Video Frame Buffer IP

IP Version: v1.7.0

User Guide

FPGA-IPUG-02137-1.6

December 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

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AMBA	Advanced Microcontroller Bus Architecture
AXI	Advanced eXtensible Interface
BPC	Bits Per Color
BPP	Bits Per Pixel
CPP	Colors Per Pixel
CPU	Central Processing Unit
DDR	Double Data Rate
DRAM	Dynamic Random Access Memory
EBR	Embedded Block RAM
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HDL	Hardware Description Language
IP	Intellectual Property
LUT	Look-Up Table
PDC	Physical Design Constraint
PLL	Phase-Locked Loop
PPC	Pixels Per Clock
RAM	Random Access Memory
RGB	Red Green Blue
RTL	Register Transfer Level
UVSI	Unified Video Streaming Interface
YCbCr	Luminance, Chrominance (Y, Cb, Cr)

1. Introduction

1.1. Overview of the IP

The Video Frame Buffer IP Core buffers video data in external memory to be displayed on output devices such as computer monitors, projectors, and others. The Video Frame Buffer IP Core supports image sizes up to 4K x 4K with YCbCr 4:2:2, 4:4:4 and RGB video formats. The IP supports dynamic parameter updating through the AXI4-Lite interface or native parameter bus interface, which can be configured to operate on a different clock from the IP. Simple frame rate conversion is employed to support different input and output frame rates.

1.2. Quick Facts

Table 1.1 presents a summary of the Video Frame Buffer IP Core.

Table 1.1. Quick Facts

IP Requirements	Supported Devices	CrossLink™-NX, Certus™-NX (LFD2NX-9, LFD2NX-15, LFD2NX-17, LFD2NX-25, LFD2NX-28, LFD2NX-40), CertusPro™-NX, Lattice Avant™, Certus-N2
	IP Changes[1]	For a list of changes to the IP, refer to the Video Frame Buffer IP Release Notes (FPGA-RN-02042).
Resource Utilization	Supported User Interface	AXI4 Memory Interface, AXI4-Lite Interface, Universal Video Streaming Interface, Native Memory Interface, Native Parameter Bus Interface, Native Video Interface
	Resources	See the Resource Utilization section
Design Tool Support	Lattice Implementation	IP Core v1.7.0 – Lattice Radiant™ software 2025.2
	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.

1.3. IP Support Summary

Table 1.2. Video Frame Buffer IP Support Readiness for CertusPro-NX Devices

Only AXI4 interface, AXI4-Lite interface, and Unified Video Streaming interface are used in hardware validation.

Video Format	Video Frame Width x Height	Bit per Color	iclk oclk mem_clk (MHz)	Frame Rate Conversion	AXI Data Width	Memory Base Address	DDR Memory Burst Length	Command Burst Count	Radiant Timing Model	Hardware Validated
YCbCr4:2:2	64 x 64	8	200 — 150	Off	32	16'h0000	8	1	Final	Yes
	1920 x 1080[1]	8	200, 150 150 150	On	32	16'h0000	8	1	Final	Yes
	2560 x 1080	12	200, 20 120, 200 150	On	32	16'h0000	8	1	Final	Yes

Video Format	Video Frame Width x Height	Bit per Color	iclk oclk mem_clk (MHz)	Frame Rate Conversion	AXI Data Width	Memory Base Address	DDR Memory Burst Length	Command Burst Count	Radiant Timing Model	Hardware Validated
	2560 x 1080	12	200 200 150	On	32	16'h0000	2	2	Final	Yes
YCbCr4:4:4 or RGB	64 x 64	10	200 — 150	Off	32	16'hFFF8	2	2	Final	Yes
		16	200 — 150	Off	64	16'h0000	8	1	Final	Yes
		16	200 — 150	Off	64	16'hFFF0	2	2	Final	Yes
		16	200 200 150	On	64	16'h0000	8	1	Final	Yes
	1920 x 1080	8	90, 150, 20 150, 90, 200 150	On	32	16'h0000	8	1	Final	Yes
		8	150 150 150	On	32	16'h0000	2	2	Final	Yes
Single Color	4096 x 2160	10	200, 20 120, 200 150	On	32	16'h0000	8	1	Final	Yes
	4096 x 2160	16	200 200 150	On	64	16'h0000	2	2	Final	Yes
YCbCr4:2:2	1920 x 1080	8	200 200 150	Off	128	16'h0000	4	2	Final	No
YCbCr4:4:4 or RGB	2560 x 1080	12		Off	128	16'h0000	4	2	Final	No
Single Color	4096 x 2160	10		Off	128	16'h0000	4	4	Final	No

Note:

1. Tested Register Access (AXI4-Lite) using pclk_i at 20 MHz and 100 MHz.

1.4. Features

The key features of Video Frame Buffer IP Core include:

- Supports single color, YCbCr 4:2:2, YcbCr 4:4:4, and RGB video formats
- Supports input and output resolutions of 64 × 64 to 4 k × 4 k pixels
- Supports serial and parallel pixel processing
- Supports frame rate conversion
- Supports dynamic parameter update of frame size
- Supports configurable parameter bus clock
- Supports configurable memory bus width and base address
- Supports configurable memory burst length and burst count

- Configurable internal FIFO type and depth
- Supports 8, 10, 12, or 16-bit color depth per plane
- Supports optional AXI4-Lite interface for register access
- Supports optional AXI4 interface for memory interface
- Supports optional Unified Video Streaming Interface (UVSI)

1.5. Licensing and Ordering Information

The Video Frame Buffer IP is provided at no additional cost with the Lattice Radiant software.

1.6. Hardware Support

Refer to the Example Design section for more information on the boards used.

1.7. Minimum Device Requirements

Refer to the Resource Utilization section.

1.8. Naming Conventions

1.8.1. Nomenclature

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

1.8.2. 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

1.8.3. Host

The logic unit inside the FPGA interacts with the Video Frame Buffer IP Core.

1.8.4. Attribute

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

2. Functional Description

2.1. IP Architecture Overview

The Video Frame Buffer IP Core receives input video data, stores it in the external memory and outputs it based on the timing controlled by the dout_enable_i or axis_vid_tready_i signal. The IP stores data into memory in different width format as required by the application. The IP also provides synchronization of data across different clock domains and different format domains.

The IP provides a simple parameter bus for dynamic frame size updating. It also implements a flexible memory interface which can be connected to Lattice memory controller IP cores.

An optional AMBA bus interface is also provided — AXI4-Lite for parameter bus, Unified Video Streaming Interface for video interface, and AXI4 for memory controller interface.

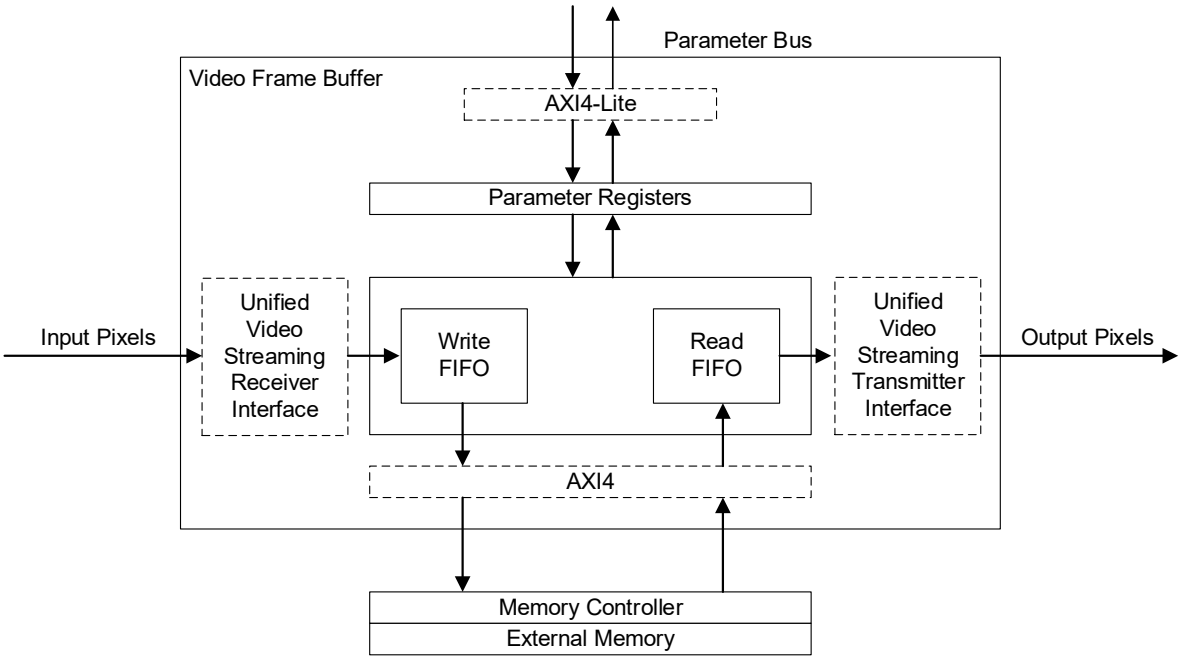

Figure 2.1. Video Frame Buffer IP Core Functional Diagram

The IP supports continuous data streams from and to external interfaces in different clock domains using asynchronous Write and Read FIFOs. Input pixels are packed and stored into the asynchronous double clock Write FIFO first. The pixels are then sent to an external memory controller to be written to the memory. After an entire video frame has been stored in the external memory, frame reading starts. The pixels read from external memory are stored in the asynchronous Read FIFO and transferred to output interface clock domain. After unpacking, pixels are output from the Video Frame Buffer IP Core. In the video frame buffer, several clock sources are involved. The memory interface operates on a separate memory clock. When frame rate conversion is enabled, there are two clocks in the video data path: input pixel sample clock and output pixel sample clock. When frame rate conversion is disabled, the video data path operates at input pixel sample clock rate. The parameter bus runs on a separate parameter bus clock.

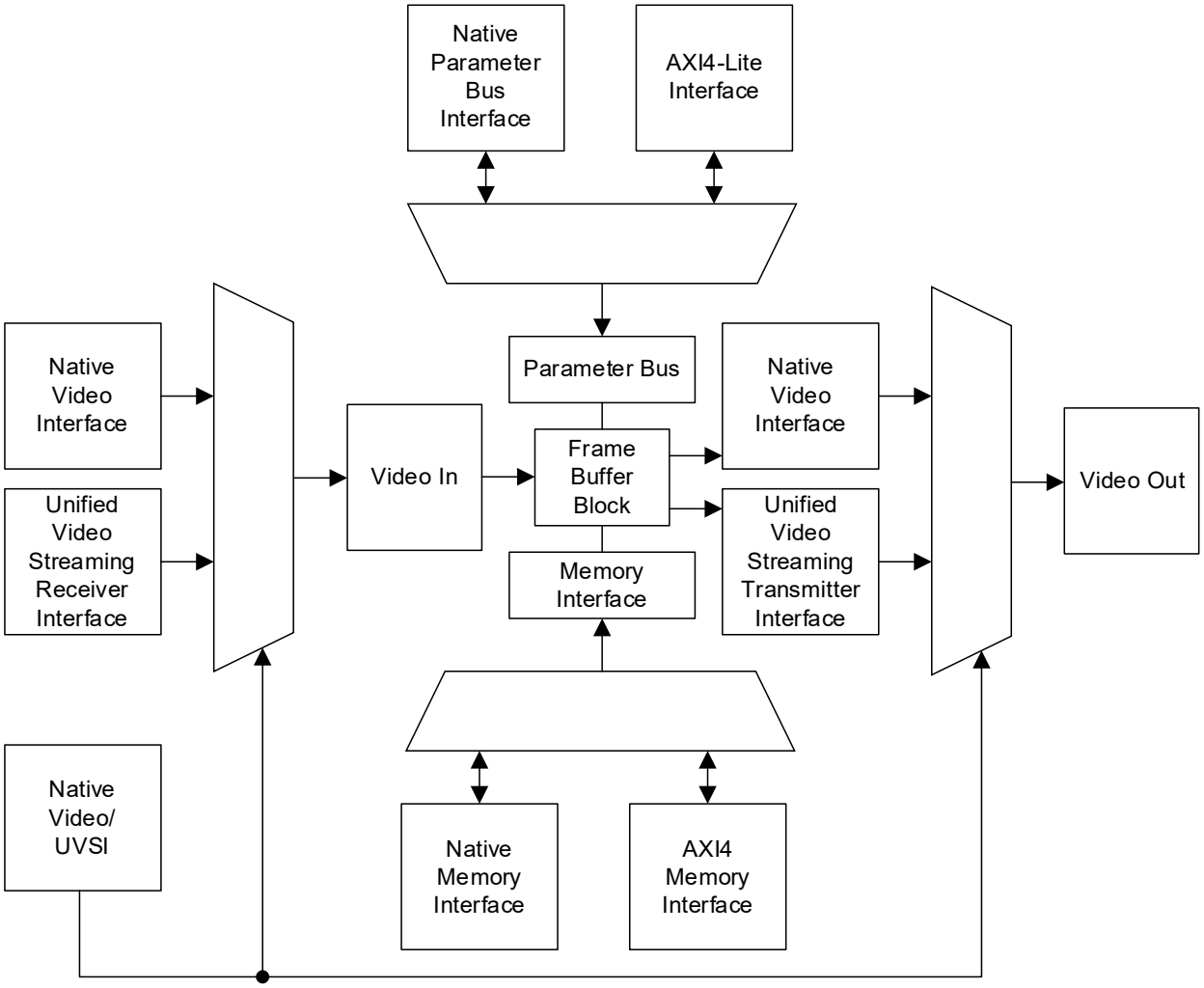

Figure 2.2. Video Frame Buffer IP Core I/O

2.2. Frame Rate Conversion

The Video Frame Buffer IP Core provides a simple frame rate conversion. When frame rate conversion is enabled, the IP output data path runs at the output pixel clock rate. The output frame rate is controlled by the output pixel sample clock and dout_enable_i or axis_vid_tready_i signal. When there is no new video frame, the IP outputs the last frame repeatedly. When frame rate conversion is ON, the IP requires to store 2 frames first before the IP begins to output frames. When frame rate conversion is disabled, the IP output data path runs on the input pixel sample clock. The output frame is generated directly from the input video stream. If there is no new input video frame, the IP stops generating output data.

2.3. Dynamic Parameter Updating

The Video Frame Buffer IP Core provides AXI4-Lite parameter bus and native parameter bus ports for internal parameter update at run time. The parameters are double-buffered to avoid adverse effect to the IP when they are changed. The new values are buffered and transferred to the working registers when the IP is ready to accept a new configuration. The UPDATE register is used to indicate when the new values are consumed and when the buffers can accept new data.

When AXI4-lite is enabled, dynamic parameter updating is allowed through control and status registers. Table 5.1 shows the registers for configuration of the IP.

All the parameter registers (Frame Height and Frame Width) can be written to only when the UPDATE register bit is 0. When the UPDATE bit is set to 1, the parameters FRMWIDTH and FRMHEIGHT take effect when the frmsync_in_i or axis_vid_tuser[0] signal is active, indicating a new input frame is arriving. After updating the internal parameters, the IP resets the UPDATE bit to indicate that the parameter registers are now empty and can take on new values.

2.4. Memory Bandwidth and Size

The Video Frame Buffer IP Core stores and retrieves pixels to and from the external memory using memory burst write and read commands. When native memory interface is enabled and DDR2 memory is used for external memory, one burst operation with the number of bits equals (burst_length × burst_count / 2) × memory_data_width bit, cannot exceed the size of a single video line. A single video line is transferred through multiple burst write/read transactions internally.

When frame rate conversion is inactive, the Video Frame Buffer IP Core needs a two-frame memory storage space; when frame rate conversion is active, a three-frame storage space is required. The total external memory size the IP requires can be viewed on the Video Frame Buffer IP user interface.

The required memory bandwidth when using native memory interface is input pixel data rate plus output pixel data rate.

For example, for parallel 8-bit YCbCr 4:2:2 pixels, if the input pixel sample clock is 74.25 MHz and output pixel sample clock is 148.5 MHz, the required bandwidth is 2 × 8 × (74.25 + 148.5) = 3564 bit × MHz. If the memory data width is 32, the required memory clock is (3564 / 32) = 111.375 MHz.

2.5. Clocking

2.5.1. Video Input/Output Timing

The Video Frame Buffer IP Core supports single color, YCbCr 4:2:2, YCbCr 4:4:4, or RGB video format.

For YCbCr 4:4:4 or RGB video format in Native Video, the three planes are interleaved for serial processing and combined on the din_i and dout_o ports for parallel processing.

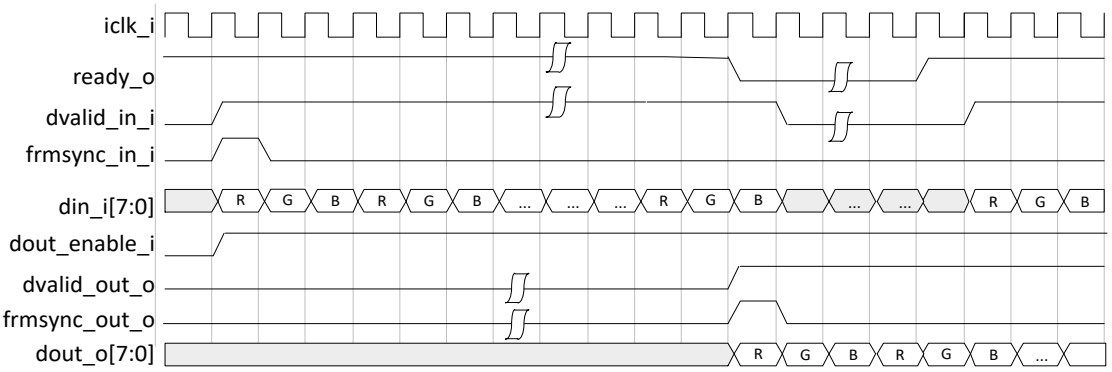

Figure 2.3. Timing Diagram of the RGB Serial Processing (8-bit Pixel)

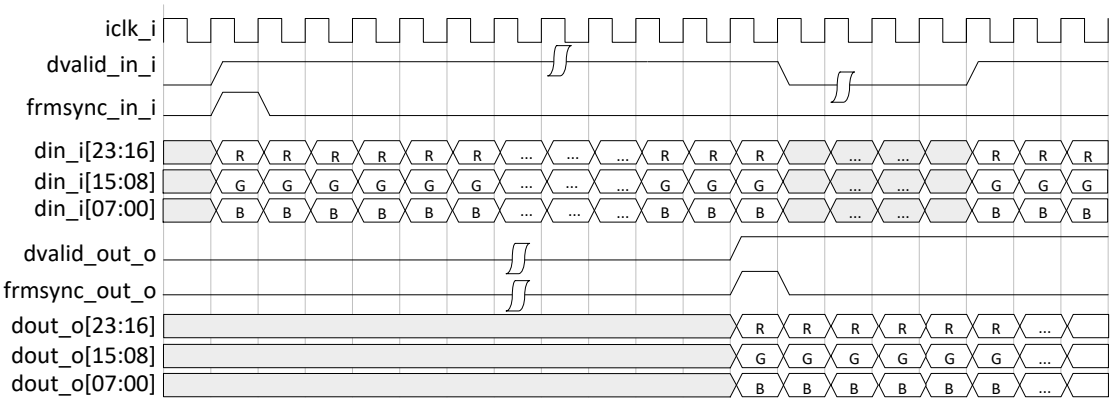

Figure 2.4. Timing Diagram of the RGB Parallel Processing (8-bit Pixel)

For YCbCr 4:2:2 video serial processing, the input and output sequence is Cb, Y, Cr, Y, ... For parallel processing, the Y plane occupies the upper bits of the din_i and dout_o ports, and the Cb and Cr planes the lower bits. Cb and Cr planes are interleaved in the lower half, and Cb comes before Cr.

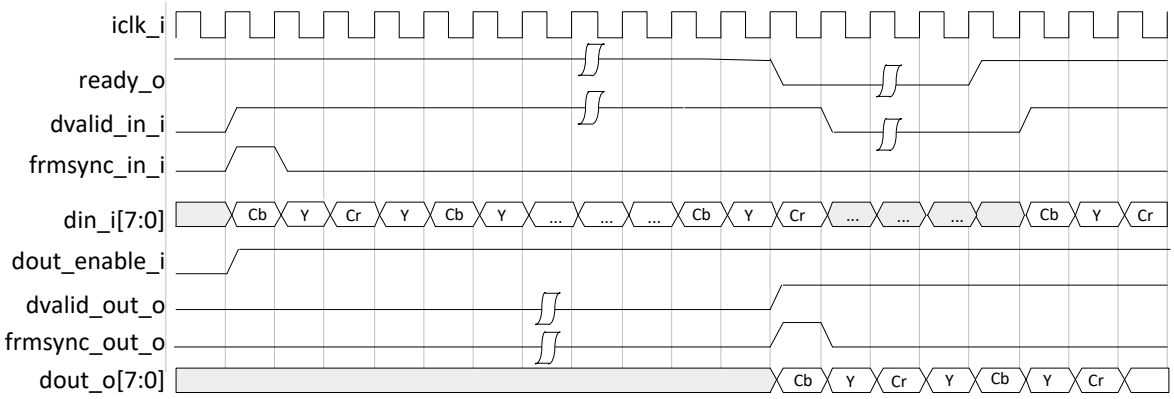

Figure 2.5. Timing Diagram of the YCbCr 4:2:2 Serial Processing (8-bit Pixel)

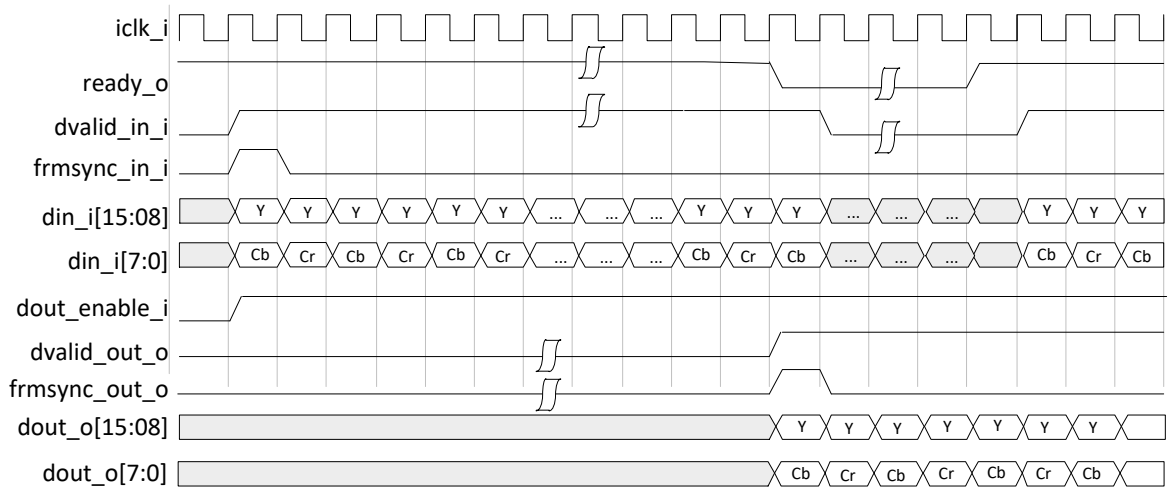

Figure 2.6. Timing Diagram of the YCbCr 4:2:2 Parallel Processing (8-bit Pixel)

When dout_enable_i is de-asserted, the IP stops outputting data. Similarly, when dout_enable_i is asserted, the IP begins outputting data. The assertion and de-assertion of dout_enable_i can be used to generate a horizontal blank and a vertical blank depending on the output video format.

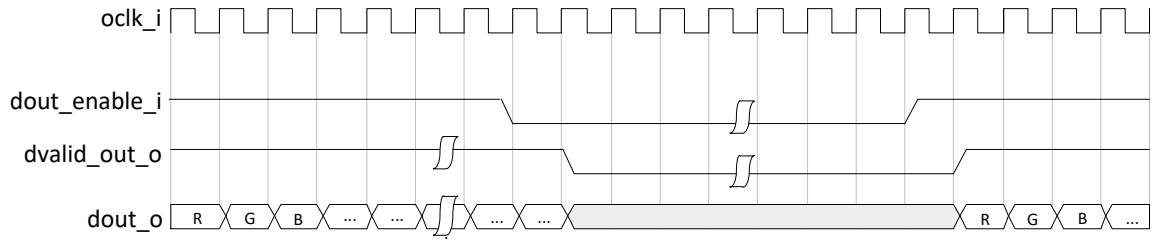

Figure 2.7. dout_enable_i Control Timing

2.5.2. Video Frame Timing

The following diagram shows the frame timing when frame rate conversion is disabled. After accepting one input frame, the IP starts generating frames. The output frame is driven by the input frame. When a new input frame arrives before the current output frame is finished, the current output frame is dropped and a new output frame starts. After the last input frame, the video frame buffer stops generating output frames. For details, refer to the Design Considerations section.

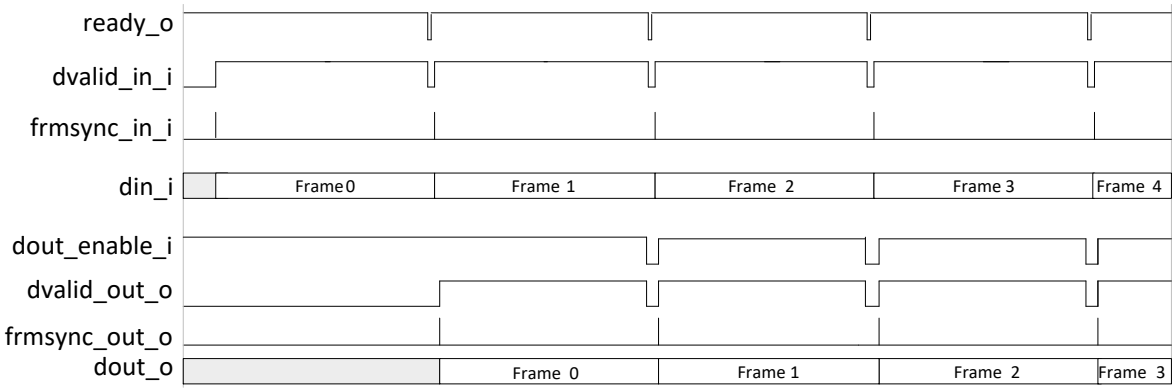

Figure 2.8. Output Frame Rate Same as Input Frame Rate

The following diagram shows the frame timing when the output frame rate is twice the input frame rate. Output frames run on a separate output pixel clock. The output frame rate is determined by the output pixel clock and dout_enable_i or axis_vid_tready_i signal. The IP generates output frames based on the oldest pending input frames. When a new input frame is received, the IP pushes out the oldest frame and exports the second oldest frame. When there is no new frame input, the IP exports the stored frames and then keeps exporting the last frame repeatedly. When the external memory is fully occupied by the unconsumed frames where there is no more space for new frames, the IP overwrites the latest input frame.

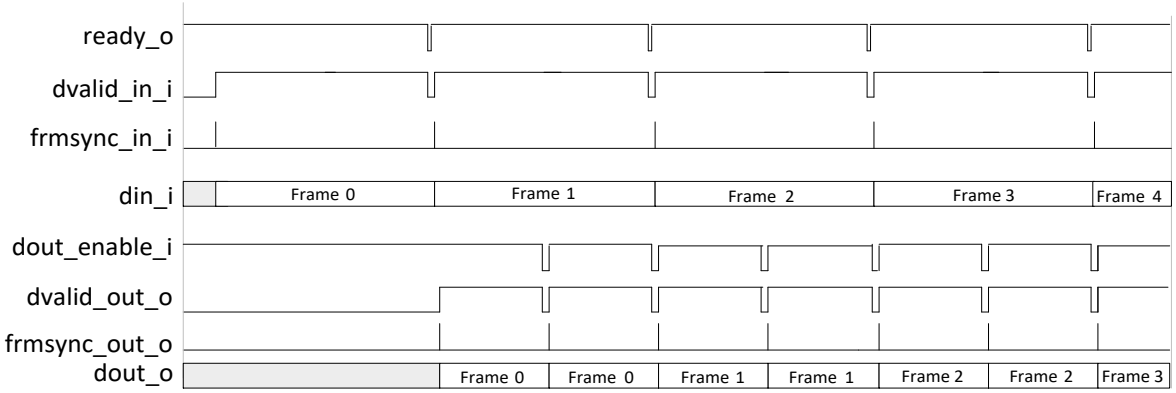

Figure 2.9. Output Frame Rate is Twice the Input Frame Rate

The following diagram shows the frame timing when the output frame rate is two times slower than the input frame rate. The Video Frame Buffer can store at most 2 frames at a time. The IP accepts only a new frame when the IP is complete writing at least one of the stored frames to the external memory, or when the IP is not holding any frames. When a new frame is sent to the IP when the output clock is slower and the IP is still holding 2 frames, the new frame is dropped.

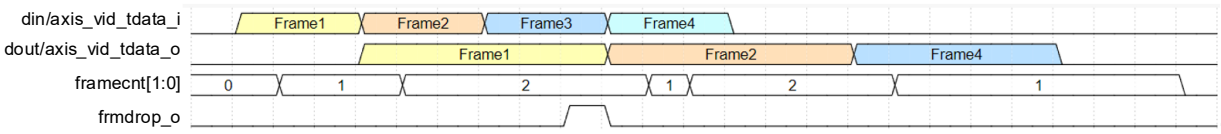

Figure 2.10. Frame Dropping when Frame Rate Conversion = 1

2.5.3. Memory Interface Timing

When the internal write FIFO is half full, the IP triggers memory write operation cycles. The memory write operation continues until the write FIFO is empty. Write FIFO width is the same as the data bus width which is set from the GUI. Data is stored to the write FIFO during the positive edge of the input pixel clock and read with the memory clock.

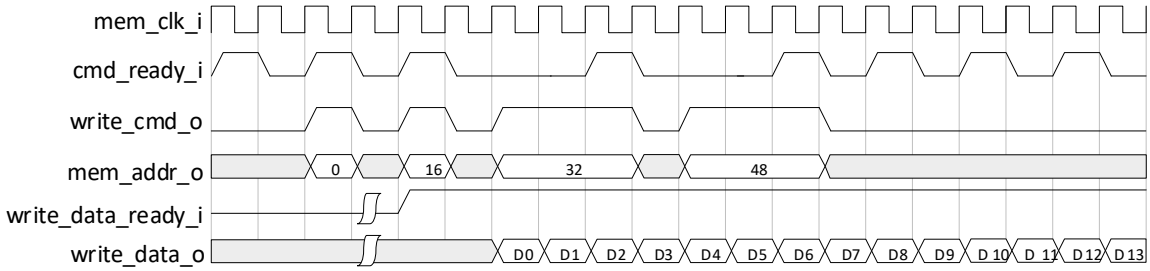

Figure 2.11. Timing Diagram for Memory Write Operation

When the internal read FIFO is less than half full, and there is no ongoing write operation, the memory read burst operation starts. Read burst operation continues until the read FIFO is half full.

The memory write and read operations are in bursts. The memory write and read operations switch at the end of the burst cycle.

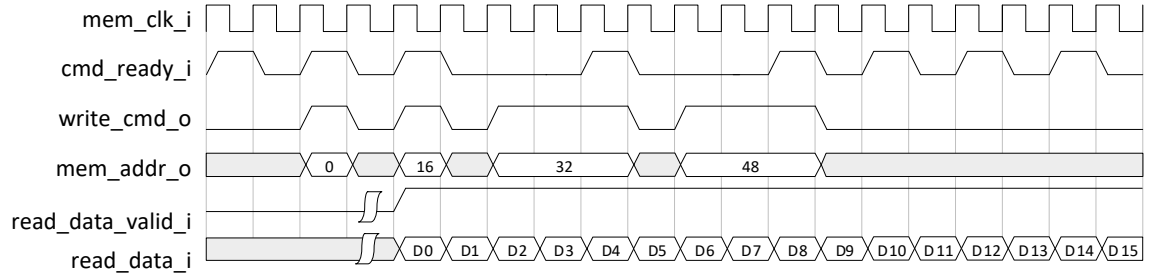

Figure 2.12. Timing Diagram for Memory Read Operation

The AXI4 Memory interface write operation is triggered whenever the Write FIFO holds any data. The memory write operation continues until the write FIFO is empty.

2.5.4. Dynamic Parameter Updating

In the following diagram, FW and FH are the video frame width and height.

The parameter bus can be configured to operate on a separate clock. By default, the bus operates at the input pixel clock rate. The parameter registers are writable only when the UPDATE register is 0. When the UPDATE register bit is set to 1, the IP updates the internal parameter registers with the new values at the next active frmsync_in_i or axis_vid_tuser_i[0] and reset the UPDATE register bit.

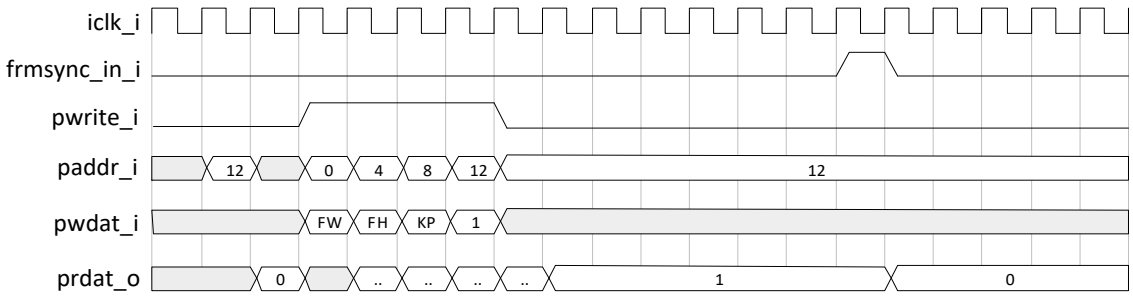

Figure 2.13. Timing Diagram for Dynamic Parameter Updating (Parameter Bus Width = 32)

2.5.5. Clocking and Reset Overview

The IP contains 2 resets — one active low asynchronous reset and an active high synchronous reset. The synchronous reset takes effect on the positive edge of the input pixel clock domain for signals in this clock domain. For signals in other clock domains, the synchronous reset takes effect at the third positive edge of the corresponding clock after two positive edges of the input pixel clock at the latest. The IP uses iclk_i and mem_clk_i for all configurations while output sample clock (oclk_i) is used only when frame rate conversion is on and is used to sample the read FIFO. The pclk_i clock input is a separate parameter bus clock used for dynamic parameter.

2.5.6. Clock Domains and Clock Domain Crossing

The following diagram shows the clock domain crossings of the IP when separate bus parameter clock is used. The read clock for the Read FIFO can either be from iclk_i or oclk_i clock domain. If the frame rate conversion option is turned on, the read clock for the Read FIFO is from the ock_i clock domain. The entire IP is controlled by the active low asynchronous reset and the active high synchronous reset.

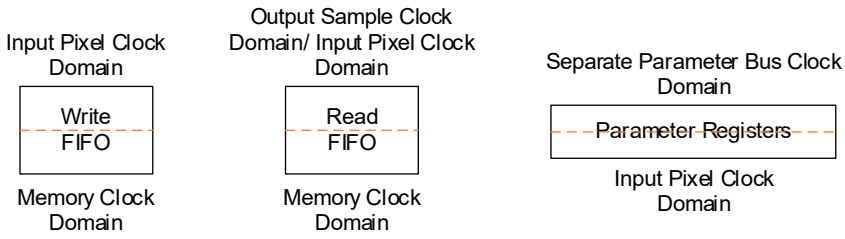

Figure 2.14. Clock Domains and Clock Domain Crossing Diagram

2.6. User Interfaces

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
Video interface	Native video Unified Video Streaming Interface	Video interface can either be set to Native Video which is described in the Video Input/Output section or through Unified Video Streaming Interface which is described in the Unified Video Streaming Interface section.

User Interface	Supported Protocols	Description
Memory interface	Native memory AXI4	Memory interface can either be set to Native Memory which is described in the Memory Interface section or through AXI4. For details, refer to the AMBA AXI protocol specification in the Arm Developer web page.
Dynamic parameter updating	Native parameter bus AXI4-Lite	When using dynamic parameter updating, the control and status registers can be written and read through either the Native Parameter Bus as described in the Parameter Register Read/Write Interface section or through AXI4-Lite. For details, refer to the AMBA AXI-Lite protocol specification in the Arm Developer web page.

2.6.1. Video Input/Output

The Video Frame Buffer IP Core uses a simple handshaking method to pass pixel data into and out of the IP. The IP asserts the ready output when the IP is ready to receive data. When the driving module has data to pass to the video frame buffer, the module asserts the IP dvalid_in_i port and at the same time placing the input video data on the din_i port. The frmsync_in_i input is driven to a 1 during the clock cycle when the very first active pixel is placed on the din_i bus.

Similarly, dvalid_out_o is active when valid output pixels are available on dout_o, and frmsync_out_o marks the first pixel in an output frame.

The input signal dout_enable_i enables the IP to generate output pixels. When dout_enable_i is low (inactive), the IP stops generating output pixels.

2.6.2. Memory Interface

The Video Frame Buffer IP Core implements a flexible memory interface which operates on a separate clock from the main core.

2.6.2.1. Native Memory Interface

The native memory interface supports the DDR2 and DDR3 memory controller. When connecting to DDR2 memory controller, the burst length and burst count of video frame buffer have the same values as the DDR2 memory controller. When connecting to DDR3 memory controller, the burst length of video frame buffer has the same value as the DDR3 memory controller, while the burst count of video frame buffer is half as that of DDR3 memory controller.

The video frame buffer assumes a memory byte addressing scheme. When connecting the memory controller, the address connection adjusts according to the DDR memory data width.

Assuming the DDR2 and DDR3 memory has 27 bits address width, the address connection can be as follows:

```
ddr_addr = {1'b0,mem_addr_o[25:0]}; // for 8-bit DDR memory only
ddr_addr = {2'b00,mem_addr_o[25:1]}; // for 16-bit DDR memory only
ddr_addr = {3'b000,mem_addr_o[25:2]}; // for 32-bit DDR memory only
```

The corresponding command is as follows:

```
ddr_cmd = (write_cmd_o) ? 4'b0010 : 4'b0001; // (write_cmd_o) ? WRITE : READ
ddr_cmd_valid = (write_cmd_o || read_cmd_o);
```

To get the best throughput (about 5% increase compared to normal connection), you must fine tune the combination of row, bank, or column address to get the best throughput.

To get maximum throughput on the memory bus (about 5% increase compared to normal connection), you need to steer clear of write-to-precharge, read-to-precharge, or precharge-to-active during row switching. You can access each bank in turn by using write or read with auto-precharge command to close current bank immediately after current burst write or read. To adopt this policy, you must fine tune the combination of row, bank, or column address to get the best throughput.

For example, when DDR2 memory data width is 16, row size is 14, column size is 10, bank size is 8, burst length is 8, and burst count is 1, which means the memory controller user interface data width is 32, row address is 14 bits, column

address is 10 bits, bank address is log2(bank size) = 3 bits and log2(burst length × burst count) = 3 bits. The memory controller address mapping inside the memory controller IP core is as follows:

```
ddr_addr = {row_addr[13:0], bank_addr[2:0], col_addr[9:0]}
```

To get the best throughput, first increase col_addr[2:0] to utilize the burst operation, then increase bank_addr[2:0], followed by col_addr[9:3], and finally the row_addr. The interface to the DDR2 and DDR3 memory is similar to:

```
ddr_addr = {2'b00, mem_addr_o[25:14], mem_addr_o[6:4], mem_addr_o[13:7],
mem_addr_o[3:1]};
```

The corresponding command is as follows:

```
ddr_cmd = (write_cmd_o) ? 4'b0100 : 4'b0011; // (write_cmd_o) ? WRITEA : READA
```

As the memory controller user interface data width is 32, mem_addr_o[1:0] is always zero.

The IP arbitrates between memory write and read operations, ensuring only a write_cmd_o or a read_cmd_o is asserted at any given time.

Two clock cycles after the write_data_ready_i is asserted, data becomes available on the write_data_o port. The parameter *Data_rdy to Write Data Delay* of memory controller must be set to 2.

The cmd_ready_i can be asserted once every two clock cycles, and must have at least a one-cycle interval, which is consistent with the Lattice DDR Memory Controller IP cores.

2.6.2.2. AXI4 Memory Interface

When the IP is configured to use AXI4 as a memory interface, the AXI4 signals are exposed in the top-level ports instead of the native memory interface. Refer to Table 4.1 for AXI burst length and burst size computation based on command burst count and DDR memory burst length you set. Refer to the AMBA AXI-Lite protocol specification in the Arm Developer web page for detailed information.

2.6.3. Parameter Register Read/Write Interface

The parameter bus data width is configured based on the system CPU's data width.

When the IP is configured to use AXI4-Lite as register interface, the AXI4-Lite signals are exposed in the top-level ports instead of the parameter register interface. Refer to the AMBA AXI-Lite protocol specification in the Arm Developer web page for detailed information.

2.6.4. Unified Video Streaming Interface

The Unified Video Streaming Interface is compliant with the AMBA AXI4-Stream Protocol Specification. This interface is only available when parallel processing is enabled. When Unified Video Streaming Receiver Interface is enabled, pixel data is received through the Unified Video Streaming Receiver Interface.

Refer to the AMBA AXI-stream protocol specification in the Arm Developer web page for detailed information.

The size of axis_vid_tdata_i and axis_vid_tdata_o depends on the value of TD_WD and PPC, where PPC is the Pixel Per Clock – this IP can only support 1 PPC, and TD_WD (axis_vid_tdata width) depends on the following attributes:

- BPP, Bits Per Pixel == CPP × BPC.
- CPP, Colors Per Pixel. Red, Green, and Blue for RGB. Luma, Blue minus Luma, and Red minus Luma for YCbCr. 3 is used for RGB and YCbCr 4:4:4, while 2 for YCbCr 4:2:2.
- BPC, Bits Per Color. Minimum width of each component is 8 -bits. This IP can support BPC of 8, 10, 12, and 16 bits.

The TD_WD is calculated as ceil(BPP/8) × 8 which is the data width per pixel. The size of axis_vid_tdata_i is calculated as TD_WD × PPC.

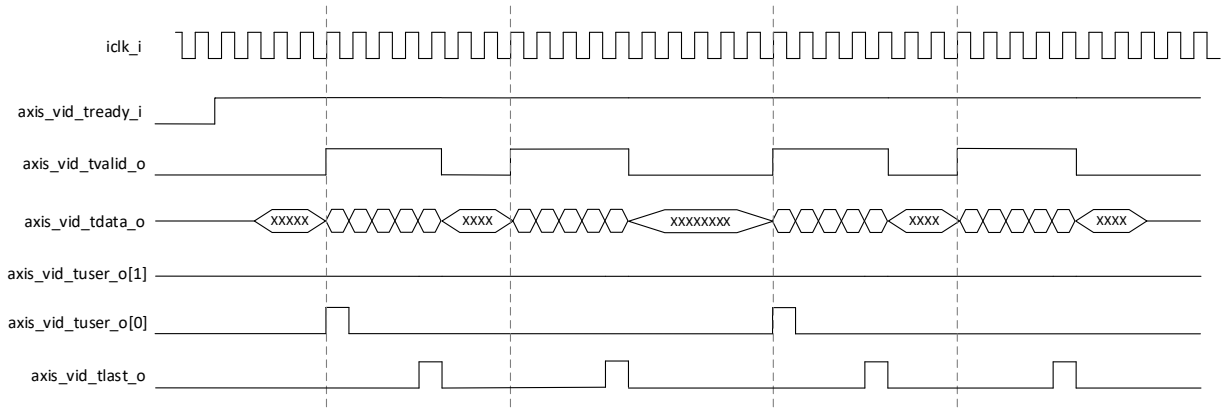

Figure 2.15. UVSI Tx and Rx Signals Example Waveform

Figure 2.15 shows the waveform when UVSI is enabled. The valid pixels are accompanied by axis_tvalid_o and the transaction is valid when handshake with axis_vid_tready_i is made. The waveform also shows the assertion of axis_vid_tuser_o [0] at start of frame and axis_vid_tlast_o for end of line. Horizontal blanking occurs after every line while vertical blanking occurs before every start of frame. For Video Frame Buffer, the axis_vid_tready_o signal deasserts after last pixel of row is received or write FIFO is almost full (Write FIFO Depth – 3 if Memory Bus Width / Data Port Width (TD_WD) <2 and Write FIFO Depth – 2 otherwise).

For axis_vid_tdata mapping, the video packet data is mapped across the TDATA bytes with the LSB of the first color component of the first pixel in bit 0. Byte align each pixel when mapping multiple pixels in parallel. When a pixel does not perfectly fill a given number of bytes, pad MSBs with don't care data. Refer to the following diagrams.

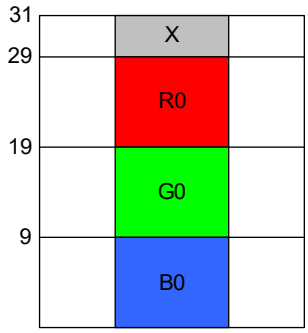

Figure 2.16. RGB with 10 BPC

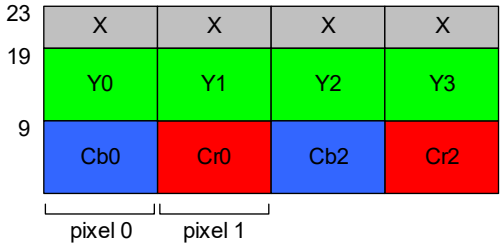

Figure 2.17. YCbCr422 with 10 BPC

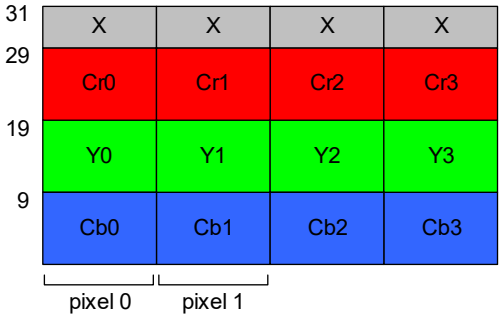

Figure 2.18. YCbCr444 with 10 BPC

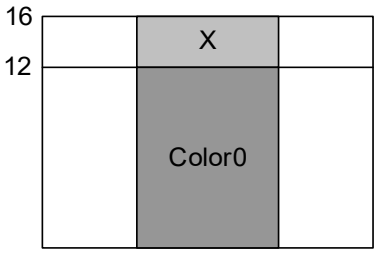

Figure 2.19. Single Color with 12 BPC

3. IP Parameter Description

The configurable attributes of the Video Frame Buffer IP Core are shown in Table 3.1. You can configure the IP by setting the attributes accordingly in the IP Catalog Module/IP Wizard of the Lattice Radiant software.

Wherever applicable, default values are in bold.

Table 3.1. Attributes Table

Attribute	Selectable Values	Description	Dependency on Other Attributes
Architecture			
Frame Dimensions			
Video format	Single color, **YCbCr4:2:2**, YCbCr4:4:4 or RGB	Defines the format of video stream.	—
Video frame width	64–4096, **720**	Selects the video frame width which is the horizontal number of pixels. Only values from 64 to 4096 are allowed.	—
Video frame height	64–4096, **480**	Selects the video frame height which is the horizontal number of pixels. Only values from 64 to 4096 are allowed.	—
Parallel processing	Off, **On**	Determines whether the IP processes video color planes in parallel.	Off when *Video format* is equal to Single color, editable otherwise.
Dynamic Parameter Updating	**Off**, AXI4-lite, Native Parameter Bus	Determines whether the IP supports parameters updating at runtime. Refer to the Dynamic Parameter Updating section for more information. When *dynamic parameter updating* is enabled, the Max video frame width and Max video frame height specify the largest frame size the IP needs to support.	—
Frame rate conversion	**Checked**, Unchecked	Enables frame rate conversion. When enabled, the output video stream runs on a separate clock. Refer to the Frame Rate Conversion section for more information.	—
Video Interface	**Native video**, Unified Video Streaming	Set to *Native Video* as described in the Video Input/Output section or through Unified Video Streaming Interface as described in the Unified Video Streaming Interface section.	*Unified Video Streaming* is only available when *Parallel processing* is on.
I/O Specification			
Input Data			
Input pixel width (Bits per Color)	**8**,10,12,16	Sets the bit width of all color planes comprising the incoming pixel.	—
Memory Interface			
Memory bus width	8,16,**32**,64,128	Sets the data width of memory interface.	Selectable values are calculated based on *Video format*, *Parallel processing*, and *Input pixel width*.

Attribute	Selectable Values	Description	Dependency on Other Attributes
Memory base address	Memory address width x00000000–Memory address width xFFFFFFFF, **Memory address width 'b00000000**	Sets the base address of the external memory space used. Value must be in binary format and must not exceed memory address width.	Base address must be aligned to axi_awsize_o and axi_arsize_o value.
Memory address width	Output Value, **21**	Specifies the address width of the memory interface. This is generated automatically by the user interface.	Calculated based on *Memory size needed*.
Memory size needed	Output Value, **2073600**	Specifies the size of the external memory space needed. This is generated automatically by the user interface.	Calculated based on *Video frame width*, *Video frame height*, *Frame rate conversion*, *Command burst count*, *DDR memory burst length*, *Memory bus width*, and *Input pixel width*.
Memory interface	**Native memory**, AXI4	Set to *Native Video* as described in the Memory Interface section or through *AXI4*. For details, refer to the AMBA AXI protocol specification in the Arm Developer web page.	—
Miscellaneous ports			
Miscellaneous Signals	Checked, **Unchecked**	Enables the input frame re-sync flag, including frame drop and repeat flag when *frame rate conversion* is on.	Only selectable when *Dynamic Parameter Updating* is off.
Implementation			
Read FIFO type	**EBR**, Distributed	Selects EBR or Distributed RAM for the internal read FIFO type of the frame buffer module.	—
Write FIFO type	**EBR**, Distributed	Selects EBR or Distributed RAM for the internal write FIFO type of the frame buffer module.	—
Read FIFO depth	32,**64**,128,256,512	Selects the depth of the internal Read FIFO.	Both Read and Write FIFO must be set to the same value. Selectable values are calculated based on video format, input pixel width, video frame width, video frame height, memory bus width, command burst count, and DDR memory burst length.
Write FIFO depth	32,**64**,128,256,512	Selects the depth of the internal write FIFO.	Both Read and Write FIFO must be set to the same value. Selectable values are calculated based on video format, input pixel width, video frame width, video frame height, memory bus width, command burst count, and DDR memory burst length.
DDR memory burst length	2,4,**8**	Selects the burst length value.	—

Attribute	Selectable Values	Description	Dependency on Other Attributes
Command burst count	**1**,2,4,8	Selects the burst count value for working with the memory controller.	Selectable values are calculated based on *DDR memory burst length*. Only 2, 4, and 8 are available for burst length of 2, otherwise all selectable values are available.

4. Signal Description

Table 4.1 lists the input and output signals for Video Frame Buffer IP Core.

Table 4.1. Video Frame Buffer IP Core Signal Description

Port	I/O	Width	Clock Domain	Default	Description
Clock and Reset					
rstn_i	In	1	—	—	Asynchronous active-low reset signal.
iclk_i	In	1	iclk_i	—	Input pixel sample clock.
mem_clk_i	In	1	mem_clk_i	—	Memory write and read clock. This is also the clock source of the AXI4 interface.
pclk_i	In	1	pclk_i	—	Clock source for the AXI4-Lite and Parameter Bus Interface.
oclk_i	In	1	oclk_i	—	Output sample clock, available when *Frame rate conversion* is checked.
sr_i	In	1	iclk_i	—	Active-high synchronous reset signal.
Native Video Interface[1]					
frmsync_in_i	In	1	iclk_i	—	New input video frame indicator, active-high.
dvalid_in_i	In	1	iclk_i	—	Input video data valid signal, active-high.
din_i	In	When *Video format* = YCbCr4:2:2 and *Parallel processing* is checked, size is 2 × *Input pixel width.* When *Video format* = YCbCr4:4:4 or RGB and *Parallel processing* is checked, size is 3 × *Input pixel width.* When *Parallel processing* is unchecked, size is equal to *Input pixel width.*	iclk_i	—	Input video data in frame format.
ready_o	Out	1	iclk_i	b0	When high, indicating the Video Frame Buffer IP Core can accept more input data. The signal ready_o deasserts after last pixel of row is received or write FIFO is almost full (threshold)[9].
dout_enable_i	In	1	iclk_i	—	Input from down-stream module to enable output data, active high.

Port	I/O	Width	Clock Domain	Default	Description
dout_o	Out	When *Video format* = YCbCr4:2:2 and *Parallel processing* is checked, size is 2 × *Input pixel width.* When *Video format* = YCbCr4:4:4 or RGB and *Parallel processing* is checked, size is 3 × *Input pixel width.* When *Parallel processing* is unchecked, size is equal to *Input pixel width.*	oclk_i	b0	Output video pixel in frame format.
dvalid_out_o	Out	1	oclk_i	b0	Output video pixel valid signal, active high.
frmsync_out_o	Out	1	oclk_i	b0	New output frame indicator, active high.
Native Memory Interface[2]					
cmd_ready_i	In	1	mem_clk_i	—	Input from memory controller indicating the IP ready to accept a new command, active high.
mem_addr_o	Out	Size depends on *Video frame width, Video frame height, Frame rate conversion, Command burst count, DDR memory burst length, Memory bus width*, and *Input pixel width*.	mem_clk_i	Memory Base Address	Memory read/write address.
read_cmd_o	Out	1	mem_clk_i	b0	Memory read command, active-high.
read_data_i	In	8, 16, 32, 64, 128	mem_clk_i	—	Read data output from memory.
read_data_valid_i	In	1	mem_clk_i	—	Read data valid indicator from memory controller, active high.
write_cmd_o	Out	1	mem_clk_i	b0	Memory write command, active-high.
write_data_o	Out	8, 16, 32, 64, 128	mem_clk_i	b0	Write data to memory.
write_data_ready_i	In	1	mem_clk_i	—	Input from memory controller indicating that the IP ready to accept new write data, active high.

Port	I/O	Width	Clock Domain	Default	Description
AXI4 Memory Interface[4]					
axi_araddr_o	Out	Size depends on *Video frame width*, *Video frame height*, *Frame rate conversion*, *Command burst count*, *DDR memory burst length*, *Memory bus width*, and *Input pixel width*.	mem_clk_i	b0	AXI4 read address channel: Read address signal.
axi_arlen_o	Out	8	mem_clk_i	b0	AXI4 read address channel: Burst length signal. Supports up to burst length 64 only. AxLEN = [(Command burst count × DDR memory burst length)/2] – 1

axi_arsize_o	Out	3	mem_clk_i	b0	AXI4 read address channel: Burst size signal.
					If (Memory bus width/8) ≤ 4,
					AxSIZE = ‘h(Memory bus width/8)/2
					If (Memory bus width/8) = 8,
					AxSIZE = ‘h3
					If (Memory bus width/8) = 16,
					AxSIZE = ‘h4
					If (Memory bus width/8) = 32,
					AxSIZE = ‘h5
					If (Memory bus width/8) = 64,
					AxSIZE = ‘h6

axi_arburst_o	Out	2	mem_clk_i	b0	AXI4 read address channel: Burst type signal. Only INCR is supported.
axi_arvalid_o	Out	1	mem_clk_i	b0	AXI4 read address channel: Read address valid signal.
axi_arready_i	In	1	mem_clk_i	—	AXI4 read address channel: Read address ready signal.
axi_arprot_o	Out	3	mem_clk_i	3’b000	Read channel protection signal.
axi_rdata_i	In	8, 16, 32, 64, 128	mem_clk_i	—	AXI4 read data channel: Read data signal.
axi_rlast_i	In	1	mem_clk_i	—	AXI4 read data channel: Read last signal.
axi_rvalid_i	In	1	mem_clk_i	—	AXI4 read data channel: Read valid signal.
axi_rready_o	Out	1	mem_clk_i	b0	AXI4 read data channel: Read ready signal.

Port	I/O	Width	Clock Domain	Default	Description
axi_awaddr_o	Out	Size depends on *Video frame width, Video frame height, Frame rate conversion, Command burst count, DDR memory burst length, Memory bus width,* and *Input pixel width*.	mem_clk_i	b0	AXI4 write address channel: Write address signal.
axi_awlen_o	Out	8	mem_clk_i	b0	AXI4 write address channel: Burst length signal. AxLEN = [(Command burst count × DDR memory burst length)/2] – 1

axi_awsize_o	Out	3	mem_clk_i	b0	AXI4 write address channel: Burst size signal.
					If (Memory bus width/8) ≤ 4,
					AxSIZE = ‘h(Memory bus width/8)/2
					If ((Memory bus width/8) == 8),
					AxSIZE = ‘h3
					If ((Memory bus width/8) == 16),
					AxSIZE = ‘h4
					If ((Memory bus width/8) == 32),
					AxSIZE = ‘h5
					If ((Memory bus width/8) == 64),
					AxSIZE = ‘h6

axi_awburst_o	Out	2	mem_clk_i	b0	AXI4 write address channel: Burst type signal.
axi_awvalid_o	Out	1	mem_clk_i	b0	AXI4 write address channel: Write address valid signal.
axi_awready_i	In	1	mem_clk_i	—	AXI4 write address channel: Write address ready signal.
axi_awprot_o	Out	3	mem_clk_i	3’b000	Write channel protection signal.
axi_wdata_o	Out	8, 16, 32, 64, 128	mem_clk_i	b0	AXI4 write data channel: Write data signal.
axi_wstrb_o	Out	axi_wdata_o bus width/8	mem_clk_i	b0	AXI4 write data channel: Write strobe signal.
axi_wlast_o	Out	1	mem_clk_i	b0	AXI4 write data channel: Write last signal.
axi_wvalid_o	Out	1	mem_clk_i	b0	AXI4 write data channel: Write valid signal.
axi_wready_i	In	1	mem_clk_i	—	AXI4 write data channel: Write ready signal.
axi_bvalid_i	In	1	mem_clk_i	—	AXI4 write response channel: Write response valid signal.
axi_bready_o	Out	1	mem_clk_i	b0	AXI4 write response channel: Response ready signal.
Native Parameter Bus Interface[3]					
pwrite_i	In	1	pclk_i	—	Parameter bus write enable.
paddr_i	In	5	pclk_i	—	Parameter bus address.
pwdat_i	In	32	pclk_i	—	Parameter bus write data.
prdat_o	Out	32	pclk_i	b0	Parameter bus read data.

Port	I/O	Width	Clock Domain	Default	Description
Unified Video Streaming Transmitter Interface[6]					
axis_vid_tdata_o	Out	TD_WD × PPC[7]	oclk_i	b0	Output video pixel in AXI4-Stream format.
axis_vid_tvalid_o	Out	1	oclk_i	b0	AXI4-Stream data valid.
axis_vid_tuser_o	Out	2	oclk_i	b0	AXI4-Stream user signal, bit 0 is used to indicate new video frame. This signal is qualified with the axis_tx_tvalid_o signal, bit 1 is reserved.
axis_vid_tready_i	In	1	oclk_i	—	AXI4-Stream signal indicating that the receiver is ready to accept data transfer.
axis_vid_tlast_o	Out	1	oclk_i	b0	AXI4-Stream signal indicating end of line.
Unified Video Streaming Receiver Interface[6]					
axis_vid_tdata_i	In	TD_WD × PPC[7]	iclk_i	—	Input video pixel in AXI4-Stream format.
axis_vid_tvalid_i	In	1	iclk_i	—	AXI4-Stream data valid.
axis_vid_tuser_i	In	2	iclk_i	—	AXI4-Stream user signal, bit 0 is used to indicate new video frame. This signal is qualified with the axis_tx_tvalid_i signal, bit 1 is reserved.
axis_vid_tready_o	Out	1	iclk_i	b0	AXI4-Stream signal indicating that the IP can accept data transfer. The axis_vid_tready_o signal may deassert after the last pixel of row is received or write FIFO threshold[10] is reached.
axis_vid_tlast_i	In	1	iclk_i	b0	AXI4-Stream signal indicating end of line.
AXI4-Lite Parameter Bus Interface[5]					
axil_awaddr_i	In	5	pclk_i	—	Write address bus.
axil_awvalid_i	In	1	pclk_i	—	Write address valid.
axil_awready_o	Out	1	pclk_i	b0	Write address acknowledge.
axil_wdata_i	In	32	pclk_i	—	Write data bus.
axil_wvalid_i	In	1	pclk_i	—	Write data valid.
axil_wready_o	Out	1	pclk_i	b0	Write data acknowledge.
axil_wstrb_i	In	4	pclk_i	—	Write strobe signal. Only 4'b1111 input is supported by the IP.
axil_bvalid_o	Out	1	pclk_i	b0	Write response valid.
axil_bready_i	In	1	pclk_i	—	Write response acknowledge.
axil_araddr_i	In	5	pclk_i	—	Read address bus.
axil_arvalid_i	In	1	pclk_i	—	Read address valid.
axil_arready_o	Out	1	pclk_i	b1	Read address acknowledge.
axil_rdata_o	Out	32	pclk_i	b0	Read data output.
axil_rvalid_o	Out	1	pclk_i	b0	Read data/response valid.
axil_rready_i	In	1	pclk_i	—	Read data acknowledge.

Port	I/O	Width	Clock Domain	Default	Description
Miscellaneous Signals[8]					
frm_drop_o	Out	1	iclk_i	b0	Available when miscellaneous signals are checked and frame rate conversion is on. This flag is asserted when new input frame is received while previous input frame is not yet completely written to memory.
frm_resync_o	Out	1	iclk_i	b0	Available when miscellaneous signals are checked. Flag is asserted when frmsync_in_i or axis_vid_tuser_i[0] is asserted while the input has not yet transmitted the complete frame. When this flag is asserted, contents of the write FIFO are reset and pixels from the old frame are dropped.
frm_repeat_o	Out	1	oclk_i	b0	Available when miscellaneous signals are checked and frame rate conversion is on. Flag is asserted during repeat frames of frame conversion. The frm_repeat_o is asserted at the start of each repeating frame.

Notes:

1. Available when Video interface == Native video.
2. Available when Memory interface == Native memory.
3. Available when Dynamic Parameter Updating == Native Parameter Bus.
4. Available when Memory interface == AXI4.
5. Available when Dynamic Parameter Updating == AXI4-Lite.
6. Available when Video interface == UVSI.
7. The TD_WD is calculated as ceil(BPP/8) × 8 which is the data width per pixel. The size of axis_vid_tdata_i is calculated as TD_WD × PPC. TD_WD (Data Width) BPP (Bits per Pixel) PPC (Pixels per Clock).
8. Available when Miscellaneous Signals == Checked.
9. Threshold is Write FIFO Depth – 2 if Memory Bus Width / Data Port Width (din_i width for Native Video) <2 and Write FIFO Depth – 1 otherwise.
10. Threshold is Write FIFO Depth – 3 if Memory Bus Width / Data Port Width (TD_WD) <2 and Write FIFO Depth – 2 otherwise.

5. Register Description

Table 5.1. General Configuration Registers

Address Offset	Name	Description	Access Type	Default
0x0000	FRMWIDTH	Frame width register	RW	Frame width – 1
0x0004	FRMHEIGHT	Frame height register	RW	Frame height – 1
0x0008	RSVD	Reserved	RO	32'h0
0x000C	UPDATE	Update parameter enable register	RW	32'h0
0x0010	FRM_RESYNC	Frame resync register	RO and W1C	32'h0
0x0014	FRAME_CONV_STATUS	Frame conversion status register	RO and W1C	32'h0

Table 5.2. FRMWIDTH Register (Address 0x0000)

Field	Name	Description	Access	Default
[31:0]	FRMWIDTH	The FRMWIDTH value must be frame width – 1. The minimum value is 63, and the maximum value is the maximum frame width specified on the IP user interface minus 1. The default value is the maximum value.	RW	Frame width – 1

Table 5.3. FRMHEIGHT Register (Address 0x0004)

Field	Name	Description	Access	Default
[31:0]	FRMHEIGHT	The FRMWIDTH value must be (frame width – 1). The minimum value is 63, and the maximum value is the maximum frame width specified on the IP user interface minus 1. The default value is the maximum value.	RW	Frame width – 1

Table 5.4. UPDATE Register (Address 0x000C)

Field	Name	Description	Access	Default
[31:1]	RSVD	Reserved.	RO	31'h0
[0]	UPDATE	The value can be 0 or 1 for bit 0. Setting this bit to 1 triggers the IP to start updating the FRMWIDTH and FRMHEIGHT registers. The IP resets the UPDATE register to 0 after the parameters have taken effect. UPDATE register is reset two input pixel clock cyles and two parameter bus clock cycles after frmsync_in or axis_vid_tuser_i[0] is asserted.	RW	1'h0

Table 5.5. FRM_RESYNC Register (Address 0x0010)

Field	Name	Description	Access	Default
[31:1]	RSVD	Reserved.	RO	31'h0
[0]	FRM_RESYNC	Shows status when frame resync is asserted. Bit is asserted when frmsync_in_i or axis_vid_tuser_i[0] is asserted while the input has not yet transmitted the complete frame. When this bit is asserted, contents of the write FIFO are reset and pixels from the old frame are dropped. The FRM_RESYNC register (address 0x0010) can be sampled two input pixel clock cycles and four parameter bus clock cycles after frmsync_in or axis_vid_tuser_i[0] is asserted.	W1C	1'h0

Table 5.6. FRAME_CONV_STATUS Register (Address 0x0014)

Field	Name	Description	Access	Default
[31:2]	RSVD	Reserved.	RO	30'h0
[1]	FRM_DROP	Bit is asserted when new input frame is received while previous input frame is not yet completely written to memory. For the FRAME_CONV_STATUS register (address 0x0014), FRM_DROP can be sampled five input pixel clock cycles and four parameter bus clock cycles after the last pixel is written to memory.	W1C	1'h0
[0]	FRM_REPEAT	Bit is asserted during repeat frames of frame conversion. The frm_repeat_o bit is asserted at the start of each repeating frame. For the FRAME_CONV_STATUS register (address 0x0014), FRM_REPEAT can be sampled five output pixel clock cycles and four parameter bus clock cycles after the last pixel is read from the memory.	W1C	1'h0

6. Example Design

The Video Frame Buffer example design allows you to compile, simulate, and test the Video Frame Buffer IP on the following Lattice evaluation board:

- CertusPro-NX Versa Board (LFCPNX-VERSA-EVN)

6.1. Example Design Supported Configuration

The following table summarizes the IP parameter configuration that are reflected in the Video Frame Buffer Example Design. For steps on how to generate the Video Frame Buffer IP Core, refer to the Designing with the IP section.

Table 6.1. Video Frame Buffer IP Configuration Supported by the Example Design

Attribute	Supported Settings
Video Format	YCbCr:4:2:2
Video Frame Width	64
Video Frame Height	64
Parallel Processing	On
Dynamic Parameter Updating	Off
Frame Rate Conversion	Off
Input Pixel Width (Bits per Color)	8
Memory Bus Width	32
Memory Base Address	15'b000000000000000
Memory Interface	AXI
Video Interface	Unified Video Streaming
Miscellaneous signals	Off
Read FIFO type	EBR
Write FIFO type	EBR
Write/Read FIFO depth	64
DDR memory burst length	8
Command burst count	1

6.2. Overview of the Example Design and Features

The key features of the example design are as follows:

- Continuous pixel input and pixel compare
  - When SW1 (rstn_i) is deasserted, continuous pixel input and pixel compare begins. Pixel pattern changes from frame-to-frame. Input pattern can be toggled using SW3 (sw_pattern_i).
- Status LEDs
  - Pixel comparator asserts LED0 when a pixel compare error is detected. LED1 (pll_lock_o) and LED2 (init_done_o) provide the status of the LPDDR4 memory controller for Nexus devices.

6.3. Example Design Components

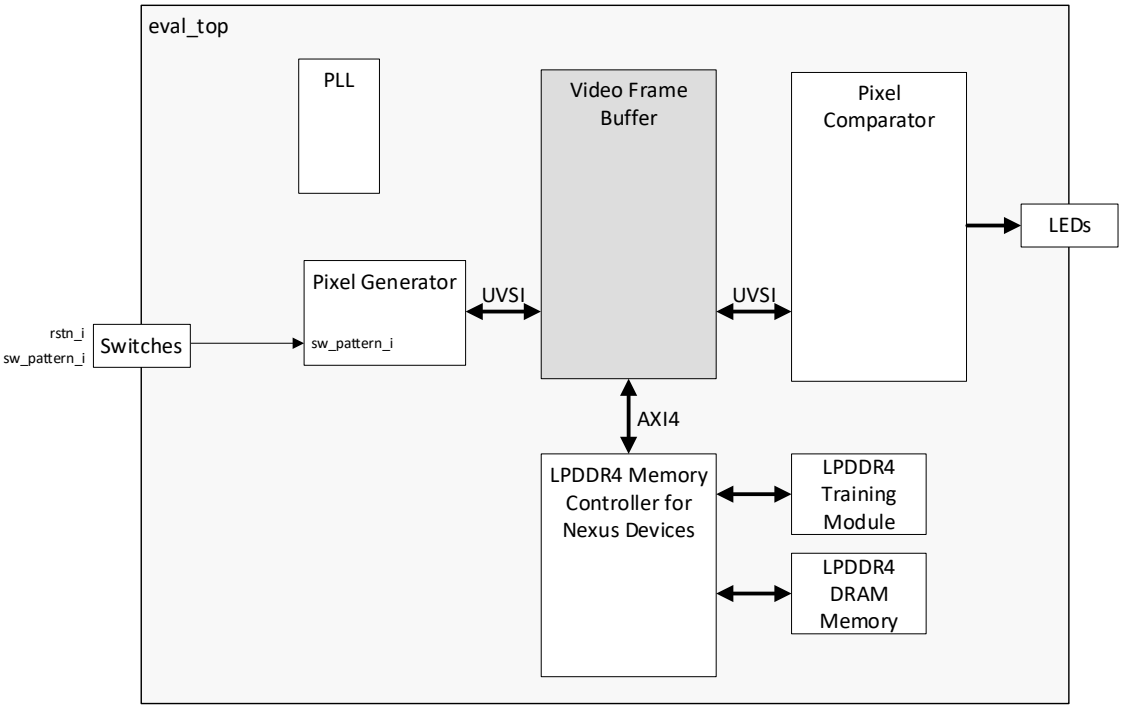

Figure 6.1. Video Frame Buffer Example Design Block Diagram

Note: The rstn_i port is the reset switch for all the blocks in the example design. The PLL provides all the necessary clocks for the Video Frame Buffer example design.

The synthesizable example design consists of the following main blocks:

- Video Frame Buffer
- LPDDR4 memory controller for Nexus Devices
- Pixel generator block
- Pixel comparator
- PLL
- LPDDR4 training module
- LPDDR4 DRAM memory

6.3.1. Video Frame Buffer

Refer to the Introduction section.

6.3.2. LPDDR4 Memory Controller for Nexus Devices

Refer to the LPDDR4 Memory Controller IP Core for Nexus Devices User Guide (FPGA-IPUG-02127).

6.3.3. Pixel Generator

This block generates pixel data and continuously sends entire frames to the Video Frame Buffer. To control the pixel data generated, SW3 (sw_pattern_i) can be used. When this signal is deasserted, the pixel generator sends increasing and decreasing pixel value from one frame to the next. When this signal is asserted, the pixel generator sends running 0 and running 1 pattern for pixel values from one frame to the next.

6.3.4. Pixel Comparator

This module checks and compares the pixel data received from and sent to the Video Frame Buffer. If the pixel comparator encounters a data mismatch within a frame, the comparator asserts the error port, and the port remains asserted until the reset is asserted.

6.3.5. PLL

The PLL generates clocks for all the modules in the Video Frame Buffer example design. The PLL reference clock comes from the 125 MHz differential clock source of the CertusPro-NX Versa Board (LFCPNX-VERSA-EVN).

6.3.6. LPDDR4 Training Module

The LPDDR4 training module is used to perform the training sequence of the LPDDR4 memory controller for Nexus devices.

6.3.7. LPDDR4 DRAM Memory

The LPDDR4 DRAM memory is used to store pixels received by the Video Frame Buffer. This is the on-board memory of the CertusPro-NX Versa Board (LFCPNX-VERSA-EVN).

6.4. Generating and Using the Example Design

You can use the Lattice Radiant software to generate and use the example design. A sample Lattice Radiant software project file for the CertusPro-NX device is provided in the package.

Table 6.2. Design File List for the Example Design

Attribute	Description
eval/VFB_Example_Design/source	Contains all the design modules needed from validation, including testbench files for functional simulation. Contains the pixel generator, pixel comparator, and LPDDR4 training module.
eval/VFB_Example_Design/mem_controller_0	Pre-generated LPDDR4 memory controller.
eval/VFB_Example_Design/pll_0	Pre-generated general PLL soft IP.
eval/VFB_Example_Design/vfb_0	Pre-generated Video Frame Buffer.
eval/VFB_Example_Design/VFB_Example_Design.rdf	Sample Lattice Radiant software project in RDF format.
eval/VFB_Example_Design/source/post_syn.pdc	Sample post-synthesis constraint file in PDC format for the example design. Pin location constraints are pre-generated for the Certus-NX Versa Board (LFCPNX-VERSA-EVN) only.

To use the example design sample project, follow these steps:

1. Open the sample project *<PROJECT_DIRECTORY>/<IP_INST>/eval/VFB_Example_Design/VFB_Example_Design.rdf* provided.

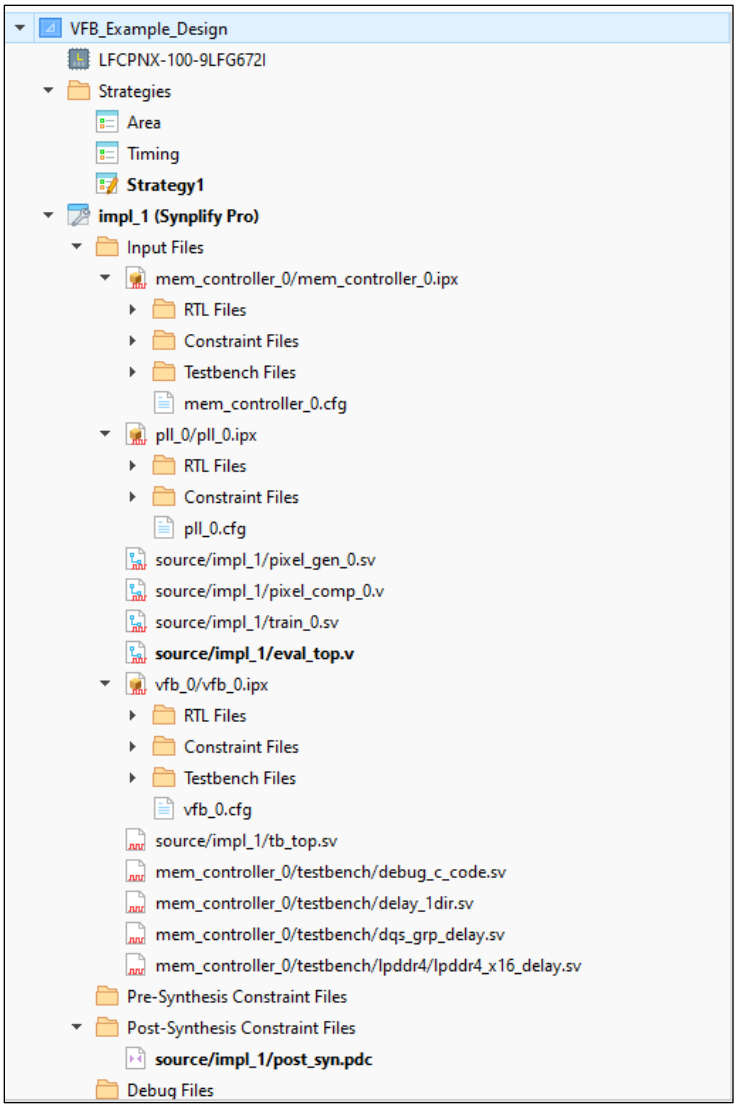

Figure 6.2. Sample File List for the Example Design

2. Run the implementation flow.

   The sample project includes all the files required by the example design including the PDC file. You must have a thorough understanding of the effect of any modification to modify the settings of the example design project provided. The example design works only when using the pre-generated settings for the specific device stated.

6.5. Simulating the Example Design

To run the functional simulation, follow these steps:

1. Make sure that testbench file tb_top.sv is included in the **Input Files** section. Set the file to include in Simulation only, as shown in the following diagram.

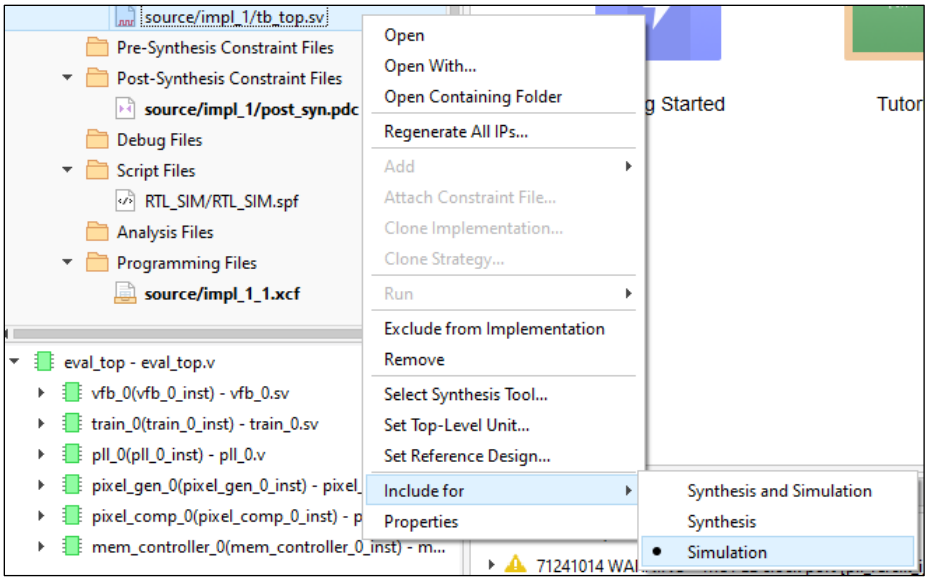

Figure 6.3. Testbench Top File

2. Click the button on the **Toolbar** to initiate the **Simulation Wizard** shown in the following diagram.

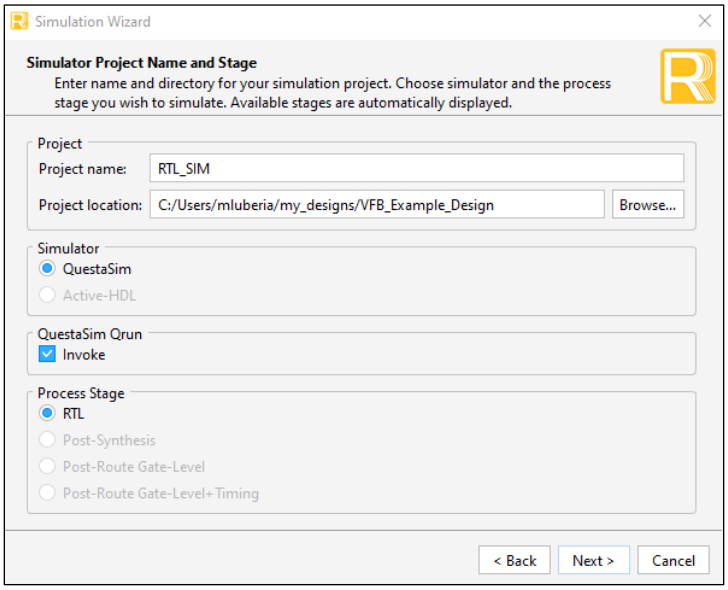

Figure 6.4. Simulation Wizard GUI

3. Click **Next** to open the **Add and Reorder Source** window.

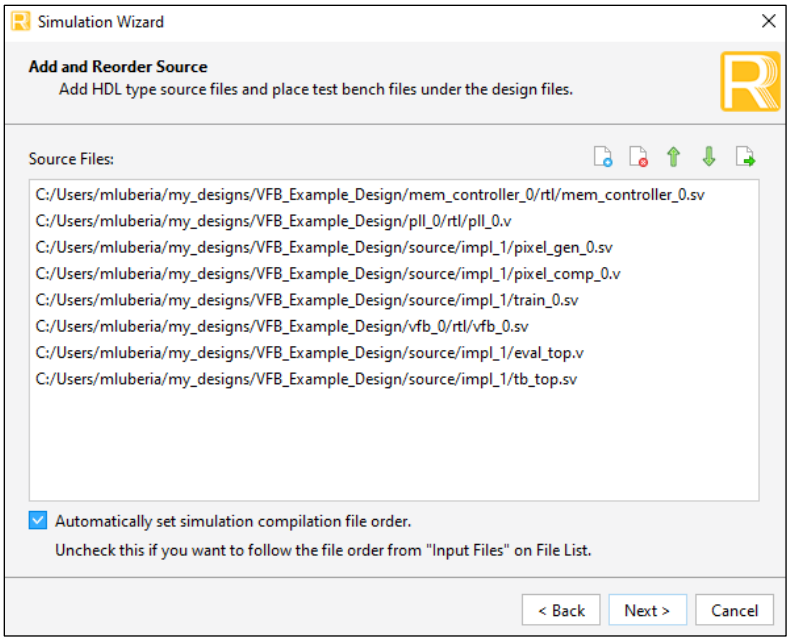

Figure 6.5. Add and Reorder Source Window

4. Click **Next** to open the **HDL Paser Log** window, which shows tb_top as the Simulation Top Module.

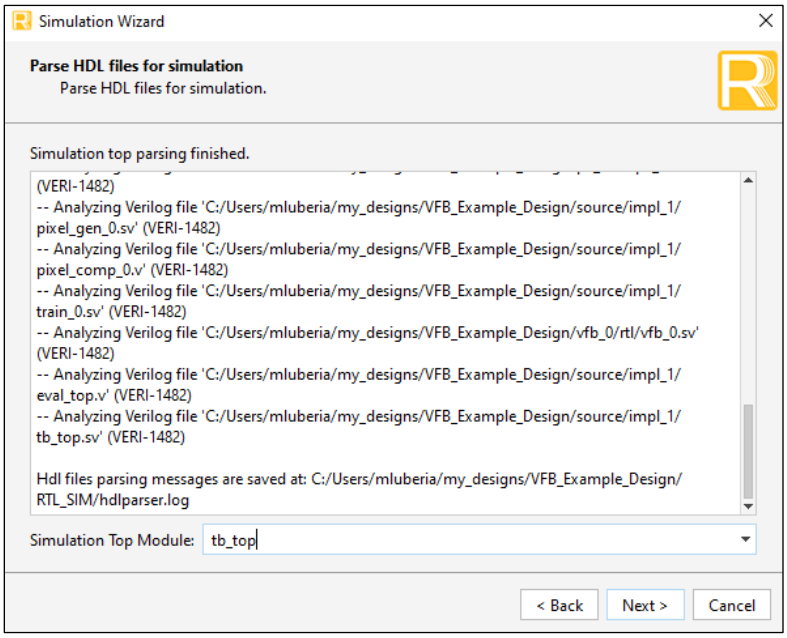

Figure 6.6. Simulation Top Module

5. Click **Next**. The **Summary** window opens.
6. Set **Run Simulation** to 0 to ensure the simulation runs completely. Click **Finish** to run the simulation.

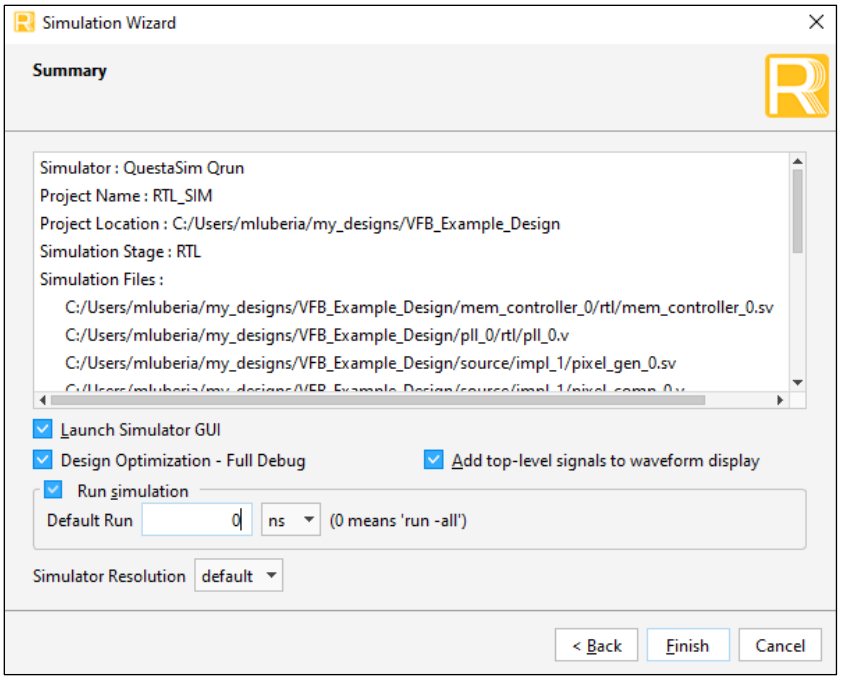

Figure 6.7. Simulation Wizard Summary Window

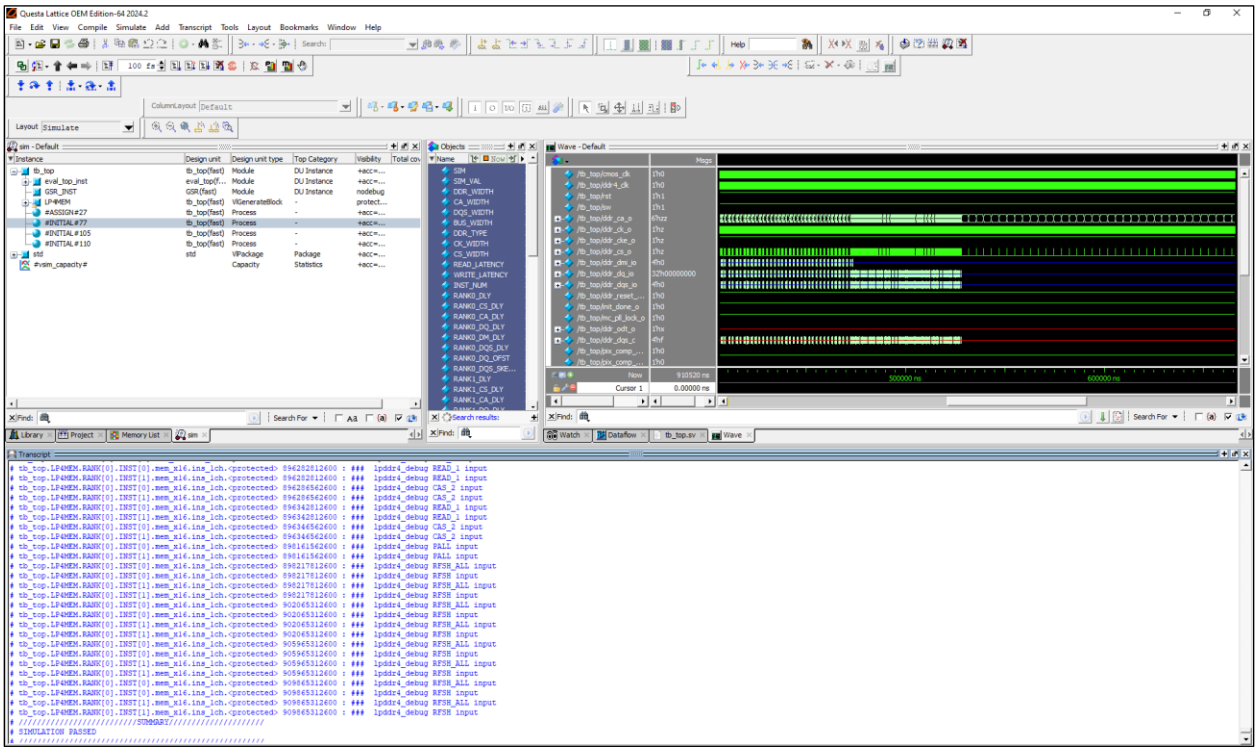

Figure 6.8. Simulation Result

6.6. Hardware Testing

When the implementation flow is complete, a bitstream file is generated. The bitstream can be loaded to the device using the Lattice Radiant Programmer by clicking the icon and Program Device icon. When the bitstream is successfully loaded on to the device, you can proceed with the tests using the provided switches and LEDs.

The top-level example design wrapper file, eval_top.v, provides ports for a PLL reference clock, LED output for init_done_o (LED2), pll_lock_o (LED1), and error port (LED0).

To run the example design, deassert the SW1 (rstn_i) signal. When the LPDDR4 memory controller training is complete, the pixel generator begins to send data to the Video Frame Buffer IP continuously. The pixel data is then sent to and from the LPDDR4 memory controller through the AXI4 interface. When the Video Frame Buffer IP begins to output pixel data, the pixel comparator module continuously checks for any unexpected data output.

To change the pixel data pattern, SW3 (sw_pattern_i) can be flipped. When this signal is deasserted, the pixel generator sends increasing and decreasing pixel value from one frame to the next. When this signal is asserted, the pixel generator sends running 0 and running 1 pattern for pixel values from one frame to the next.

To induce error into the pixel data generated by the pixel generator, SW2 (err_sw_i) signal can be asserted to produce an unrecognized pattern of data.

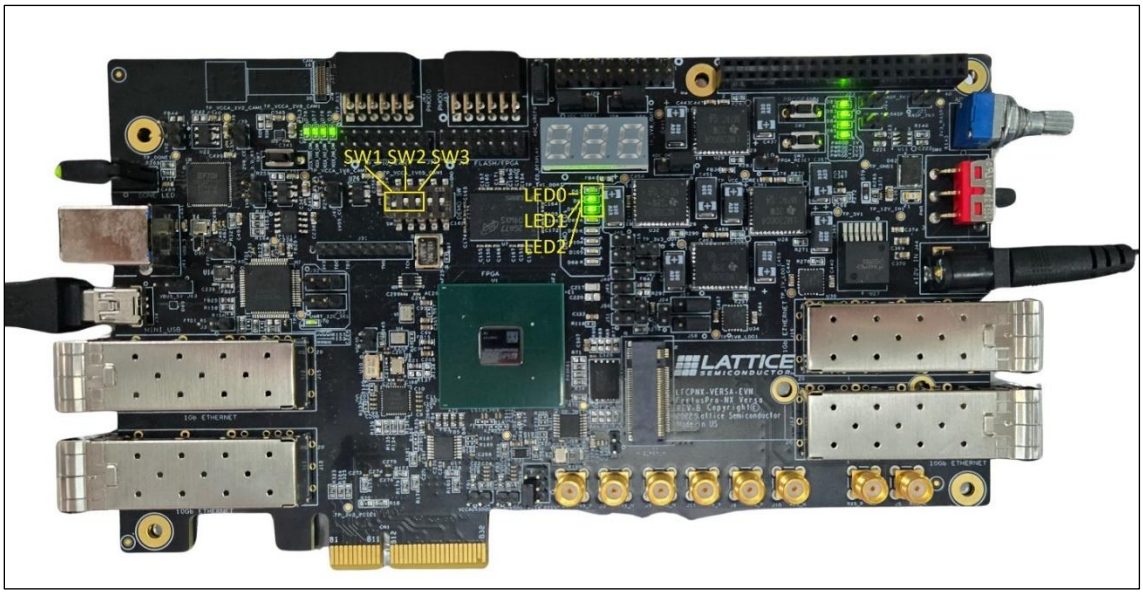

Figure 6.9. Video Frame Buffer Example Design on the CertusPro-NX Versa Board

7. Designing with the IP

This section provides information on how to generate the Video Frame Buffer IP Core 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.

7.1. Generation and Synthesis

The Lattice Radiant software allows you to customize and generate modules and IPs and integrate them into the device architecture.

To generate the Video Frame Buffer IP Core, follow these steps:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click on **Video Frame Buffer** under **IP, DSP** category. The **Module/IP Block Wizard** opens as shown in Figure 7.1.
3. Enter values in the **Component name** and the **Create in** fields and click **Next**.

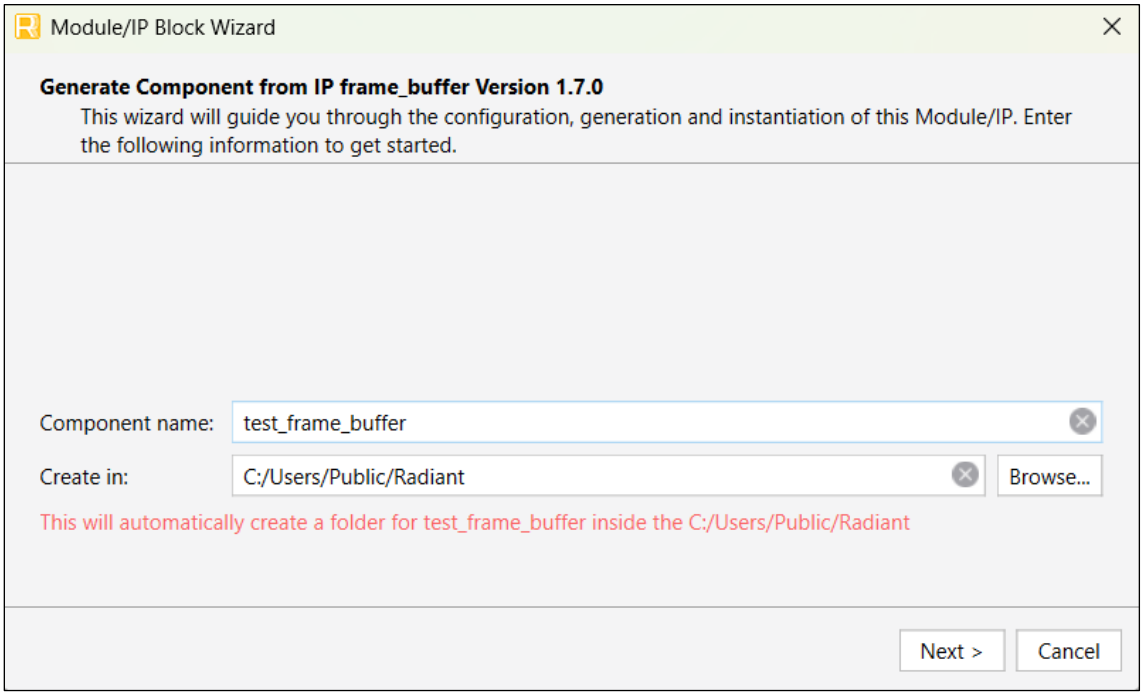

Figure 7.1. Module/IP Block Wizard

4. In the next **Module/IP Block Wizard** window, customize the selected Video Frame Buffer IP Core using drop-down menus and check boxes. As a sample configuration, see Figure 7.2. For configuration options, see the IP Parameter Description section.

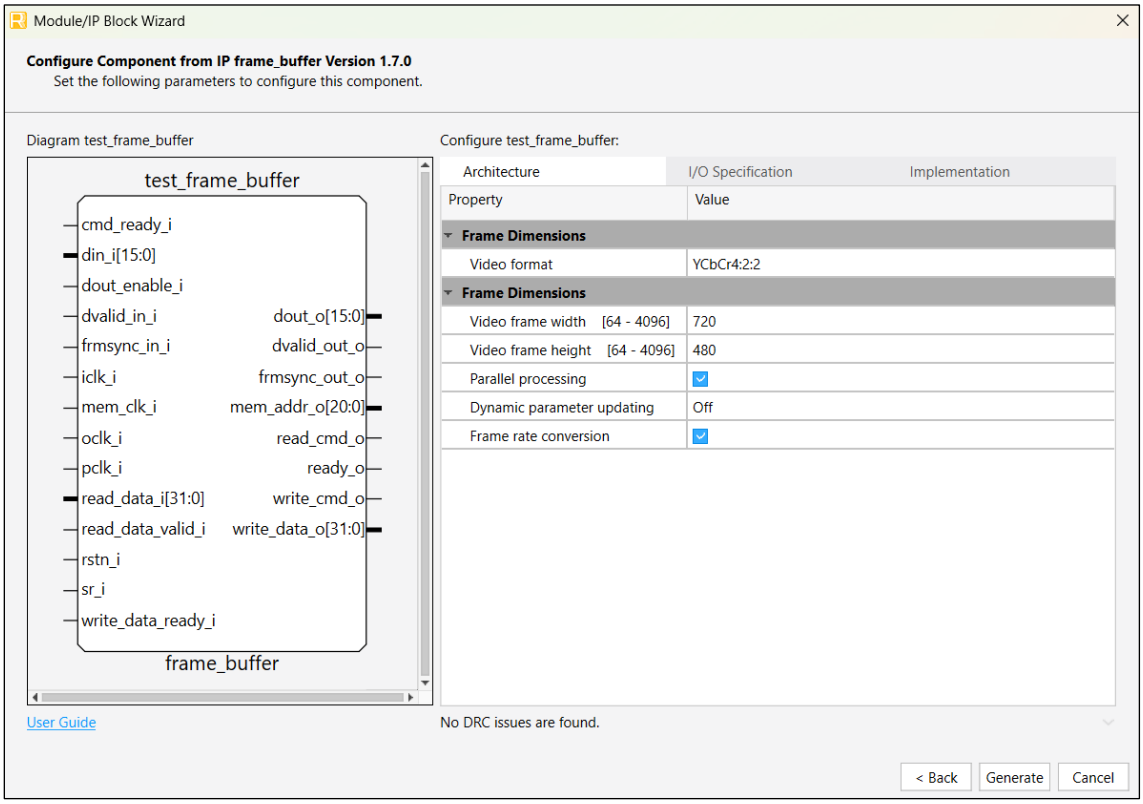

Figure 7.2. Configure User Interface of Video Frame Buffer IP Core

5. Click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results as shown in Figure 7.3.

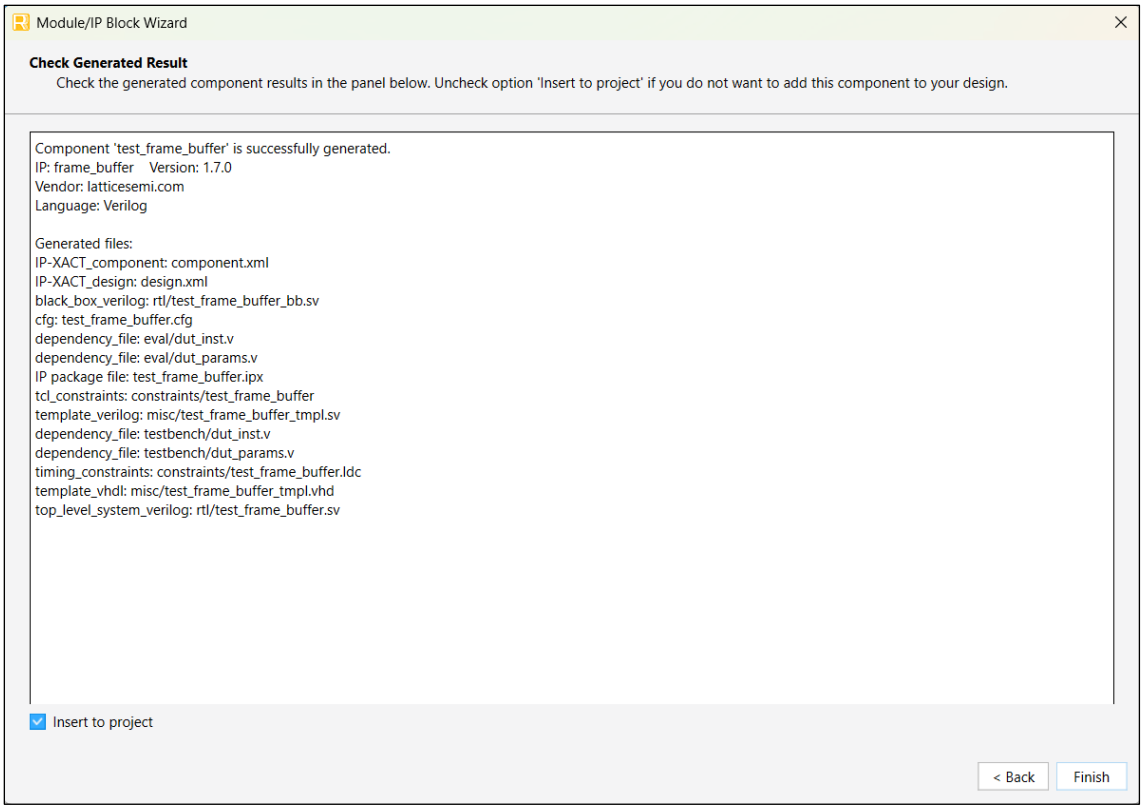

Figure 7.3. Check Generated Result

6. Click the **Finish** button. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 7.1.

The generated Video Frame Buffer IP Core package includes the closed-box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. 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. You may also use this example as the starting template for your the top-level design. The generated files are listed in Table 7.1.

Table 7.1. Generated File List

Attribute	Description
<Component name>.ipx	Contains the information on the files associated to 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
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	Provide instance templates for the IP core

7.2. Constraining the IP

You need to provide proper timing and physical design constraints to ensure that your design meets the desired performance goals on the FPGA. Add the content of the following IP constraint file to your design constraints:

<IP_Instance_Path>/<IP_Instance_Name>/eval/constraint.pdc.

The constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. You can modify the constraints in this file with thorough understanding of the effect of each constraint.

Refer to Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059) for details on how to constraint your design.

7.3. Running Functional Simulation

To run functional simulation, follow these steps:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard**.

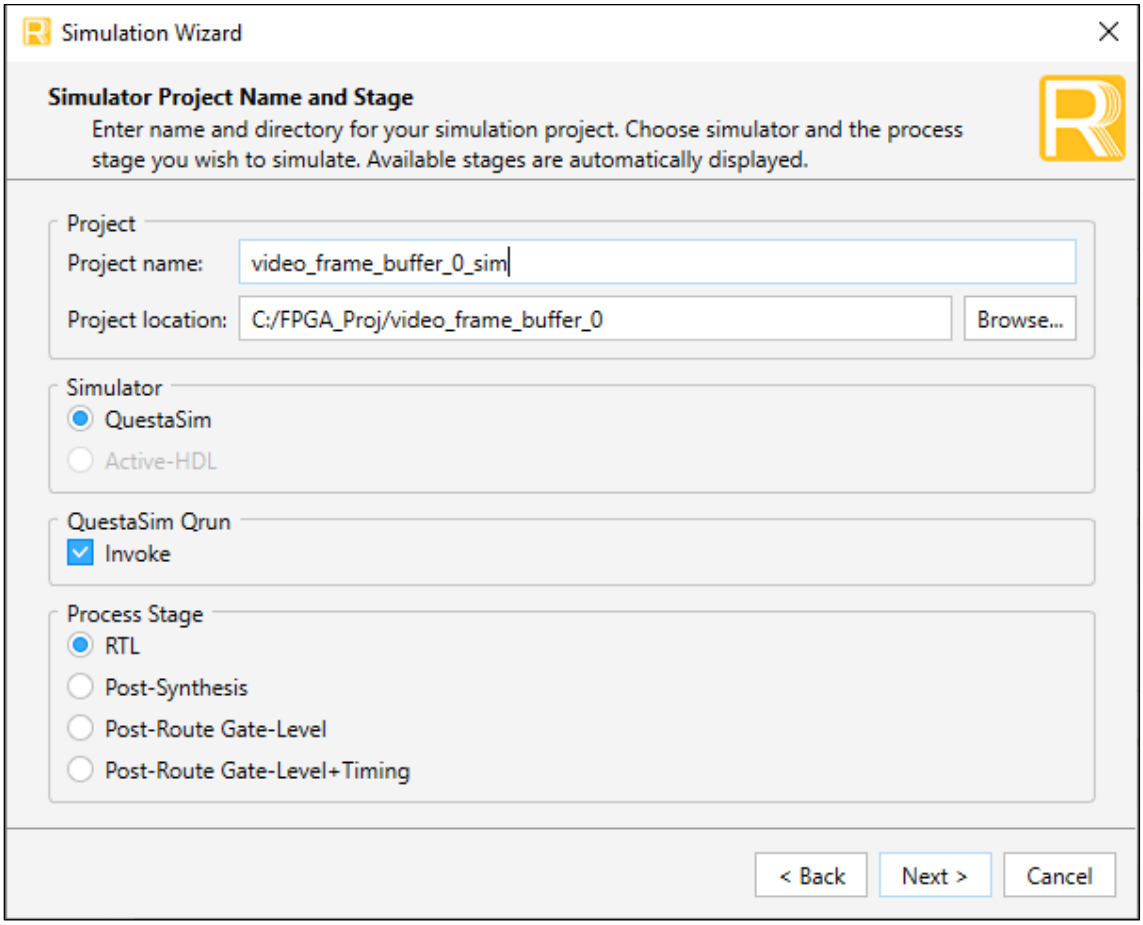

Figure 7.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window.

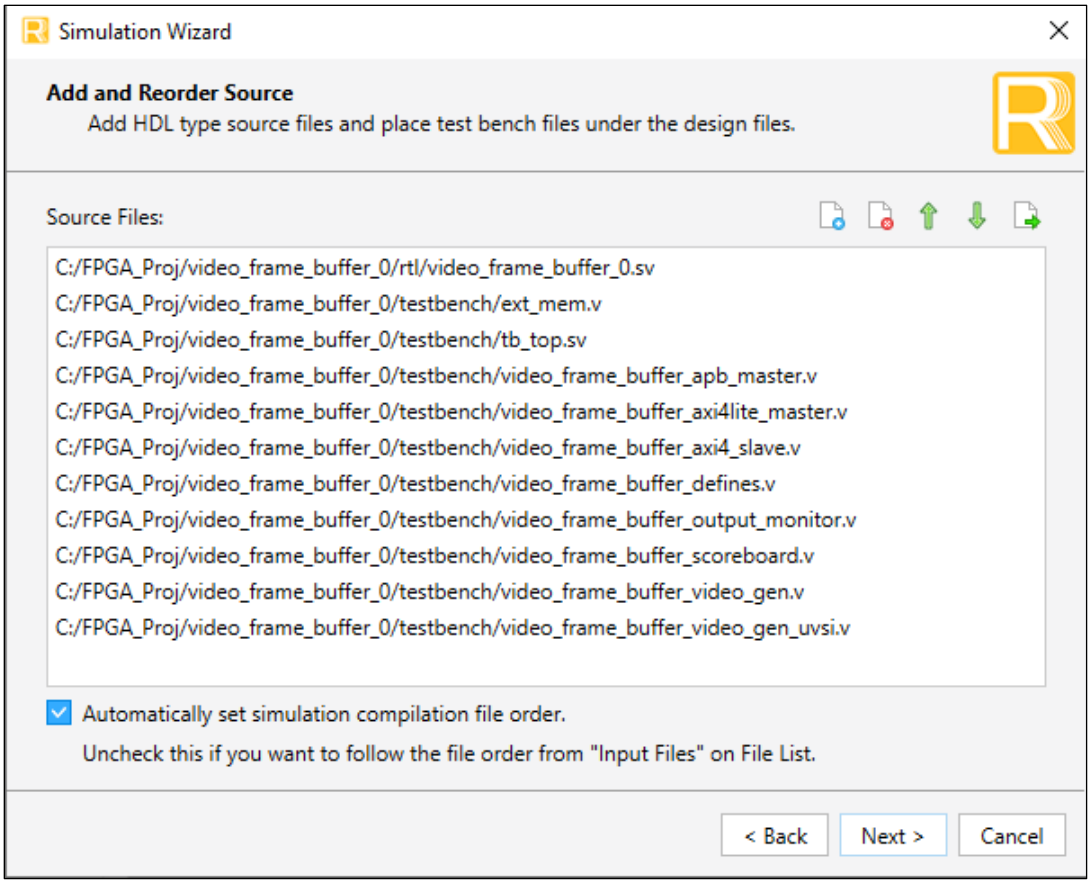

Figure 7.5. Adding and Reordering Source

3. Click **Next**. The **Summary** window is shown. Click **Finish** to run the simulation.

   Note: It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant software suite. The following diagram shows the example simulation result.

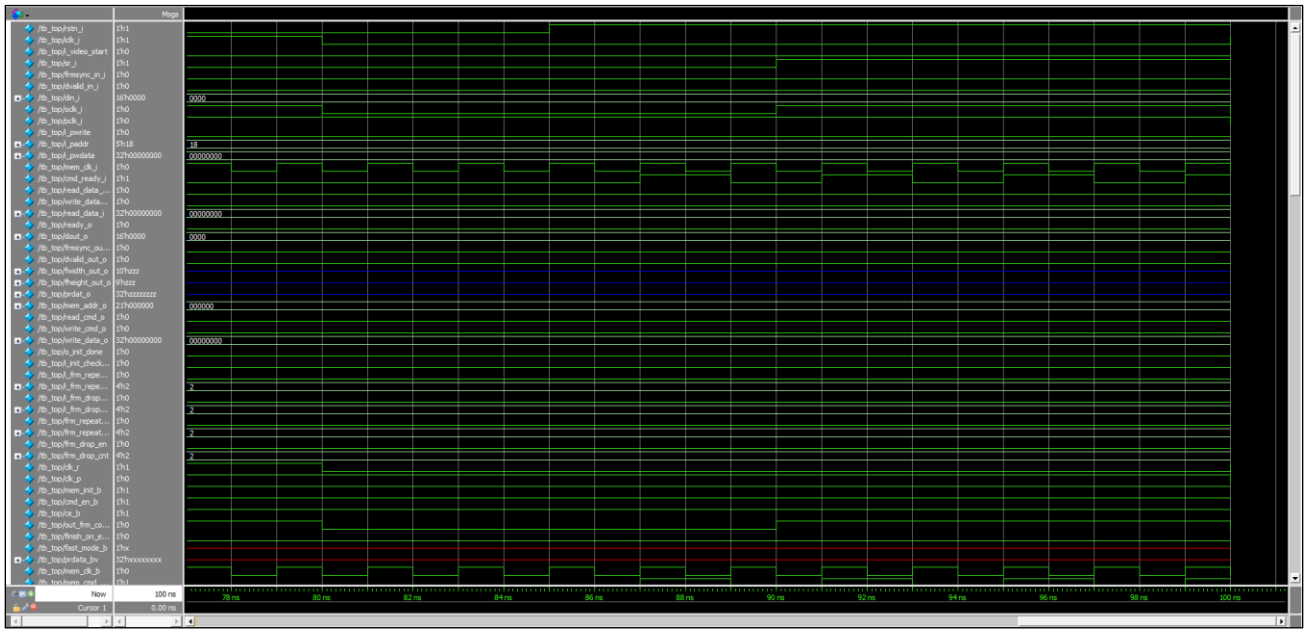

Figure 7.6. Simulation Waveform

8. Debugging

You can check the status of the IP using the FRM_RESYNC register (address 0x0010), FRAME_CONV_STATUS register (address 0x0014), or through the Miscellaneous ports.

9. Design Considerations

- To avoid the current output frame from dropping when a new input frame is received by the IP, you can perform one of the following:
  - Provide sufficient blanking between frames when frame rate conversion is off.
  - Enable frame rate conversion.
- It is recommended using the asynchronous reset (rstn_i) instead of the synchronous reset (sr_i) to avoid issues when using the IP.

9.1. Limitations

- When *Frame rate conversion* == Checked, axis_vid_tvalid_o may de-assert when the signal axis_vid_tready_i is de-asserted. This may cause the IP to not output pixels continuously.
- Some configurations may fail Static Timing Analysis when compiling your design using LSE. If this happens, consider compiling your design using Synopsis Synplify Pro.
- Some IP configurations may have slower Fmax when using devices with speed grade 7. The following Fmax value is approximate and may vary depending on the system-level design:
  - CrossLink-NX and Certus-NX devices: 191 MHz (mem_clk) for a target frequency of 200 MHz for *Dynamic Parameter Updating == AXI4-lite*.

Appendix A. Resource Utilization

The following tables show the configuration and resource utilization for the devices using Synplify Pro of the Lattice Radiant software version 2025.2.

Table A.1. Resource Utilization for the LAV-AT-G70-1LFG1156I Device

Configuration	Fmax (MHz) i_clk	Fmax (MHz) o_clk	Fmax (MHz) mem_clk	Registers	LUTs	EBRs
Default	250	250	250	804	1,217	2
Memory Interface = AXI4 Video Interface = UVSI Dynamic = AXI4-Lite Others = Default	250	250	250	1,898	2,479	2
Memory Interface = AXI4 Video Interface = UVSI Dynamic = AXI4-Lite FIFO Type = Distributed Others = Default	200	250	250	1,898	2,479	2

Notes:

1. Fmax is generated when the FPGA design only contains the Video Frame Buffer IP Core and the target frequency is 200 MHz. These values may be reduced when user logic is added to the FPGA design.
2. Fmax is generated using multiple iterations of Place and Route.

Table A.2. Resource Utilization for the LFCPNX-100-7LFG672I Device

Configuration	Fmax (MHz) i_clk	Fmax (MHz) o_clk	Fmax (MHz) mem_clk	Registers	LUTs	EBRs
Default	200	200	191.13	735	1,068	2
Memory Interface = AXI4 Video Interface = UVSI Dynamic = AXI4-Lite Others = Default	200	200	200	1,818	2,370	2
Memory Interface = AXI4 Video Interface = UVSI Dynamic = AXI4-Lite FIFO Type = Distributed Others = Default	200	200	200	1,818	2,370	2

Notes:

1. Fmax is generated when the FPGA design only contains the Video Frame Buffer IP Core and the target frequency is 200 MHz. These values may be reduced when user logic is added to the FPGA design.
2. Fmax is generated using multiple iterations of Place and Route.

Table A.3. Resource Utilization for the LN2-CT-20ES-1ASG410I Device

Configuration	Fmax (MHz) i_clk	Fmax (MHz) o_clk	Fmax (MHz) mem_clk	Registers	LUTs	EBRs
Default	250	250	250	804	1,217	2
Memory Interface = AXI4 Video Interface = UVSI Dynamic = AXI4-Lite Others = Default	250	250	250	1,898	2,479	2
Memory Interface = AXI4 Video Interface = UVSI Dynamic = AXI4-Lite FIFO Type = Distributed Others = Default	250	250	250	1,896	2,479	2

Notes:

1. Fmax is generated when the FPGA design only contains the Video Frame Buffer IP Core and the target frequency is 200 MHz. These values may be reduced when user logic is added to the FPGA design.
2. Fmax is generated using multiple iterations of Place and Route.

References

- Video Frame Buffer IP Release Notes (FPGA-RN-02042)
- LPDDR4 Memory Controller IP Core for Nexus Devices User Guide (FPGA-IPUG-02127)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Arm Developer web page
- Certus-NX web page
- Certus-N2 web page
- CertusPro-NX web page
- CrossLink-NX web page
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Video Frame Buffer IP Core 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 1.6, IP v1.7.0, 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 IP version.
	  • Removed earlier IP versions.
	• Updated the Licensing and Ordering Information section.

Functional Description	Removed the conditions where the IP outputs pixels continuously in the Frame Rate Conversion section.

Designing with the IP	• Added a note on IP version in GUI in the Designing with the IP section.
	• Updated the following figures:
	  • Figure 7.1. Module/IP Block Wizard
	  • Figure 7.2. Configure User Interface of Video Frame Buffer IP Core
	  • Figure 7.3. Check Generated Result

Design Considerations	Updated Design Considerations and Limitations sections.
Resource Utilization	Updated resource utilization for the latest software version.
References	Updated references.

Revision 1.5, IP v1.6.0, July 2025

Section	Change Summary

Introduction	• Updated Table 1.1. Quick Facts as follows:
	  • Renamed *Supported FPGA Families* to *Supported Devices*.
	  • Removed the *Targeted Devices* row.
	  • Added IP version.
	• Changed from *Multi-site Perpetual* to *Single Seat Perpetual* in Table 1.3. Ordering Part Number.

Designing with the IP	Updated the following figures: • Figure 7.1. Module/IP Block Wizard • Figure 7.2. Configure User Interface of Video Frame Buffer IP Core • Figure 7.3. Check Generating Result • Figure 7.4. Simulation Wizard • Figure 7.5. Adding and Reordering Source • Figure 7.6. Simulation Waveform
Design Considerations	Added the Limitations section and added limitations.
Resource Utilization	Updated resource utilization for the latest software version.
References	Updated references.

Revision 1.4, IP v1.5.0, December 2024

Section	Change Summary

Introduction	• Updated Table 1.1. Quick Facts as follows:
	  • Added Certus-N2 device.
	  • Added targeted devices: LFD2NX-9, LFD2NX-28, and LAV-AT-E30.
	  • Added IP changes.

Section	Change Summary
	• Updated IP version. • Added Supported User Interface. • Added the IP Support Summary section. • Updated Table 1.3. Ordering Part Number as follows: • Added OPN for Certus-N2 devices. • Changed from *Single Machine Annual* to *Single Seat Annual*. • Added the Hardware Support section.
Functional Description	• Updated the Memory Bandwidth and Size section to specify the use of native memory interface. • Updated the description on synchronous reset in the Clocking and Reset Overview section. • Updated the Memory Interface section to split in the following subsections: • Native Memory Interface • AXI4 Memory Interface
Signal Description	Added the *axi_awprot_o*, *axi_arprot_o*, and *axil_wstrb_i* ports in Table 4.1. Video Frame Buffer IP Core Signal Description.
Example Design	Added this section.
Resource Utilization	• Updated Table A.1. Resource Utilization for the LAV-AT-G70-1LFG1156I Device. • Added the following tables: • Table A.2. Resource Utilization for the LFCPNX-100-7LFG672I Device • Table A.3. Resource Utilization for the LN2-CT-20-1ASG410I Devic • Removed table: *Resource Utilization for the LFCPNX-100-7LFG672C Device*
References	Updated references.

Revision 1.3, June 2024

Section	Change Summary
Acronyms in This Document	Added definition for UVSI.
Introduction	• Reworked section 1 *Introduction* and renamed to subsection 1.1 Overview of the IP. • Reworked subsection 1.1 *Quick Facts* and moved to subsection 1.2 Quick Facts. • Reworked subsection 1.2 *Features* and moved to subsection 1.3 Features. • Reworked subsection 3.1 *Licensing the IP* and subsection 3.5 *Hardware Evaluation* and renamed to subsection 1.4 Licensing and Ordering Information. • Reworked subsection 1.3 *Ordering Part Number* and moved to subsection 1.4.1 Ordering Part Number. • Moved subsection 1.4 *Conventions* and renamed to subsection 1.5 Naming Conventions. • Added subsection 1.6 Minimum Device Requirements.
Functional Description	• Reworked subsection 2.1 *Overview* and renamed to subsection 2.1 IP Architecture Overview. • Reworked subsection 2.4 *Frame Rate Conversion* and moved to subsection 2.2 Frame Rate Conversion. • Reworked subsection 2.5 *Dynamic Parameter Updating* and moved to subsection 2.3 Dynamic Parameter Updating. • Reworked subsection 2.6 *Memory Bandwidth and Size* and moved to subsection 2.4 Memory Bandwidth and Size. • Added subsection 2.5 Clocking. • Reworked subsection 2.8.1 *Video Input/Output Timing* and moved to subsection 2.5.1 Video Input/Output Timing. • Reworked subsection 2.8.2 *Video Frame Timing* and moved to subsection 2.5.2 Video Frame Timing. • Reworked subsection 2.8.3 *Memory Interface Timing* and moved to subsection 2.5.3 Memory Interface Timing. • Reworked subsection 2.8.4 *Dynamic Parameter Updating* and moved to subsection 2.5.4

Section	Change Summary

	Dynamic Parameter Updating.
	• Added the following subsections:
	  • 2.5.5 Clocking and Reset Overview
	  • 2.5.6 Clock Domains and Clock Domain Crossing
	• Added subsection 2.6 User Interfaces.
	• Reworked subsection 2.7.1 *Video Input/Output* and moved to subsection 2.6.1 Video Input/Output.
	• Reworked subsection 2.7.2 *Memory Interface* and moved to subsection 2.6.2 Memory Interface.
	• Reworked subsection 2.7.3 *Parameter Register Read/Write Interface* and moved to subsection 2.6.3 Parameter Register Read/Write Interface.
	• Added subsection 2.6.4 Unified Video Streaming Interface.

IP Parameter Description	Reworked subsection 2.3 *Attributes Summary* and renamed to subsection 3 IP Parameter Description.
Signal Description	Reworked subsection 2.2 *Signal Description* and moved to subsection 4 Signal Description.
Register Description	Added this section.
Designing with the IP	• Moved section 3 *Core Generation, Simulation, and Validation* to section 6 Designing with the IP. • Reworked subsection 3.2 *Generation and Synthesis* and moved to subsection 6.1 Generation and Synthesis. • Moved subsection 3.4 *Constraining the IP* to subsection 6.2 Constraining the IP. • Moved subsection 3.3 *Running Functional Simulation* to subsection 6.3 Running Functional Simulation.
Debugging	Added this section.
Design Considerations	Added this section.
Resource Utilization	Reworked section content.
References	Updated references.

Revision 1.2, February 2024

Section	Change Summary
All	• Renamed document from *Video Frame Buffer IP Core - Lattice Radiant Software* to *Video Frame Buffer IP*. • Performed minor formatting and typo edits.
Disclaimers	Updated disclaimers.
Inclusive Language	Added inclusive language boilerplate.
Acronyms in This Document	Added acronym definitions for AMBA and AXI.
Introduction	• Added Lattice Avant devices in Table 1.1. Quick Facts. • Added support for AXI interfaces in the Features section. • Added the Ordering Part Number section.

Functional Description	• Added information on AMBA bus interface in the Overview section.
	• Updated the following figures:
	  • Figure 2.1. Video Frame Buffer IP Core Functional Diagram
	  • Figure 2.2. Video Frame Buffer IP Core I/O
	• Updated Table 2.1. Video Frame Buffer IP Core Signal Description as follows:
	  • Updated description for mem_clk_i and pclk_i.
	  • Add notes for ports.
	  • Added ports for AXI4 Manager Interface and AXI4-Lite Subordinate Interface.
	• Updated Table 2.2. Attributes Table as follows:
	  • Added attributes: Memory interface, Parameter bus interface, and Video interface.
	  • Updated the dependency on other attributes for Read FIFO depth and Write FIFO depth attributes.

Section	Change Summary
	• Updated the description for Read FIFO depth and Write FIFO depth attributes in Table 2.3. Attributes Descriptions. • Added information on AXI in the Memory Interface and Parameter Register Read/Write Interface sections.
Core Generation, Simulation, and Validation	• Changed black box to closed-box in the Generation and Synthesis section. • Added the Constraining the IP section.
References	Updated references.
Technical Support Assistance	Added link to the Lattice Answer Database.

Revision 1.1, June 2021

Section	Change Summary
All	Minor adjustments in formatting.
Introduction	Updated content, including Table 1.1 to add CertusPro-NX support.
References	Updated this section to add CertusPro-NX web page.

Revision 1.0, October 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com