



Video Scaler IP

IP Version: 1.3.0

User Guide

FPGA-IPUG-02234-1.3

July 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language [FAQ 6878](#) for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents.....	3
Acronyms in This Document	6
1. Introduction	7
1.1. Quick Facts	7
1.2. Features.....	7
1.3. Conventions.....	8
1.3.1. Nomenclature.....	8
1.3.2. Signal Name	8
2. Functional Description.....	9
2.1. Key Concepts	9
2.1.1. Algorithm and Supported Filter Kernels	9
2.2. Block Diagram.....	10
2.3. Clocking Architecture	11
2.4. AXI-Stream Receiver	11
2.5. AXI-Stream Transmitter.....	13
2.6. Native Video Timing Diagram.....	16
2.7. Blanking Pass-through.....	17
2.7.1. Blanking Pass through When Using AXI-Stream	17
2.7.2. Blanking Pass Through When Using Native Interface	17
2.8. YCbCr Video Format Timing	18
2.9. AXI-Lite Subordinate.....	19
2.9.1. Write Operation	19
2.9.2. Read Operation	19
3. Signal Description	20
3.1. Video Scaler IP with AXI Stream Interface.....	20
3.2. Video Scaler IP with Native Interface	22
4. Parameter Settings & Attributes Summary	24
4.1. Architecture Section.....	24
4.2. Implementation Section.....	27
5. Register Description.....	29
6. IP Core Generation, Simulation, and Validation	31
6.1. Licensing the IP Core	31
6.2. Generation and Synthesis	31
6.3. Running Functional Simulation	34
6.3.1. Limitations of Simulation.....	38
7. Design Considerations	39
7.1. Known Issues.....	39
8. Ordering Part Number	40
Appendix A. Resource Utilization	41
References	43
Technical Support Assistance	44
Revision History	45

Figures

Figure 2.1. Video IP Architecture Diagram	9
Figure 2.2. Scaling Process with 4-tap Bicubic Filter	10
Figure 2.3. Video Scaler IP Core Block Diagram	11
Figure 2.4. AXI-Stream Receiver Signals for PPC=1 (RGB Format)	12
Figure 2.5. AXI-Stream Receiver Signals for PPC=2 (RGB Format)	12
Figure 2.6. AXI-Stream Receiver Signals for PPC=4 (RGB Format)	13
Figure 2.7. Bit Allocation for BPC=8 and PPC=1 for RGB Input	13
Figure 2.8. Bit Allocation for BPC=10 and PPC=1 for RGB Input	13
Figure 2.9. Bit Allocation for BPC=8 and PPC=2 for RGB Input	13
Figure 2.10. Bit Allocation for BPC=8 and PPC=4 for RGB Input	13
Figure 2.11. AXI-Stream Transmitter Signals for PPC=1 (RGB Format and BPC=8)	14
Figure 2.12. AXI-Stream Transmitter Signals for PPC=2 (RGB Format and BPC=8)	14
Figure 2.13. AXI-Stream Transmitter Signals for PPC=4 (RGB Format and BPC=8)	15
Figure 2.14. Native Input Video Timing Diagram	16
Figure 2.15. Native Output Video Timing Diagram	16
Figure 2.16. AXI-Stream Receiver with Blanking Related Signals for PPC=1(RGB Format)	17
Figure 2.17. Native Input Video Timing Diagram with Blanking Related Signals	18
Figure 2.18. YCbCr 4:2:2 Parallel Scaling (PPC=2)	19
Figure 4.1. Architecture Tab	24
Figure 4.2. Implementation Tab	27
Figure 6.1. Module/IP Block Wizard	31
Figure 6.2. Configuration User Interface for Video Scaler IP	32
Figure 6.3. Check Generating Result	33
Figure 6.4. Simulation Wizard	34
Figure 6.5. Adding and Reordering Source	35
Figure 6.6. Run Simulation Value of 0 for Run All	36
Figure 6.7. Sample Simulation Wave Form	36
Figure 6.8. Data Check Passed	37
Figure 6.9. Testing AXI-Lite Register Read/Write Interface	37
Figure 6.10. Dynamically Configurable Parameters in the Code	38
Figure 6.11. Simulation Showing Passed for Matching the Dynamic Parameters Same as GUI Entered Parameters	38
Figure 6.12. Simulation Showing Completed for Difference Between Dynamic and GUI Entered Parameters	38

Tables

Table 1.1. Video Scaler IP Quick Facts	7
Table 2.1. AXI-Lite Channels and Signals.....	19
Table 3.1. Description of Width Parameters	20
Table 3.2. Video Scaler IP Signal Description	20
Table 3.3. Video Scaler IP Native Interface I/O	22
Table 4.1. Attributes Table for Architecture Tab	25
Table 4.2. Attributes Description for Architecture Tab	26
Table 4.3. Attributes Table for Implementation Tab	27
Table 4.4. Attributes Description for Implementation Tab.....	28
Table 5.1. Summary of Configuration and Status Registers	29
Table 6.1. Generated Files List	33
Table 8.1. Ordering Part Numbers for Video Scaler IP Core	40
Table A.1. Resource Utilization using LFCPNX-100-8LFG672C Certus-Pro NX Device	41
Table A.2. Resource Utilization using LFD2NX-40-8MG121C Certus-NX Device.....	41
Table A.3. Resource Utilization using LIFCL-33-8USG84C CrossLink-NX Device	42
Table A.4. Resource Utilization using LAV-AT-E70-2LFG676C Lattice Avant Device	42
Table A.5. Resource Utilization using LN2-CT-20ES-1ASG410I Certus-N2 Device	42

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AXI	Advanced eXtensible Interface
BD	Blanking Data
BPC	Bits Per Component
CDC	Clock Domain Crossing
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HBD	Horizontal Blanking Data
PPC	Pixel Per Clock
RGB	Red Green Blue
RTL	Register Transfer Level
VBD	Vertical Blanking Data

1. Introduction

The Video Scaler IP Core is used to scale up or scale down the resolution of a video stream. The IP supports scaling from an arbitrary input resolution to a wide range of output resolutions as configured by the user. The IP supports four scaling algorithms namely, nearest neighborhood, bilinear interpolation, bicubic interpolation, and multi-tap Lanczos filter. The Lanczos filter supports multiple taps from 4 to 12. The Video Scaler IP core supports Red Green Blue (RGB), YCbCr 4:4:4, and YCbCr 4:2:2 video formats.

This IP supports either a native video interface or a standard Advanced eXtensible Interface (AXI)-Stream interface for the input and output of video data streams. There is also an option to support run time, dynamic programming of parameters through an AXI-Lite interface.

1.1. Quick Facts

[Table 1.1](#) presents a summary of Video Scaler IP.

Table 1.1. Video Scaler IP Quick Facts

IP Requirements	Supported Devices	Lattice Avant™, CrossLink™-NX, Certus™-NX (LFD2NX-17, LFD2NX-40, LFD2NX-35, LFD2NX-65), CertusPro™-NX, Certus-N2, MachXO5™-NX (LFMXO5-35, LFMXO5-35T, LFMXO5-65, LFMXO5-65T)
	IP Changes	For a list of changes to the IP, refer to the Video Scaler IP Release Notes (FPGA-RN-02063) .
Resource Utilization	Resources	See the Resource Utilization section
Design Tool Support	Lattice Implementation	IP v1.1.0 – Lattice Radiant™ software 2022.1 or later IP v1.2.0 – Lattice Radiant software 2024.2 IP v1.3.0 – Lattice Radiant software 2025.1
	Synthesis	Lattice Synthesis Engine Synopsys® Synplify Pro® for Lattice
	Simulation	For a list of supported simulators, see the Lattice Radiant Software User Guide.

1.2. Features

The key features of the Video Scaler IP are:

- Configurable input/output resolution from 32 to 4096.
- Choice of four scaling algorithms: Nearest neighbor, Bilinear, Bicubic, and Lanczos.
- Choice of AXI-Stream or native video interface for video data input and output.
- Configurable number of filter-taps for Lanczos coefficient set.
- Configurable number of phases for Bicubic, and Lanczos coefficient sets.
- Configurable pixels per clock (PPC): 1,2, and 4.
- Configurable bits per color: 8, 10, 12, and 16.
- Coefficient width selectable from 6 to 18.
- Multiple color spaces: RGB, YCbCr 4:4:4, and YCbCr 4:2:2.
- Optional AXI-Lite interface for dynamic update of some IP parameters.
- Blanking data passes through to the output.

1.3. Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog Hardware Description Language (HDL).

1.3.2. Signal Name

Signal Names that end with:

- `_n` are active low.
- `_i` are input signals.
- `_o` are output signals.
- `_io` are bi-directional input/output signals.

2. Functional Description

2.1. Key Concepts

A video scaler maps input video frames to output frames of a different size. The output frame size can be either smaller or bigger than the input size and each X and Y axes can have a different scaling ratios. In a simpler up scaling by an integer factor, the output frame has all the pixels from the input frame and additionally more pixels interspersed between the input pixels. The newly interspersed pixels are determined by the scaler using one of the chosen algorithms.

If the scale up factor is fractional number, not all input pixels are used as is in the output frame, but instead more target pixels are computed based on the input pixels. For the simpler down scaling by an integral factor, the output frame is constructed by removing alternate pixels from the input frame. Since the downscaling is a decimation process, a two-dimensional low pass filtering is required before decimation to bandlimit the input image frequency. The algorithms used for downscaling include the low pass filtering process before decimation. As in upscaling, down scaling by a fractional factor may not use the input pixels as is to construct the target frame but computes more of the target pixels based on input pixel values.

The algorithms used for scaling are essentially performing low pass filtering. The term “filter” is used in this document to refer to the scaling process and the term “coefficients” is used to refer to the constants used in the filter arithmetic. In some of the algorithms, the granularity of the grid between two known value pixels is referred to as “phase”.

The high-level architecture of the Video Scaler core is shown in [Figure 2.1](#).

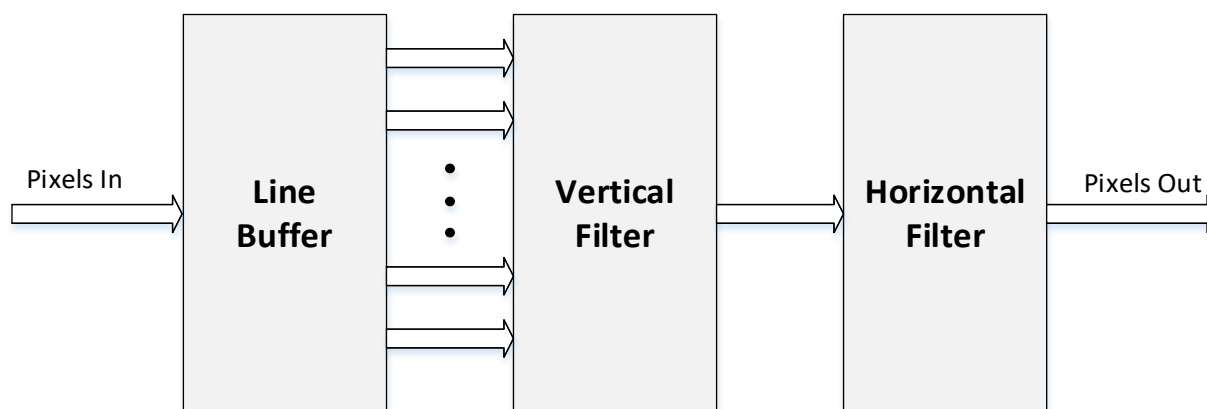


Figure 2.1. Video IP Architecture Diagram

The Video Scaler IP core allows different scaling factors for the horizontal and vertical dimensions. It performs vertical and horizontal scaling in two steps. The input pixel data is first stored in the line buffer. The size of the line buffer is dictated by the number of the vertical filter taps and the maximum input frame width. Pixel data is read out of the line buffer and passed to the vertical filter column by column. Likewise, the vertical filter coefficients are read out of the coefficient memories and passed to the vertical filter for processing along with the pixel data. The row outputs from the vertical filter are then passed to the horizontal filter to generate the output pixels. Output precision control is then performed on the final output pixel value.

2.1.1. Algorithm and Supported Filter Kernels

Video scaling is a process of generating pixels that do not exist in the original image. To compute an output pixel from a set of fixed input pixels, it is necessary to map the output pixel to the input pixel grid and estimate the location of the output pixel relative to the input pixels. The algorithm approximates the output pixel value by using a filter with coefficients weighted accordingly.

The Video Scaler IP core supports four adaptive scaling kernels: 1-tap nearest neighbor, 2-tap bilinear, 4-tap bicubic, and configurable multi-tap Lanczos filters. The range for the size of the Lanczos filters is from 4 taps to 12 taps.

Filter coefficients are generated at compile time when the kernel is configured. A brief description of the four algorithms is given below.

Nearest neighbor: Nearest neighbor is the most basic method of all. It requires the least processing time because it only uses one pixel that is the closest one to the interpolated point for the interpolation.

Bilinear: Bilinear algorithm considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value.

Bicubic: An n-tap Bicubic filter uses n adjacent input pixels to interpolate an output pixel. The spaces between the adjacent pixels are divided into a configurable number of positions called phases so that the output pixel is mapped into one of these positions. The weights of the coefficients are determined by the positions or phases and how close they are to the output pixel. The higher the number of phases, the more accurate is the interpolated output pixel. An example 4-tap bicubic filter method is depicted in Figure 2.2.

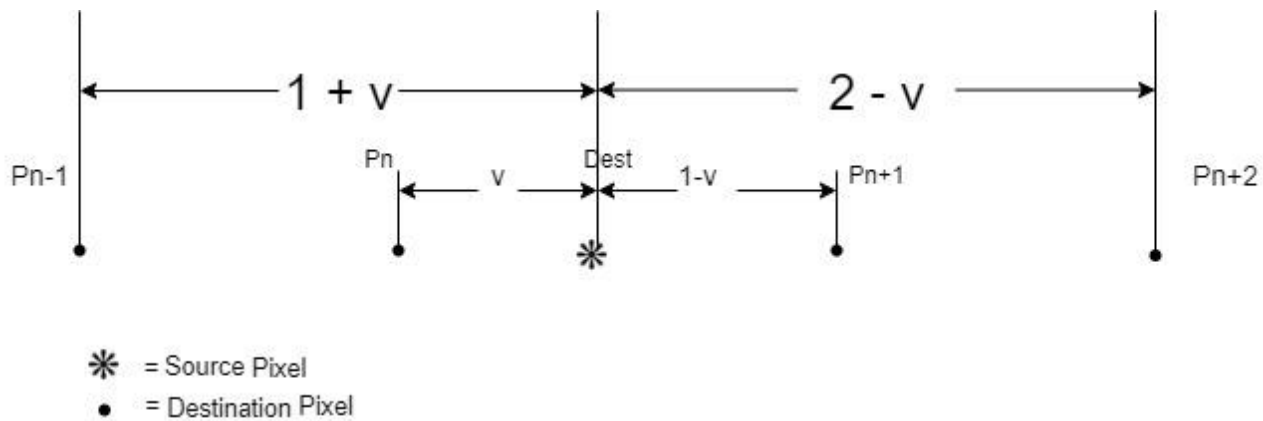


Figure 2.2. Scaling Process with 4-tap Bicubic Filter

The equation for determining the destination pixel with a 4-tap bicubic filter is shown below.

$$\text{Dest} = P_{n-1} * \text{coeff}(1+v) + P_n * \text{coeff}(v) + P_{n+1} * \text{coeff}(1-v) + P_{n+2} * \text{coeff}(2-v)$$

Lanczos: Lanczos algorithm is similar to bicubic except that it uses sinc function for interpolation and variable filter taps ranging from 4 to 12 for both horizontal and vertical filters.

2.2. Block Diagram

The data flow diagram for Video scaler is shown in Figure 2.3. AXI-stream Rx and Tx interfaces are attached to the input and output of the scaler core. There are First In First Out (FIFOs) added on the input and output sides to cross the data between clock domains as well as to merge and split data for 2 and 4 pixels per clock support. As Figure 2.3 shows there is a blanking data extraction module at the input of the IP. The blanking data is extracted and passed out through separate dedicated outputs. The dynamic configuration of the IP and register access are supported through an AXI-Lite interface.

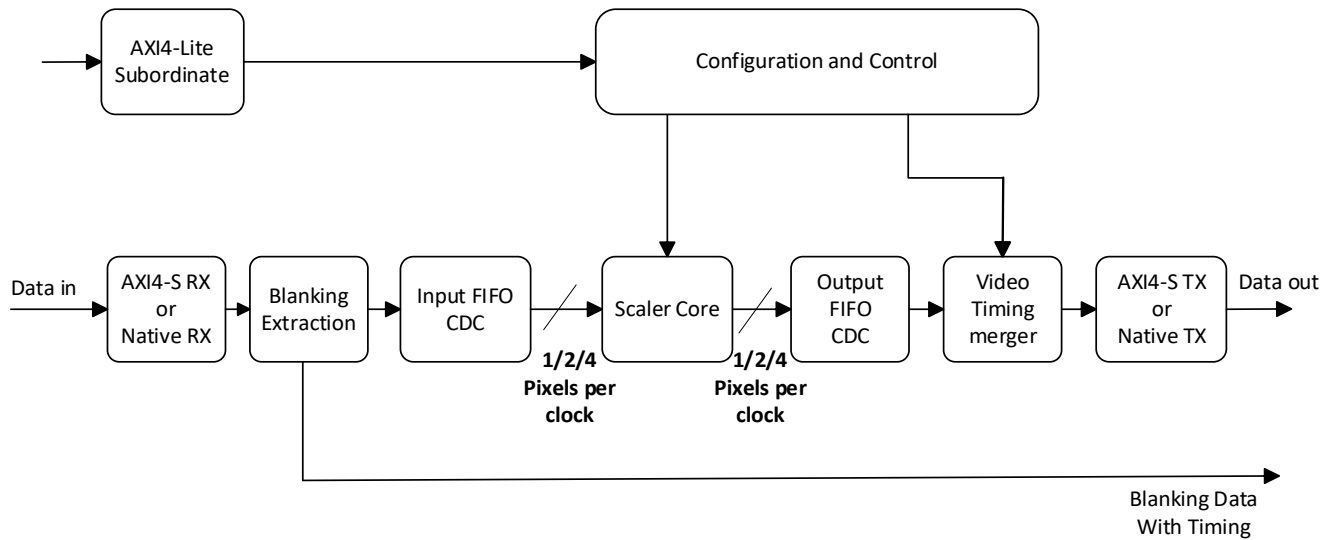


Figure 2.3. Video Scaler IP Core Block Diagram

The Video Scaler IP core supports in-system re-programming of the input and output frame sizes via AXI4-lite interface. If dynamic reconfiguration is enabled, then the maximum input and output frame resolutions need to be specified so that line buffer and various counters can be configured appropriately.

2.3. Clocking Architecture

The Video Scaler design includes 4 clock domains:

1. Receiver clock domain: all blocks left of the Input FIFO Clock Domain Crossing (CDC) module in [Figure 2.3](#).
2. Core clock domain: all blocks between the Input FIFO CDC module and the Output FIFO CDC module
3. Transmitter clock domain: all blocks after the Output FIFO CDC module
4. Configuration clock domain: AXI-Lite Subordinate and the Configuration and Control modules

When the “Dynamic Reconfiguration” parameter (refer to [Table 4.1](#)) is checked, the core clock is made available to the user as an input port, else it is internally connected to the faster of the two clocks depending on the fixed parameters entered in the Graphical User Interface (GUI). For example, if the IP is configured for upscaling, and “Dynamic Reconfiguration” is not checked the core clock is internally connected to the transmit clock. If the IP is configured for downscaling, the core clock is connected to the receive clock. If the “Dynamic Reconfiguration” is checked, the core clock is driven by the user through an input port.

The transmit clock and receive clock rates must be equal to or higher than the respective pixel clock rates. The core clock, if available, must be equal to or higher than the maximum of the transmit and receive clocks. The configuration clock is independent of the data flow, so there is no speed restriction on it.

2.4. AXI-Stream Receiver

The AXI-Stream Rx block is the protocol receiver for AXI-Stream. The AXI-Stream handshake signals and user defined TUSER bus specifies frame boundaries and active data windows respectively. The user defined signal `rx_tuser_i[0]` is used to indicate the start of frame, with a value of 1 being asserted during the first pixel of a frame. Similarly, the input `rx_tuser_i[1]` indicates the active part of a line, with a value of 1 indicating the active part and 0, the blanking part. The input `rx_tuser_i[2]` is used to indicate the end of frame being high during the last active pixel of a frame. The `rx_tlast_i` signal is used to mark the end of a line, being asserted high during the last valid clock cycle of each line.

The IP core uses a simple handshake to pass pixel data into the core. The core asserts its `rx_tready_o` output when it is ready to receive data. When the driving module has data to give the core, it drives the core’s `rx_tvalid_i` port to a 1 synchronously with the rising edge of the `clk` signal, providing the input pixel data on port `rx_data_i`. The `rx_tuser_i` (Start of

frame) input should be driven to a 1 during the clock cycle when the first pixel of the first row in the incoming video frame is active.

The output signal, rx_tready_o is normally set to 1, except when the internal FIFO is full. The IP GUI allows the user to set the input FIFO depth automatically or based on user input. If Automatic FIFO Depth is checked (refer to Table 4.2), the FIFO depth is equal to (Input active pixels/ pixels per clock). The rx_tstrobe_i signal which identifies the valid bytes in a data beat is not used by this IP.

The timing diagrams for the AXI-Stream receiver for different PPC values are shown in Figure 2.4, Figure 2.5, and Figure 2.6.

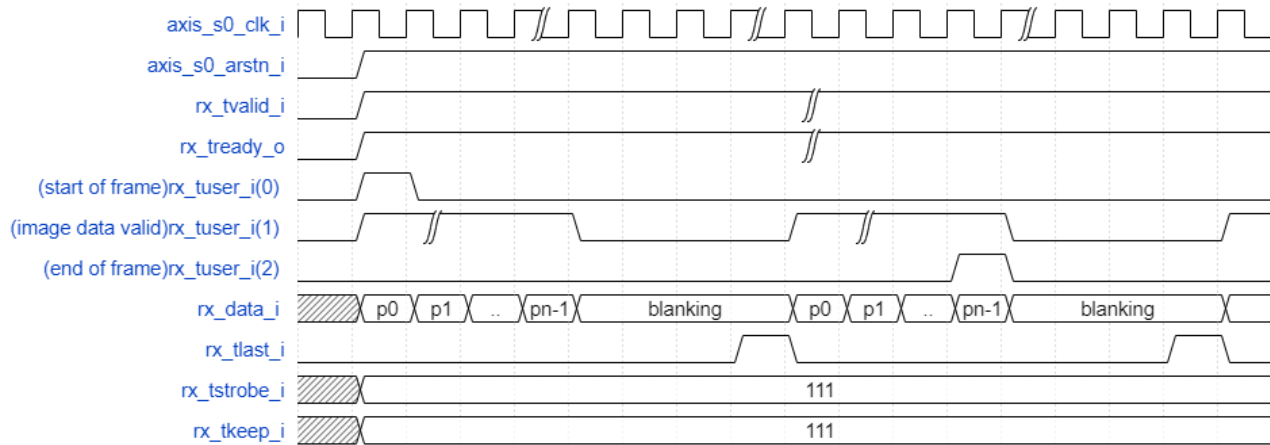


Figure 2.4. AXI-Stream Receiver Signals for PPC=1 (RGB Format)

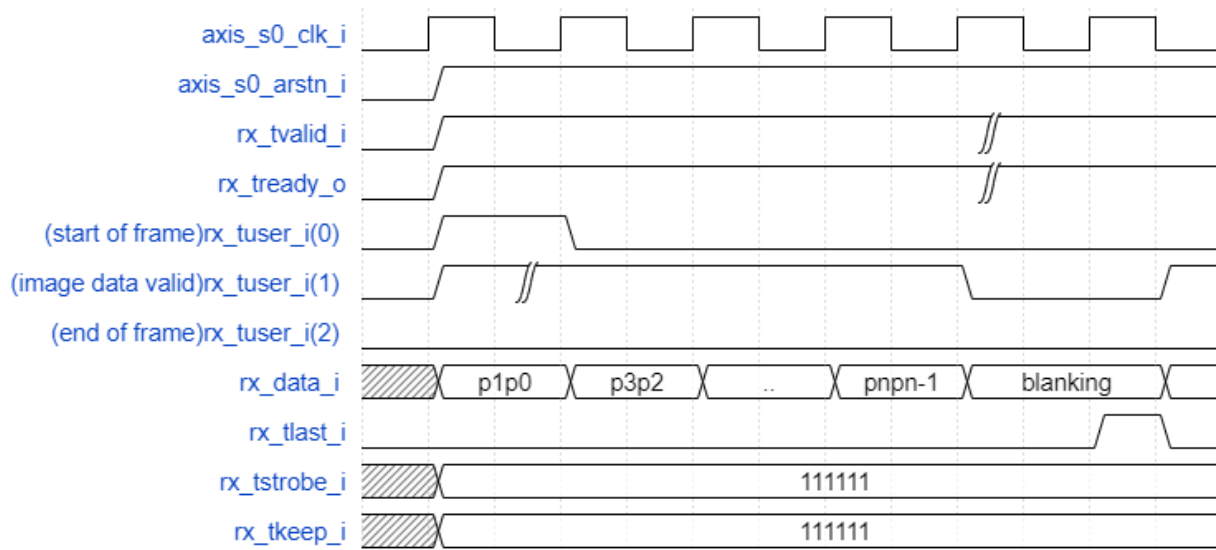


Figure 2.5. AXI-Stream Receiver Signals for PPC=2 (RGB Format)

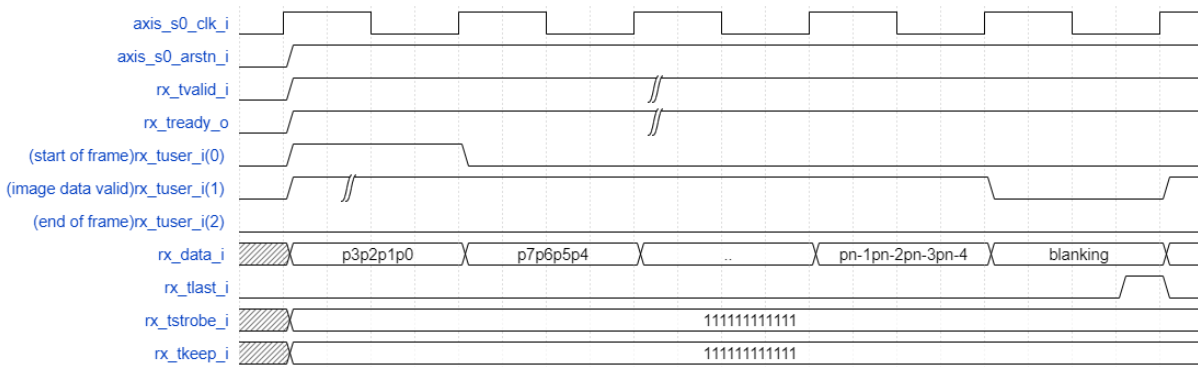


Figure 2.6. AXI-Stream Receiver Signals for PPC=4 (RGB Format)

The organization of the color component data in the rx_data_i bus is described here. The size of the rx_tdata_i bus depends on the parameters Bits Per Component (BPC) and PPC (refer to Table 4.1). These parameters determine the width of the bus, which is equal to the next higher multiple of 8 of $(PPC \times BPC \times 3)$. Pixel data of $(PPC \times BPC \times 3)$ bits is mapped to the lower part of the bus, with the remaining upper bits assigned to zeros.

The mapping of bits for the case of PPC=1 and BPC=8 is shown in Figure 2.7. The mappings for other cases are shown in Figure 2.8 to Figure 2.10.

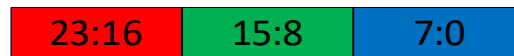


Figure 2.7. Bit Allocation for BPC=8 and PPC=1 for RGB Input



Figure 2.8. Bit Allocation for BPC=10 and PPC=1 for RGB Input



Figure 2.9. Bit Allocation for BPC=8 and PPC=2 for RGB Input

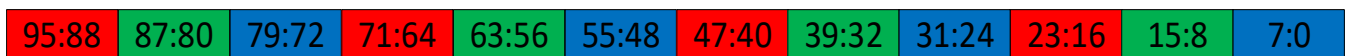


Figure 2.10. Bit Allocation for BPC=8 and PPC=4 for RGB Input

2.5. AXI-Stream Transmitter

The AXI Stream Transmitter module maps the video data to AXI-Stream data bus for transmission. The transmitter sends out the data in RGB or YCbCr format. Here, red is in the upper bits followed by green and blue. The user-defined bus tx_tuser_o is used to transmit sync signals. AXI-Stream transmitter sends out data only when tx_tready_i is high. If tx_tready_i goes low, data is held in Output FIFO.

For AXI Stream handshaking, tx_tvalid_o is active when valid output pixel data is available on tx_data_o, and tx_tuser_o (start of frame) marks the first pixel, first row of the output video frame. When the input signal tx_tready_i is asserted, the core outputs video pixels.

The AXI-Stream handshake signals and user-defined TUSER bus specifies frame boundaries and active data windows respectively. The user-defined sideband signal TUSER is used to specify the frame boundaries and data validity. Specifically, tx_tuser_o[0] indicates the start of the frame, with this bit asserting high during the first pixel of a frame. The output tx_tuser_o[1] specifies the active part of a line, with a value of 1 indicating the active part and 0, the blanking part. The output tx_tuser_o[2] indicates the end of the frame. This signal will be high during the last active pixel of a frame. The output tx_tlast_o indicates the end of a total line. This is asserted during the last pixel of the entire line. The tx_tstrobe_o

signal defines the valid bytes in a data beat, that is, the valid bytes in tx_tdata_o. The width of tx_tdata_o depends on the parameters PPC, BPC, and Video format.

For RGB and YCbCr444, the width of AXI Bus is equal to the quantity $(PPC*BPC*3)$ if it is a multiple of 8 or else the next higher multiple of 8 of the quantity $(PPC*BPC*3)$. For YCbCr422, the width of AXI Bus is equal to the quantity $(PPC*BPC*2)$ if it is a multiple of 8 or else the next higher multiple of 8 of the quantity $(PPC*BPC*2)$.

The timing diagram for AXI-Stream Transmitter is shown in Figure 2.11, Figure 2.12, and Figure 2.13.

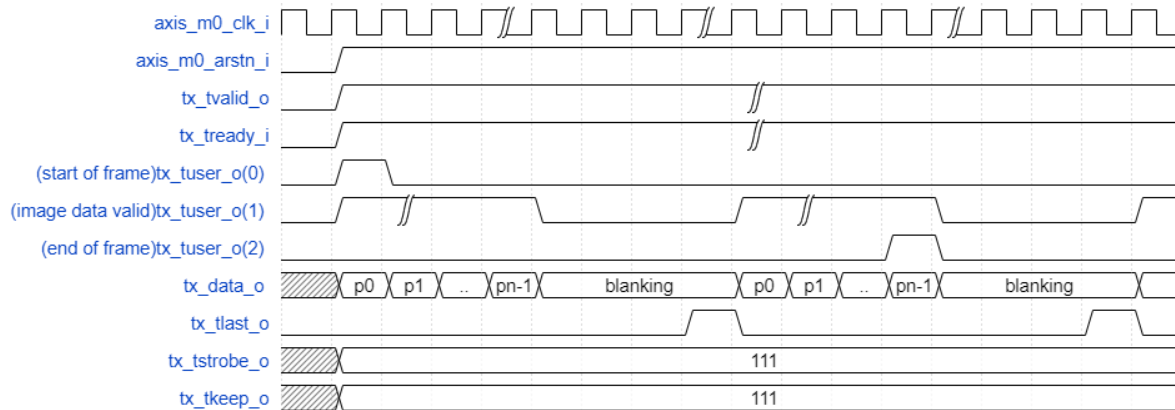


Figure 2.11. AXI-Stream Transmitter Signals for PPC=1 (RGB Format and BPC=8)

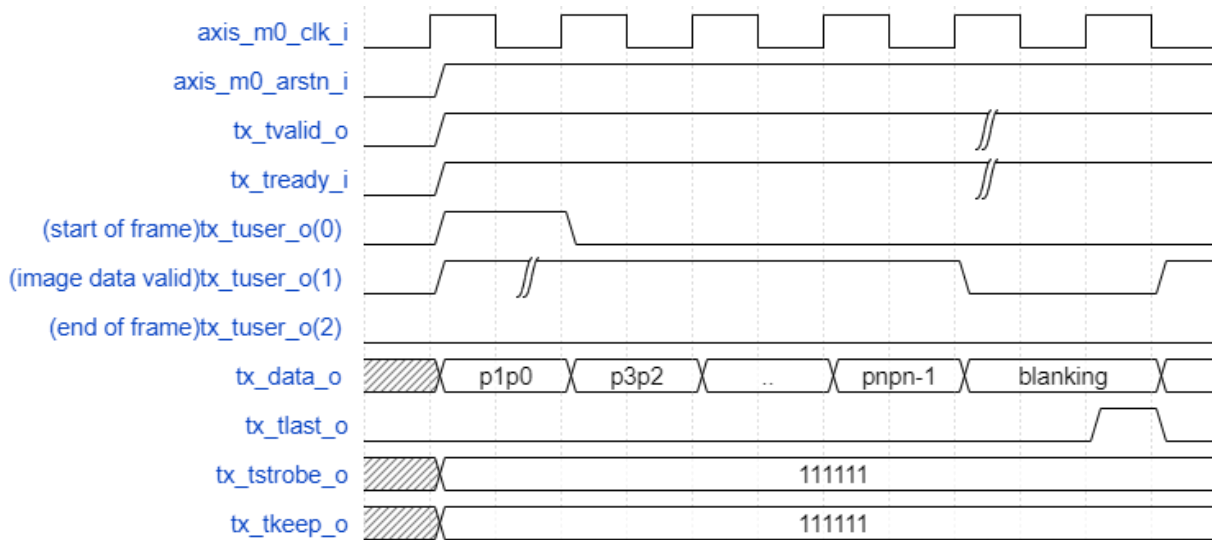


Figure 2.12. AXI-Stream Transmitter Signals for PPC=2 (RGB Format and BPC=8)

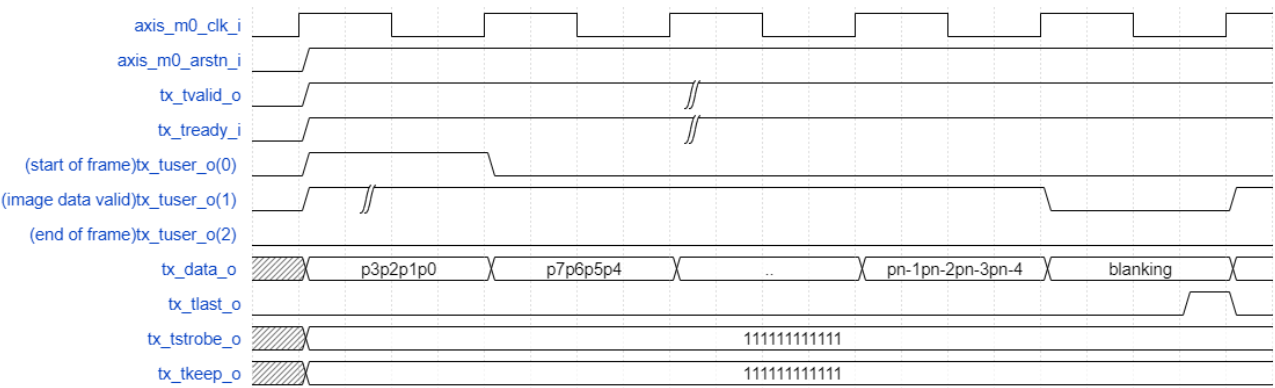


Figure 2.13. AXI-Stream Transmitter Signals for PPC=4 (RGB Format and BPC=8)

2.6. Native Video Timing Diagram

The Video Scaler IP core uses a simple handshake to pass pixel data into the core. The core asserts its `vid_ready_rx` output when it's ready to receive data. When the driving module has data to give the core, it drives the core's `vid_dvalid_i` port to a 1 synchronously with the rising edge of the `vid_clk_rx` signal, providing the input pixel data on port `vid_data_i`. The `vid_sof_i` (Start of frame) input should be driven to a 1 during the clock cycle when the first pixel of the first row in the incoming video frame is active. The `vid_hblank_i` will be driven high for horizontal blanking period after the active pixels in a row are completed. It will continue to be high for all the rows in the frame. The `vid_vblank_i` will be driven high for vertical blanking period in the frame. The timing diagram for native input video stream is shown below in Figure 2.14.

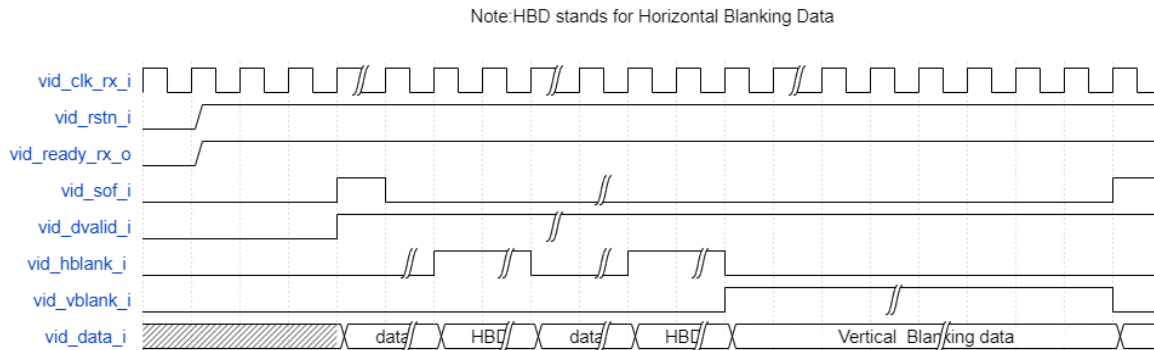


Figure 2.14. Native Input Video Timing Diagram

The Video Scaler IP core uses a simple handshake to pass pixel data out of the core. If the `vid_ready_tx` input signal is 1 and the core has data to transmit, it drives the `vid_dvalid_o` port to a 1 synchronously with the rising edge of the `vid_clk_tx` signal, providing the output pixel data on port `vid_data_o`. The `vid_sof_o` (Start of frame) output is driven to a 1 during the first clock cycle of the first row of the outgoing video frame when the first pixel of the first row in the outgoing video frame is active. The `vid_hblank_o` is driven high for horizontal blanking period after the active pixels in a row are completed and it will continue this way for all the rows in the frame. The `vid_vblank_o` is driven high for vertical blanking period in the frame. The timing diagram for native output video stream is shown below in Figure 2.15.

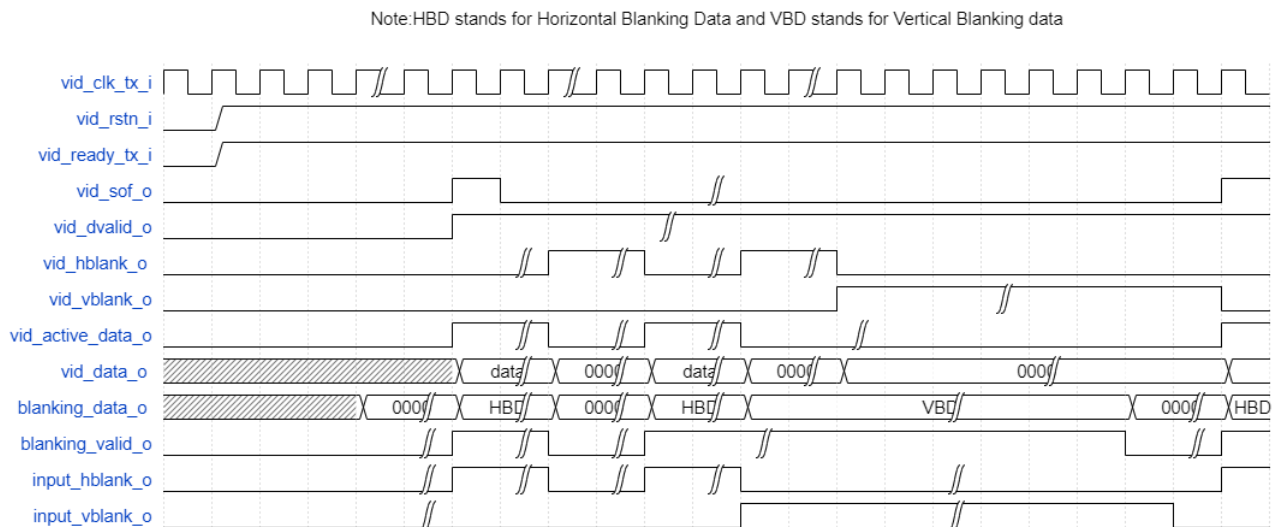


Figure 2.15. Native Output Video Timing Diagram

2.7. Blanking Pass-through

2.7.1. Blanking Pass through When Using AXI-Stream

To preserve the original blanking information without any distortion or interference, the following process is implemented: The input video blanking data is extracted from the AXI Stream and synchronized with the blanking valid signal. This ensures that the proper horizontal blanking and vertical blanking time are maintained. The resulting blanking data is then output by the IP for utilization by subsequent modules.

Figure 2.16 illustrates a timing diagram that showcases the extraction of blanking data from the input AXI Stream data. The signals `blanking_data_o`, `blanking_valid_o`, `input_hblank_o`, and `input_vblank_o` are the outputs of the Video Scaler IP, carrying the blanking data along with the corresponding timing information, indicated by the valid signal.

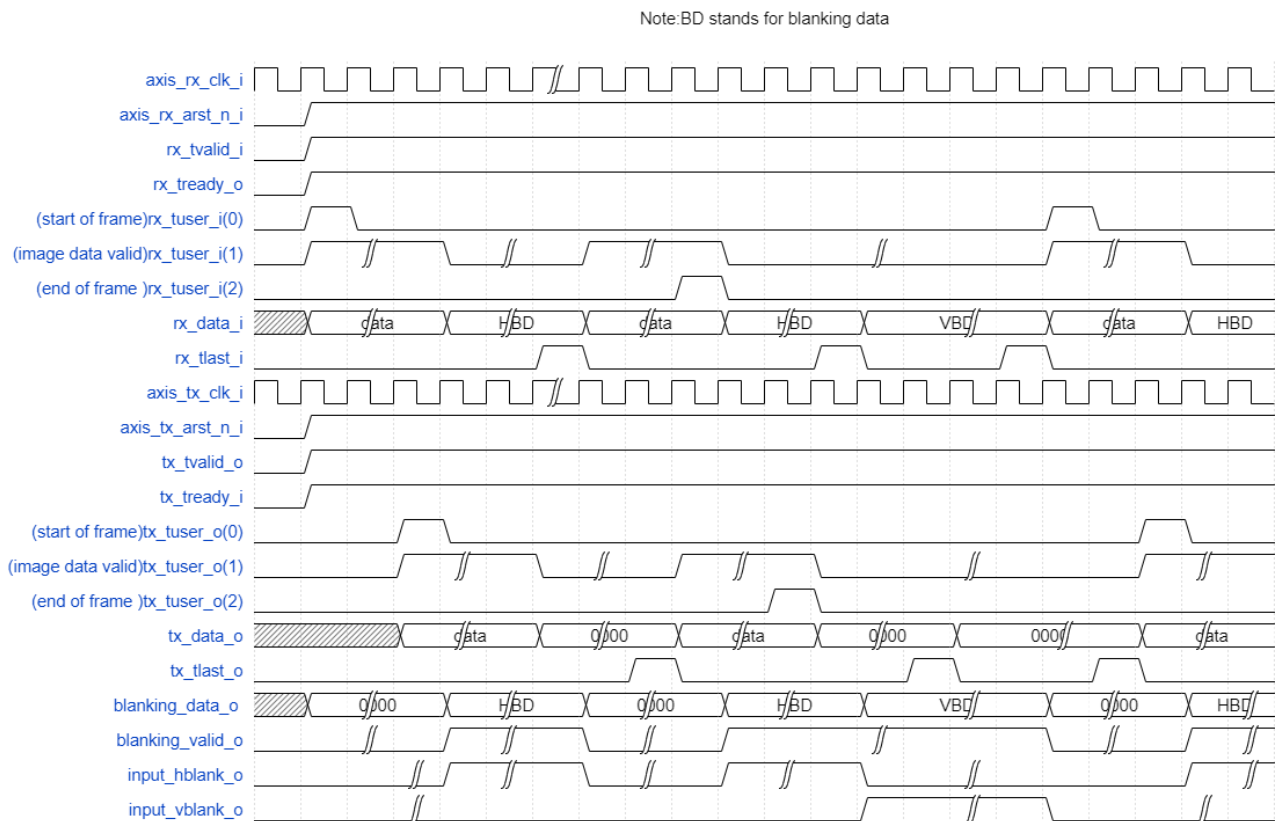


Figure 2.16. AXI-Stream Receiver with Blanking Related Signals for PPC=1(RGB Format)

2.7.2. Blanking Pass Through When Using Native Interface

To ensure the preservation of original blanking information without distortion or interference, the input video blanking data is extracted from the native data and sent out with a synchronized blanking valid signal. This synchronization allows the passage of unaltered blanking information to the output. The resulting blanking data, which includes appropriate horizontal and vertical blanking time can be utilized by the subsequent modules to extract or process information contained in the blanking data.

Figure 2.17 is a timing diagram that illustrates the extraction of blanking data from the Native Video data. The signals `blanking_data_o`, `blanking_valid_o`, `input_hblank_o`, and `input_vblank_o` are the outputs of the Video Scaler IP, carrying the blanking data along with the corresponding timing information, indicated by the valid signal.

Note: HBD stands for Horizontal Blanking Data and VBD stands for Vertical Blanking data



Figure 2.17. Native Input Video Timing Diagram with Blanking Related Signals

2.8. YCbCr Video Format Timing

The Video Scaler IP supports YCbCr 4:2:2, YCbCr 4:4:4 and RGB video formats.

For YCbCr 4:2:2 scaling, the Y plane occupies the upper bits of the rx_data_i and tx_data_o ports, and the Cb and Cr planes occupy the lower bits. Cb and Cr planes are interleaved in the lower half, and Cb comes before Cr.

Figure 2.18 shows the AXI transmitter for YCbCr 4:2:2 video format. The organization of the color component data in the rx_data_i bus is described here. The size of the rx_data_i bus depends on the parameters BPC and PPC (refer to Table 4.1). These parameters determine the width of the bus, which is equal to the next higher multiple of 8 of $(PPC \times BPC \times 3)$. Pixel data of $(PPC \times BPC \times 3)$ bits is mapped to the lower part of the bus, with the remaining upper bits assigned to zeros.

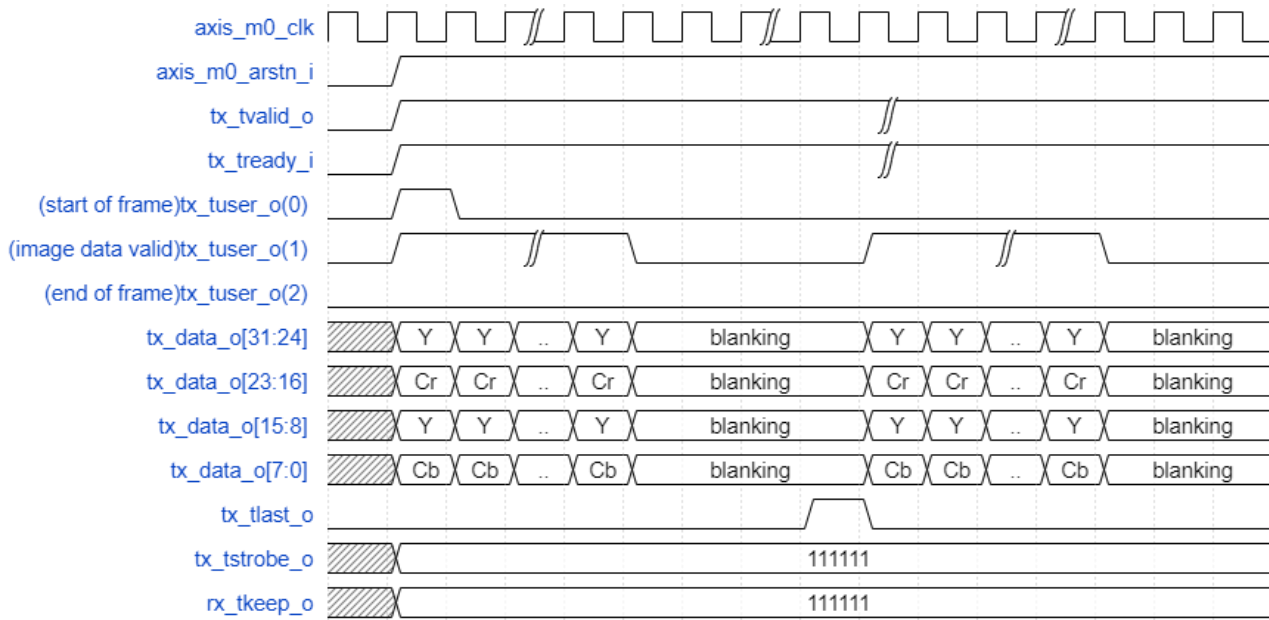


Figure 2.18. YCbCr 4:2:2 Parallel Scaling (PPC=2)

2.9. AXI-Lite Subordinate

AXI-Lite consists of five channels. Table 2.1 shows the channels and key signals in each channel.

Table 2.1. AXI-Lite Channels and Signals

Channel	Key Signals
Write address	aw_valid_i, aw_address_i, aw_ready_o
Write data	w_valid_i, w_data_i, w_ready_o
Write response	b_valid_o, b_response_o, b_ready_i
Read address	ar_valid_i, ar_address_i, ar_ready_o
Read data	r_data_o, r_valid_o, r_response_o, r_ready_i

2.9.1. Write Operation

The write operation sequence is described below.

1. The manager asserts aw_valid_i along with the address.
2. The subordinate asserts the aw_ready_i and captures the address.
3. The manager asserts w_valid_i along with the write data.
4. The subordinate asserts w_ready_i and captures the data.
5. The subordinate sends an OKAY response and asserts b_valid_o.
6. The manager asserts b_ready_i, completing the transaction.

2.9.2. Read Operation

The write operation sequence is described below.

1. The manager asserts ar_valid_i signal along with the address.
2. The subordinate captures the address by asserting the ar_ready_i.
3. The subordinate sends the r_data_o along with r_valid_o and r_response_o.
4. The manager asserts r_ready_i, completing the transaction.

3. Signal Description

3.1. Video Scaler IP with AXI Stream Interface

The widths used for the signal buses in the interface table are defined in [Table 3.1](#). The input/output interface signals for Video Scaler IP with AXI Stream Interface are indicated in [Table 3.2](#).

Table 3.1. Description of Width Parameters

Width Parameter Name	Description
AXI_STREAM_DATA_WIDTH	Width of AXI-Stream bus is equal to the next multiple of 8 of the quantity $(PPC \times BPC \times 3)$ for RGB and YCbCr444 format. And for YcbCr422, it is the next multiple of 8 $(PPC \times BPC \times 2)$.
NATIVE_DATA_WIDTH	Width of Native bus is equal to the product of $(PPC \times BPC \times 3)$ for RGB and YcbCr444 format. And for YcbCr422, it is equal to the product of $(PPC \times BPC \times 2)$.
AXI_LITE_ADDR_WIDTH	32
AXI_LITE_DATA_WIDTH	32

Table 3.2. Video Scaler IP Signal Description

Port Name	I/O	Width	Default	Description
Blanking Pass-through Signals				
frm_invalid_o	Out	1	0	This signal is asserted high if the SOF of current frame is not aligned to the first pixel data. There will be no output from the IP when this is asserted high. It will be de-asserted on the next frame's SOF if it is correctly aligned.
blanking_data_o	Out	[AXI_STREAM_DATA_WIDTH-1:0]	{{AXI_STREAM_DATA_WIDTH {1'b0}}}	Input data received during blanking period.
blanking_valid_o	Out	1	0	This signal asserts high when valid blanking data passed to output.
input_vblank_o	Out	1	0	Input Frame vertical blanking period.
input_hblank_o	Out	1	0	Input Frame horizontal blanking period.
Clock and Reset				
axis_s0_arst_n_i	In	1	N/A	axis_s0_arst_n_i is an active low global reset signal.
axis_m0_arst_n_i	In	1	N/A	axis_m0_arst_n_i is an active low global reset signal.
axi_lite_rst_n_i	In	1	N/A	axi_lite_rst_n_i is an active low global reset signal.
axis_s0_clk_i	In	1	N/A	axis_s0_clk_i is the global clock signal. All signals are sampled on the rising edge of axis_s0_clk_i.
axis_m0_clk_i	In	1	N/A	AXI stream TX clock.
axi_lite_clk_i	In	1	N/A	AXI-Lite clock
core_clk_i	In	1	N/A	If dynamic reconfiguration is enabled, core_clk_i needs to be connected to the faster clock depending on whether the IP is configured for upscaling or downscaling
AXI-Stream Rx				
rx_data_i	In	[AXI_STREAM_DATA_WIDTH-1:0]	N/A	rx_data_i is the primary payload that is used to provide the data that is passing across the interface.

Port Name	I/O	Width	Default	Description
rx_tuser_i	In	3	N/A	Sideband information transmitted alongside the data-stream. <ul style="list-style-type: none"> rx_tuser_i[0] – start of frame. rx_tuser_i[1] – active image data. rx_tuser_i[2] – end of frame.
rx_tlast_i	In	1	N/A	rx_tlast_i indicates the end of line for each horizontal line.
rx_tvalid_i	In	1	N/A	rx_tvalid_i indicates that the Rx is driving a valid transfer. Atransfer takes place when both TVALID and TREADY are asserted.
rx_tready_o	Out	1	0	rx_tready_o indicates that the Rx can accept a transfer in the current cycle.
rx_tstrobe_i	In	[(AXI_STREAM_DATA_WIDTH/8)-1:0]	N/A	rx_tstrobe_i is the byte qualifier that indicates whether the content of the associated byte of axi_data_i is processed as a data byte or position byte.
rx_tkeep_i	In	[(AXI_STREAM_DATA_WIDTH/8)-1:0]	N/A	rx_tkeep_i is the byte qualifier that indicates whether the content of the associated bytes of rx_tkeep_i is processed as part of the data stream. Associated bytes that have the rx_tkeep_i byte qualifier deasserted are null bytes and can be removed from the data stream.
AXI-Stream Tx				
tx_tvalid_o	Out	1	0	tx_tvalid_o indicates that the Tx is driving a valid transfer. Atransfer takes place when both TVALID and TREADY are asserted.
tx_tready_i	In	1	N/A	tx_tready_i indicates that the Tx can accept a transfer in the current cycle.
tx_data_o	Out	[AXI_STREAM_DATA_WIDTH-1:0]	{AXI_STREAM_DATA_WIDTH{1'b0}}	tx_data_o is the primary payload that is used to provide the data that is passing across the interface.
tx_tuser_o	Out	3	3'b000	Sideband information transmitted alongside the data stream. <ul style="list-style-type: none"> tx_tuser_o[0] - start of frame. tx_tuser_o[1] - valid image data. tx_tuser_o[2] - end of frame.
tx_tlast_o	Out	1	0	tx_tlast_o indicates the end of line for each horizontal line.
tx_tstrobe_o	Out	[(AXI_STREAM_DATA_WIDTH/8)-1:0]	{(AXI_STREAM_DATA_WIDTH/8) {1'b0}}	tx_tstrobe_o is the byte qualifier that indicates whether the content of the associated byte of rx_data_i is processed as a data byte or position byte.
tx_tkeep_o	Out	[(AXI_STREAM_DATA_WIDTH/8)-1:0]	{(AXI_STREAM_DATA_WIDTH/8) {1'b0}}	tx_tkeep_o is the byte qualifier that indicates whether the content of the associated bytes of tx_data_o is processed as part of the data stream. Associated bytes that have the tx_tkeep_o byte qualifier de-asserted are null bytes and can be removed from the data stream.
AXI-Lite Subordinate				
aw_valid_i	In	1	N/A	Write address valid. This signal indicates that the channel is signaling valid write address and control information.
aw_address_i	In	[AXI_LITE_ADDR_WIDTH-1:0]	N/A	Write address.

Port Name	I/O	Width	Default	Description
aw_ready_o	Out	1	0	Write address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals.
w_valid_i	In	1	N/A	Write valid. This signal indicates valid write data.
w_data_i	In	[AXI_LITE_DATA_WIDTH-1:0]	N/A	Write data.
w_ready_o	Out	1	0	Write ready. This signal indicates that the subordinate can accept the write data.
b_valid_o	Out	1	0	Write response valid. This signal indicates that the channel is signaling a valid write response.
b_response_o	Out	2	2'b00	Write response. This signal indicates the status of the write transaction
b_ready_i	In	1	N/A	Response ready. This signal indicates that the manager can accept a write response.
ar_valid_i	In	1	N/A	Read address valid. This signal indicates that the channel is signaling valid read address.
ar_address_i	In	[AXI_LITE_ADDR_WIDTH-1:0]	N/A	Read address.
ar_ready_o	Out	1	0	Read address ready. This signal indicates that the subordinate is ready to accept an address.
r_data_o	Out	[AXI_LITE_DATA_WIDTH-1:0]	{[AXI_LITE_DATA_WIDTH-1:0]}	Read data.
r_valid_o	Out	1	0	Read valid. This signal indicates that the channel is signaling the required read data.
r_response_o	Out	2	2'b00	Read response. This signal indicates the status of the read transfer.
r_ready_i	In	1	N/A	Read ready. This signal indicates that the Manager can accept the read data and response information.
ar_prot_i	In	3	N/A	Protection type. This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
aw_prot_i	In	3	N/A	Protection type. This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
w_strb_i	In	4	N/A	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.

3.2. Video Scaler IP with Native Interface

The input/output interface signals for Video Scaler IP with Native Interface are indicated in [Table 3.3](#).

Table 3.3. Video Scaler IP Native Interface I/O

Port Name	I/O	Width	Default	Description
vid_sof_i	In	1	N/A	Indicates Start of frame for the video input.
vid_clk_rx_i	In	1	N/A	vid_clk_rx is the global clock signal. All input signals are sampled on the rising edge of vid_clk_rx.
vid_clk_tx_i	In	1	N/A	vid_clk_tx is the global clock signal. All output signals are sampled on the rising edge of vid_clk_tx.

Port Name	I/O	Width	Default	Description
vid_data_i	In	[NATIVE_DATA_WIDTH-1:0]	N/A	vid_data_i is the primary payload that is used to provide the data that is passing across the interface.
vid_active_data_i	In	1	N/A	vid_active_data_i indicates that the active pixel data receive over the native stream.
vid_dvalid_i	In	1	N/A	vid_dvalid_i indicates that the Rx is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.
vid_hblank_i	In	1	N/A	vid_hblank_i indicates timing for line blanking period of input video stream.
vid_ready_rx_o	Out	1	0	vid_ready_rx indicates that the Rx can accept a transfer in the current cycle.
vid_rstn_i	In	1	N/A	vid_rstn is an active low global reset signal.
vid_vblank_i	In	1	N/A	vid_vblank_i indicates timing for frame blanking period of input video stream.
vid_sof_o	Out	1	0	Indicates Start of frame for the video output.
vid_data_o	Out	[NATIVE_DATA_WIDTH-1:0]	{NATIVE_DATA_WIDTH{1'b0}}	vid_data_o is the primary payload that is used to provide the data that is passing across the interface.
vid_dvalid_o	Out	1	0	vid_dvalid_o indicates that the Tx is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.
vid_active_data_o	Out	1	0	This signal asserts when the active pixel data transmitted over the native stream.
vid_ready_tx_i	In	1	N/A	vid_ready_tx indicates that the Tx can accept a transfer in the current cycle.
vid_hblank_o	Out	1	0	vid_hblank_o indicates timing for line blanking period of input video stream.
vid_vblank_o	Out	1	0	vid_vblank_o indicates timing for line blanking period of input video stream.
blanking_data_o	Out	[NATIVE_DATA_WIDTH-1:0]	[NATIVE_DATA_WIDTH{1'b0}]	Input data received during blanking period.
blanking_valid_o	Out	1	0	This signal asserts high when valid blanking data passed to output.
input_vblank_o	Out	1	0	Input Frame vertical blanking period.
input_hblank_o	Out	1	0	Input Frame horizontal blanking period.

4. Parameter Settings & Attributes Summary

The configurable attributes of the Video Scaler IP are shown and described in this section. Video Scaler IP GUI has two sections, Architecture and Implementation. The attributes can be configured through the IP Catalog's Module/IP wizard of the Lattice Propel Builder software or Lattice Radiant software.

4.1. Architecture Section

Figure 4.1 shows the sample user interface of architecture tab of the Video scaler IP and the attributes set through the user interface are described in Table 4.1 and Table 4.2.

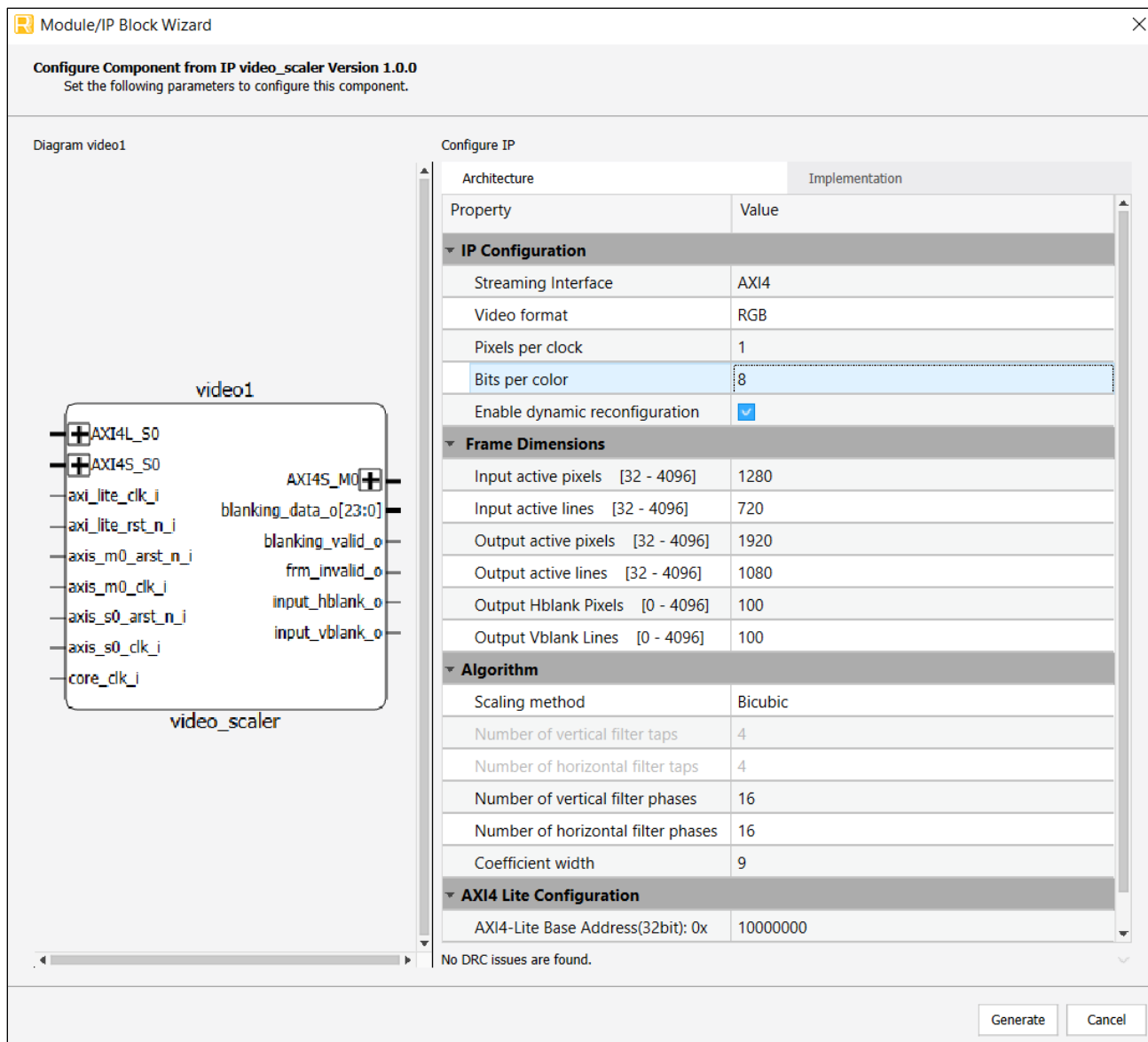


Figure 4.1. Architecture Tab

Table 4.1. Attributes Table for Architecture Tab

Attribute	Selectable Values	Default	Dependency on other attributes
Architecture			
IP Configuration			
Streaming Interface	AXI4, Native	AXI4	...
Video format	RGB, YCbCr4:4:4, YCbCr4:2:2	RGB	...
Pixels per clock	1,2,4	1	...
Bits per color	8,10,12,16	8	...
Enable Dynamic Reconfiguration	Checked, Unchecked	Checked	...
Frame Dimensions			
Input active Pixels	32-4096	1280	...
Input active lines	32-4096	720	...
Output active pixels	32-4096	1920	...
Output active lines	32-4096	1080	...
Output Hblank Pixels	0-4096	100	...
Output Vblank lines	0-4096	100	...
Algorithm			
Scaling Method	Nearest Neighbor, Bilinear, Bicubic, Lanczos	Bicubic	...
Number of vertical filter taps	4-12(Lanczos)	4	1. Editable only when Scaling Method==Lanczos. 2. Fixed to 4 when Scaling Method==Bicubic. 3. Fixed to 2 when Scaling Method==Bilinear. 4. Fixed to 1 when Scaling Method==Nearest.
Number of horizontal filter taps	4-12(Lanczos)	4	1. Editable only when Scaling Method==Lanczos. 2. Fixed to 4 when Scaling Method==Bicubic. 3. Fixed to 2 when Scaling Method==Bilinear. 4. Fixed to 1 when Scaling Method==Nearest.
Number of vertical filter phases	16,32,64,128,256,512	16	1. Editable only when Scaling Method == Lanczos or Bicubic. 2. Fixed to 16 when Scaling Method ==Bilinear or Nearest.
Number of horizontal filter phases	16,32,64,128,256,512	16	1. Editable only when Scaling Method == Lanczos or Bicubic. 2. Fixed to 16 when Scaling Method ==Bilinear or Nearest.
Coefficient width	6-18	9	...
AXI Lite Configuration			
AXI Lite Base Address (32bits)	N/A	0x10000000	...

Table 4.2. Attributes Description for Architecture Tab

Attribute	Description
IP Configuration	
Streaming Interface	Select the streaming interface type whether AXI Stream Interface or Native video interface.
Video format	Defines the format of a video stream. It can be RGB, YCbCr4:4:4, YCbCr4:2:2.
Pixels per clock	Defines the number of pixels per clock used for processing.
Bits per color	Sets the bit width of the incoming pixel values and may vary between 8 and 16, inclusive.
Enable Dynamic Reconfiguration	Checkbox determines whether the core supports Dynamic Scaling.
Frame Dimensions	
Input active Pixels	Defines the input video frame width for fixed scaling. When dynamic reconfiguration is enabled, it will be the maximum configurable "input active pixels" over the AXI-Lite interface.
Input active lines	Defines the input video frame height for fixed scaling. When dynamic reconfiguration is enabled, it will be the maximum configurable "input active lines" over the AXI-Lite interface.
Output active pixels	Defines the output video frame width for fixed scaling. When dynamic reconfiguration is enabled, it will be the maximum configurable "output active pixels" over the AXI-Lite interface.
Output active lines	Defines the output video frame height for fixed scaling. When dynamic reconfiguration is enabled, it will be the maximum configurable "output active lines" over the AXI-Lite interface.
Output Hblank Pixels	Defines the output video frame Hblank width for fixed scaling. When dynamic reconfiguration is enabled, it will be the maximum configurable "output hblank pixels" over the AXI-Lite interface.
Output Vblank Lines	Defines the output video frame Vblank width for fixed scaling. When dynamic reconfiguration is enabled, it will be the maximum configurable "output vblank lines" over the AXI-Lite interface.
Algorithm	
Scaling Method	It selects the scaling algorithm of the core.
Number of Vertical Filter Taps	Number of taps for the vertical filter. This number varies between 4 and 12 only for the Lanczos.
Number of Horizontal Filter Taps	Number of taps for the horizontal filter. This number varies between 4 and 12 only for the Lanczos.
Number of Vertical Filter Phases	It is a power of 2 number which provides the setting for the number of phases of the vertical filter and may vary between 16 and 512. Higher number of vertical phases results in larger vertical coefficient storage.
Number of Horizontal Filter Phases	It is a power of 2 number which provides the setting for the number of phases of the horizontal filter and may vary between 16 and 512. Higher number of horizontal phases results in larger horizontal coefficient storage.
Coefficient width	Set the bit width of the coefficients to any number between 6 and 18 inclusive.
AXI Lite Configuration	
AXI Lite Base Address (Hex)	Sets the base address of AXI Lite in hexadecimal format.

4.2. Implementation Section

Figure 4.2 shows the sample user interface of architecture tab of the Video scaler IP and the attributes set through the user interface are described in Table 4.3 and Table 4.4.

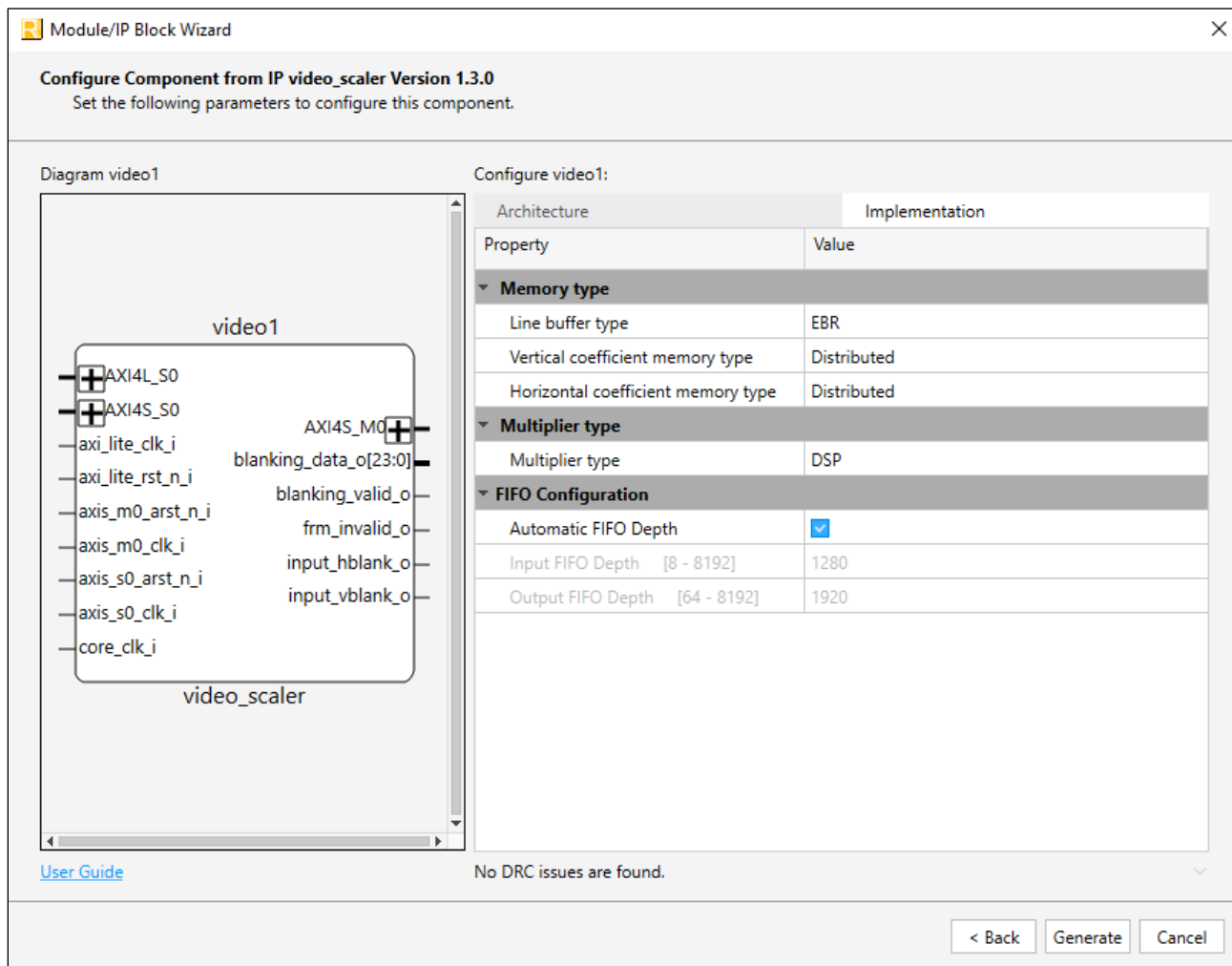


Figure 4.2. Implementation Tab

Table 4.3. Attributes Table for Implementation Tab

Attribute	Selectable Values	Default	Dependency on other attribute
Implementation			
Memory Type			
Line buffer type	EBR, Distributed	EBR
Vertical coefficient memory type	EBR, Distributed	Distributed	Editable only when Scaling Method==Lanczos or Bicubic.
Horizontal coefficient memory type	EBR, Distributed	Distributed	Editable only when Scaling Method==Lanczos or Bicubic.
Multiplier type			
Multiplier type	DSP, LUT	DSP
FIFO Configuration			
Automatic FIFO Depth	Checked, Unchecked	Checked

Attribute	Selectable Values	Default	Dependency on other attribute
Input FIFO Depth	8-8192	720
Output FIFO Depth	64-8192	1280

Table 4.4. Attributes Description for Implementation Tab

Attribute	Descriptions
Memory type	
Line buffer type	Selects memory type for the line buffer implementation.
Vertical coefficient memory type	Selects memory type for the vertical coefficient memory.
Horizontal coefficient memory type	Selects memory type for the horizontal coefficient memory.
Multiplier type	
Multiplier type	Selects the multiplier type to be used on scaling.
FIFO Configuration	
Automatic FIFO Depth	Determines FIFO depth automatically based on frame dimensions and Pixel per clock. Input FIFO depth is equal to Input frame Width divided by Pixel per clock and Output FIFO depth is equal to Output frame width divided by Pixel per clock. Output FIFO depth cannot be less than 64. The FIFO depth can be manually entered if the user wants more buffering to allow for a uniform input and output throughput without any dips in the rx_tready_o/ vid_ready_rx_o/tx_tvalid_o/ native_dvalid_o signals or the depth can be decreased from the default values to optimize space.
Input FIFO Depth	User configuration depth for Input FIFO.
Output FIFO Depth	User configuration depth for Output FIFO.

5. Register Description

The Video Scaler IP core supports in-system updates of input and output frame sizes through AXI-Lite interface access. The CSR registers are listed below [Table 5.1](#).

Table 5.1. Summary of Configuration and Status Registers

Offset	Register Name	Access	Default Value	Register Description	
				Bit	Field Description
0x0000	FRMWIDTH	RW	"Input frame width"	[31:0]	Input frame width register – The FRMWIDTH value must be the input frame width. The minimum value is 32, and the maximum value is the maximum input frame width specified on the IP user interface. The default value is the maximum value. Input framewidth must be an even number for YCbCr4:2:2 format.
0x0004	FRMHEIGHT	RW	"Input frame height"	[31:0]	Input frame height register – The FRMHEIGHT must be the input frame height. The minimum value is 32, and the maximum value is the maximum input frame height specified on the IP user interface. The default value is the maximum value.
0x0008	OUTWIDTH	RW	"Output frame width"	[31:0]	Output frame width register – The OUTWIDTH must be the output frame width. The minimum value is 32, and the maximum value is the maximum output frame width specified on the IP user interface. The default value is the maximum value. Output frame width must be an even number for YCbCr4:2:2 format.
0x000C	OUTHEIGHT	RW	"Output frame height"	[31:0]	Output frame height register – The OUTHEIGHT must be the output frame height. The minimum value is 32, and the maximum value is the maximum output frame height specified on the IP user interface. The default value is the maximum value.
0x0010	VSFACTOR	RW	Calculated value for GUI entered input and output resolution	[31:0]	Vertical scaling factor register – $VSFACTOR = ((FRMHEIGHT) * (1 << VFCBPWIDTH)) / OUTHEIGHT$. Where $VFCBPWIDTH = \max(\log_2(\text{maximum output frame height}), \log_2(\text{number of vertical filter phases}))$.
0x0014	HSFACTOR	RW	Calculated value for GUI entered input and output resolution	[31:0]	Horizontal scaling factor register – $HSFACTOR = ((FRMWIDTH) * (1 << HFCBPWIDTH)) / OUTWIDTH$. Where $HFCBPWIDTH = \max(\log_2(\text{maximum output framewidth}), \log_2(\text{number of the horizontal filter phases}))$.
0x0018	HBLANKOUT	RW	"Output Hblank pixels"	[31:0]	Output HBLANK timing duration : in terms of number of pixels.
0x001C	VBLANKOUT	RW	"Output Vblank lines"	[31:0]	Output VBLANK timing duration : in terms of number of vertical lines.
0x0020	UPDATE	RW	0	[31:0]	Update parameters enable register – When the writing of the parameter registers is complete, you should set this register to 1 to enable the parameter's status. The default value is 0. When the core updates the new parameters, it resets this register to 0.

The parameter registers can be written only when the UPDATE register bit is 0. When the UPDATE bit is set to 1, the eight parameters inside the core are updated with the new values when the Start of frame (SOF) signal is active, indicating a new input frame is arriving. After updating its internal parameters, the core resets the UPDATE bit to 0 to indicate that the parameter registers are now empty and can take on new values.

When dynamic parameter updating is enabled, you need to configure the largest input and output frame sizes the system expects to handle. This is so that the line buffer and various counters within the core can be configured properly. The core uses the default values for the input and output frame sizes to start until the driving block updates the parameters in the subsequent frames. The default values are the maximum input frame size and the maximum output frame size.

For most cases, VFCBPWIDTH equals to \log_2 (output frame height) for fixed scaler and \log_2 (maximum output frame height) for dynamic scaler. HFCBPWIDTH equals \log_2 (output frame width) for fixed scaler and \log_2 (maximum output frame width) for dynamic scaler. When the number of vertical phases is greater than the maximum output frame height, the VFCBPWIDTH equals \log_2 (number of the vertical filter phases). When the number of horizontal phases is greater than the maximum output frame width, the HFCBPWIDTH equals \log_2 (number of horizontal filter phases).

For the nearest neighbor and bilinear kernels, when the VFCBPWIDTH is smaller than the vertical coefficient bit width, its value should be replaced by the vertical coefficient bit width. Similarly, when the HFCBPWIDTH is smaller than the horizontal coefficient bit width, its value should be replaced by the horizontal coefficient bit width. The values of VFCBPWIDTH and HFCBPWIDTH can be found in the generated parameter value file: <project_dir>\<ip_inst_name>\testbench\dut_params.v.

6. IP Core Generation, Simulation, and Validation

This section provides information on how to generate the Video Scaler IP using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant software user guide.

6.1. Licensing the IP Core

An IP core-specific license string is required to enable full use of the Video Scaler IP in a complete, top-level design. The user can fully evaluate the IP core through functional simulation and implementation (synthesis, map, place, and route) without an IP license string. This IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core, which operate in hardware for a limited time (approximately four hours) without requiring an IP license string.

6.2. Generation and Synthesis

The Lattice Radiant software allows the user to customize and generate modules and IPs and integrate them into device's architecture. The procedure for generating the Video Scaler IP in Lattice Radiant software is described below.

To generate the Video Scaler IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the IP Catalog tab, double-click on **Video Scaler** under the **Audio_Video_and_Image_Processing** category.
 - a. The **Module/IP Block Wizard** opens as shown in [Figure 6.1](#).
3. Enter values in the **Component name** and the **Create in** fields.
4. Click **Next**.

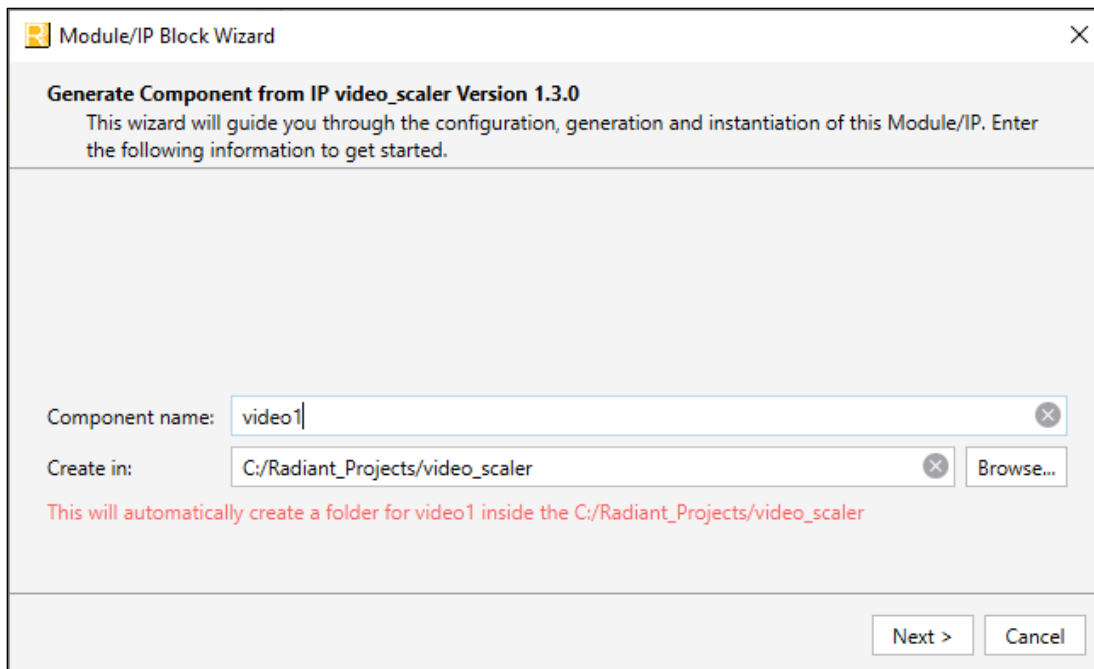


Figure 6.1. Module/IP Block Wizard

5. In the **Module/IP Block Wizard** page, customize the selected Video Scaler IP. As a sample configuration, see [Figure 6.2](#). For configuration options, see [Table 4.1](#) and [Table 4.2](#).

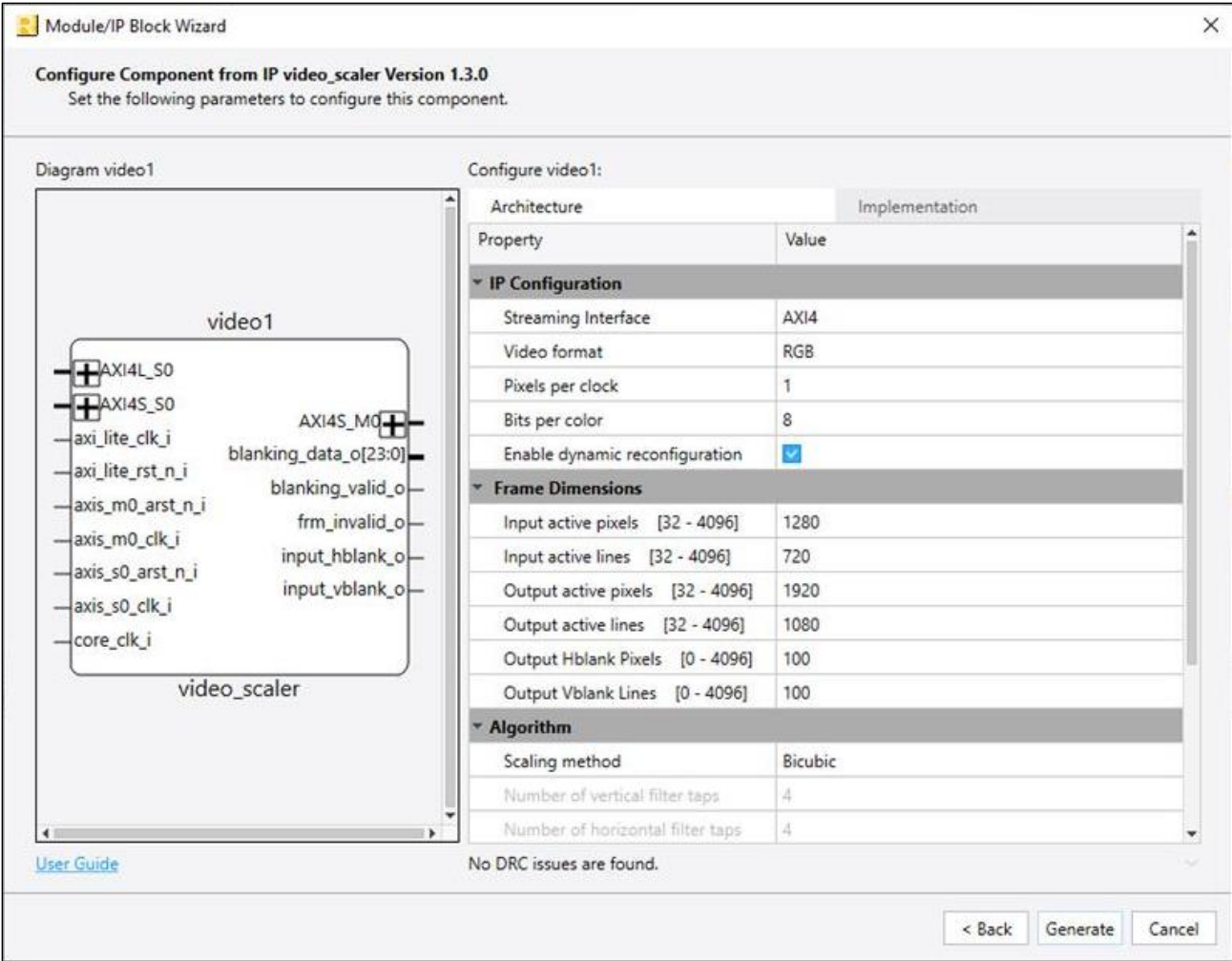


Figure 6.2. Configuration User Interface for Video Scaler IP

6. Click **Generate**. The **Check Generated Result** page opens, showing design block messages and results as shown in Figure 6.3.

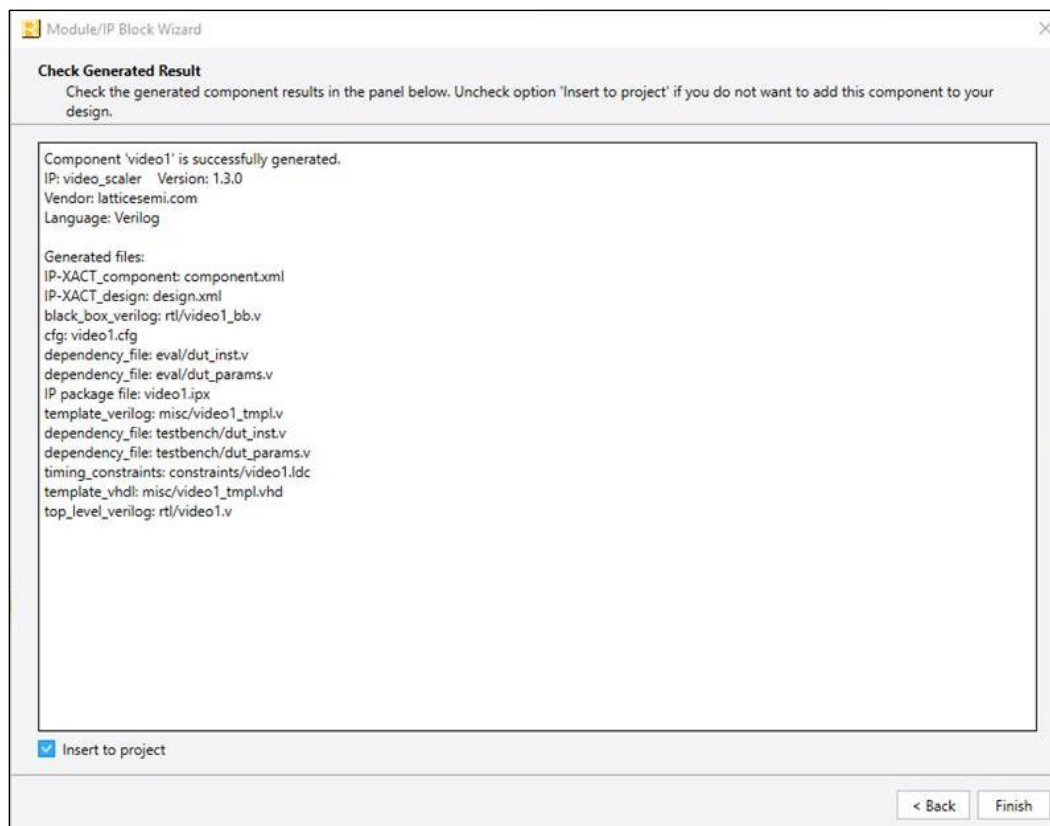


Figure 6.3. Check Generating Result

- Click **Finish**. All the generated files are placed under the directory paths in the Create in and the **Component name** fields are shown in Figure 6.1.


The generated Video Scaler IP package includes the closed-box (<Component name>_bb.v). An example Register Transfer Level (RTL) top-level reference source file (<Component name>.v) that can be used as an instantiation template for the IP core is also provided. This top-level reference may also be used as the starting template for the top-level complete design. The generated files are listed in Table 6.1.

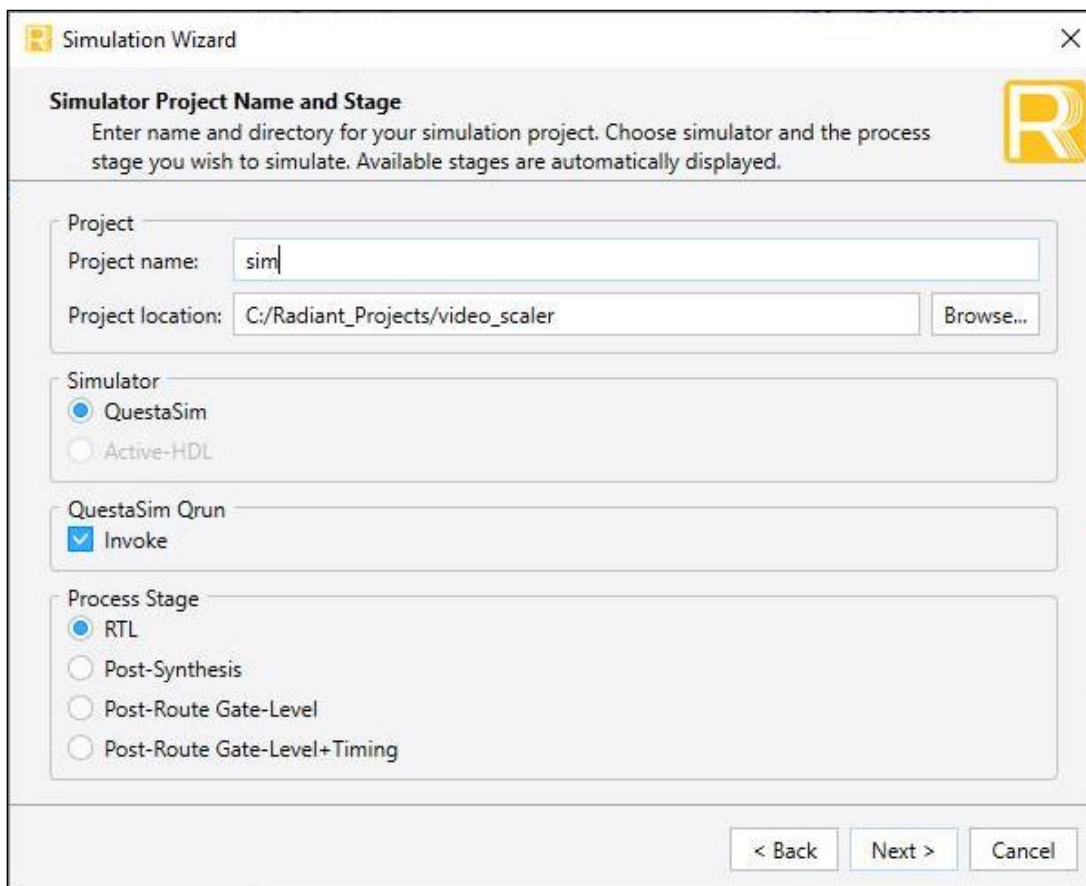
Table 6.1. Generated Files List

Attribute	Description
<Component name>.ipx	Contains the information on the files associated with the generated IP.
<Component name>.cfg	Contains the parameter values used in IP configuration.
component.xml	Contains the ipxact: component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	Provides an example RTL top file that instantiates the IP core.
rtl/<Component name>_bb.v	Provides the synthesis closed-box.
testbench/input_file_<Video format>	Input_file for RTL simulation. Where <Video format> is red, green and blue for RGB format and Y, Cb and Cr for YCbCr format.
testbench/golden_file_<Video format>	Golden file is the reference file for comparison with output rtl files. Where <Video format> is the red, green, blue for RGB format and Y, Cb and Cr for YCbCr format.
input.bmp	Input image for simulation.
golden.bmp	Scaled image output.

6.3. Running Functional Simulation

Running functional simulation can be performed after the IP is generated. The following steps should be taken.

1. Click the  button located on the toolbar to initiate the **Simulation Wizard** shown in [Figure 6.4](#).



The **Simulation Wizard** dialog box is shown. It has a title bar with a close button. The main area is titled **Simulator Project Name and Stage** and includes a description: "Enter name and directory for your simulation project. Choose simulator and the process stage you wish to simulate. Available stages are automatically displayed." There is a large 'R' logo in the top right corner of the main area.

The dialog is divided into several sections:

- Project**:
 - Project name:** A text field containing "sim".
 - Project location:** A text field containing "C:/Radiant_Projects/video_scaler" and a **Browse...** button.
- Simulator**:
 - ☒ **QuestaSim**
 - ☐ **Active-HDL**
- QuestaSim Qrun**:
 - ☒ **Invoke**
- Process Stage**:
 - ☒ **RTL**
 - ☐ **Post-Synthesis**
 - ☐ **Post-Route Gate-Level**
 - ☐ **Post-Route Gate-Level+Timing**

At the bottom right, there are three buttons: **< Back**, **Next >**, and **Cancel**.

Figure 6.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** page as shown in [Figure 6.5](#).

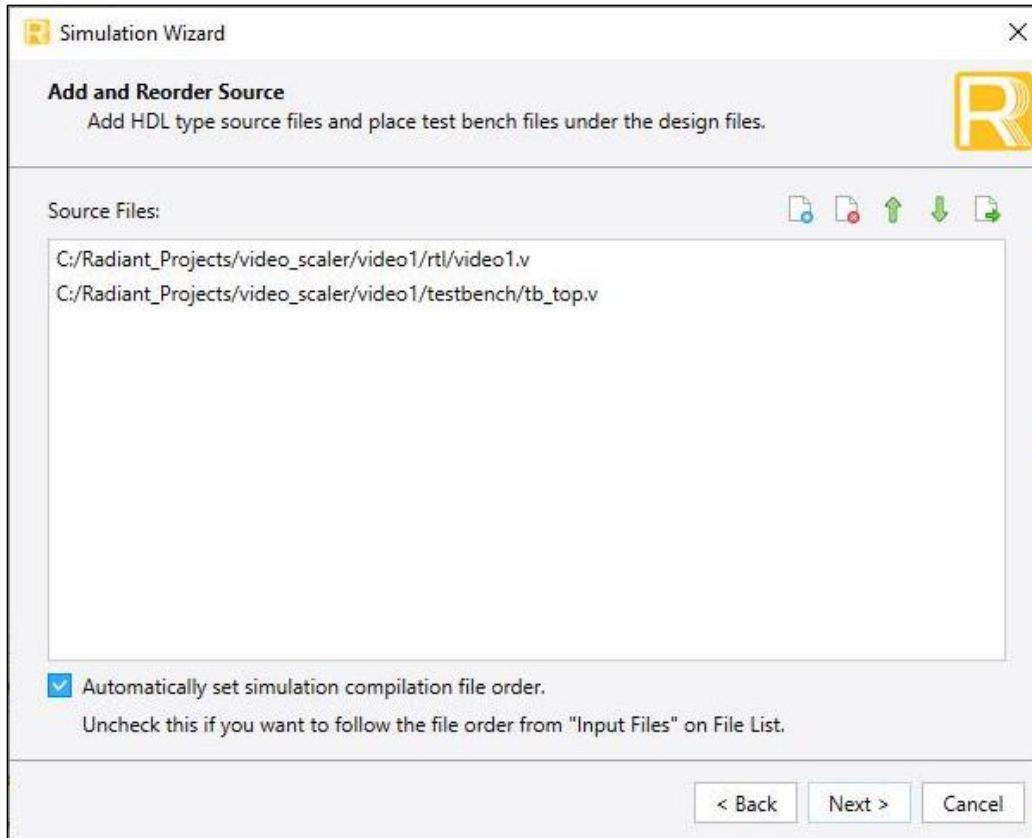


Figure 6.5. Adding and Reordering Source

3. Change the simulation time to **0 ns**, which means run -all. See [Figure 6.6](#).

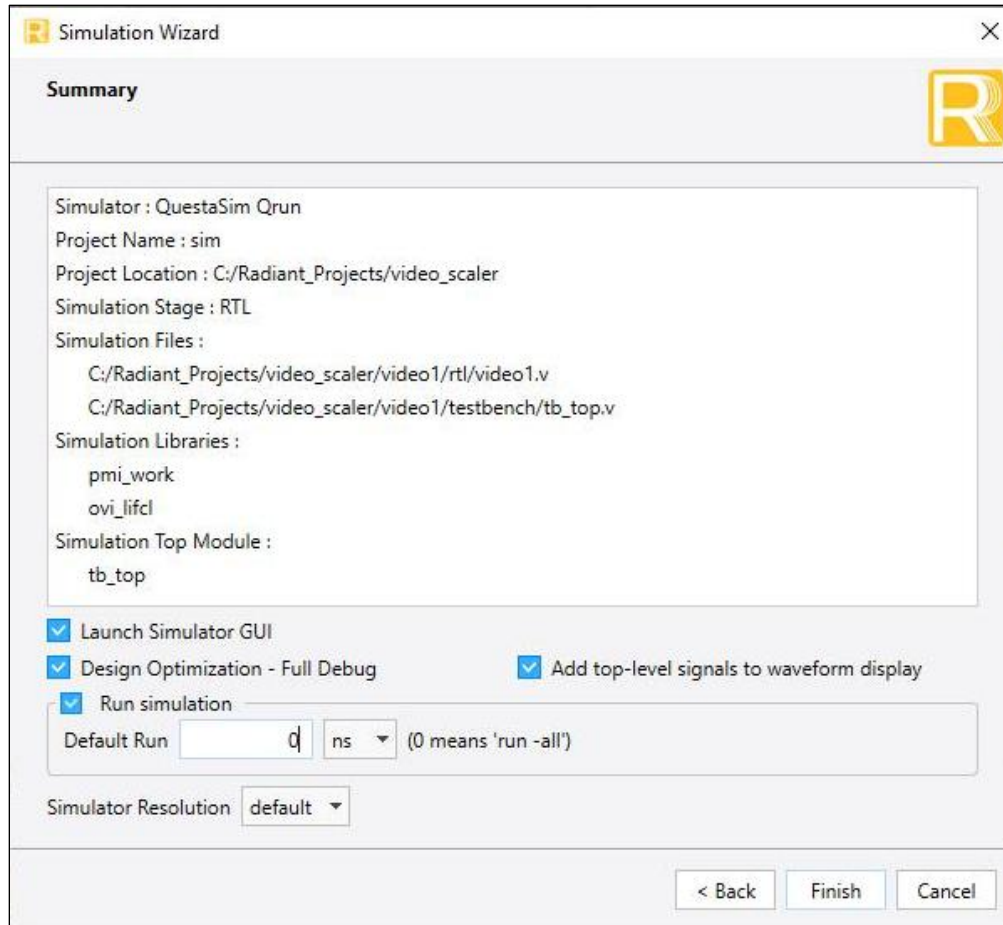


Figure 6.6. Run Simulation Value of 0 for Run All

4. Click **Next**. The **Summary** window opens.
5. Click **Finish** to run the simulation. The result of the simulation in the example is provided in Figure 6.7.

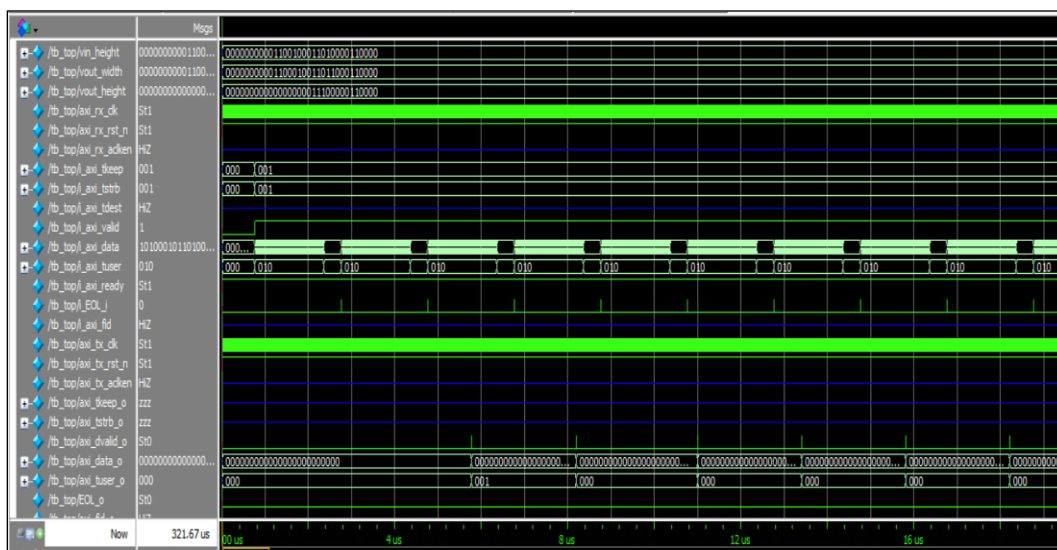


Figure 6.7. Sample Simulation Wave Form

Note: Testbench also includes a data comparator/checker. The data check completed indicates the correctness of the test. The results of data checker displayed in Figure 6.8.

```
# INFO: Waiting for Start of frame for      2 frame
# INFO: Waiting for Start of frame for      2 frame
# Input frame completed at:                7892645
# INFO: Waiting for Start of frame for      2 frame
# INFO: Waiting for Start of frame for      2 frame
# INFO: Waiting for Start of frame for      2 frame
# INFO: Start of frame Detected for         2 frame
# INFO: Opening files for writing RTL output data
# INFO: Creating RGB files
# INFO: RGB files created
# INFO: No of Rows = 1280 and No of Columns = 720
# INFO:          2 out of 2 Frames Completion Detected
# Input frame completed at:                15779842
# INFO: Data Checker started at:           15779842
# INFO: Data Check Completed at:           15779842
# ++++++ SIMULATION PASSED ++++++
# ** Note: $finish      : D:/video_loyce/project1/videoscaler1/testbench/testcases.vh(128)
# Time: 15779841840 ps  Iteration: 1  Instance: /tb_top
```

Figure 6.8. Data Check Passed

Note: If we enable the dynamic reconfiguration, then it will run the AXI-Lite register read/write interface.

It is done by writing the register addresses below with some data and checking by comparing the register data with the expected data. READ CHECK PASS indicates the passing of AXI-Lite register read/write interface test. AXI-Lite register read/write interface sample can be seen in Figure 6.9.

```
# .main_pane.wave.interior.cs.body.pw.wf
# Testing AXI-Lite register read/write interface
# Write Config Started
# INFO: Register Addr : 01000000 Data :    80
# INFO: Register Addr : 01000004 Data :    50
# INFO: Register Addr : 01000008 Data :    80
# INFO: Register Addr : 0100000c Data :    80
# INFO: Register Addr : 01000010 Data :   113
# INFO: Register Addr : 01000014 Data :    85
# INFO: Register Addr : 01000018 Data :    20
# INFO: Register Addr : 0100001c Data :    50
# INFO: Register Addr : 01000020 Data :     1
# Read check started
# READ CHECK PASS: exp_data =      80, read_data =      80
# READ CHECK PASS: exp_data =      50, read_data =      50
# READ CHECK PASS: exp_data =      80, read_data =      80
# READ CHECK PASS: exp_data =      80, read_data =      80
# READ CHECK PASS: exp_data =   113, read_data =   113
# READ CHECK PASS: exp_data =      85, read_data =      85
# READ CHECK PASS: exp_data =      20, read_data =      20
# READ CHECK PASS: exp_data =      50, read_data =      50
# READ CHECK PASS: exp_data =      1, read_data =      1
```

Figure 6.9. Testing AXI-Lite Register Read/Write Interface

6.3.1. Limitations of Simulation

```

////////////////////////////////////
task base_test(input [1:0]ppc,[13:0]vsize,[13:0]hsize,[2:0]bpp,[7:0]no_of_frames
    reg error0;
begin
    FRMWIDTH      = DYNAMIC == "TRUE" ? VINWIDTH  : VINWIDTH;
    FRMHEIGHT     = DYNAMIC == "TRUE" ? VINHEIGHT : VINHEIGHT;
    OUTWIDTH      = DYNAMIC == "TRUE" ? VOUTWIDTH  : VOUTWIDTH;
    OUTHEIGHT     = DYNAMIC == "TRUE" ? VOUTHEIGHT : VOUTHEIGHT;
    VSFACTOR1     = DYNAMIC == "TRUE" ? VSFACTOR   : VSFACTOR;
    HSFACTOR1     = DYNAMIC == "TRUE" ? HSFACTOR   : HSFACTOR;
    HBLANKOUT1    = DYNAMIC == "TRUE" ? HBLANKOUT  : HBLANKOUT;
    VBLANKOUT1    = DYNAMIC == "TRUE" ? VBLANKOUT  : VBLANKOUT;
    UPDATE        = DYNAMIC == "TRUE" ? 'd1       : 'd0;

```

Figure 6.10. Dynamically Configurable Parameters in the Code

If dynamic reconfiguration is enabled in GUI and all the dynamically re-configurable parameters (testbench/testcases.vh: L50-L57) are equal to the GUI entered parameters, then it will run the internal data checker to compare RTL output with the golden test vectors and if matched, “Simulation passed” is displayed in the transcript. Refer to Figure 6.11.

```

# INFO: No of Rows = 200 and No of Columns = 100
# INFO: 2 out of 2 Frames Completion Detected
# Input frame completed at: 526953
# INFO: Data Checker started at: 526953
# INFO: Data Check Completed at: 526953
# ++++++ SIMULATION PASSED ++++++

```

Figure 6.11. Simulation Showing Passed for Matching the Dynamic Parameters Same as GUI Entered Parameters

If dynamic reconfiguration is enabled in GUI and any one of the dynamically configurable parameters is not equal to the GUI entered parameters, then it will not run the data checker and “Simulation completed” is displayed in the transcript. Refer to Figure 6.12.

```

# INFO: RGB files created
# INFO: No of Rows = 80 and No of Columns = 80
# INFO: 2 out of 2 Frames Completion Detected
# Input frame completed at: 465639
# ++++++ SIMULATION COMPLETED ++++++

```

Figure 6.12. Simulation Showing Completed for Difference Between Dynamic and GUI Entered Parameters

7. Design Considerations

The Video Scaler IP may not produce the correct output for the following frame dimensions using Nearest as the scaling method:

- Input active pixels: 1920
- Input active lines: 1080
- Output active pixels: 3840
- Output active lines: 2160

For the above-mentioned frame dimensions, use an alternate scaling method such as Bilinear, Bicubic, and Lanczos.

7.1. Known Issues

Simulation hangs when you set *Native Interface* == *NATIVE* in the Video Scaler IP v1.3.0.

8. Ordering Part Number

The Ordering Part Number (OPN) for this IP Core are the following:

Table 8.1. Ordering Part Numbers for Video Scaler IP Core

OPN	Device Family	License Type
VIDEO-SCALER-CPNX-UT	CertusPro-NX	Single Seat Perpetual License
VIDEO-SCALER-CPNX-US	CertusPro-NX	Single Seat Annual License
VIDEO-SCALER-CTNX-UT	Certus-NX	Single Seat Perpetual License
VIDEO-SCALER-CTNX-US	Certus-NX	Single Seat Annual License
VIDEO-SCALER-CN2-UT	Certus-N2	Single Seat Perpetual License
VIDEO-SCALER-CN2-US	Certus-N2	Single Seat Annual License
VIDEO-SCALER-CNX-UT	CrossLink-NX	Single Seat Perpetual License
VIDEO-SCALER-CNX-US	CrossLink-NX	Single Seat Annual License
VIDEO-SCALER-XO5-UT	MachXO5-NX	Single Seat Perpetual License
VIDEO-SCALER-XO5-US	MachXO5-NX	Single Seat Annual License
VIDEO-SCALER-AVE-UT	Lattice Avant-AT-E	Single Seat Perpetual License
VIDEO-SCALER-AVE-US	Lattice Avant-AT-E	Single Seat Annual License
VIDEO-SCALER-AVG-UT	Lattice Avant-AT-G	Single Seat Perpetual License
VIDEO-SCALER-AVG-US	Lattice Avant-AT-G	Single Seat Annual License
VIDEO-SCALER-AVX-UT	Lattice Avant-AT-X	Single Seat Perpetual License
VIDEO-SCALER-AVX-US	Lattice Avant-AT-X	Single Seat Annual License

Appendix A. Resource Utilization

The following tables show the sample resource utilization of the Video Scaler IP Core.

The results are based on the Synplify Pro and Lattice Radiant software version 2025.1. The major parameters for each configuration that are different from the IP GUI defaults are shown in the table entries.

Table A.1. Resource Utilization using LFCPNX-100-8LFG672C Certus-Pro NX Device

Configuration				Frequency		Resource Utilization			
Pixels Per Clock	Bits Per Component	Resolutions	Scaling Method	Clock RX_max (MHz)	Clock TX_max (MHz)	Registers	LUTs	EBRs	DSP
1	8	720p -> 1080p	Nearest	200	200	1929	2623	7	—
1	8	720p -> 1080p	Bilinear	200	200	2697	3836	9	12
1	8	720p -> 1080p	Bicubic	200	200	3067	5370	11	12
1	16	1080p -> 720p	Bicubic	200	200	3925	7676	27	24
1	16	720p -> 1080p	Bicubic	192	200	3899	7166	20	24
1	8	720p -> 1080p	Lanzcos	200	200	3056	5392	11	12
2	8	720p -> 1440p	Lanzcos	200	200	4555	9480	16	26
2	8	1440p -> 720p	Lanzcos	200	200	4537	9818	23	26
4	8	720p -> 2160p	Lanzcos	200	200	8134	21275	31	56
4	8	2160p -> 720p	Lanzcos	196	197	8166	22267	64	56

Table A.2. Resource Utilization using LFD2NX-40-8MG121C Certus-NX Device

Configuration				Frequency		Resource Utilization			
Pixels Per Clock	Bits Per Component	Resolutions	Scaling Method	Clock RX_max (MHz)	Clock TX_max (MHz)	Registers	LUTs	EBRs	DSP
1	8	720p -> 1080p	Nearest	200	200	1929	2623	7	—
1	8	720p -> 1080p	Bilinear	200	200	2697	3836	9	12
1	8	720p -> 1080p	Bicubic	200	200	3067	5370	11	12
1	16	1080p -> 720p	Bicubic	200	200	3925	7676	27	24
1	16	720p -> 1080p	Bicubic	200	200	3899	7166	20	24
1	8	720p -> 1080p	Lanzcos	200	200	3056	5392	11	12
2	8	720p -> 1440p	Lanzcos	197	200	4555	9480	16	26
2	8	1440p -> 720p	Lanzcos	199	200	4537	9818	23	26
4	8	720p -> 2160p	Lanzcos	196	195	8134	21275	31	56
4	8	2160p -> 720p	Lanzcos	Not fitting					

Table A.3. Resource Utilization using LIFCL-33-8USG84C CrossLink-NX Device

Pixels Per Clock	Bits Per Component	Resolutions	Scaling Method	Frequency		Resource Utilization			
				Clock RX_max (MHz)	Clock TX_max (MHz)	Registers	LUTs	EBRs	DSP
1	8	720p -> 1080p	Nearest	200	198	1929	2623	7	—
1	8	720p -> 1080p	Bilinear	193	184	2697	3836	9	12
1	8	720p -> 1080p	Bicubic	200	191	3067	5370	11	12
1	16	1080p -> 720p	Bicubic	195	170	3925	7676	27	24
1	16	720p -> 1080p	Bicubic	195	170	3899	7166	20	24
1	8	720p -> 1080p	Lanzcos	200	193	3056	5392	11	12
2	8	720p -> 1440p	Lanzcos	199	200	4555	9480	16	26
2	8	1440p -> 720p	Lanzcos	193	179	4537	9818	23	26
4	8	720p -> 2160p	Lanzcos	191	191	8134	21275	31	56
4	8	2160p -> 720p	Lanzcos	Not fitting					

Table A.4. Resource Utilization using LAV-AT-E70-2LFG676C Lattice Avant Device

Configuration				Frequency		Resource Utilization			
Pixels Per Clock	Bits Per Component	Resolutions	Scaling Method	Frequency		Registers	LUTs	EBRs	DSP
				Clock RX_max (MHz)	Clock TX_max (MHz)				
1	8	720p -> 1080p	Nearest	250	250	1884	2597	4	—
1	8	720p -> 1080p	Bilinear	250	250	2743	3754	5	12
1	8	720p -> 1080p	Bicubic	250	250	3142	5369	6	8
1	16	1080p -> 720p	Bicubic	250	250	3952	7402	14	24
1	16	720p -> 1080p	Bicubic	250	250	3944	7130	11	24
1	8	720p -> 1080p	Lanzcos	250	250	3137	5423	6	8
2	8	720p -> 1440p	Lanzcos	250	250	4892	9492	9	17
2	8	1440p -> 720p	Lanzcos	250	250	4909	9799	12	17
4	8	720p -> 2160p	Lanzcos	250	250	7938	20819	16	37
4	8	2160p -> 720p	Lanzcos	246	250	7977	21976	34	37

Table A.5. Resource Utilization using LN2-CT-20ES-1ASG410I Certus-N2 Device

Configuration				Frequency		Resource Utilization			
Pixels Per Clock	Bits Per Component	Resolutions	Scaling Method	Frequency		Registers	LUTs	EBRs	DSP
				Clock RX_max (MHz)	Clock TX_max (MHz)				
1	8	720p -> 1080p	Nearest	250	250	1884	2597	4	—
1	8	720p -> 1080p	Bilinear	250	250	2743	3754	5	12
1	8	720p -> 1080p	Bicubic	247	250	3142	5369	6	8
1	16	1080p -> 720p	Bicubic	250	250	3952	7402	14	24
1	16	720p -> 1080p	Bicubic	250	250	3944	7130	11	24
1	8	720p -> 1080p	Lanzcos	250	250	3137	5423	6	8
2	8	720p -> 1440p	Lanzcos	250	250	4892	9492	9	17
2	8	1440p -> 720p	Lanzcos	241	250	4909	9799	12	17
4	8	720p -> 2160p	Lanzcos	250	250	7938	20819	16	37
4	8	2160p -> 720p	Lanzcos	249	250	7977	21976	34	37

References

- [Video Scaler IP Release Notes \(FPGA-RN-02063\)](#)
- [Lattice Radiant Timing Constraints Methodology \(FPGA-AN-02059\)](#)
- [CrossLink-NX](#) web page
- [Certus-NX](#) web page
- [Certus-N2](#) web page
- [CertusPro-NX](#) web page
- [MachXO5-NX](#) web page
- [Avant-E](#) web page
- [Avant-G](#) web page
- [Avant-X](#) web page
- [Lattice Radiant Software](#) web page
- [Lattice Insights](#) for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.3, IP v1.3.0, July 2025

Section	Change Summary
Introduction	Updated Table 1.1. Video Scaler IP Quick Facts as follows: <ul style="list-style-type: none"> Renamed <i>Supported FPGA Families</i> to <i>Supported Devices</i>. Removed the <i>Targeted Devices</i> row. Added IP version.
Parameter Settings & Attributes Summary	Updated Figure 4.2. Implementation Tab .
IP Core Generation, Simulation, and Validation	Updated the following figures: <ul style="list-style-type: none"> Figure 6.1. Module/IP Block Wizard Figure 6.2. Configuration User Interface for Video Scaler IP Figure 6.3. Check Generating Result Figure 6.4. Simulation Wizard Figure 6.5. Adding and Reordering Source Figure 6.6. Run Simulation Value of 0 for Run All
Design Considerations	Added the Known Issues section.
Ordering Part Number	Updated Table 8.1. Ordering Part Numbers for Video Scaler IP Core as follows: <ul style="list-style-type: none"> Added OPN for MachXO5-NX devices. Changed <i>Multi-site Perpetual</i> to <i>Single Seat Perpetual</i>.
Resource Utilization	Updated resource utilization for the latest software version.
References	Updated references.

Revision 1.2, IP v1.2.0, December 2024

Section	Change Summary
Introduction	Updated Table 1.1. Video Scaler IP Quick Facts as follows: <ul style="list-style-type: none"> Added Certus-N2 device. Added IP changes. Updated IP version.
IP Core Generation, Simulation, and Validation	Changed <i>black box</i> to <i>closed-box</i> in the Generation and Synthesis section.
Design Considerations	Added this section.
Ordering Part Number	Updated Table 8.1. Ordering Part Numbers for Video Scaler IP Core as follows: <ul style="list-style-type: none"> Added OPN for Certus-N2 devices. Changed from <i>Single Machine Annual</i> to <i>Single Seat Annual</i>.
Resource Utilization	Added resource utilization for the LN2-CT-20-1ASG410I device.
References	Added links to the Certus-N2 web page and IP release notes.

Revision 1.1, December 2023

Section	Change Summary
Disclaimers	Updated disclaimers.
Inclusive Language	Added inclusive language boilerplate.
Introduction	Updated Table 1.1. Video Scaler IP Quick Facts as follows: <ul style="list-style-type: none"> Added Lattice Avant devices. Updated IP version from v1.0.0 to v1.1.0.
Ordering Part Number	Added OPNs for Lattice Avant devices.

Section	Change Summary
Resource Utilization	<ul style="list-style-type: none">Updated resource utilization for CertusPro-NX, Certus-NX, and CrossLink-NX devices.Added resource utilization for Lattice Avant device.
References	Updated references.

Revision 1.0, August 2023

Section	Change Summary
All	Initial release.



www.latticesemi.com