is a horizontal 2:1 decimation operation, implemented using a low-pass FIR filter to suppress chroma aliasing. In order to evaluate output pixel $o_{x,\,y,}$ the FIR filter in the core convolves COEFk_HPHASE0, where k is the coefficient index, $i_{x,y}$ are pixels from the input image, and $[\,]^M{}_m$ represents rounding with clipping at M, and clamping at m. DW is the Data Width or number of bits per video component. N_{taps} is the number of filter taps.

  Equation 11: **K**

$$o_{x,\,y} = \left[\sum_{k=0}^{N_{taps}-1} i_{x-k,\;y}\, COEFk\_HPHASE0\right]_0^{2^{DW}-1}$$

In phase 0, COEF00_HPHASE0 is the coefficient applied to the most recent input sample in the filter. Figure YUV 4:2:2 Format illustrates coefficient use for a 5 tap filter example, with simplified nomenclature a= COEF00_HPHASE0, b= COEF01_HPHASE0, c= COEF02_HPHASE0, d=COEF03_HPHASE0, and e= COEF04_HPHASE0.

Figure 15: **4:4:4 to 4:2:2 Coefficient Configuration**



The predefined filter coefficients are [0.25 0.5 0.25].

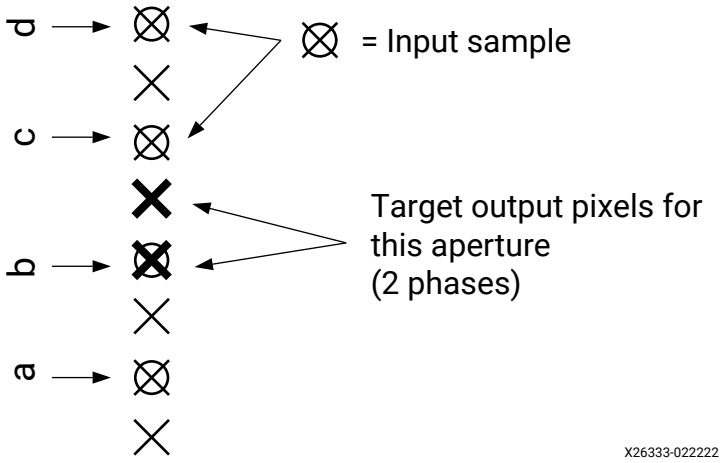
- **Convert 4:2:0 to 4:2:2:** This conversion is a 1:2 vertical interpolation operation, implemented using a 2-phase polyphase FIR filter. One of the two output pixels is co-sited with one of the input sample. The ideal output is achieved simply by replicating this input sample.To evaluate output pixel $o_{x,y}$, the FIR filter in the core convolves COEFk_VPHASEp_y, where k is the coefficient index, p_y is the interpolation phase, $i_{x,y}$ are pixels from the input image, and [ ]M m represents rounding with clipping at M, and clamping at m. DW is the Data Width or number of bits per video component. Ntaps is the number of filter taps.

Equation 12: **L**

$$o_{x,\,y} = \left[\sum_{k=0}^{N_{taps}-1} i_{x,\,y-k}\; COEFk\_VPHASE0 \right]_{0}^{2^{DW}-1}$$

In phase 1, COEF00_VPHASE1 is the coefficient applied to the most recent input sample in the filter. The following figure illustrates coefficient use for a four tap filter example, with simplified nomenclature a= COEF00_VPHASE1, b= COEF01_VPHASE1, c= COEF02_VPHASE1, and d= COEF03_VPHASE1.

Figure 16: **4:2:0 to 4:2:2 Coefficient Configuration**



The predefined filters use the coefficients [0.5 0.5] to interpolate one of the output samples. The other output sample is a replication of the input sample.

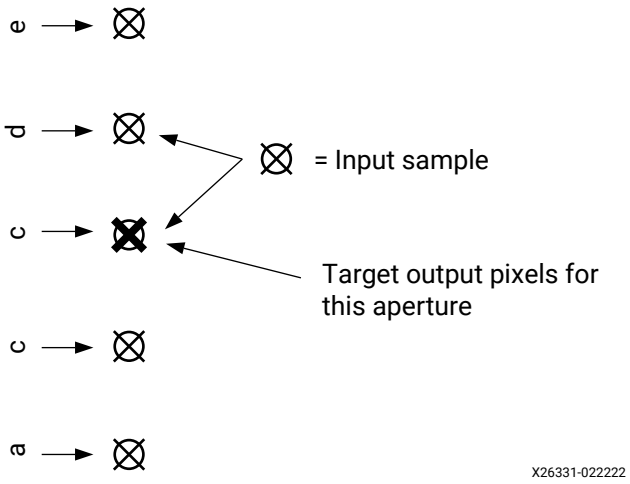
- **Convert 4:2:2 to 4:2:0:** This conversion is a vertical 2:1 decimation operation, implemented using a low-pass FIR filter to suppress chroma aliasing. In order to evaluate output pixel $o_{x,y}$, the FIR filter in the core convolves COEFk_VPHASE0, where k is the coefficient index, $i_{x,y}$ are pixels from the input image, and [ ]M m represents rounding with clipping at M, and clamping at m. DW is the Data Width or number of bits per video component. N_{taps} is the number of filter taps.

Equation 13: **M**

$$o_{x,\,y} = \left[\sum_{k=0}^{N_{taps}-1} i_{x-k,\;y}\; COEFk\_VPHASE0 \right]_{0}^{2^{DW}-1}$$

In phase 0, COEF00_VPHASE0 is the coefficient applied to the most recent input sample in the filter. The following figure illustrates coefficient use for a five tap filter example, with simplified nomenclature a= COEF00_VPHASE0, b= COEF01_VPHASE0, c= COEF02_VPHASE0, d= COEF03_VPHASE0, and e= COEF04_VPHASE0.

Figure 17: **4:2:2 to 4:2:0 Coefficient Configuration**



The predefined filter coefficients are [0.25 0.5 0.25].

- **Convert 4:2:0 to 4:4:4:** This conversion performs interpolation both vertically and horizontally. This is equivalent to a 2D separable filter implemented by cascading the 4:2:0 to 4:2:2 block and the 4:2:2 to 4:4:4 block. Quantized vertical filter results are filtered by the horizontal filter, which in turn quantizes results back to the [0 .. 2DW-1] range. (DW is the Data Width or number of bits per video component.)

  Intermediate 4:2:2 chroma values are computed using Equation 13: M. The resulting computation is shown in the following equation.

Equation 14: **N**

$$t_{x,\,y} = \left[\sum_{k=0}^{N_{Vtaps}-1} t_{x,\,y-k}\, COEFk\_VPHASE0 \right]_{0}^{2^{DW}-1}$$

Next, the values are filtered according to Equation 1: A. The resulting computation is shown in the following equation.

Equation 15: **O**

$$o_{x,\,y} = \left[\sum_{k=0}^{N_{taps}-1} i_{x-k,\,y}\, COEFk\_HPHASEp_X \right]_{0}^{2^{DW}-1}$$

The predefined filter coefficients are the same as defined in Convert 4:2:0 to 4:2:2 and Convert 4:2:2 to 4:4:4. In the vertical direction, one input sample is replicated, and the other is interpolated with the filter [0.5 0.5]. The same then happens in the horizontal direction.

- **Convert 4:4:4 to 4:2:0:** This conversion performs decimation by 2 both vertically and horizontally. This is equivalent to a 2D separable filter implemented by cascading the 4:4:4 to 4:2:2 block and the 4:2:2 to 4:2:0 block. Quantized horizontal filter results are filtered by the vertical filter, which in turn quantizes results back to the [0 .. 2DW-1] range. (DW is the Data Width or number of bits per video component.)

  Intermediate 4:2:2 chroma values are computed using the Equation 2: B. The resulting computation is shown in the following equation.

  Equation 16: **P**

  $$t_{x,\,y} = \left[\sum_{k=0}^{N_{Htaps}-1} i_{x-k,\,y}\, COEFk\_HPHASE0 \right]_0^{2^{DW}-1}$$

  Next, these values are filtered according to Equation 4: D. The resulting computation is shown in the following equation.

  Equation 17: **Q**

  $$o_{x,y} = \left[\sum_{k=0}^{N_{taps}-1} t_{x,y-k}\, COEFk\_VPHASE0 \right]_0^{2^{DW}-1}$$

  The predefined filter coefficients are the same as defined in Convert 4:4:4 to 4:2:2 and Convert 4:2:2 to 4:2:0. In the horizontal direction, decimation is performed with the filter [0.25 0.5 0.25]. The same then happens in the vertical direction.

- **Resampling Filters:** The upsampling and downsampling performed during the chroma format conversion is implemented with low pass filters for the interpolation and anti-aliasing.

  The chroma resampling function offers a horizontal filter with a maximum of 10 taps and two phases, as well as a vertical filter with a maximum of 10 taps and two phases. For conversions requiring up/down sampling in both horizontal and vertical directions, 2D separable filters are offered.

  The number of taps selected must be even (4, 6, 8, or 10). Depending on the conversion type and filter size selected, a subset of the coefficients can be used by setting the unnecessary coefficients to zero.

  Each coefficient has 16 bits in 2's complement format: 4 integer bits (one sign bit) and 12 fractional bits. The sign bit is the MSB. For example, a coefficient with a value of 1 is represented with this bit vector

  ```
  [0001000000000000].
  ```

The coefficients should sum to exactly 1 to achieve unity gain. If they sum to less than 1, some loss of dynamic range is observed.

- **Computation Bit Width Growth:** Full precision (DATA_WIDTH+16+log2(N_{Taps}) bits) is maintained during the horizontal and/ or vertical FIR convolution operation.

  FIR filter outputs are rounded to DATA_WIDTH bits by adding half an output LSB in the full precision domain prior to truncation. Clipping and clamping of the output data prevents overflows and underflows. Data is clipped and clamped at 2^{DATA_WIDTH}-1 and 0.

- **Edge Padding:** The edge pixels of images are replicated prior to filtering to avoid image artifacts.

***Note*:** Configure input and output resolutions of the Video Processing Subsystem IP (all modes of the core) to enable the bypass mode.

Clocking

The AXI Streaming, AXI Memory-Mapped, and AXI Lite interfaces can be run at their own clock rate. Therefore, three seperate clock interfaces are provided named `aclk_axis`, `aclk_axi_mm`, and `aclk_ctrl`, respectively.

Pixel throughput of the Video Processing Subsystem is defined by the product of the clock frequency times the Samples per Clock setting in the GUI. With a clock frequency of 300 MHz for `aclk_axis`, and a four sample per clock configuration, the Video Processing Subsystem is capable of a 1200 mega pixel throughput rate, which is sufficient to handle 8k resolutions at 30 Hz.

Resets

The `aresetn_ctrl` signal is the active-Low reset input signal of the IP. The reset signal must be synchronous to the `aclk_ctrl` signal. Each time this reset is asserted, it should be asserted for a minimum of 16 clock cycles of the slowest clock. All registers are reset to power-on conditions; all queues are flushed; all internal logic returned to power-on conditions.

The `aresetn_io_axis` signal is an outgoing signal that can be used to hold IPs in reset when the Video Processing Subsystem is not ready to consume data on the streaming input. This reset signal is synchronous to the `aclk_axis` signal.

For Full-fledged and Scaler Only modes along with `aresetn_ctrl` signal, there is an internal gpio reset signal to reset the sub cores of vpss. To control the internal gpio, you have to get the internal gpio configuration, where you can set/reset the IP.

Design Flow Steps

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

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

Customizing and Generating the Subsystem

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

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

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

1. In the Flow Navigator, click on **Create Block Diagram** or **Open Block Design** under the IP Integrator heading.
2. Right click in the diagram and select Add IP.

   A searchable IP catalog opens. You can also add IP by clicking on the Add IP button on the left side of the IP Integrator Block Design canvas.
3. Click on the IP name and press the **Enter** key on your keyboard or double click on the IP name.

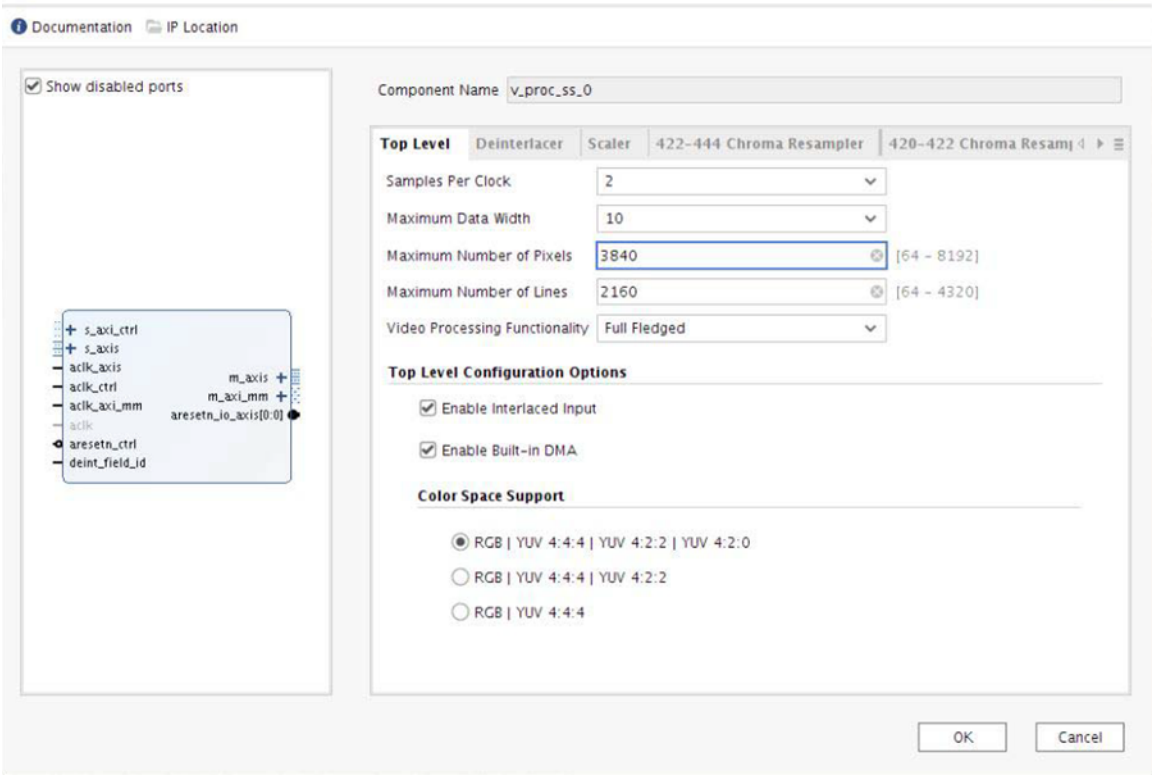
4. Double-click the selected IP block or select the Customize Block command from the right-click menu.

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

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

Top Level Settings Tab

Figure 18: **Top Level Tab**



The parameters on the Top Level tab are as follows:

- **Component Name:** Component name is set automatically by IP integrator.
- **Samples Per Clock:** Select 1, 2, 4, or 8 pixel wide interface for the IP configurations without deinterlacer. If you enable deinterlacer, only 1,2, and 4 samples per clock are supported.
- **Maximum Data Width:** Select 8, 10, 12, or 16-bit color depth.

- **Maximum Number of Pixels:** Specify the maximum number of pixels per scanline. Permitted values are from 64 to 8192. Specifying this value is necessary to establish the depth of internal line buffers. Using a tight upper-bound results in optimal block RAM usage. The active frame size can be programmed during runtime via the AXI4-Lite interface and driver API. The number of active columns must be less than or equal to the Maximum Number of Pixels.
- **Maximum Number of Lines:** Specify the maximum number of scanlines per frame. Permitted values are from 64 to 4320. The active frame size can be programmed during runtime via the AXI4-Lite interface and driver API. The number of active rows must be less than or equal to the Maximum Number of Lines.
- **Video Processing Functionality:** The following sets of processing functionality are offered:
  - **Full Fledged:** The Full Fledged design indicates deinterlacing (optional), scaling, color matrix operations such as color space conversion and correction, chroma resampling (optional), and frame rate conversion by means of drop/repeat (optional). The Full Fledged configuration is the most resource intensive configuration, but allows for stripping down the full configuration by excluding deinterlacing, frame rate conversion, and chroma resampling. If deinterlacing is not included, only progressive inputs can be supported. Similarly, the Color Space Support option (explained below), allows for stripping down the Video Processing Subsystem to handle only RGB and YUV 4:4:4, therefore eliminating the need for horizontal and vertical chroma resampling. In the Full Fledged configuration, you have full configurability over the included functions, for example, the algorithm and/or number of taps used for chroma resampling can be changed.
  - **Scaler Only:** The subsystem is configured to perform the scaling function with an option for video format conversion between RGB, YUV 4:4:4, YUV 4:2:2, and YUV 4:2:0.

    ***Note*:** Together with Enable Color Space Conversion, and Color Space Support for RGB, YUV 4:4:4, YUV 4:2:2, YUV 4:2:0, the Scaler Only configuration is effectively a scaler with any to any format conversion. In terms of resources, this is a cost-effective alternative to the Full Fledged configuration as long as there is no need for better than basic chroma resampling quality and no need for programming the color space conversion coefficients.

    In this mode the coefficients are fixed. The following equations are implemented in the IP source code and are YUV to RGB equations:

    Cr = V - (1<<(HSC_BITS_PER_COMPONENT-1))

    Cb = U - (1<<(HSC_BITS_PER_COMPONENT-1))

    R = Y + ((Cr * 1733) >> 10)

    G = Y - ((Cb * 404 + Cr * 595) >> 10)

    B = Y + ((Cb * 2081) >> 10)

    These are RGB to YUV equations:

    Y = (306*R + 601*G + 117*B)>>10

U = (1<<(HSC_BITS_PER_COMPONENT-1)) + (((B-Y)*504)>>10)

V = (1<<(HSC_BITS_PER_COMPONENT-1)) + (((R-Y)*898)>>10)

***Note*:**

1. HSC_BITS_PER_COMPONENT = C_MAX_DATA_WIDTH (here C_MAX_DATA_WIDTH can be set by user through the GUI).
2. Calculated values are clipped to 0 to Maximum allowed value in that color depth, that is, C_MAX_DATA_WIDTH. For example, if the maximum data width is 10, the value is clipped in 0 to 1023.

- **Deinterlacing Only:** The subsystem is configured to perform only the deinterlacing function.
- **Color Space Conversion Only:** The subsystem is configured to perform color correction functions and color space conversion between RGB and YUV 4:4:4 with options to also support conversion to and from YUV 4:2:2 and YUV 4:2:0, meaning the Color Space Conversion Only configuration can handle any format conversion.
- **420-422 Chroma Resampling Only:** The subsystem is configured to perform only the vertical chroma resampling function.
- **422-444 Chroma Resampling Only:** The subsystem is configured to perform only the horizontal chroma resampling function.

- **Top Level Configuration Options:** When the Full Fledged Video Processing Functionality is selected, the following additional configurations are available:
  - **Enable Interlaced Input:** Select this checkbox if interlaced support is desired. In this case, the subsystem includes the Deinterlacer subcore.
  - **Enable Built-in DMA:** Select this checkbox to include a Video DMA Engine in the subsystem.

    ***Note*:** Samples per clock should only be 1 if interlaced Input or Built-in DMA is selected.
  - **Use UltraRAM for Line Buffers:** In AMD UltraScale+™ devices, line buffers can be stored in UltraRAM instead of Block RAM.
  - **Color Space Support:** Select the color spaces for which support is desired. If YUV 4:2:2 is selected in the Full Fledged configuration, the Horizontal Chroma Resampler subcore is included. If YUV 4:2:0 is selected, the Vertical Chroma Resampler subcore is also included.

Deinterlacer Settings

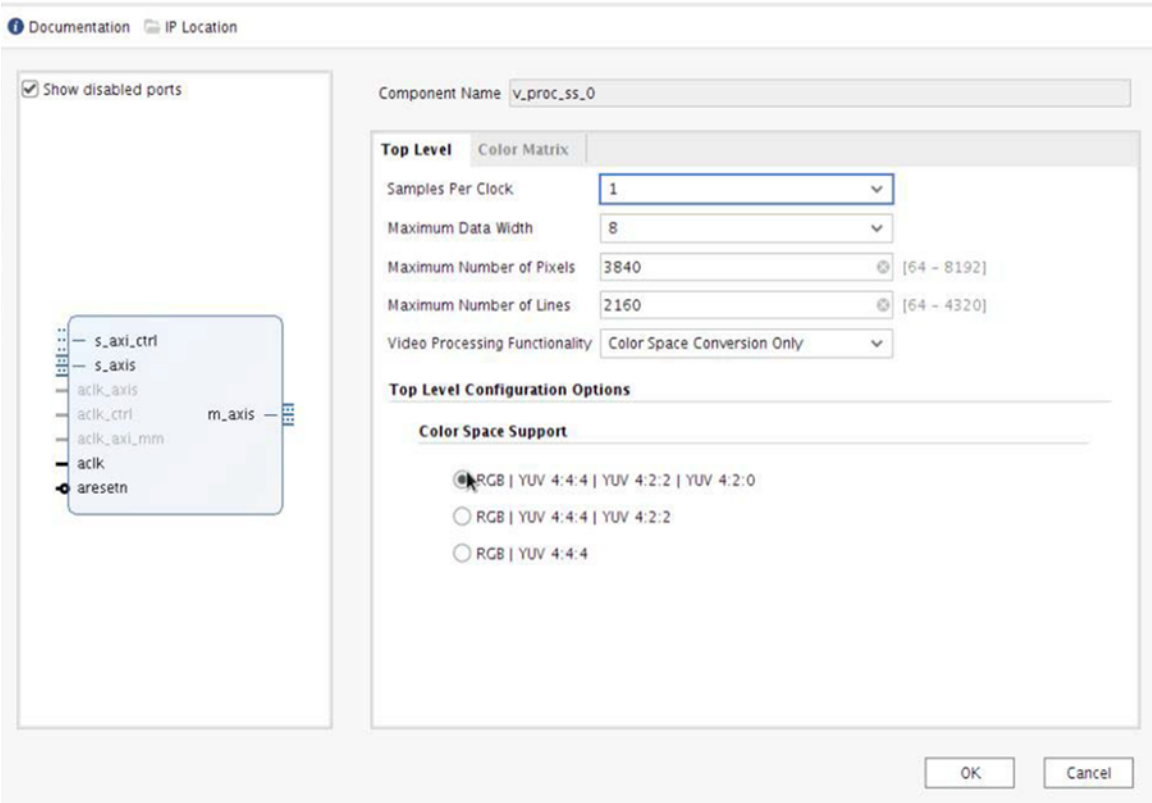
Figure 19: **Deinterlacer Tab**

The parameter on the Deinterlacer tab is as follows:

- **Enable Motion Adaptive Deinterlacing:** Selecting this option allows support for all six deinterlacing algorithms: Line Doubling, Weave, Vertical Temporal Linear Interpolation, Vertical Temporal Median, Median, Bilinear Interpolation. If this checkbox is not selected, only the following two algorithms are supported: Bob and Bilinear Interpolation.

Scaler Settings

Figure 20: **Top Level Scaler Options**



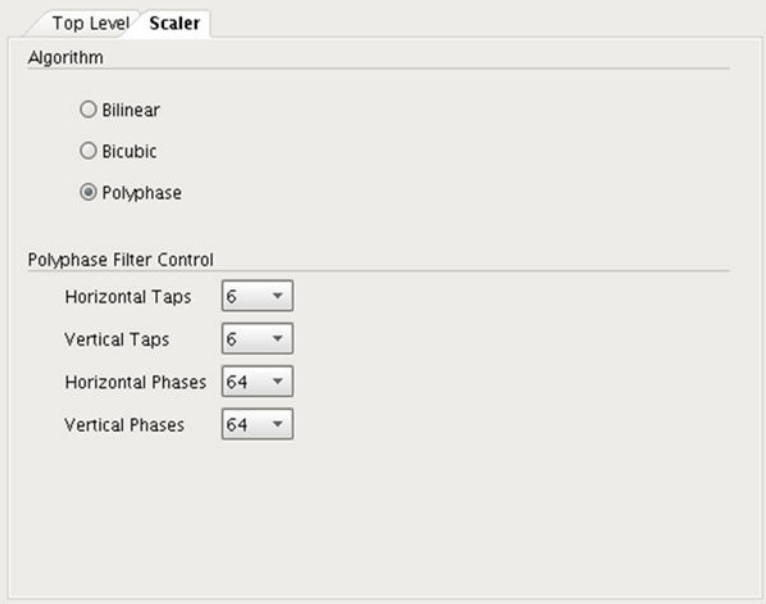
When the Video Processing Functionality is chosen to be Scaler Only, the following options are available:

- **Enable Color Space Conversion:** When selected, the Scaler Only subcore supports run-time configurable color space conversion from RGB to YUV 4:4:4 and vice versa between the input and output video streams.

Note: The coefficients are fixed and not programmable.

- **Color Space Support:** Select the color spaces for which support is desired. If YUV 4:2:2 is selected, basic horizontal chroma resampling is included. If YUV 4:2:0 is selected, basic vertical chroma resampling is also included. The combination of Enable Color Space Conversion and Color Space Support for RGB, YUV 4:4:4, YUV 4:2:2, YUV 4:2:0 basically turns the Scaler Only configuration into a scaler with any to any format conversion.

Figure 21: **Scaler Tab**



The parameters on the Scaler tab are as follows:

- **Algorithm:** The Scaler comes in three different quality levels each at different levels of resource usage.
  - **Bilinear:** Bilinear scaling uses bilinear interpolation to calculate pixels.
  - **Bicubic:** Bicubic scaling is a little bit more demanding compared to bilinear scaling, and produces smoother pictures with less artifacts.
  - **Polyphase:** The picture quality (and resource usage) of a polyphase scaler depends largely on the number of filter taps used and number of filter phases used.
- **Polyphase Filter Control:** When polyphase filtering is selected, the horizontal and vertical taps and phases are defined here.
  - **Horizontal Taps:** Select 6, 8, 10, or 12 horizontal filter taps.
  - **Vertical Taps:** Select 6, 8, 10, or 12 vertical filter taps.
  - **Horizontal Phases:** Only 64 phases is currently supported.
  - **Vertical Phases:** Only 64 phases is currently supported.

Chroma Resampler Settings

The Video Processing Subsystem supports chroma resampling between 4:4:4 and 4:2:2 formats as well as between 4:2:2 and 4:2:0.

Figure 22: **4:2:2 – 4:4:4 Chroma Resampling Tab**

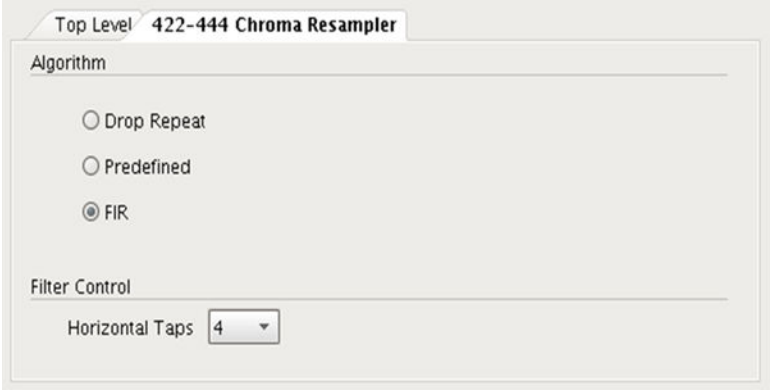
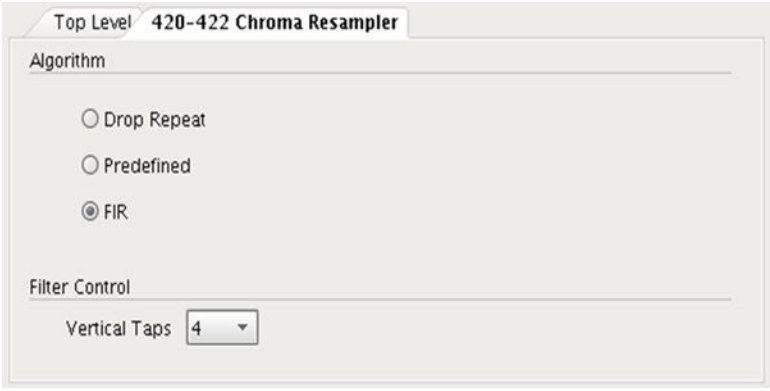


Figure 23: **4:2:0 – 4:2:2 Chroma Resampling Tab**



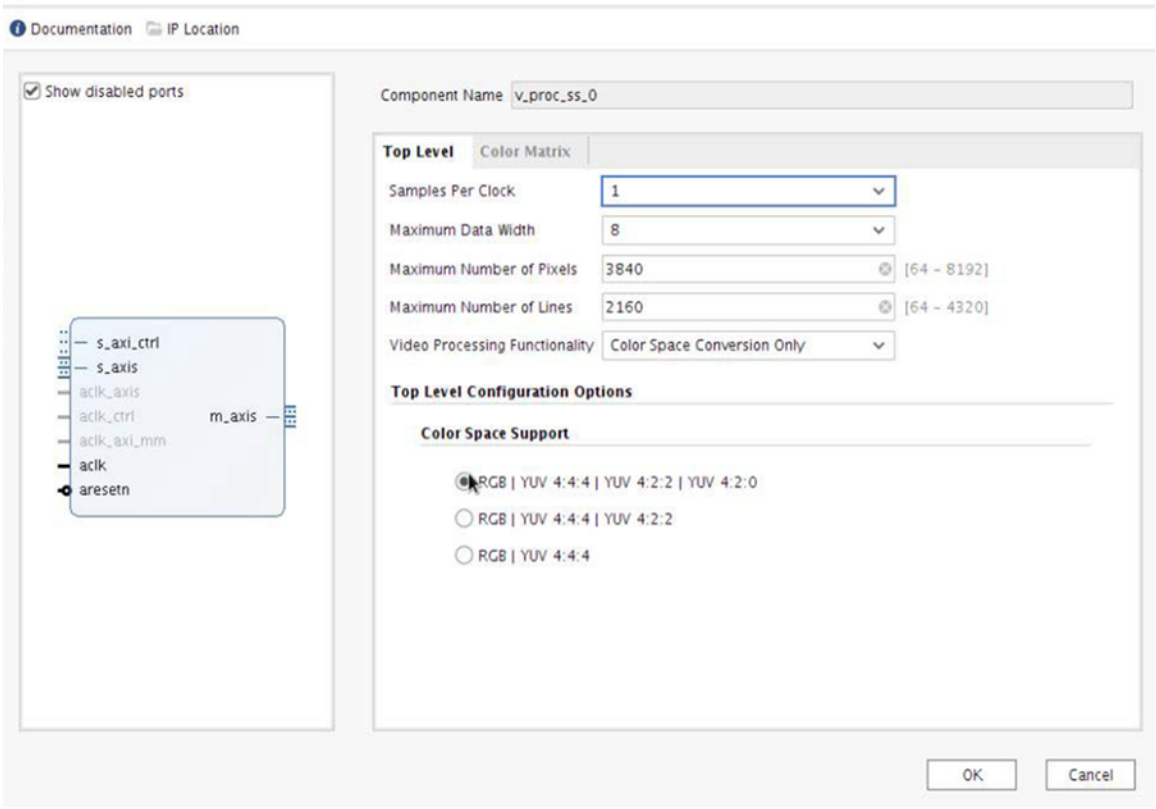
The parameters on the Chroma Resampler tabs are as follows:

- **Algorithm:**
  - **Drop Repeat:** Using the drop option results in down conversion with no filter. Some samples are passed directly to the output, but others are dropped entirely, as appropriate. This occurs on a line-by-line basis and on a pixel-by-pixel basis. The replicate option is available in all upconverters. It applies in both vertical and horizontal domains as appropriate. Using the replicate option results in up conversion with no filter. Replication of the previous input sample occurs instead.
  - **Predefined:** Filters are predefined and not programmable. The filters use only power-of-two coefficients so no DSP48s are necessary. Linear interpolation is employed for the low pass filters used for anti-aliasing and interpolation.
  - **FIR:** Used to program the filter coefficients through the AXI4-Lite interface.
- **Filter Control:** When FIR filtering is selected, the number of Horizontal and Vertical taps are defined here.
  - **Horizontal Taps:** Select 4, 6, 8, or 10 horizontal filter taps.

- **Vertical Taps:** Select 4, 6, 8, or 10 vertical filter taps.

Color Matrix Settings

Figure 24: **Top Level Color Matrix Option**



When Color Space Conversion Only is selected in Video Processing Functionality, the following option is available:

- **Color Space Support:** Select the color spaces for which support is desired. If YUV 4:2:2 is selected, basic horizontal chroma resampling is included. If YUV4:2:0 is selected, basic vertical chroma resampling is also included.

Figure 25: **Color Matrix Tab**

The parameter on the Color Matrix tab is:

- **Enable Demo Window:** This feature allows a different set of matrix coefficients to be used within a user-defined window in the image.

User Parameters

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

Table 23: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter[1]	User Parameter[1]	Default Value
Top Level		
Samples Per Clock	C_SAMPLES_PER_CLK	2
Maximum Data Width	C_MAX_DATA_WIDTH	10
Maximum Number of Pixels	C_MAX_COLS	3840
Maximum Number of Lines	C_MAX_ROWS	2160
Video Processing Functionality	C_TOPOLOGY	1
Scaler Only	0	
Full Fledged	1	
Deinterlacing Only	2	
Color Space Conversion Only	3	
420-422 Chroma Resampling Only	4	
422-444 Chroma Resampling Only	5	
Enabled Interlaced Input	C_ENABLE_INTERLACED	true
Enable Built-in DMA	C_ENABLE_DMA	true

Table 23: **Vivado IDE Parameter to User Parameter Relationship** *(cont'd)*

Vivado IDE Parameter[1]	User Parameter[1]	Default Value
Use UltraRAM for Line Buffers	USE_URAM	0
Color Space Support	C_COLORSPACE_SUPORT	0
RGB | YUV 4:4:4 | YUV 4:2:2 | YUV 4:2:0	0	
RGB | YUV 4:4:4 | YUV 4:2:2	1	
RGB | YUV 4:4:4	2	
Deinterlacer		
Enable Motion Adaptive Deinterlacing	C_DEINT_MOTION_ADAPTIVE	true
AXIMM Address Width	AXIMM_ADDR_WIDTH	32
Number Read/Write Outstanding	AXIMM_NUM_OUTSTANDING	16
Transaction Burst Length	AXIMM_BURST_LENGTH	16
Scaler		
Algorithm	C_SCALER_ALGORITHM	Polyphase
Bilinear	0	
Bicubic	1	
Polyphase	2	
Horizontal Taps	C_H_SCALER_TAPS	6
Vertical Taps	C_V_SCALER_TAPS	6
Horizontal Phases	C_H_SCALER_PHASES	64
Vertical Phases	C_V_SCALER_PHASES	64
Enable 4:2:2 Color Format	C_SCALER_ENABLE_422	true
Chroma Resampler		
Horizontal Algorithm	C_H_CHROMA_ALGORITHM	FIR
Drop Repeat	0	
Predefined	1	
FIR	2	
Vertical Algorithm	C_V_CHROMA_ALGORITHM	FIR
Drop Repeat	0	
Predefined	1	
FIR	2	
Horizontal Taps	C_H_CHROMA_TAPS	4
Vertical Taps	C_V_CHROMA_TAPS	4
Color Matrix		
Enable Demo Window	C_CSC_ENABLE_WINDOW	true
Enable 4:2:2 Color Format	C_CSC_ENABLE_422	true

Notes:

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

Output Generation

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

Constraining the Subsystem

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

Required Constraints

The only constraints required are clock frequency constraints for the AXI4-Stream video interfaces clock, `aclk_axis`, AXI4-Lite control interface clock, `aclk_ctrl`, and memory subsystem clock, `aclk_axi_mm`. Paths from AXI4-Lite signals should be constrained with a `set_false_path`, causing setup and hold checks to be ignored for AXI4-Lite signals. These constraints are provided in the XDC constraints files included with the IP.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP subsystem.

Clock Frequencies

This section is not applicable for this IP subsystem.

Clock Management

This section is not applicable for this IP subsystem.

Clock Placement

This section is not applicable for this IP subsystem.

Banking

This section is not applicable for this IP subsystem.

Transceiver Placement

This section is not applicable for this IP subsystem.

I/O Standard and Placement

This section is not applicable for this IP subsystem.

Simulation

Simulation of this core is not supported.

Synthesis and Implementation

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

Detailed Example Design

An IP integrator example design is provided to demonstrate the Video Processing Subsystem capabilities.

The supported platforms are listed in the following table.

Table 24: **Supported Platforms**

Development Boards	Additional Hardware	Processor
KC705	N/A	MicroBlaze™
ZCU102	N/A	A53
ZCU104	N/A	A53
ZCU106	N/A	A53
VCK190	N/A	CIPS

To open the example project, perform the following:

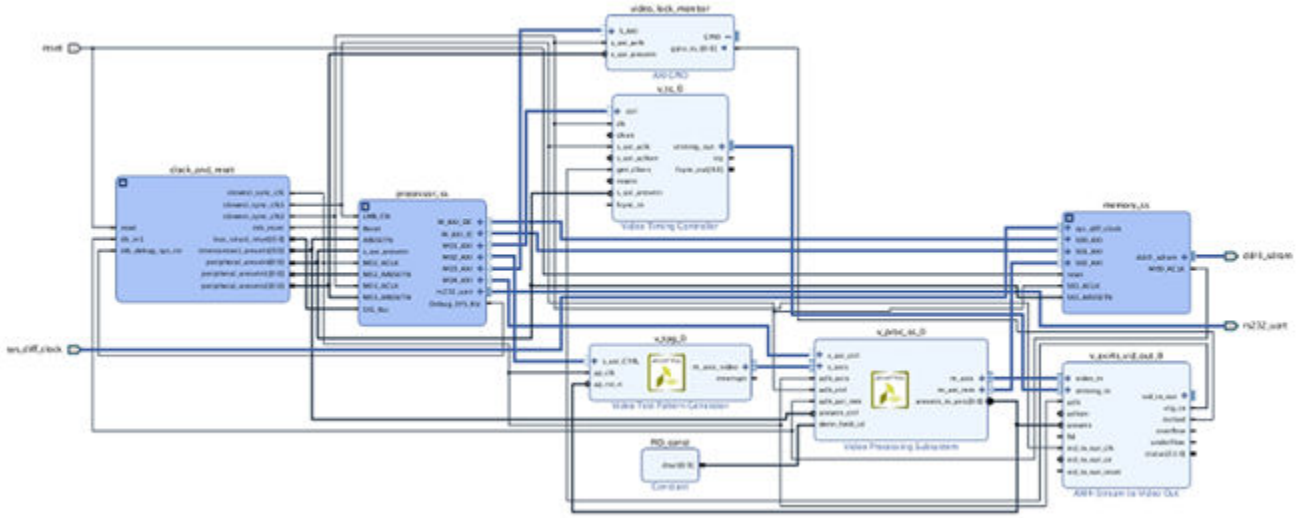
1. Add the IP to an IP integrator Block Design canvas.
2. After the IP has been instantiated in the block design, double-click on it to customize it.
3. Right-click the selected IP block and select Open IP Example Design from the menu.
4. In the Open IP Example Design window, select example project directory and click **OK**.

   A new session of the AMD Vivado™ IDE opens that shows the example design in the Design Sources window.

Full Fledged Video Processing Design

The following figure shows the top level block diagram. In the video path is a video test pattern generator and an AXI4-Stream to Video Out core. Furthermore, a processor is controlling the IPs, and a memory interconnect with MIG are interfacing with external DDR.

Figure 26: **Top-level Full Fledged Video Processing Design**



Video Path

A video test pattern generator feeds into the Video Processing Subsystem. The output of the Video Processing Subsystem is connected to the AXI4-Stream to Video Out IP.

Memory Subsystem

The memory subsystem in the Full Fledged and Deinterlacing Only designs consists of an AXI-MM interconnect that is a 3:1 cross-bar that feeds into the MIG. The 3 ports feeding into the cross bar are the data and instruction cache ports from the processor, and the memory port of the Video Processing Subsystem. For the Scaler Only, Color Space Conversion Only, and Chroma Resampling Only designs, the interconnect is a 2:1 cross bar.

Processor Subsystem

A processor is used for controlling the IPs. The example design uses the default configuration which is not optimized for performance as CPU load is not critical. The MicroBlaze has 8KB data and instruction caches added. The MicroBlaze processor is being run at 100 MHz clock speed.

Clocking

The Video Processing Subsystem uses three clocks that are derived through a clock generator from the 200 MHz system clock that is available on the AMD Kintex™ 7 KC705 board. The memory subsystem is running at this system clock at 200 MHz. This system clock feeds into the clock wizard IP, which derives a 300 MHz and 100 MHz clock at the output. The 300 MHz clock drives the AXI4-Stream video interfaces. The 100 MHz clock drives the AXI4-Lite control interface, and also drives the processor.

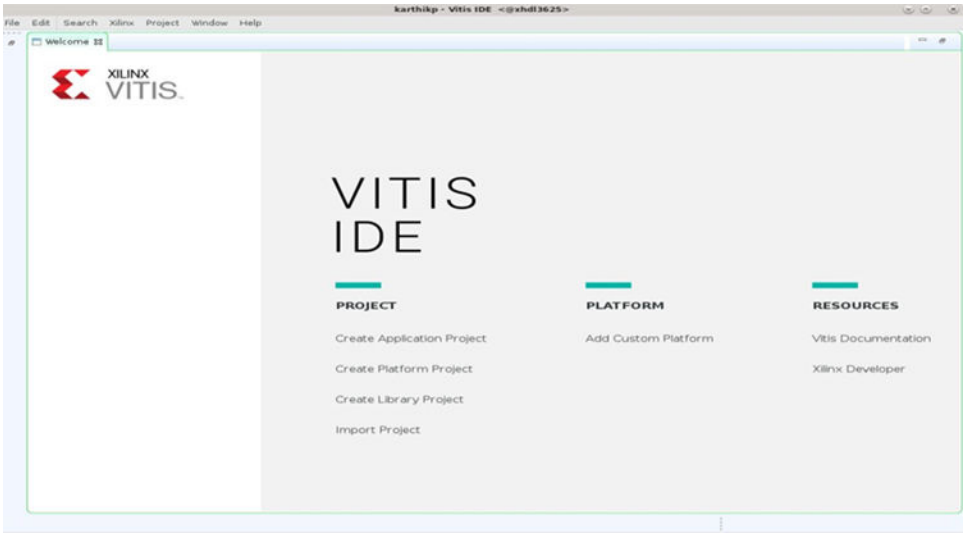
Example Design Software

The synthesizeable example design requires both Vivado and AMD Vitis™ software platform.

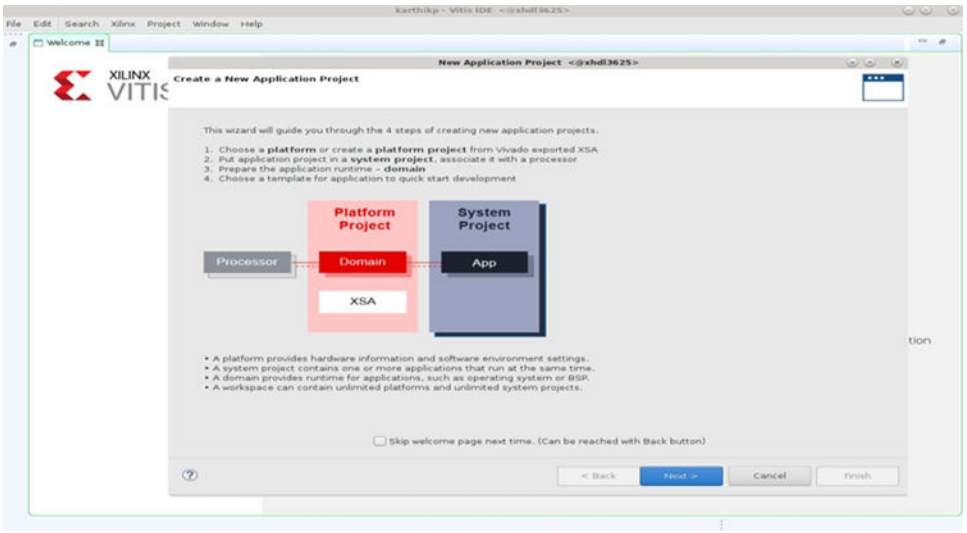
The first step is to run synthesis, implementation, and bitstream generation in Vivado. After all those steps are done, select **File > Export > Export Hardware**. In the window, select Include bitstream, select an export directory and click **OK**.

Perform the following to generate the elf file (executable and linkable file) from the Vitis software platform.

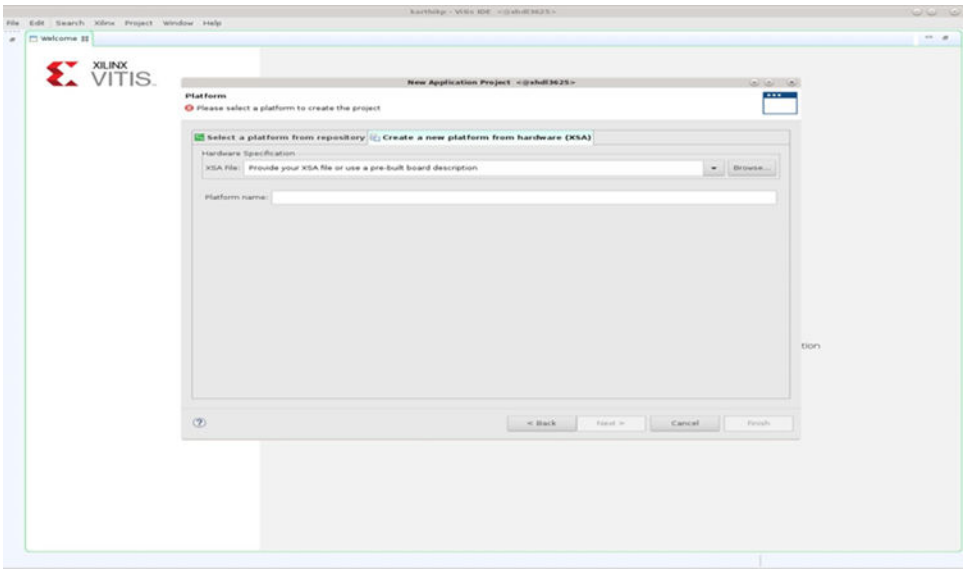
1. Open the Vitis software platform.



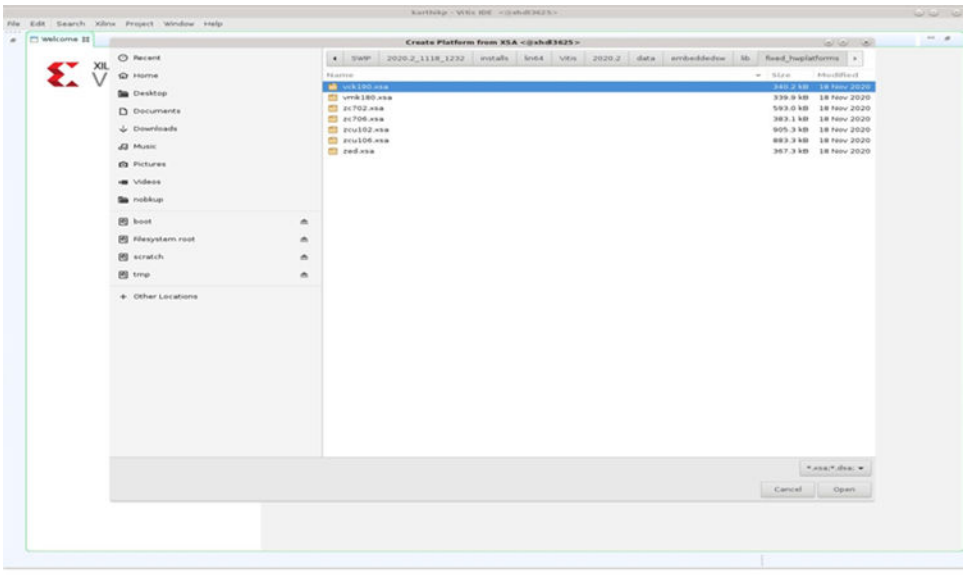
2. Select **File → New Application Project**.



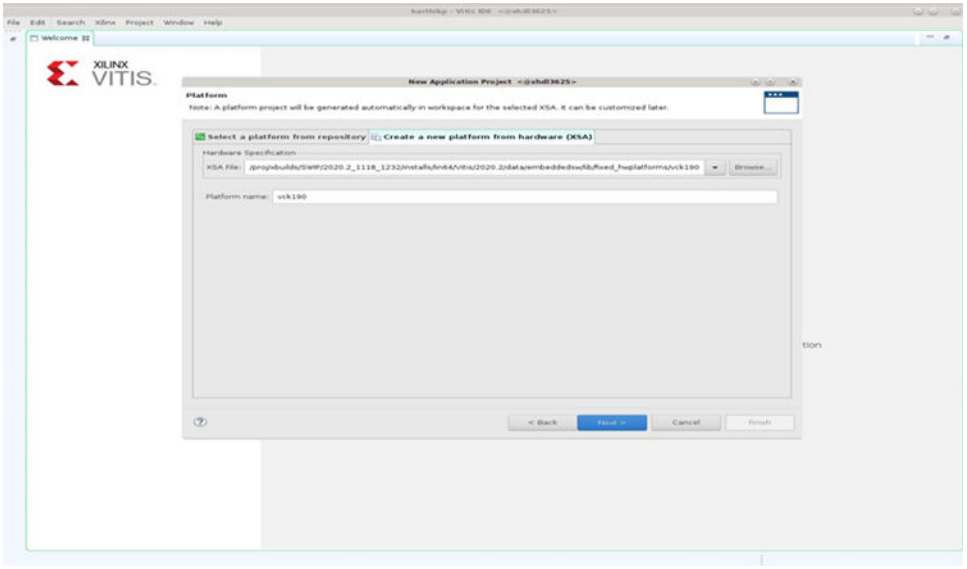
3. Select a platform to create the project.



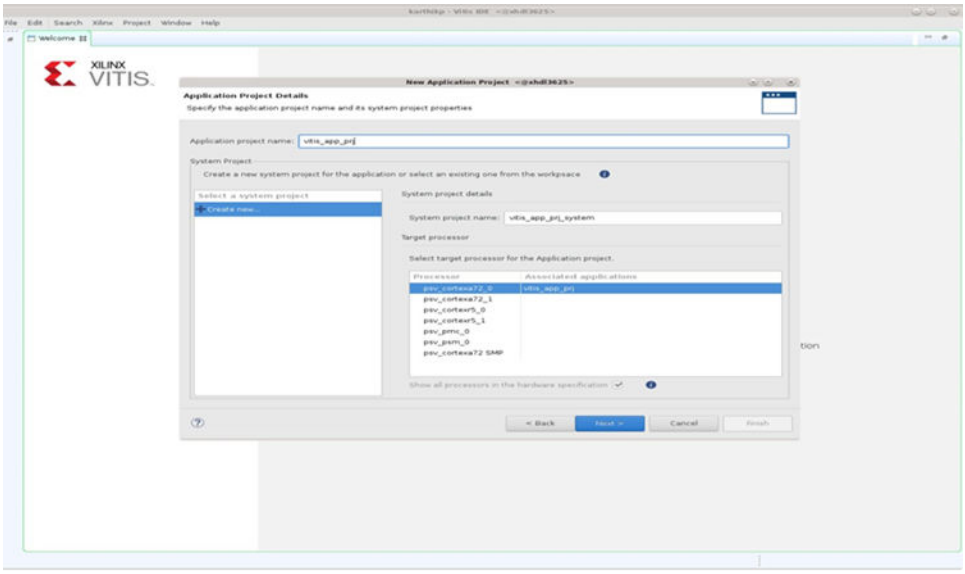
4. Select the required xsa.



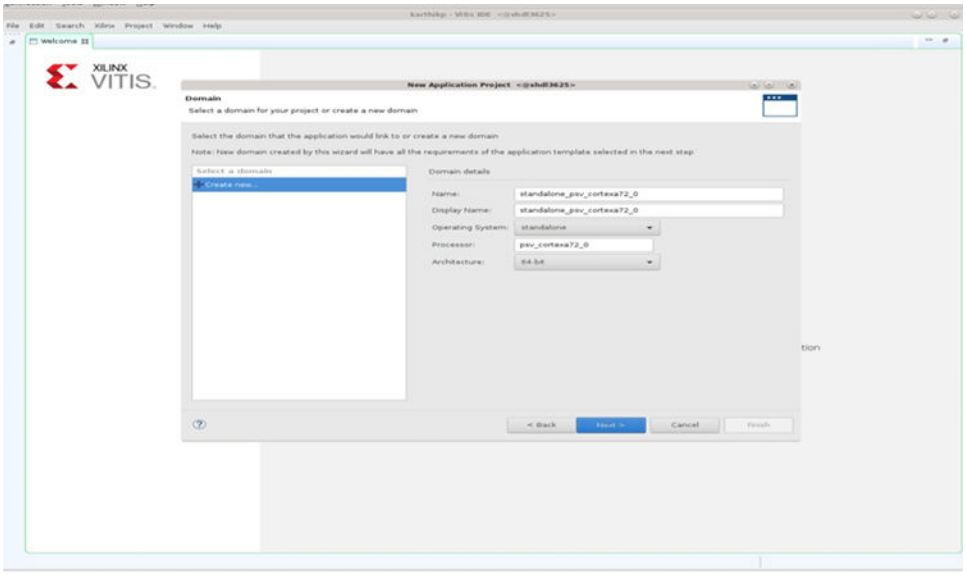
5. Click **Next**.



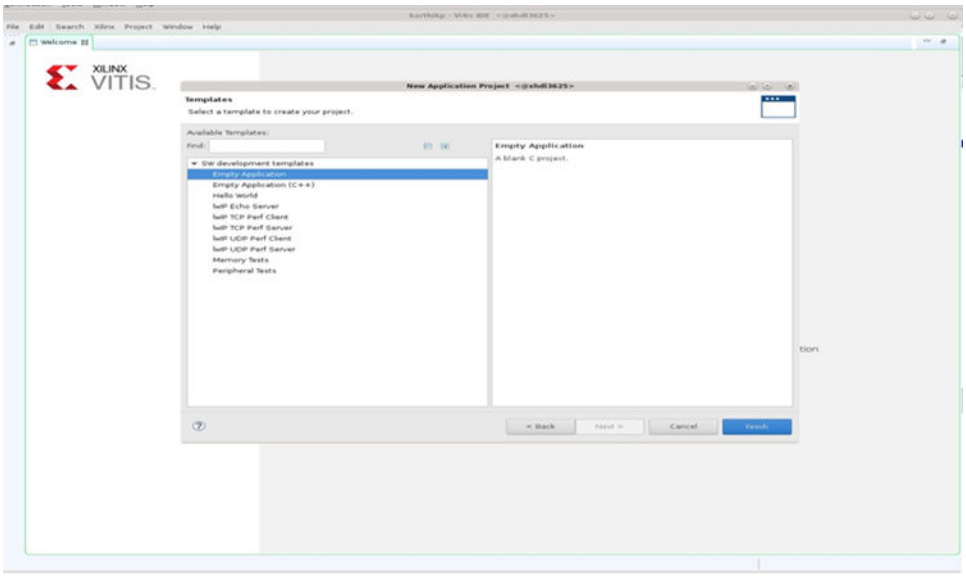
6. Name the application.



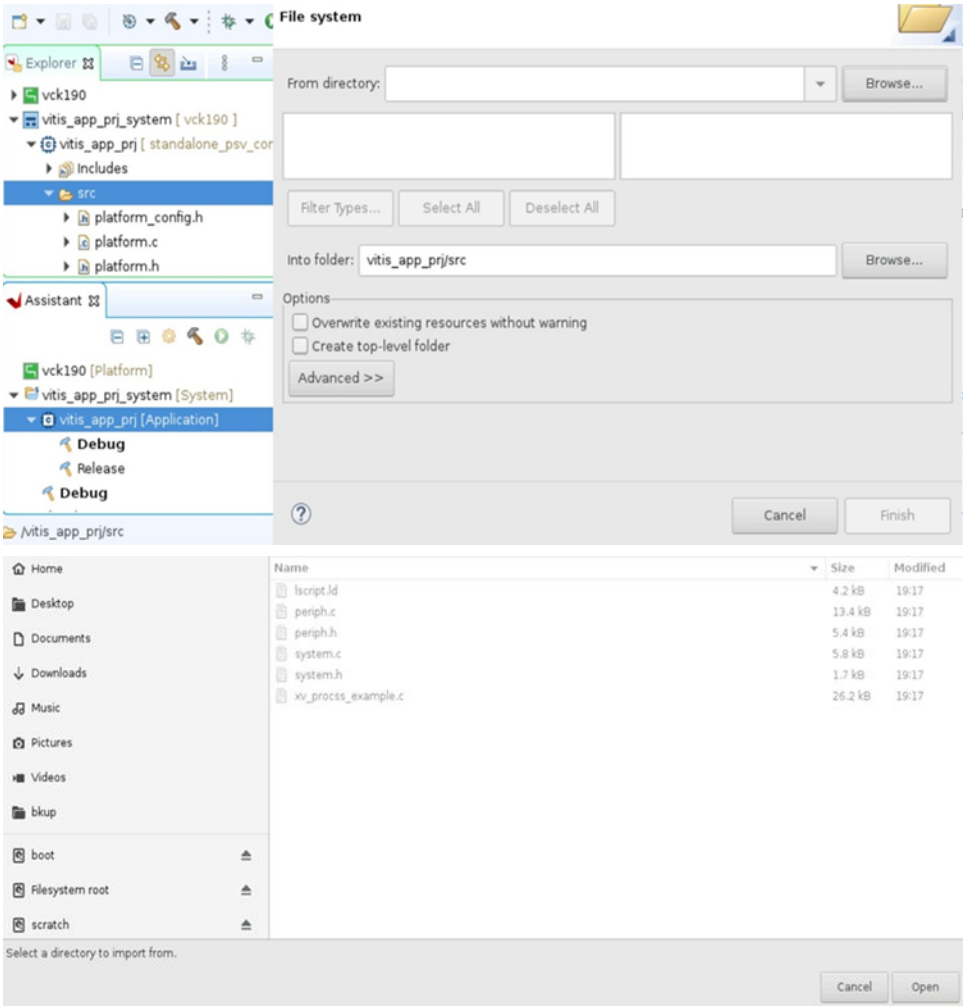
7. Select the processor and click **Next**.



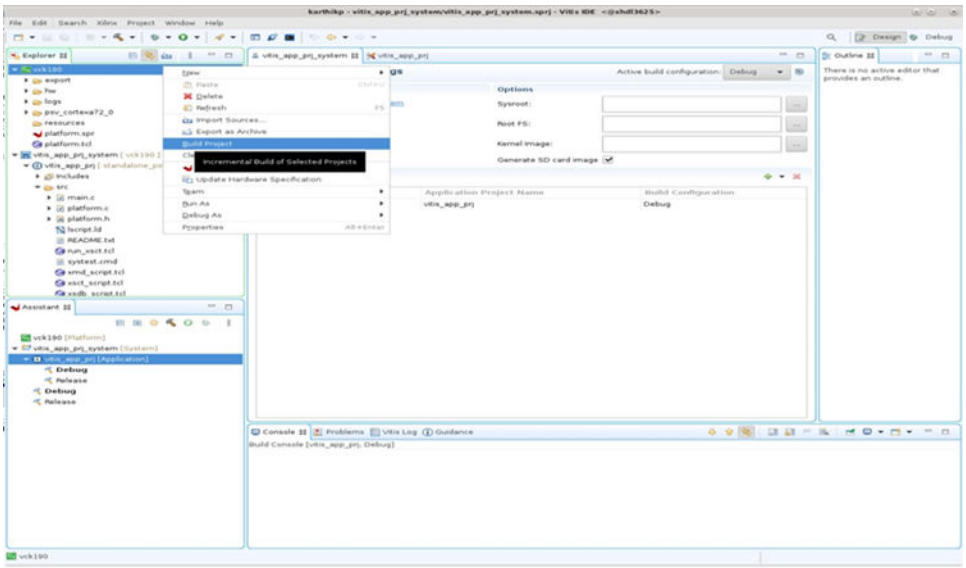
8. Select the empty application.



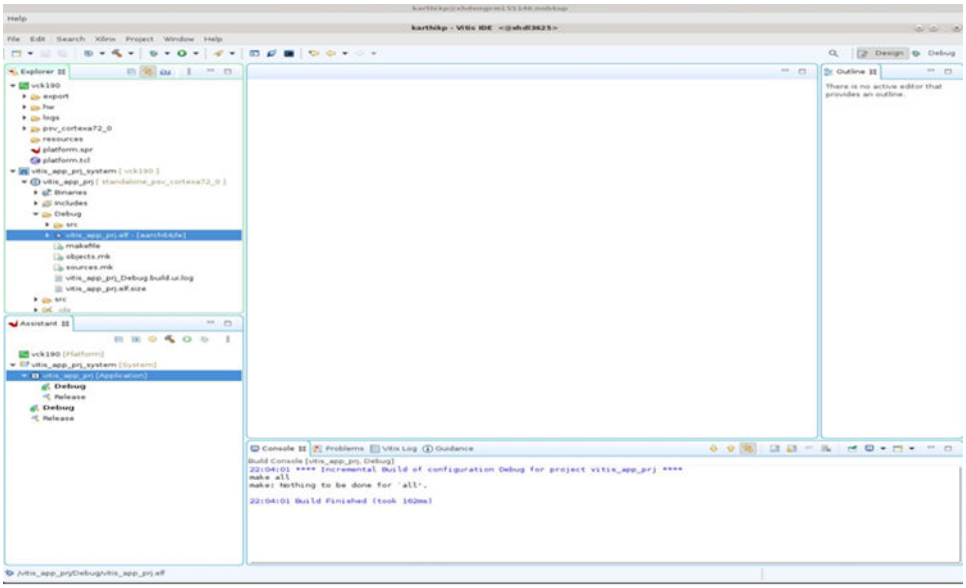
9. Import the required files.



10. Build the project.



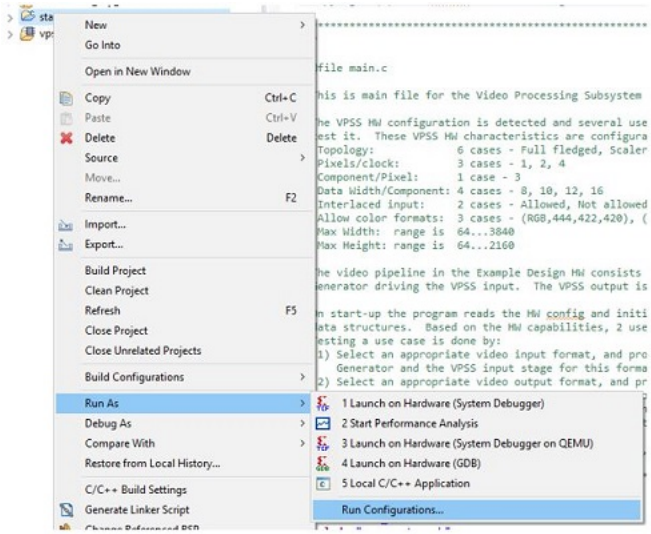
11. For the elf file, check the debug folder.



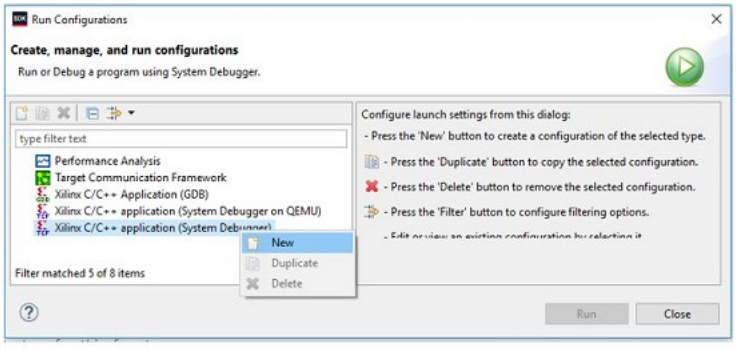
Next, perform the following to run the application:

1. Connect a USB cable from the host PC to the USB JTAG port. Ensure the appropriate device drivers are installed.
2. Connect a second USB cable from the host PC to the USB UART port. Ensure that USB UART drivers are installed.
3. Connect the evaluation board to the power supply slot.
4. Switch on the board.

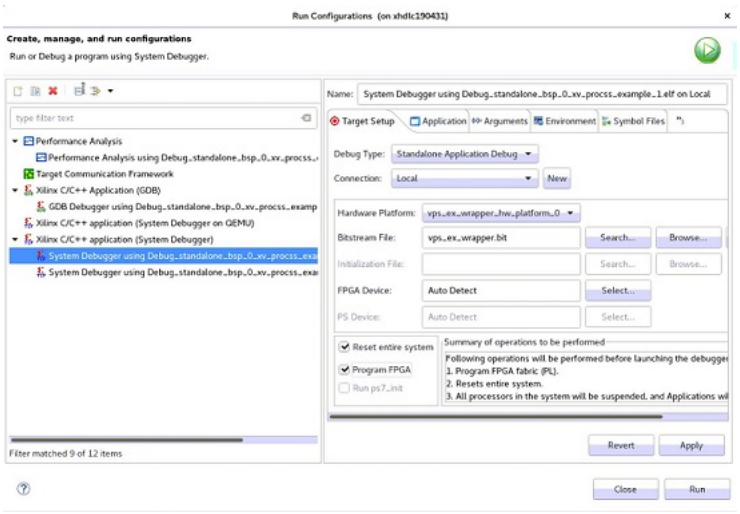
5. Start a terminal program (for example, Hyper Terminal) on the host PC with the following settings for the standard COM port:
   a. Baud Rate: 115200
   b. Data Bits: 8
   c. Parity: None
   d. Stop Bits: 1
   e. Flow Control: None
6. Right-click the **xv_procss_example** application in the project explorer window and click **Build Project**.
7. Right-click the **xv_procss_example** application and click **Run As > Run Configurations**.



8. In the Run Configurations window, right-click **Xilinx C/C++ application (System Debugger)** and click **New**.



9. Enable Program FPGA, if targeting ZCU102/ZCU104/ZCU106, ensure to enable Run `psu_init`.



10. Click **Run** to program the FPGA and launch application on the board.

    Next, perform the following steps to run the software application:

> **IMPORTANT!** *To do so, make sure that the hardware is powered on and a Digilent Cable or an USB Platform Cable is connected to the host PC. Also, ensure that a USB cable is connected to the UART port of the board.*

1. Launch Vitis software platform.
2. Set workspace to `vpss_example` folder in prompted window. The Vitis project opens automatically. (If a welcome page shows up, close that page.)
3. Download the bitstream into the FPGA by selecting **Xilinx Tools > Program FPGA**. The Program FPGA dialog box opens.
4. Ensure that the Bitstream field shows the bitstream file generated by Tcl script, and then click **Program**.

   ***Note*:** The DONE LED on the board turns green if the programming is successful.
5. A terminal program (HyperTerminal or PuTTY) is needed for UART communication. Open the program, choose appropriate port, set baud rate to 115200 and establish Serial port connection.
6. Select and right-click the application `vpss_example_design` in Project_Explorer panel.
7. Select **Run As > Launch on Hardware (GDB)**.
8. Select Binaries and Qualifier in window and click **OK**.

   The example design test result are shown in terminal program.

For more information, visit https://www.xilinx.com/products/design-tools/vitis.html.

When executed on the board, the example application determines the Video Processing Subsystem topology and sets the input and output stream configuration accordingly. The test pattern generator IP is used to generate the input stream. Video Lock Monitor IP then monitors the output of the subsystem (to vidout) to determine if lock is achieved and present the status (Pass/Fail) on the terminal.

Appendix A

Upgrading

This appendix contains information about upgrading to a more recent version of the IP subsystem.

Upgrading in the Vivado Design Suite

Changes from v2.2 to 2.3

Video Processing Subsystem v2.3 is a direct replacement for v2.2. The IP features remain same but Vitis HLS tool synthesizes the IP instead of Vivado HLS tool.

AMD Versal™ adaptive SoC Example Design is provided along with the IP.

V2.0 Parameter Changes

The Video Processing Subsystem v2.0 supports more configurability for the underlying subcores. Version 2.0 supports Deinterlacing Only, Color Space Conversion Only, and Chroma Resampling Only configurations.

The default Video Processing functionality has changed from Scaler Only to Full Fledged.

Chroma Resampling is now divided into 4:4:4-4:2:2 and 4:2:2-4:2:0. The algorithm and filter taps can be selected independently in the horizontal and vertical directions.

The drivers have been updated for v2.0. For more information, see Appendix C: Application Software Development

Appendix B

Debugging

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

Finding Help with AMD Adaptive Computing Solutions

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

Documentation

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

Answer Records

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

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

- Product name
- Tool message(s)

- Summary of the issue encountered

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

Master Answer Record for the Subsystem

AR: 65449.

Technical Support

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

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

To ask questions, navigate to the Community Forums.

Debug Tools

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

Vivado Design Suite Debug Feature

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

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

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

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

Simulation Debug

Simulation of this core is not supported.

Hardware Debug

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

General Checks

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

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.
- If your outputs go to 0, check your licensing.

Interface Debug

AXI4-Lite Interfaces

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

- The `s_axi_aclk` and `aclk` inputs are connected and toggling.
- The interface is not being held in reset, and `s_axi_areset` is an active-Low reset.
- The interface is enabled, and `s_axi_aclken` is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.

- If the simulation has been run, verify in simulation and/or a Vivado Lab Edition capture that the waveform is correct for accessing the AXI4-Lite interface.

AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions:

- If transmit `<interface_name>_tready` is stuck Low following the `<interface_name>_tvalid` input being asserted, the core cannot send data.
- If the receive `<interface_name>_tvalid` is stuck Low, the core is not receiving data.
- Check that the `aclk` inputs are connected and toggling.
- Check that the AXI4-Stream waveforms are being followed.
- Check core configuration.

Appendix C

Application Software Development

Device Drivers

The Video Processing Subsystem driver abstracts the included video processing elements and presents the processing chain as a black-box. It shields you from the internal workings of included subcores and thus provides an out-of-the-box solution for video.

The subsystem driver is a bare-metal driver. It dynamically manages the data and control flow through the processing elements, based on the input/output stream configuration set at run time. Internally, it relies on included subcore layer 1 drivers to configure the IP hardware block and layer 2 drivers to provide an abstracted view of the feature set provided by each subcore.

Dependencies

A video common driver is delivered as part of the Vitis software platform. Most video IPs have master/slave AXI4-Stream interfaces so the concept of Video Streams as an interface parameter for drivers for such cores was introduced.

The video_common defines the following features:

- Enumerations for video specific details like color format, color depth, frame rate, etc.
- Video stream, video timing and video window data types.
- Video Mode Table with predefined resolutions and their timing details.
- Utility APIs that can access data from the mode table and work with stream data types.

Architecture

The subsystem driver provides an easy-to-use, well-defined API to help integrate the subsystem in an application without having to understand the underlying complexity of configuring each and every subcore.

Subsystem driver consists of the following:

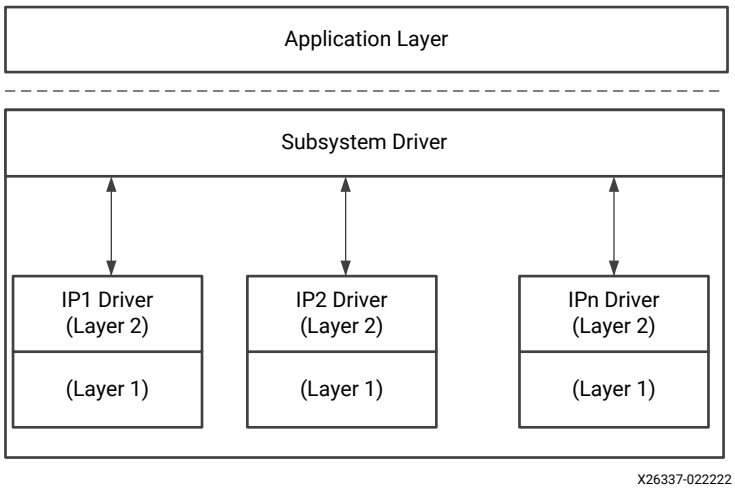
- Subsystem layer: Controls the data/control flow of AXI4-Stream through the processing cores. It uses support functions to examine the input and output stream properties and make appropriate decision to implement the determined use case.
- Subcore drivers: Every included subcore has a driver associated with it and provides 2 layers of abstraction
    - Layer-1: Implements API's to peek/poke IP registers at HW level.
    - Layer-2: Implements feature set to abstract core functionality.

For example, the color space conversion IP at the core level contains a 3 x 3 coefficient matrix which must be manipulated to do the required conversion, for example RGB to YUV.

- Layer-1 provides APIs to read/write these 3 x 3 coefficient registers. You are responsible for computing the required coefficient values and program the specific registers.
- Layer-2 implements the feature set that can be built upon layer-1 and is much easier to use:
    - API to convert RGB to YUV or vice-versa.
    - API to set/get brightness, contrast, saturation, and gain.
    - Auto translation between user setting of 0-100 to IP offered range.

The following figure shows the Video Processing Subsystem architecture.

Figure 27: **Subsystem Driver Architecture**



Prerequisites

For Full Fledged topology, the subsystem requires external memory to store video buffers. You must specify the DDR address for the buffer storage. The API to use is:

```
void XVprocSs_SetFrameBufBaseaddr(XVprocSs *InstancePtr, UINTPTR addr);
```

Usage

The subsystem driver itself is not an active driver and relies on application software to make use of the provided APIs to configure it. Application software is responsible to monitor the system for external inputs and communicate changes to input/output stream properties to the subsystem through provided APIs, and trigger the subsystem auto reconfiguration process.

To integrate and use the Video Processing Subsystem driver in a user application, the following steps must be followed:

1. Include the subsystem header file `xvprocss.h` that defines the subsystem object.
2. Declare an instance of the subsystem object in the application code:

   ```
   XVprocSs VprocInst;
   ```

3. Initialize system peripherals (timer, interrupt controller, UART, etc.) in the application.

4. If delay routine is defined at application level using a timer, this must be registered with the subsystem:

```
void XVprocSs_SetUserTimerHandler(XVprocSs *InstancePtr,
 XVidC_DelayHandler CallbackFunc,
 void *CallbackRef);
```

   If the application is not using the timing peripheral to implement the timeouts, skip this step. The Subsystem driver internally uses the platform-specific delay handler.

5. Initialize Video Processing Subsystem at Power-On:

```
int XVprocSs_CfgInitialize(XVprocSs *InstancePtr,
 XVprocSs_Confg *CfgPtr,
UINTPTR EffectiveAddr);
```

   Accesses the HW configuration determines the included subcores and initializes them to the power on default. The input and output stream resolution is set to 1080p at 60 Hz, RGB, and color depth is 10-bit. Finally all the cores are reset and the subsystem Ready flag is set.

6. Configure subsystem:

```
int XVprocSs_SetSubsystemConfig(XVprocss *InstancePtr);
```

   This API is the entry point and trigger mechanism for the subsystem processing control flow. It examines the input and output stream properties and validates them against the supported feature set. If the use case is supported, the driver determines the subcores required to implement the use case, builds a data flow network, configures cores in the network, and starts the video pipe.

Now, the subsystem is ready for use and the application software monitors changes in the system input or output and communicates those changes to the video subsystem.

Appendix D

Additional Resources and Legal Notices

Finding Additional Documentation

Documentation Portal

The AMD Adaptive Computing Documentation Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Documentation Portal, go to https://docs.xilinx.com.

Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

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

***Note*:** For more information on DocNav, refer to the *Documentation Navigator User Guide* (UG968).

Design Hubs

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

- In DocNav, click the **Design Hubs View** tab.
- Go to the Design Hubs web page.

Support Resources

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

References

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *ISE to Vivado Design Suite Migration Guide* (UG911)
6. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
7. *Vivado Design Suite User Guide: Implementation* (UG904)
8. AXI4-Stream Video IP and System Design Guide (UG934)
9. *Vivado Design Suite: AXI Reference Guide* (UG1037)
10. *AXI Video Direct Memory Access LogiCORE IP Product Guide* (PG020)
11. *AXI GPIO LogiCORE IP Product Guide* (PG144)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
02/21/2024 Version 2.4	
AXI-Lite Control Interface	Added a note.
10/19/2022 Version 2.4	
Deinterlace Only	Added register names.
04/27/2022 Version 2.3	
Chapter 3: Product Specification	Updated tables.

Section	Revision Summary
10/27/2021 Version 2.3	
Chapter 4: Designing with the Core	• Deinterlacer features updated. • MM interface updated from 32 bits for deinterlacer to optional 32/64. • MM interface can be 128/256/512 for full fledged design.
08/06/2021 Version 2.3	
General updates	Entire document.
02/05/2021 Version 2.3	
Chapter 6: Detailed Example Design	Updated images.
02/04/2021 Version 2.3	
Chapter 5: Design Flow Steps	Added Deinterlacing information.
07/08/2020 Version 2.1	
Feature Summary	Updated feature summary.
AXI4-Stream Video	Updated AXI4-Stream Video section.
Chapter 5: Design Flow Steps	Updated Samples Per Clock description.
Horizontal Scaler	Updated Horizontal Scaler Registers table.
12/17/2019 Version 2.1	
General Updates	Updated SDK instances to Vitis software platform.
Chapter 6: Detailed Example Design	Updated the section with the Vitis software platform flow.
05/22/2019 Version 2.1	
Performance	Updated tables Video Processing Subsystem Resource Utilization, Color Space Conversion OnlyRegisters, and 422-444 Chroma Resampling Only Registers.
Example Design Software	Updated the section.
	• Added resolution support up to 8192x4320. • Added deinterlacer support for 32-bit and 64-bit memory address.
01/30/2018 Version 2.0	
General Updates	• Added register descriptions for individual IPs in the video processing subsystem. • Added descriptions for individual IPs (scaler, deinterlacer). • Added reset descriptions for the video processing subsystem.
04/04/2018 Version 2.0	
General Updates	• Added option to use UltraRAM for line buffers on UltraScale+ devices. • Added support for ZCU102, ZCU104, and ZCU106 boards in the example design.

Section	Revision Summary
10/04/2017 Version 2.0	
General Updates	Added format conversion to Color Space Conversion Only functionality (convert between RGB, YUV 4:4:4, YUV 4:2:2, and YUV 4:2:0)
04/05/2017 v2.0	
General Updates	Updated for GUI screens and description.
10/05/2016 Version 2.0	
General Updates	Added video format conversion to Scaler Only configuration, added YUV 4:2:0 support to Video Deinterlacer, added pass through mode to Chroma Resamplers. Updated Xilinx automotive applications disclaimer.
04/06/2016 Version 2.0	
General Updates	Additional configurability features.
11/18/2015 Version 1.0	
General Updates	Added support for UltraScale+ devices.
10/19/2015 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices

AUTOMOTIVE APPLICATIONS DISCLAIMER

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

Copyright