Automatic White Balance IP

IP Version: 1.4.1

User Guide

FPGA-IPUG-02204-1.4

June 2026

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. 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 use older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AWB	Automatic White Balance
AXI	Advanced eXtensible Interface
BPP	Bits Per Pixel
CSR	Configuration and Status Registers
DSP	Digital Signal Processor
EBR	Embedded Block RAM
FIFO	First-In First-Out
IP	Intellectual Property
LUT	Look-Up Table
PPC	Pixels Per Clock
RGB	Red, Green, and Blue
RTL	Register Transfer Level

1. Introduction

The Lattice Semiconductor Automatic White Balance (AWB) IP automatically compensates for illumination temperature-based color differences, such that white actually appears white.

The operation is performed mostly in the Bayer domain, but it can also be done in RGB domain. AWB is a pixel-based operation that uses image statistics. Since pixel correction is scene-based, calibration is not required. White balancing is a two-step process: determination of the nature of the illuminant and image correction based on the illuminant. The methods available for AWB differ in the way the illuminant temperature is determined. The image correction part involves multiplying the pixel components with the correction factor. The AWB IP provides AXI4-Stream interfaces for the input and output video streams. The IP can process up to 16 bits per pixel (BPP) and 4 pixels per clock (PPC). An optional AXI4-Lite interface is provided to dynamically configure the parameters and to read out the status of the IP.

1.1. Quick Facts

Table 1.1 presents a summary of the AWB IP.

Table 1.1. Quick Facts

IP Requirements	Supported Devices	CrossLink™-NX, CertusPro™-NX , Certus™-NX (LFD2NX-17, LFD2NX-40, LFD2NX-9, LFD2NX-28, LFD2NX-25, LFD2NX-35, LFD2NX-65, LFD2NX-15P, LFD2NX-25P), MachXO5™-NX (LFMXO5-35, LFMXO5-35T, LFMXO5-65, LFMXO5-65T, LFMXO5-25P), Lattice Avant™, Lattice Certus™-N2
	IP Changes[1]	For a list of changes to the IP, refer to the Automatic White Balance IP Release Notes (FPGA-RN-02055).
Resource Utilization	Resources	See Appendix A. Resource Utilization
Design Tool Support	Lattice Implementation	IP v1.4.1 – Lattice Radiant™ software 2026.1
	Synthesis	Lattice Synthesis Engine
		Synopsys® Synplify Pro® for Lattice
	Simulation	For a list of supported simulators, see the Lattice Radiant software user guide.

Note:

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

1.2. Features

The following are the key features of the AWB IP:

- Supports 6, 8, 10, 12, and 16 bits per pixel
- Supports 1, 2, and 4 pixels per clock
- Supports AXI4-Stream Protocol to receive and send pixel information
- Supports AXI4-Lite Protocol to configure and control the IP

1.3. Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.3.2. Signal Names

Signal Names that end with:

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

1.4. Attributes

The names of attributes in this document are formatted in title case and italicized (Attribute Name).

2. Functional Description

2.1. Algorithms for Automatic White Balance

Various algorithms are available for performing AWB, offering different performance and complexity. A common method of AWB is using the gray world theory, which uses the mean of green components to scale the red and blue components. The gray world method shows a consistently good performance across different illuminations and multiple scenarios, and has the least hardware complexity among the algorithms.

2.2. Block Diagram

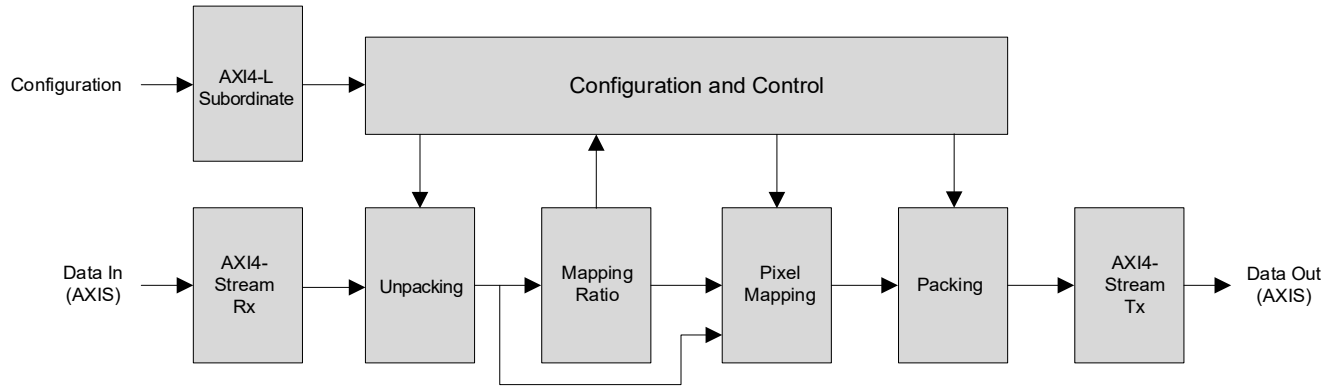

Figure 2.1. AWB IP Architectural Diagram

The data comes in through the AXI-Stream interface, is processed, and goes out through the AXI-Stream interface. There are two essential operations happening in the IP: the determination of color correction ratio based on full past frame data and the mapping of each pixel in the current frame using the color correction ratio.

The following are the main elements of the AWB hardware architecture:

- Use of AXI-Stream for streaming in video data. The input/output data includes pixel data as well as video sync signals. The pixel data is passed through the TDATA bus and the video sync signals are passed through the user-defined TUSER bus as sideband signals.
- Unpacking of the AXI-Stream data to pixel data based on the color format. This process also includes extraction of color components from the Bayer input.
- The central function of AWB is color mapping ratio determination. This is done in two steps: illuminant determination (finding the mean value of the components across the entire frame) followed by ratio determination using the mean and maximum values of the components.
- Pixel mapping maps the original component values to corrected values using the mapping ratios.
- Packing is the process of lining up the pixel values and timing information into the TDATA and TUSER buses of the AXI-Stream transmitter for output.
- AXI-Lite is used for the configuration path to write or read configuration and status registers (CSR).

2.3. AXI-Stream Receiver

The AXI-Stream Rx block is the protocol receiver for AXI-Stream. The control inputs (sync signals) are sent in through the user-defined sideband bus, TUSER. The data width of AXI-Stream must be a multiple of 8, with powers of two recommended by the specifications. The number of bytes in TDATA is determined based on pixel width and number of pixels per clock and the pixel data is packed across these bytes. The receiver processes the appropriate data bytes in the bus based on the TSTROBE input.

On the AXI-Stream handshaking, the Transmitter must not wait until TREADY is asserted before asserting TVALID. This implies that the Transmitter must independently assert TVALID without reference to TREADY. Also, the standard requires that TVALID, once asserted, must remain asserted until the handshake occurs.

The AXI-Stream handshake signals and the user-defined TUSER bus are described below.

- The user-defined sideband signal TUSER is used to specify the frame boundaries and active data windows. Specifically, rx_tuser_i[0] indicates start of frame, with this bit asserting high during the first pixel of a frame. The input rx_tuser_i[1] specifies 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] indicates end of frame. This signal must be high during the last active pixel of a frame.
- The input rx_tlast_i indicates end of a total line. This must be asserted during the last pixel of horizontal blanking.
- The rx_tready_o signal is always set to *1*, except when the Configuration and Status Registers (CSR) are being updated, at which time it is set to *0*.
- The rx_tstrobe_i signal defines the valid bytes in a data beat, that is, the valid bytes in rx_tdata_i.
- The width of rx_tdata_i depends on the parameters BPP and PPC. The width of this bus is equal to the next higher multiple of 8 of the quantity (PPC × BPP × 3).

The timing diagrams for the AXI-Stream receiver for different PPC and BPP values are shown in Figure 2.2, Figure 2.3, and Figure 2.4.

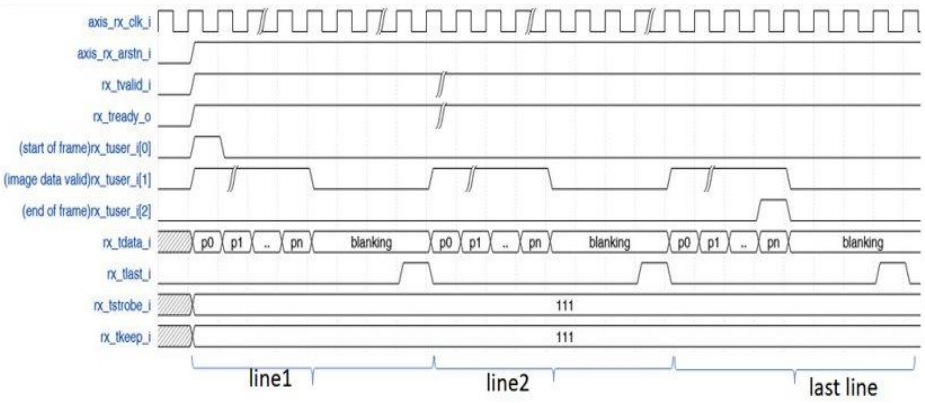

Figure 2.2. AXI-Stream Receiver Signals for PPC=1 and BPP=8 (RGB Format)

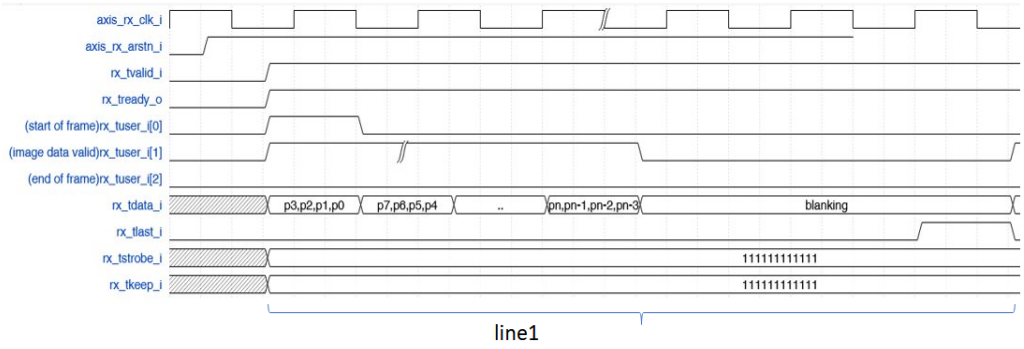

Figure 2.3. AXI-Stream Receiver Signals for PPC=4 and BPP=8 (RGB Format)

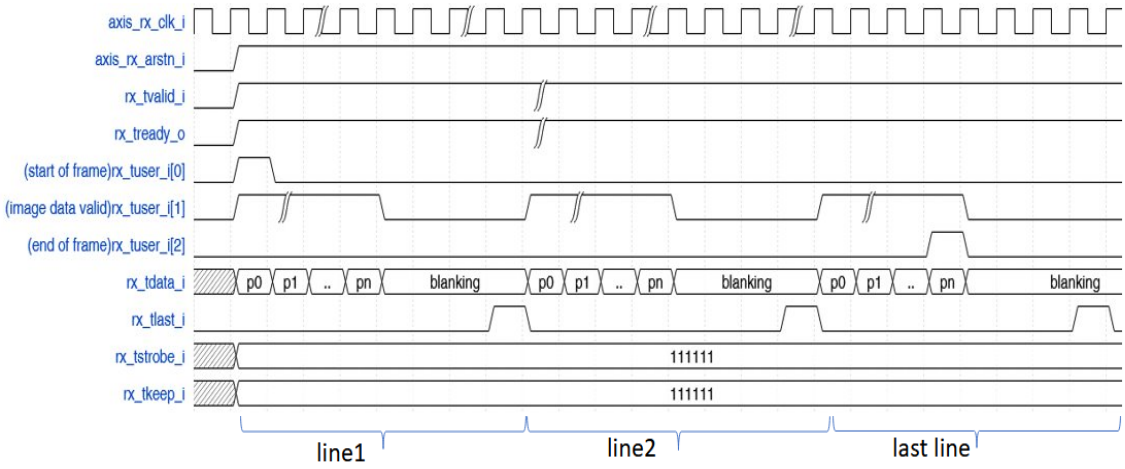

Figure 2.4. AXI-Stream Receiver Signals for PPC=1 and BPP=16 (RGB Format)

The organization of the color component data in the TDATA bus is described here. In general, the red component is mapped to the lower bits, green to the middle bits, and blue to the upper bits, each component mapped from MSB to LSB.

The mapping of bits for the case of PPC=1 and BPP=6 is shown in Figure 2.5. The mappings for other cases are shown from Figure 2.6 to Figure 2.10.

AXI-Stream bus width must be divisible by 8. Pixel data of (PPC × BPP × 3) bits is mapped to the lower part of the bus, with the remaining upper bits assigned to zeros.

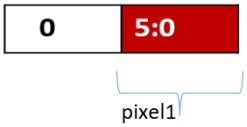

Figure 2.5. Bit Allocation for BPP=6 and PPC=1 for Bayer RGGB input

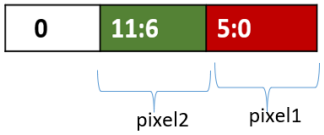

Figure 2.6. Bit Allocation for BPP=6 and PPC=2 for Bayer RGGB input

Figure 2.7. Bit Allocation for BPP=6 and PPC=4 for Bayer RGGB input

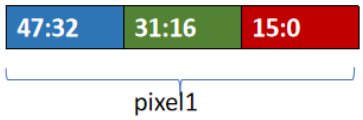

Figure 2.8. Bit Allocation for BPP=16 and PPC=1 for RGB input

Figure 2.9. Bit allocation for BPP=16 and PPC=2 for RGB input

Figure 2.10. Bit Allocation for BPP=16 and PPC=4 for RGB input

2.4. Data Unpacker

The Data Unpacker module unpacks the data from the AXI-Stream receiver into individual RGB components. The data unpacking depends on the video type as well as on the Bayer pattern, for Bayer video types. This module provides output in RGB format with data valid signals defining the valid data. The timing diagrams for the unpacker for different configurations are shown in Figure 2.11, Figure 2.12, and Figure 2.13. As seen in the figures, the data from rx_tdata_i is unpacked to red_component, green_component, and blue_component along with a valid signal.

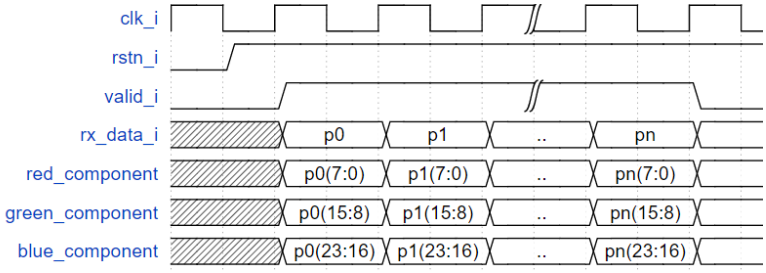

Figure 2.11. Unpacking of Pixels into Different Components for RGB Pattern

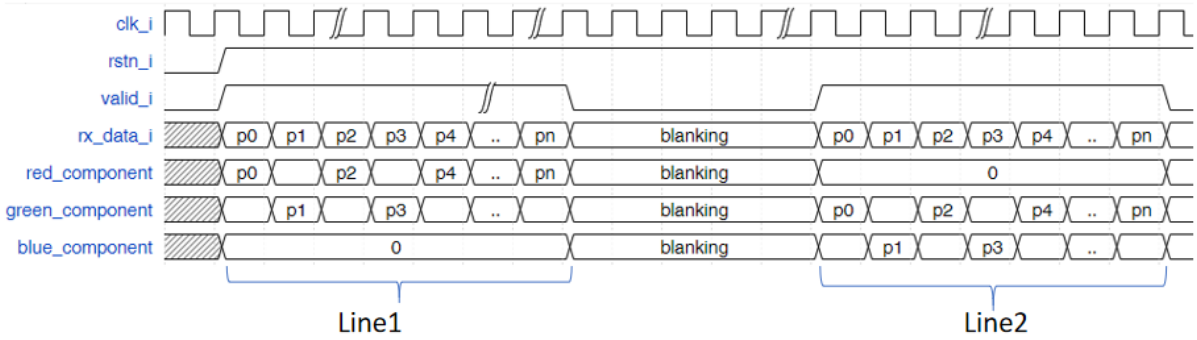

Figure 2.12. Unpacking Pixels into Different Components for Bayer Pattern (RGGB) with BPP=8 and PPC=1

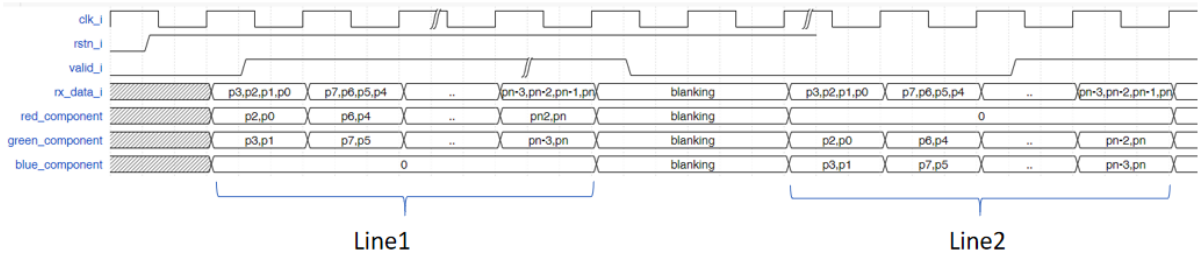

Figure 2.13. Unpacking Pixels into Different Components for Bayer Pattern (RGGB) with BPP=16 and PPC=4

2.5. Data Packer

The Data Packer module performs the reverse of the unpacking operation. Here, the pixel data is packed into multiple bytes based on the currently set parameters. The timing diagrams for the packer for different configurations are shown in Figure 2.14 and Figure 2.15.

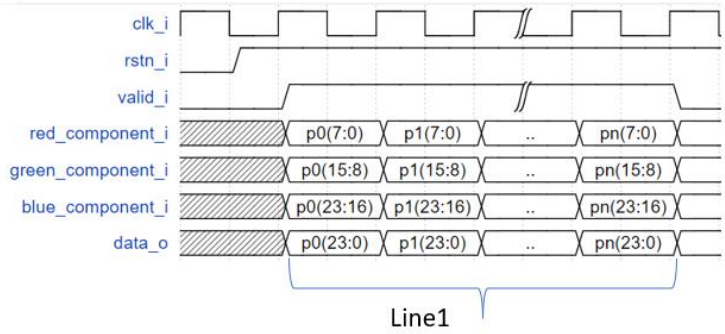

Figure 2.14. Packing of Color Components in RGB Pattern for BPP=8 and PPC=1

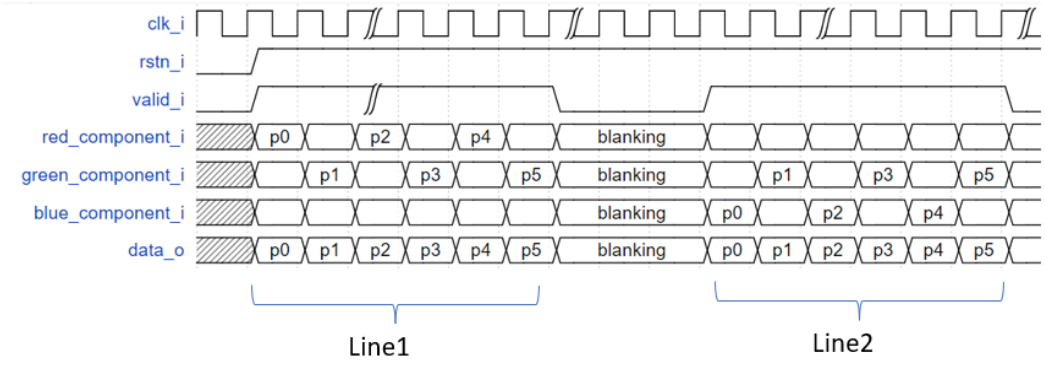

Figure 2.15. Packing of Color Components in Bayer Pattern (RGGB) for BPP=8 and PPC=1

2.6. AXI-Stream Transmitter

The AXI-Stream Transmitter module maps the video data back to bytes for transmission through AXI-Stream to the next stage pipeline.

The transmitter sends out the data in RGB format. Red is located in the lower 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 stored in the FIFO up to parameter Tx buffer depth configured through the IP user interface. If tx_tready_i is deasserted for more than the Tx buffer depth cycles, the frame is discarded.

The AXI-Stream handshake signals and user-defined TUSER bus are described below.

- The user-defined sideband signal TUSER is used to specify the frame boundaries and data validity. Specifically, tx_tuser_o[0] indicates start of 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 end of frame. This signal is high during the last active pixel of a frame.
- The output tx_tlast_o indicates 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 and BPP. The width of this bus is equal to the next higher multiple of 8 of the quantity (PPC × BPP × 3).

The timing diagrams for the AXI-Stream Transmitter are shown in Figure 2.16 and Figure 2.17.

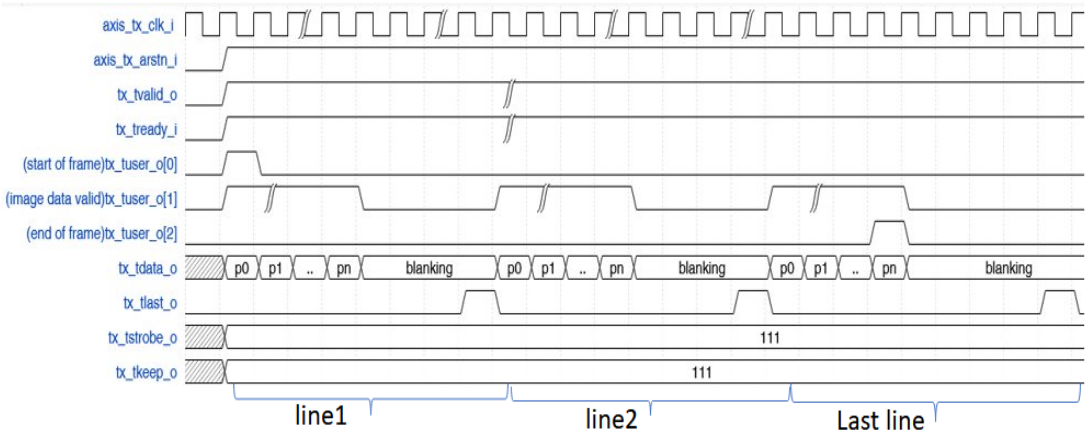

Figure 2.16. AXI-Stream Transmitter with PPC=1 and BPP=8

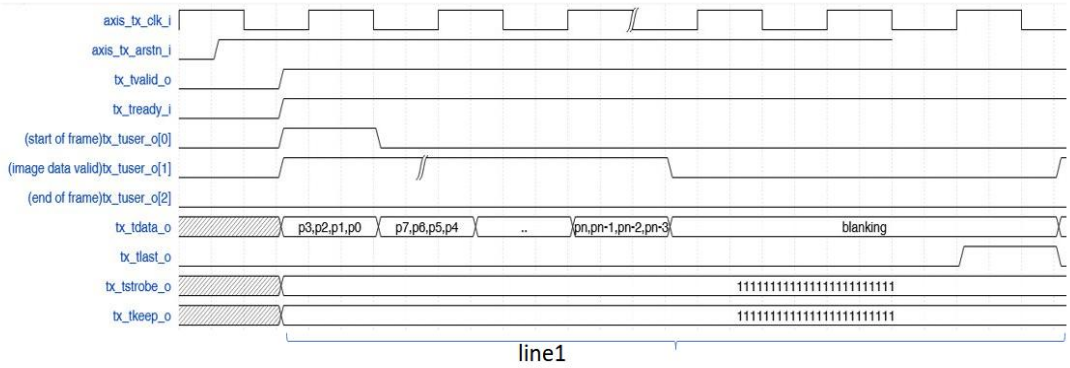

Figure 2.17. AXI-Stream Transmitter with PPC=4 and BPP=16 for RGB Pattern

2.7. Configuration and Control

The Configuration and Control block is used to dynamically configure the IP parameters. This block contains the AXI-Lite subordinate interface and a number of registers to store the dynamically configurable parameters. The parameters supported for dynamic configuration are:

- Enable/Disable AWB
- Bayer pattern
- Video type (Bayer or RGB)
- Configuration update

For details on the configuration register, refer to Table 5.1.

2.8. AXI-Lite Subordinate

The AXI-Lite interface 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

2.8.1. 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_o and captures the address.
3. The manager asserts w_valid_i along with the write data.
4. The subordinate asserts w_ready_o 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.

2.8.2. Read Operation

The read 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_o.
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

The widths used for the signal buses in the interface table are defined in Table 3.1. The input/output interface signals for AWB IP 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 next higher multiple of 8 of the quantity (PPC × BPP × 3).
AXI_LITE_ADDR_WIDTH	2
AXI_LITE_DATA_WIDTH	32

Table 3.2. AWB IP Signal Description

Port Name	I/O	Size	Description
Clock and Reset			
axi_lite_rst_n_i	I	1	AXI-Lite reset (Active low)
axis_rx_arstn_i	I	1	AXI-Stream RX reset (Active low)
axis_tx_arstn_i	I	1	AXI-Stream TX reset (Active low)
axi_lite_clk_i	I	1	AXI-Lite clock
axis_rx_clk_i	I	1	AXI-Stream RX clock
axis_tx_clk_i	I	1	AXI-Stream TX clock
AXI-Stream Rx			
rx_tdata_i	I	[AXI_STREAM_DATA_WIDTH-1:0]	TDATA is the primary payload that is used to provide the data that is passing across the interface.
rx_tuser_i	I	3	Sideband information transmitted alongside the data stream. rx_tuser_i[0] - start of frame rx_tuser_i[1] - active image data rx_tuser_i[2] - end of frame
rx_tlast_i	I	1	TLAST indicates the end of each horizontal line.
rx_tvalid_i	I	1	TVALID indicates that the transmitter is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.
rx_tstrobe_i	I	[AXI_STREAM_DATA_WIDTH-1:0]/8	TSTROBE is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte.
rx_tkeep_i	I	[AXI_STREAM_DATA_WIDTH-1:0]/8	TKEEP is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the TKEEP byte qualifier deasserted are null bytes and can be removed from the data stream.
rx_tready_o	O	1	TREADY indicates that the RX can accept a transfer in the current cycle.
AXI-Stream Tx			
tx_tvalid_o	O	1	TVALID indicates that the Transmitter is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.
tx_tready_i	I	1	TREADY indicates that the RX can accept a transfer in the current cycle.
tx_tdata_o	O	[AXI_STREAM_DATA_WIDTH-1:0]	TDATA is the primary payload that is used to provide the data that is passing across the interface.

Port Name	I/O	Size	Description
tx_tuser_o	O	3	Sideband information transmitted alongside the data stream. tx_tuser_o[0] - start of frame tx_tuser_o[1] - valid image data tx_tuser_o[2] - end of frame
tx_tlast_o	O	1	TLAST indicates the end of each horizontal line.
tx_tstrobe_o	O	[AXI_STREAM_DATA_WIDTH-1:0]/8	TSTROBE is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte.
tx_tkeep_o	O	[AXI_STREAM_DATA_WIDTH-1:0]/8	TKEEP is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the TKEEP byte qualifier deasserted are null bytes and can be removed from the data stream.
AXI-Lite Subordinate			
aw_valid_i	I	1	Write address valid. This signal indicates that the channel is signaling valid write address and control information.
aw_address_i	I	[AXI_LITE_ADDR_WIDTH-1:0]	Write address.
aw_ready_o	O	1	Write address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals.
aw_prot_i	I	[2:0]	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. (Not used).
w_valid_i	I	1	Write valid. This signal indicates valid write data.
w_data_i	I	[AXI_LITE_DATA_WIDTH-1:0]	Write data.
w_ready_o	O	1	Write ready. This signal indicates that the subordinate can accept the write data.
w_strb_i	I	[AXI_LITE_DATA_WIDTH-1:0]/8	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. (Not used)
b_valid_o	O	1	Write response valid. This signal indicates that the channel is signaling a valid write response.
b_response_o	O	2	Write response. This signal indicates the status of the write transaction.
b_ready_i	I	1	Response ready. This signal indicates that the manager can accept a write response.
ar_valid_i	I	1	Read address valid. This signal indicates that the channel is signaling valid read address.
ar_address_i	I	[AXI_LITE_ADDR_WIDTH-1:0]	Read address
ar_ready_o	O	1	Read address ready. This signal indicates that the subordinate is ready to accept an address
ar_prot_i	I	[2:0]	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. (Not used).
r_data_o	O	[AXI_LITE_DATA_WIDTH-1:0]	Read data.
r_valid_o	O	1	Read valid. This signal indicates that the channel is signaling the required read data.

Port Name	I/O	Size	Description
r_response_o	O	2	Read response. This signal indicates the status of the read transfer.
r_ready_i	I	1	Read ready. This signal indicates that the manager can accept the read data and response information.

4. Attributes Summary

The attributes set through the user interface are described in Table 4.1.

Table 4.1. Attributes Table

Attribute	Selectable Values	Default	Dependency on Other Attributes	Description
AXI-Lite				
Bits per pixel	6, 8, 10, 12, 16	8	—	Number of bits per color component of a pixel
Pixels per clock	1, 2, 4	1	—	Number of pixels streamed in or out in a clock
Enable Dynamic Configuration	Enabled, Disabled	Enabled	—	Whether Configuration and Status Registers (CSR) and AXI-Lite interface are required
Partial Resolution	Enabled, Disabled	Disabled	—	Partial strobe support for last pixel of a line for PPC2 and PPC4, when resolution is not divisible by PPC.
FIFO				
Tx buffer depth	128, 256, 512, 1024	1024	—	This value describes the FIFO depth. If tx_tready_i is deasserted for more than *TX buffer depth* cycles, the entire frame gets skipped. If tx_tready_i is asserted before the Tx buffer depth cycles, then the data flow will not be interrupted.
CSR				
Video Type	RGB, Bayer	RGB	—	Whether RGB or Bayer Pattern is enabled
Bayer Pattern	RGGB, BGGR, GRBG, GBRG	RGGB	This option is disabled when Video Type is RGB	Different Bayer patterns are selectable
AWB Enable	Enabled, Disabled	Enabled	This option is not editable when Enable dynamic Configuration is disabled	Enable (Automatic White balance operation) Disable (the output pixel data is the same as the input data)

A sample user interface for the AWB IP is shown in Figure 4.1.

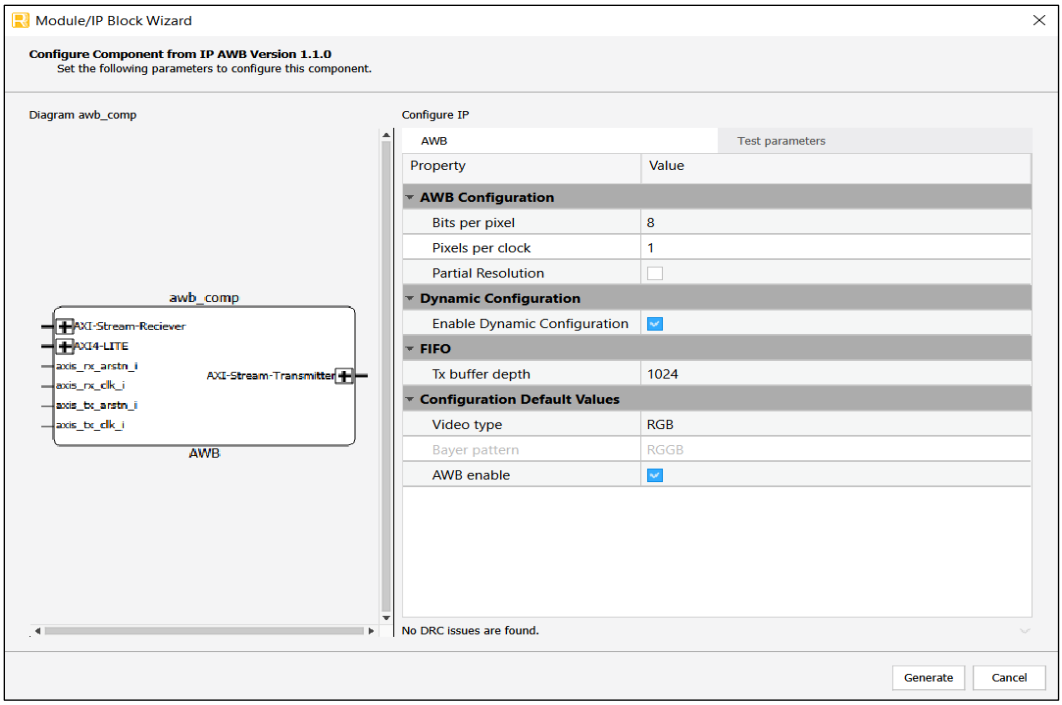

Figure 4.1. Configuration User Interface for AWB IP

5. Register Description

The registers used in the CSR module of the AWB IP are shown in Table 5.1. The CSR registers are accessible through the AXI-Lite interface.

Table 5.1. Summary of Configuration and Status Registers

Offset	Register Name	Access	Default Value	Register Description		
				Bit	Field	Field Description
Attribute Registers						
0x00	VIDEO_TYPE	RW	0	0	video_type	0 – Bayer 1 – RGB
				[31:1]	RSVD	Reserved bits
0x01	BAYER_PATTERN	RW	00	[1:0]	bayer_pattern	00 – RGGB pattern 01 – BGGR pattern 10 – GRBG pattern 11 – GBRG pattern
				[31:2]	RSVD	Reserved bits
0x02	AWB_ENABLE	RW	0	0	awb_enable	0 – enable (automatic white balance operation) 1 – disable (the output pixel data is the same as the input data)
				[31:1]	RSVD	Reserved bits
0x03	CONFIGURATION_UPDATE	RW	0	0	config_update	Set config_update to 1 after updating all the other registers. The AWB IP clears this register after reading configuration values. If the you update the configuration registers in the middle of an active frame, the updated values are read during the blanking period of the next frame.
				[31:1]	RSVD	Reserved bits

6. IP Generation, Simulation, and Validation

This section provides information on how to generate the AWB 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.

Note: The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.

6.1. Licensing the IP

The Automatic White Balance IP is provided at no additional cost with the Lattice Radiant software.

6.2. Generation and Synthesis

The Lattice Radiant software allows you to customize and generate modules and IP and integrate them into the device architecture. The following procedure describes how to generate the AWB IP in Lattice Radiant software.

1. Create a new Lattice Radiant software project or open an existing project. If the Lattice Synthesis Engine is used, refer to the guidelines in the Limitations section.
2. In the **IP Catalog** tab, double-click on **Auto White Balance** under the **Audio_Video_and_Image_Processing** category. **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**.
5. Go to *RadiantIPLocal\awb_test\testbench\tb_settings.v*. Update the file path parameter accordingly. Verify that the generated user interface matches the testbench parameters.

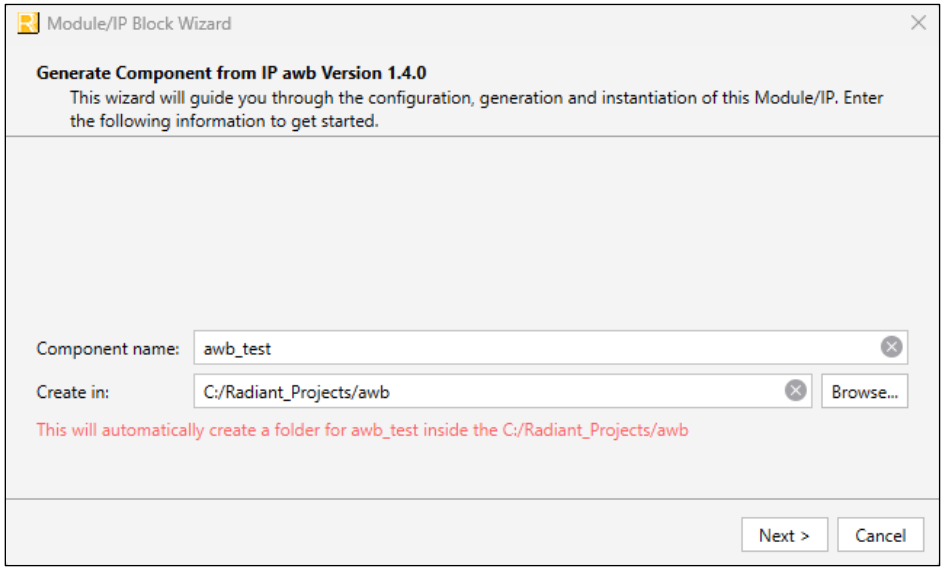

Figure 6.1. Module/IP Block Wizard

6. In the **Configure Component** dialog box, customize the selected AWB IP. As a sample configuration, see Figure 6.2. For configuration options, see Table 4.1.

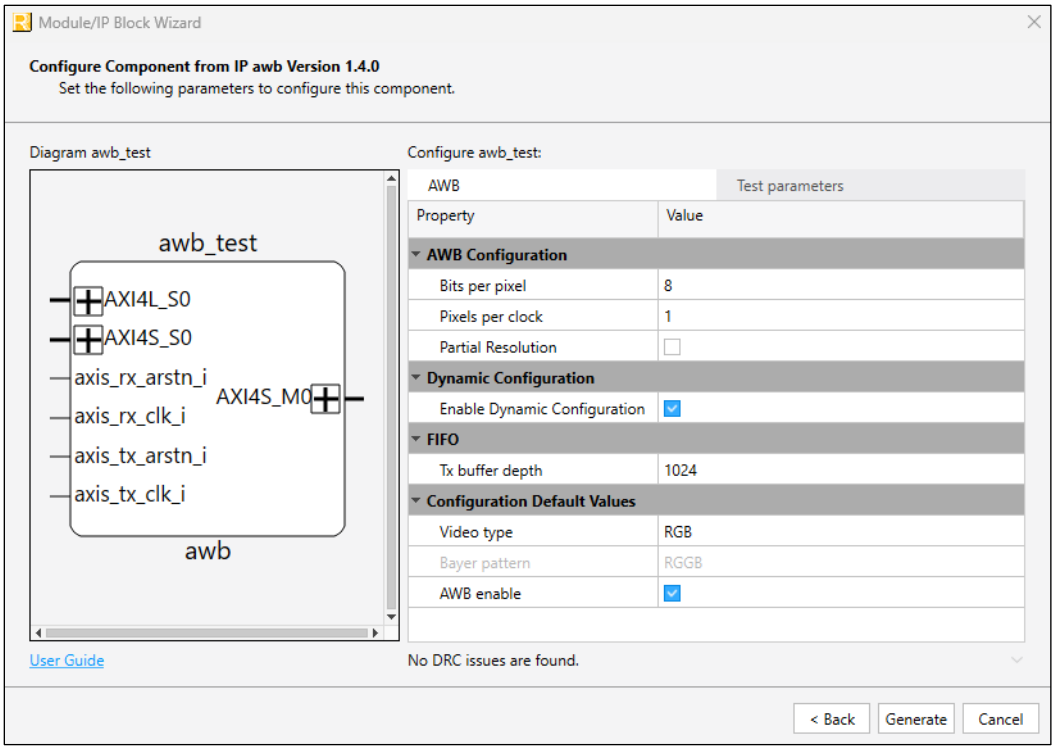

Figure 6.2. Configuration User Interface for AWB IP

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

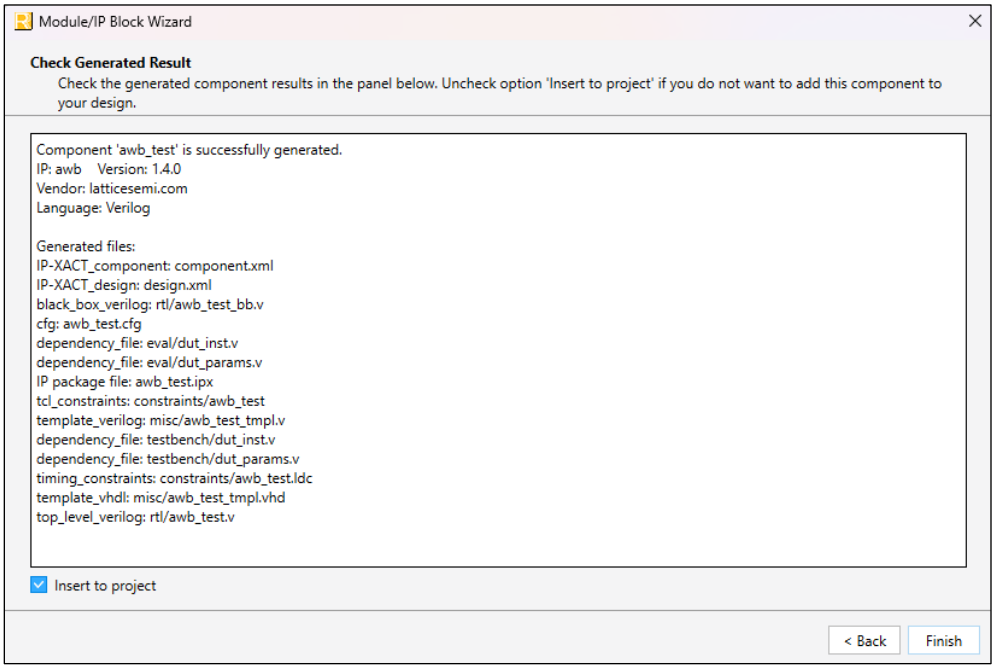

Figure 6.3. Check Generated Result

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

The generated AWB IP package includes the closed-box (*<Component name>_bb.v*). An example 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 for the complete design. The generated files are listed in Table 6.1.

Table 6.1. Generated File 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.
eval/<component_name>.pdc	This file contains post-synthesis constraints for evaluating the IP implementation flow. This may include virtual I/Os and system level clock constraints.
eval/*	This folder contains other files that may be used to evaluate the IP, including *sty (implementation strategy file), or files that are used by the testbench for simulation

6.3. Running Functional Simulation

Perform functional simulation after the IP is generated. Follow these steps.

1. Click the  button located on the toolbar to initiate the **Simulation Wizard** shown in Figure 6.4.

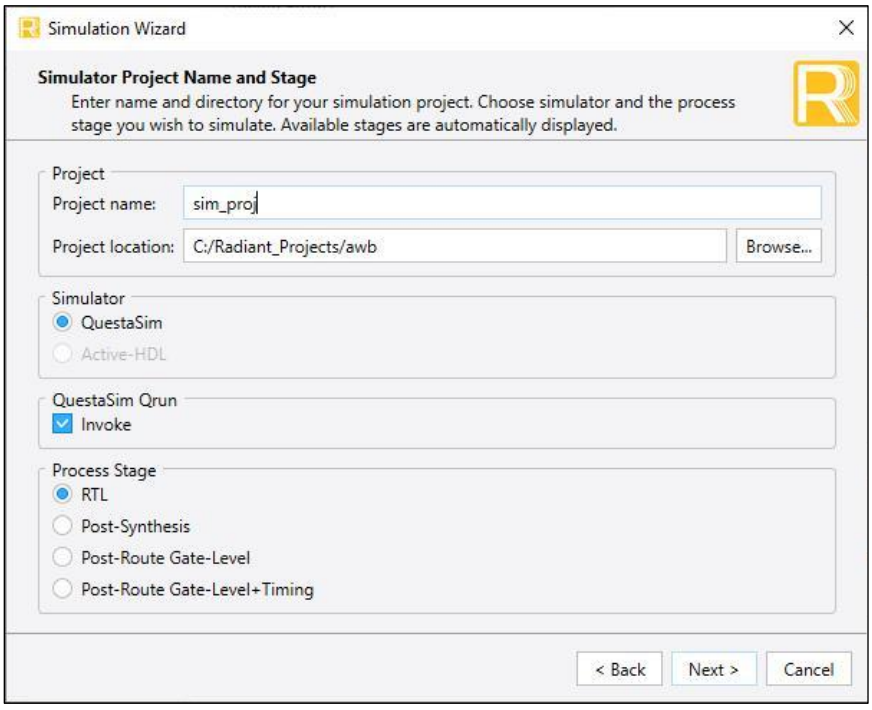

Figure 6.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** page as shown in Figure 6.5.

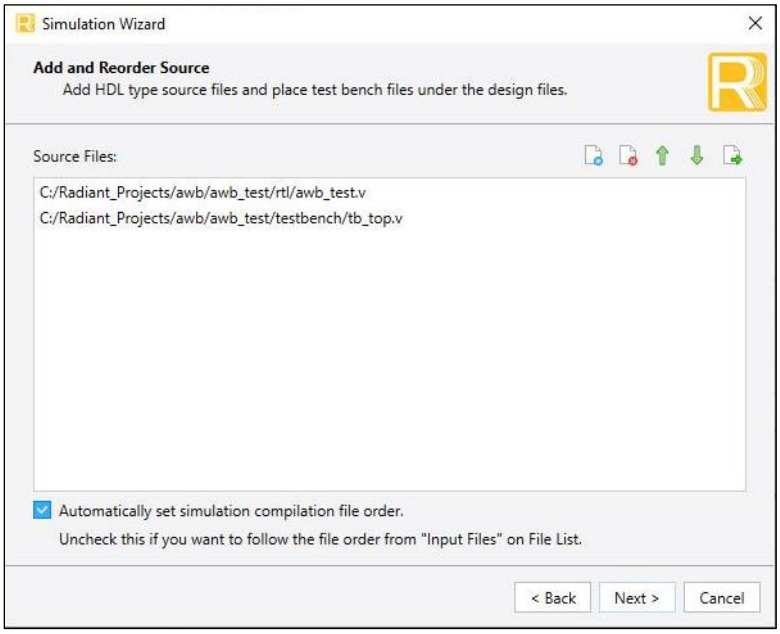

Figure 6.5. Adding and Reordering Source

3. Change the simulation time to **0** ns (equivalent to run -all).

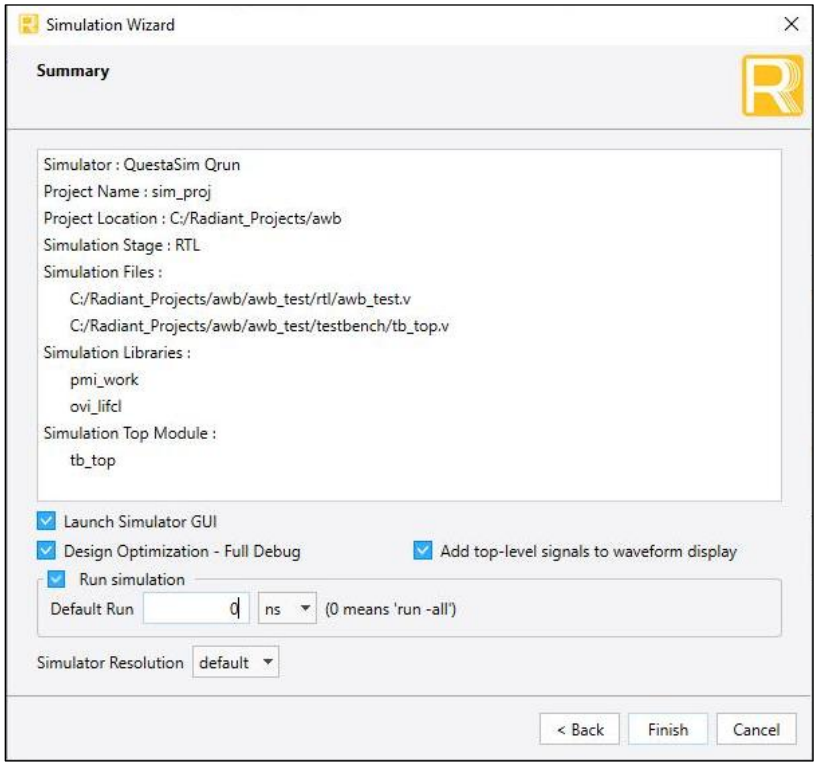

Figure 6.6. Run Simulation Value of 0 for Run All

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.

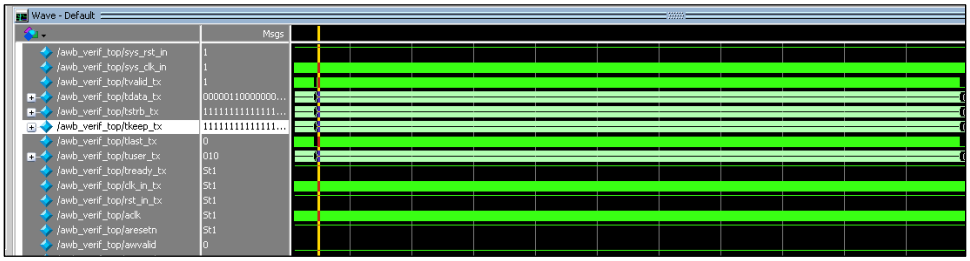

Figure 6.7. Sample Simulation Waveform

Note: The test bench also includes a data comparator and checker. Data check completed indicates correctness of the test.

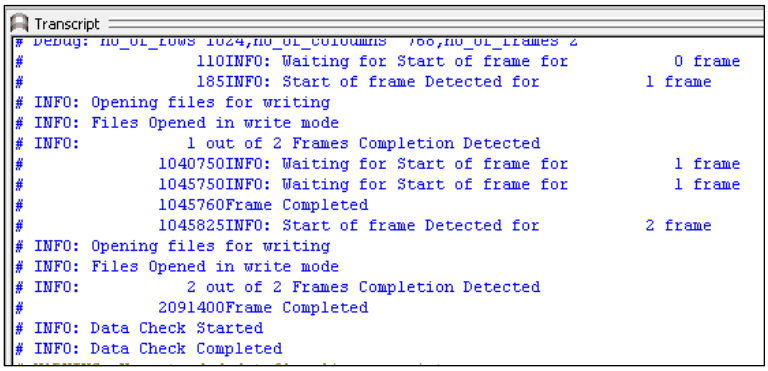

Figure 6.8. Data Check Passed

7. Design Considerations

7.1. Limitations

- When using the Lattice Synthesis Engine, ensure that the working directory does not contain spaces that can cause a synthesis error.
- This IP does not support oversampling.
- Selected IP configurations are supported only on target devices that provide a sufficient number of DSP blocks, as shown in Appendix A. Ensure that sufficient DSP blocks are available for the design to fit within the target device.

Appendix A Resource Utilization

The following tables show the resource utilization for a few select configurations for a Nexus device and Avant device at the lowest and highest speed grades. The results are based on the Synopsys Synplify Pro synthesis tool and the Lattice Radiant software 2025.2.

Table A.1. Resource Utilization using LFCPNX-100-7LFG672I — CertusPro-NX

Configuration	Clock RX Max (MHz)	Clock TX Max (MHz)	AXI-Lite Clock Max (MHz)	Registers	LUTs	DSP	sysMEM EBRs
8 bits per pixel 1 pixel per clock	153.54	138.08	200	1,258	2,009	11	5
8 bits per pixel 2 pixels per clock	158.76	137.50	200	1,485	2,341	28	7
8 bits per pixel 4 pixels per clock	150.13	136.76	200	1,924	2,870	20	13
16 bits per pixel 1 pixel per clock	151.63	136.00	200	1,518	2,226	11	7
16 bits per pixel 2 pixels per clock	148.85	142.53	200	1,999	2,802	14	13
16 bits per pixel 4 pixels per clock	148.65	135.35	200	2,888	3,925	20	23

Notes:

1. Fmax is generated using multiple iterations of Place and Route.
2. Fmax is generated when the FPGA design only contains Automatic White Balance IP core. These values may be reduced when user logic is added to the FPGA design.
3. The *distributed RAM* utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distributed among *logic*, *distributed RAM*, and *ripple logic*.

Table A.2. Resource Utilization using LAV-AT-X30-1LFG676I — Avant

Configuration	Clock RX Max (MHz)	Clock TX Max (MHz)	AXI-Lite Clock Max (MHz)	Registers	LUTs	DSP	sysMEM EBRs
8 bits per pixel 1 pixel per clock	230.73	227.38	250	1,559	2,104	9	3
8 bits per pixel 2 pixels per clock	219.35	249.50	250	1,765	2,429	12	5
8 bits per pixel 4 pixels per clock	220.344	233.37	250	2,309	3,043	18	7
16 bits per pixel 1 pixel per clock	237.59	250	250	1,823	2,343	9	5
16 bits per pixel 2 pixels per clock	215.52	246.85	250	2,418	3,058	12	7
16 bits per pixel 4 pixels per clock	228.62	236.24	250	3,480	4,320	18	13

Notes:

1. Fmax is generated using multiple iterations of Place and Route.
2. Fmax is generated when the FPGA design only contains Automatic White Balance IP core. These values may be reduced when user logic is added to the FPGA design.
3. The *distributed RAM* utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distributed among *logic*, *distributed RAM*, and *ripple logic*.

Table A.3. Resource Utilization using LFCPNX-100-9LFG672I— Certus-NX

Configuration	Clock RX Max (MHz)	Clock TX Max (MHz)	AXI-Lite Clock Max (MHz)	Registers	LUTs	DSP	sysMEM EBRs
8 bits per pixel 1 pixel per clock	198.30	177.02	200	1,258	2,006	11	5
8 bits per pixel 2 pixels per clock	190.37	170.42	200	1,485	2,340	14	7
8 bits per pixel 4 pixels per clock	176.49	178.41	200	1,929	2,883	20	13
16 bits per pixel 1 pixel per clock	195.274	163.80	200	1,524	2,246	11	7
16 bits per pixel 2 pixels per clock	189.97	157.70	200	1,999	2,808	14	13
16 bits per pixel 4 pixels per clock	174.70	159.49	200	2,888	3,917	20	23

Notes:

1. Fmax is generated using multiple iterations of Place and Route.
2. Fmax is generated when the FPGA design only contains Automatic White Balance IP core. These values may be reduced when user logic is added to the FPGA design.
3. The *distributed RAM* utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distributed among *logic*, *distributed RAM*, and *ripple logic*.

Table A.4. Resource Utilization using LAV-AT-X30-3LFG676I — Avant

Configuration	Clock RX Max (MHz)	Clock TX Max (MHz)	AXI-Lite Clock Max (MHz)	Registers	LUTs	DSP	sysMEM EBRs
8 bits per pixel 1 pixel per clock	250	250	250	1,559	2,104	9	3
8 bits per pixel 2 pixels per clock	250	250	250	1,765	2,429	12	5
8 bits per pixel 4 pixels per clock	250	250	250	2,309	3,043	18	7
16 bits per pixel 1 pixel per clock	250	250	250	1,823	2,343	9	5
16 bits per pixel 2 pixels per clock	250	250	250	2,418	3,058	12	7
16 bits per pixel 4 pixels per clock	250	250	250	3,480	2,771	18	13

Notes:

1. Fmax is generated using multiple iterations of Place and Route.
2. Fmax is generated when the FPGA design only contains Automatic White Balance IP core. These values may be reduced when user logic is added to the FPGA design.
3. The *distributed RAM* utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distributed among *logic*, *distributed RAM*, and *ripple logic*.

References

- Automatic White Balance IP Release Notes (FPGA-RN-02055)
- 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

Note: In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

Revision 1.4, IP v1.4.1, June 2026

Section	Change Summary
All	• Updated IP version to *1.4.1.* • Minor editorial fixes.
Abbreviations in This Document	• Replaced acronyms with abbreviations. • Updated the definition of BPP from *Bits Per Pixel Color Component* to *Bits Per Pixel*.
Introduction	Added LFD2NX-15P, LFD2NX-25P, and LFMXO5-25P to the supported devices in Table 1.1. Quick Facts.

Functional Description	• Corrected the following signal names in the Write Operation subsection.
	  • *aw_ready_i* to *aw_ready_o*
	  • *w_ready_i* to *w_ready_o*
	• Corrected signal name ar_ready_i to ar_ready_o in the Read Operation subsection.

Signal Description	• Corrected the signal names in the *tx_tuser_o* description from the *rx_tuser_i* form to the proper *tx_tuser_o* form in Table 3.2. AWB IP Signal Description. • Added *TDATA* to rx_tstrobe_i and tx_tstrobe_o descriptions in Table 3.2. AWB IP Signal Description.
Design Considerations	Added a new limitation stating that the *Selected IP configurations are supported only on the target devices that provide a sufficient number of DSP blocks, as shown in Appendix A. Ensure that sufficient DSP blocks are available for the design to fit within the target device.*

Revision 1.3, IP v1.4.0, December 2025

Section	Change Summary
All	• Added a note on IP version in Quick Facts and *Revision History* sections. • Performed minor formatting and editorial edits.
Acronyms in This Document	Updated list of acronyms.
Introduction	Updated Table 1.1. Quick Facts as follows: • Added Certus-N2 devices. • Added IP version. • Removed earlier IP versions.

IP Generation, Simulation, and Validation	• Added a note on IP version in GUI in the IP Generation, Simulation, and Validation section.
	• Updated the Licensing the IP section.
	• Updated the following figures:
	  • Figure 6.1. Module/IP Block Wizard
	  • Figure 6.2. Configuration User Interface for AWB IP
	  • Figure 6.3. Check Generated Result

Design Considerations	Mentioned that this IP does not support oversampling in the Limitations section.
Resource Utilization	Updated resources utilization for the latest software version.

Revision 1.2, IP v1.3.0, July 2025

Section	Change Summary
Disclaimers	Updated disclaimers.
Inclusive Language	Added inclusive language boilerplate.
Introduction	• Updated the Introduction section. • Updated Table 1.1. Quick Facts as follows:

Section	Change Summary
	• Renamed *Supported FPGA Families* to *Supported Devices*. • Removed the *Targeted Devices* row. • Added IP changes. • Added IP version.

IP Generation, Simulation, and Validation	• Updated the Generation and Synthesis section as follows:
	  • Updated the steps to refer to the guidelines when using the Lattice Synthesis Engin to generate the Automatic White Balance IP.
	  • Changed *black box* to *closed-box*.
	• Updated the following figures:
	  • Figure 6.1. Module/IP Block Wizard
	  • Figure 6.2. Configuration User Interface for AWB 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 this section.
Resource Utilization	Updated resource utilization for the latest software version.
Reference	Updated references.

Revision 1.1, December 2022

Section	Change Summary
Introduction	Updated Table 1.1. Quick Facts for below: • Added Lattice Avant to Supported FPGA Family. • Added LAV-AT-500E to Targeted Devices. • Updated IP version from v1.1.0 – Lattice Radiant™ software 3.2 or later to IP v1.1.0 – Lattice Radiant™ software 2022.1.
Functional Description	• Changed bullet from *The input rx_tlast_i indicates the end of a line. This must be asserted during the last pixel of each line* to *The input rx_tlast_ indicates end of a Total line. This must be asserted during the last pixel of horizontal blanking.* • Updated Figure 2.2. AXI-Stream Receiver Signals for PPC =1 and BPP=8 (RGB Format), Figure 2.3. AXI-Stream Receiver Signals for PPC=4 and BPP=8 (RGB Format), Figure 2.4. AXI-Stream Receiver Signals for PPC =1 and BPP=16 (RGB Format), Figure 2.15. Packing of Color Components in Bayer Pattern (RGGB) for BPP=8 and PPC=1, Figure 2.16. AXI-Stream Transmitter with PPC=1 and BPP =8, and Figure 2.17. AXI-Stream Transmitter with PPC=4 and BPP = 16 for RGB Pattern. • Updated from *The output tx_tlast_o indicates end of a line. This is asserted during the last pixel of each line.* to *The output tx_tlast_o indicates end of a total line. This is asserted during the last pixel of entire line (Horizontal Blanking)*.

Signal Description	• Changed values of Parameters AXI_LITE_ADDR_WIDTH, AXI_LITE_DATA_WIDTH from 4 to 2 and 16 to 32 in Table 3.1. Description of Width Parameters.
	• Updated Table 3.2. AWB IP Signal Description:
	  • Added Ports *axis_rx_arstn_i*, *axis_tx_arstn_i*, *axis_rx_clk_i*, and *axis_tx_clk_i* in Table 3.2. AWB IP Signal Description.
	  • Deleted ports *axo_stream_rstn_i* and *axi_stream_clk_i*.

Attributes Summary	Updated Table 4.1. Attributes Table and Figure 4.1. Configuration User Interface for AWB IP.
Register Description	Updated Offsets and Bits and deleted Table Note in Table 5.1. Summary of Configuration and Status Registers.

IP Generation, Simulation, and Validation	• Updated AWB Version from 1.0.0 to 1.1.0 in section 6.2. Generation and Synthesis.
	• Deleted below steps in Section 6.3. Running Functional Simulation:
	  • *Click Browse and choose the project folder. This is the folder with the user specified name. The appropriate user IP folder must be selected in this step for the simulation to run correctly.*
	  • *The folder location is indicated in Project location. Enter a name for the test project in*

Section	Change Summary
	Project name. • Added Figure 6.1. Module/IP Block Wizard, Figure 6.2. Configuration User Interface for AWB IP, and Figure 6.3. Check Generating Result. • Deleted Figure 6.5. Project Folder and Figure 6.6. Simulation Wizard.
Appendix A. Resource Utilization	• Updated Table A.1. Resource Utilization using LFCPNX-100-8BBG484I – CertusPro-NX, Table A.2. Resource Utilization using LAV-AT-500E-2LF1156I – Avant, and Table A.3. Resource Utilization using LFD2NX-40-8BG256I - (Certus-NX). • Added Table A.4. Resource Utilization using LIFCL-40-8BG400I- (Crosslink-NX).
Reference	Added web page for Lattice Avant.

Revision 1.0, September 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com