



MIPI CSI-2 TX Controller Core User Guide

UG-CORE-MIPI-CSI2-TX-v3.4
July 2025
www.efinixinc.com



Contents

Introduction.....	3
Features.....	3
Device Support.....	4
Resource Utilization and Performance.....	4
Release Notes.....	4
Functional Description.....	5
Ports.....	6
Pixel Clock Calculation.....	9
Control Status Registers.....	9
Pixel Encoding.....	11
MIPI TX Video Data DATA[63:0] Formats.....	12
Video Timing Parameters.....	15
Minimum Horizontal Blanking Per Line Requirement.....	16
Interleaved Data Transmission with Virtual Channels.....	17
Reset Sequence and Initialization.....	18
IP Manager.....	19
Customizing the MIPI CSI-2 TX Controller.....	20
MIPI CSI-2 TX Controller Example Design.....	22
MIPI CSI-2 TX Controller Testbench.....	24
Revision History.....	25

Introduction

The MIPI CSI-2 interface, which defines a simple, high-speed protocol, is the most widely used camera interface for mobile⁽¹⁾. Adding a MIPI interface to an FPGA creates a powerful bridge to transmit or receive high-speed video data easily to/from an application processor. The MIPI CSI-2 TX Controller core allows you to perform complex video and image processing as a part of a complete system solution.

Use the IP Manager to select IP, customize it, and generate files. The MIPI CSI-2 TX Controller core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix[®] development board.

Features

- Configurable data lanes: 1, 2, 4, or 8
- High-speed (HS) mode and Low-power (LP) mode
- Arbitrary number of payload data bytes
- HS mode byte clock frequency from 10 MHz up to 187 MHz (from 80 Mbps up to 1,500 Mbps data rate)⁽²⁾
- Continuous HS mode byte clock and discontinuous HS mode byte clock
- 8-bit HS mode data width
- Pixel format:
 - RAW: RAW6, RAW7, RAW8, RAW10, RAW12, RAW14, RAW16, RAW20, RAW24, RAW28
 - RGB: RGB444, RGB555, RGB565, RGB888
 - YUV: YUV420 8-bit (legacy), YUV420 8-bit, YUV420 10-bit, YUV420 8-bit (CSPS), YUV420 10-bit (CSPS), YUV422 8-bit, YUV422 10-bit
- User defined 8-bit data types
- Generic 8-bit long packet
- Null, blank, and embedded 8-bit non-image data
- PPI interface
- Generic frame mode and accurate frame mode
- Supports end of transmission error, start of transmission sync error, control error & LP escape error
- Supports control status register (CSR) for status and error assertion accessed through AXI4-Lite interface

⁽¹⁾ Source: MIPI Alliance.

⁽²⁾ The maximum data rate of IP depends on the devices. Refer to the respective device data sheet for more accurate information.

Device Support

Table 1: MIPI CSI-2 TX Controller Core Device Support

FPGA Family	Supported Device
Trion	-
Titanium and Topaz	All

Resource Utilization and Performance



Note: The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and can change depending on the device resource utilization, design congestion, and user design.

Table 2: Titanium Resource Utilization and Performance

MIPI CSI-2 TX Controller with 4 data lanes.

FPGA	Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	f_{MAX} (MHz) ⁽³⁾				Efinity® Version ⁽⁴⁾
				clk	axi_clk	clk_byte_HS	clk_pixel	
Ti60 F225 C4	6,072/60,800 (10.0%)	7/256 (2.7%)	1/160 (0.6%)	461	594	363	362	2021.2

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes are available on the [Efinity Downloads](#) page under each Efinity software release version.



Note: You must be logged in to the Support Center to view the IP Core Release Notes.

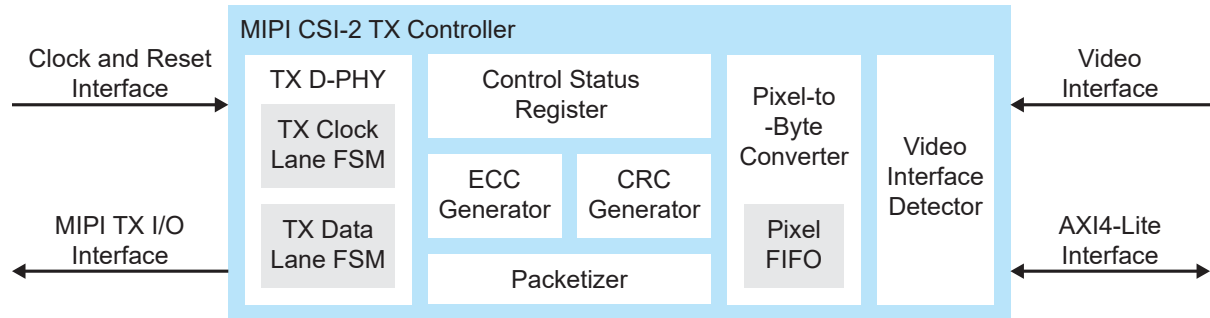
⁽³⁾ Using default parameter settings.

⁽⁴⁾ Using Verilog HDL.

Functional Description

The MIPI CSI-2 TX Controller consists of a TX D-PHY block, control status registers, ECC and CRC generators, packetizer, pixel-to-byte converter, and camera interface detector. The core has a video, AXI4-lite, MIPI TX I/O, and clock and reset interfaces.

Figure 1: MIPI CSI-2 TX Controller System Block Diagram



Ports

Table 3: Clock and Reset Ports

Port	Direction	Description
clk	Input	IP core clock consumed by controller logics. 100 MHz.
reset_n	Input	IP core reset signal.
clk_byte_HS	Input	MIPI TX parallel clock. This is a HS mode transmission clock.
reset_byte_HS_n	Input	MIPI TX parallel clock reset signal.
clk_pixel	Input	Pixel clock.
reset_pixel_n	Input	Pixel clock reset signal.
axi_clk	Input	AXI4-Lite interface clock.
axi_reset_n	Input	AXI4-Lite interface reset.



Note: Refer to the Interfaces User Guide in the [Support Center](#) for serial or parallel clock requirements.

Table 4: MIPI TX I/O interface

Port	Direction	Description
Tx_LP_CLK_P	Output	LP mode TX clock single-ended P signal.
Tx_LP_CLK_N	Output	LP mode TX clock single-ended N signal.
Tx_LP_CLK_P_OE	Output	Output enable for LP mode TX clock single-ended P signal.
Tx_LP_CLK_N_OE	Output	Output enable for LP mode TX clock single-ended N signal.
Tx_HS_enable_C	Output	Signal to enable HS mode clock lane.
Tx_LP_D_P [NUM_DATA_LANE-1:0]	Output	LP mode TX data single-ended P signal.
Tx_LP_D_N [NUM_DATA_LANE-1:0]	Output	LP mode TX data single-ended N signal.
Tx_LP_D_P_OE [NUM_DATA_LANE-1:0]	Output	Output enable for LP mode TX data single-ended P signal.
Tx_LP_D_N_OE [NUM_DATA_LANE-1:0]	Output	Output enable for LP mode TX data single-ended N signal.
Tx_HS_D_n[7:0]	Output	HS mode differential lane data bus. $n = \text{lane } 0 \text{ to } 7$
Tx_HS_enable_D [NUM_DATA_LANE-1:0]	Output	Signal to enable HS mode data lane.
Tx_HS_C [7:0]	Output	HS mode differential clock bus.

Table 5: Video Interface

All signals are clocked with `clk_pixel` and `reset_pixel_n`.


Port	Direction	Description
<code>hsync_vcx</code>	Input	Active-high horizontal sync for virtual channel. $x = \text{virtual lane } 0 \text{ to } 15$
<code>vsync_vcx</code>	Input	Active-high vertical sync for virtual channel. $x = \text{virtual lane } 0 \text{ to } 15$
<code>datatype [5:0]</code>	Input	Data type of the long packet. Sampled at Hsync rising edge.
<code>pixel_data [63:0]</code>	Input	Video Data. The actual width is dependent on pixel type. Refer to the pixel encoding table.
<code>pixel_data_valid</code>	Input	Active-high pixel data enable. Once the TX VALID signal goes high, the MIPI TX interface expects to receive pixel data every clock cycle until the entire line is sent. Additionally, the TX VALID signal must remain high for the entire line.
<code>haddr [15:0]</code>	Input	16 bit horizontal number of pixels. Sampled at Hsync rising edge.  Note: Total pixel count should be aligned to pixels-per-clock boundary to prevent ambiguity on the <code>pixel_data[63:0]</code> .
<code>line_num[15:0]</code>	Input	Line number to use. Sampled at Hsync rising edge.
<code>frame_num[15:0]</code>	Input	Frame number to use. Sampled at Vsync rising edge.

Table 6: AXI4-Lite Interface

Interface to access [Table 8: Control Status Registers](#) on page 9.

All signals are clocked with `axi_clk` and `axi_reset_n`.

Port	Direction	Description
<code>axi_awaddr [15:0]</code>	Input	AXI4-Lite write address bus.
<code>axi_awvalid</code>	Input	AXI4-Lite write address valid strobe.
<code>axi_awready</code>	Output	AXI4-Lite write address ready signal.
<code>axi_wdata [31:0]</code>	Input	AXI4-Lite write data.
<code>axi_wvalid</code>	Input	AXI4-Lite write data valid strobe.
<code>axi_wready</code>	Output	AXI4-Lite write ready signal.
<code>axi_bvalid</code>	Output	AXI4-Lite write response valid strobe.
<code>axi_bready</code>	Input	AXI4-Lite write response ready signal.
<code>axi_araddr [15:0]</code>	Input	AXI4-Lite read address bus.
<code>axi_arvalid</code>	Input	AXI4-Lite read address valid strobe.
<code>axi_arready</code>	Output	AXI4-Lite read address ready signal.
<code>axi_rdata [31:0]</code>	Output	AXI4-Lite read data.
<code>axi_rvalid</code>	Output	AXI4-Lite read data valid strobe.
<code>axi_rready</code>	Input	AXI4-Lite read data ready signal.

Table 7: Debug Interface

All signals are clocked with axi_clk and axi_reset_n.

Port	Direction	Description
mipi_debug_out[31:0]	Output	<p>Debug port. Present if the parameter MIPI_CSI2_TX_DEBUG is Enabled. The following is the list of internal signals that can be monitored.</p> <ul style="list-style-type: none"> [0] = fifo_full [1] = fifo_empty [2] = non_support_longpkt [3] = ready_to_xmit (indicator for initialization done, prior to skewcal) [4] = init_skewcal_done (indicator for skewcal process completion) [5] = TxStopState_0 [6] = TxStopState_1 [7] = TxStopState_2 [8] = TxStopState_3 [9] = TxStopState_4 [10] = TxStopState_5 [11] = TxStopState_6 [12] = TxStopState_7 [31:13] = reserved
mipi_debug_in[31:0]	Input	<p>Debug ports. Present if the parameter MIPI_CSI2_TX_DEBUG is Enabled.</p> <p>Currently no function has been implemented. Tie all input bits to zero.</p> <ul style="list-style-type: none"> [31:0] = reserved

Pixel Clock Calculation

The following formula calculates the pixel clock frequency that you need to drive the pixel clock input port, `clk_pixel`.

$$\text{PIX_CLK_MHZ} \leq (\text{DATARATE_MBPS} * \text{NUM_DATA_LANE}) / \text{PACK_BIT}$$

where:

- `PIX_CLK_MHZ` is the pixel clock in MHz
- `DATARATE_MBPS` is the MIPI data rate in Mbps
- `NUM_DATA_LANE` is the number of data lanes
- `PACK_BIT` is the Pixel data bits per pixel clock from [Pixel Encoding](#) on page 11

Control Status Registers

Table 8: Control Status Registers

Word Address Offset	Name	R/W	Width (bits)
0x00	Interrupt Status Register	Bit[1:0],[3] - R Bit[2] - R/W1C ⁽⁵⁾	4
0x04	Interrupt Enable Register	R/W	5
0x08	D-PHY stop state status for lane 0 to lane 7	R	8
0x0C	D-PHY Ultra Low-Power State (ULPS) status	R	9
0x10	D-PHY Skew Calibration Control	R/W	8
0x14	Reserved	-	-
0x18	D-PHY ULPS control signal	R/W	9

Table 9: Interrupt Status Register Definition (0x00)

Bit	Name	Description
0	Pixel FIFO full	Pixel FIFO in the pixel-to-byte converter module is full.
1	Pixel FIFO empty	Pixel FIFO in the pixel-to-byte converter module is empty.
2	Unsupported video data type	The TX controller received an unsupported video data type from the user through port datatype[5:0].
3	Initialization complete	The core asserts this signal high when tlnit timing parameter is met. TX controller is ready to send MIPI transaction.

⁽⁵⁾ Read register. Write 1 to clear the register.

Table 10: Interrupt Enable Register Definition (0x04)

Each enabled interrupt status bit is aggregated to the `irq` output port as indicator. By default, all interrupt enable registers are set to 1'b0 (disabled).

Bit	Name	Description
0	Pixel FIFO full interrupt enable	Enable interrupt generation for PixelFIFO full status bit.
1	Pixel FIFO empty full interrupt enable	Enable interrupt generation for PixelFIFO empty status bit.
2	Unsupported video data type full interrupt enable	Enable interrupt generation for Unsupportedvideo data type status bit.
3	Initialization complete full interrupt enable	Enable interrupt generation for Initialization complete status bit.

Table 11: D-PHY Ultra Low-Power State (ULPS) Status (0x0C)

Bit	Name	Description
0	TxUlpsActiveClkNot	ULPS (not) Active. The core deasserts this signal low to indicate that the clock lane is in ULPS.
8:1	TxUlpsActiveNot_7 to TxUlpsActiveNot_0	ULPS (not) Active. The core deasserts this signal low to indicate that the data lane is in ULPS.

Table 12: D-PHY Skew Calibration Control Register Definition (0x10)

Bit	Name	Description
7:0	TxSkewCalHS[7:0]	High-Speed Transmit Skew Calibration. When the register is set to 1, core asserts TxSkewCalHS signal high causes the PHY to initiate a de-skew calibration. When the register is set to 0, core deasserts TxSkewCalHS signal low causes the PHY to stop deskew pattern transmission and initiate an end-of-transmission sequence. This feature is only applicable to data rate above 1.5 Gbps. Do not apply for data rate of 1.5 Gbps and below.

Table 13: D-PHY ULPS Control Signal Register Definition (0x18)

Bit	Name	Description
0	TxUlpsClk	Transmit ULPS on Clock Lane. The core asserts this signal high to cause a clock lane module to enter the ULPS. The lane module remains in this mode until TxUlpsClk is de-asserted.
8:1	TxUlpsEsc[7:0]	Escape Mode Transmit ULPS. For LP implementations, the core asserts this and TxRequestEsc signals high to cause the lane module to enter the ULPS. The lane module remains in this mode until TxRequestEsc is de-asserted.

Pixel Encoding

Table 14: Pixel Encoding

TYPE[5:0]	Data Type	Pixels per Clock	Bits per Pixel	Pixel Data Bits per Pixel Clock
0x20	RGB444	4	12	48
0x21	RGB555	4	15	60
0x22	RGB565	4	16	64
0x24	RGB888	2	24	48
0x28	RAW6	8	6	48
0x29	RAW7	8	7	56
0x2A	RAW8	8	8	64
0x2B	RAW10	4	10	40
0x2C	RAW12	4	12	48
0x2D	RAW14	4	14	56
0x2E	RAW16	4	16	64
0x2F	RAW20	2	20	40
0x27	RAW24	2	24	48
0x26	RAW28	2	28	56
0x18	YUV420 8 bit	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	Odd line: 64 Even line: 64
0x19	YUV420 10 bit	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	Odd line: 40 Even line: 40
0x1A	Legacy YUV420 8 bit	4	8, 16	48
0x1C	YUV420 8 bit (CSPS)	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	Odd line: 64 Even line: 64
0x1D	YUV420 10 bit (CSPS)	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	Odd line: 40 Even line: 40
0x1E	YUV422 8 bit	4	8, 24	64
0x1F	YUV422 10 bit	2	10, 30	40
0x30 - 37	User defined 8 bit	8	8	64
0x13 - 0x16	Generic 8-bit long packet	8	8	64
0x12	Embedded 8-bit non image data	8	8	64

MIPI TX Video Data DATA[63:0] Formats

The format depends on the data type. New data arrives on every pixel clock.

Table 15: RAW6 (8 Pixels per Clock)

63	48	47	42	41	36	35	30	29	24	23	18	17	12	11	6	5	0
0		Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1								

Table 16: RAW7 (8 Pixels per Clock)

63	56	55	49	48	42	41	35	34	28	27	21	20	14	13	7	6	0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1									

Table 17: RAW8 (8 Pixels per Clock)

63	56	55	48	47	40	39	32	31	24	23	16	15	8	7	0
Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1								

Table 18: RAW10 (4 Pixels per Clock)

63	40	39	30	29	20	19	10	9	0
0		Pixel 4	Pixel 3	Pixel 2	Pixel 1				

Table 19: RAW12 (4 Pixels per Clock)

63	48	47	36	35	24	23	12	11	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1					

Table 20: RAW14 (4 Pixels per Clock)

63	56	55	42	41	28	27	14	13	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1					

Table 21: RAW16 (4 Pixels per Clock)

63	48	47	32	31	16	15	0
Pixel 4	Pixel 3	Pixel 2	Pixel 1				

Table 22: RAW20 (2 Pixels per Clock)

63	40	39	20	19	0
0	Pixel 2	Pixel 1			

Table 23: RAW24 (2 Pixels per Clock)

63	48	47	24	23	0
0	Pixel 2	Pixel 1			

Table 24: RAW28 (2 Pixels per Clock)

63	56	55					28	27					0
0	Pixel 2						Pixel 1						

Table 25: RGB444 (4 Pixels per Clock)

63	48		47	36			35	24			23	12		11	0
0	Pixel 4			Pixel 3			Pixel 2			Pixel 1					
–	[47:44]	[43:40]	[39:36]	[35:32]	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]			
	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue			

Table 26: RGB555 (4 Pixels per Clock)

63	60	59	45			44	30			29	15		14	0
0	Pixel 4			Pixel 3			Pixel 2			Pixel 1				
–	[59:55]	[54:50]	[49:45]	[44:40]	[39:35]	[34:30]	[29:25]	[24:20]	[19:15]	[14:10]	[9:5]	[4:0]		
	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue		

Table 27: RGB565 (4 Pixels per Clock)

63	48		47	32			31	16			15	0
Pixel 4			Pixel 3			Pixel 2			Pixel 1			
[63:59]	[58:53]	[52:48]	[47:43]	[42:37]	[36:32]	[31:27]	[26:21]	[20:16]	[15:11]	[10:5]	[4:0]	
Red	Green	Blue	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue	

Table 28: RGB888 (2 Pixels per Clock)

63	48		47	24			23	0		
0	Pixel 2			Pixel 1						
–	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]				
	Red	Green	Blue	Red	Green	Blue				

Table 29: YUV420 8 bit Odd Line (8 Pixels per Clock), Even Line (4 Pixels per Clock)

63	56	55	48	47	40	39	32	31	24	23	16	15	8	7	0
Odd Lines															
Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1								
Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1								
Even Lines															
Pixel 4	Pixel 3			Pixel 2	Pixel 1										
Y4	V3	Y3	U3	Y2	V1	Y1	U1								

Table 30: Legacy YUV420 8 bit (4 Pixels per Clock)

63	48		47	40			39	32			31	24			23	16		15	8		7	0
0	Pixel 4		Pixel 3			Pixel 2		Pixel 1														
Odd Lines	Y4	Y3	U3	Y2	Y1	U1																
Even Lines	Y4	Y3	V3	Y2	Y1	V1																

Table 31: YUV420 10 bit Odd Line (4 Pixels per Clock), Even Line (2 Pixels per Clock)

63	40 39	30 29	20 19	10 9	0
Odd Lines					
0	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1	
Even Lines					
0	Pixel 1	Pixel 2	Pixel 1		
	Y2	V1	Y1	U1	

Table 32: YUV422 8 bit (4 Pixels per Clock)

63	56 55	48 47	40 39	32 31	24 23	16 15	8 7	0
Pixel 4	Pixel 3			Pixel 2	Pixel 1			
Y4	V3	Y3	U3	Y2	V1	Y1	U1	

Table 33: YUV422 10 bit (2 Pixels per Clock)

63	40 39	30 29	20 19	10 9	0
0	Pixel 1	Pixel 2	Pixel 1		
	Y2	V1	Y1	U1	

Video Timing Parameters

The following waveforms show the video interface signals relationship.

Figure 2: Video Timing Waveform (Horizontal)

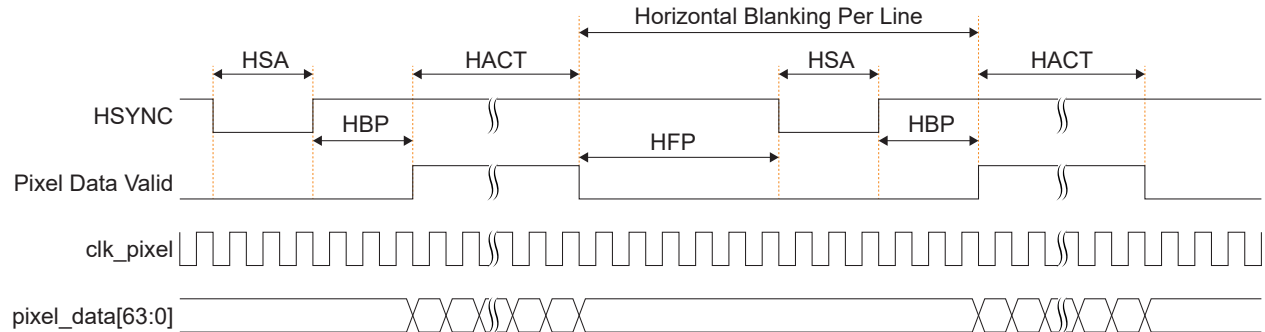


Figure 3: Video Timing Waveform (Vertical)

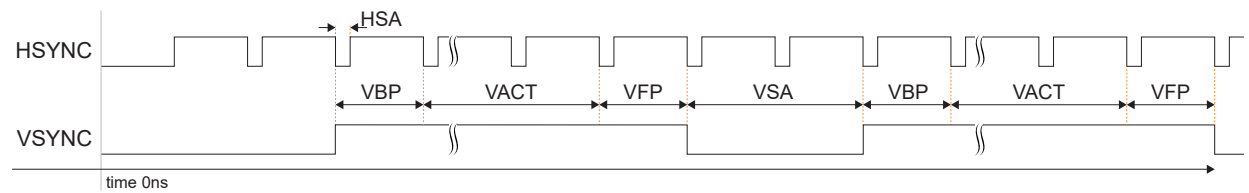


Table 34: Video Timing Parameter Definitions

MIPI Video Timing Parameters	Definition	Min	Max	Unit
HACT	Total number of pixel per line	16	8,192	Pixel
VACT	Total number of line per frame	1	8,192	Line
HSA	HSYNC pulse width	1	4,096	Pixel
HBP	Horizontal back porch	1	4,096	Pixel
HFP	Horizontal front porch	1	4,096	Pixel
VSA	VSNC pulse width	1	8,192	Line
VBP	Vertical back porch	1	8,192	Line
VFP	Vertical front porch	1	8,192	Line
Pixel Clock	Video stream pixel clock frequency in MHz	(6)	(6)	MHz
MIPI Speed	CSI-2 TX MIPI speed in Mbps	80	1,500	Mbps
No. data lane	Number of MIPI data lane	1	8	Lane

Table 34: Video Timing Parameter Definitions on page 15 describes the support range for video timing parameters. For a more precise check on timing compliance, you can refer to the **Titanium MIPI Utility**.

(6) Refer to **Pixel Clock Calculation** on page 9.

Minimum Horizontal Blanking Per Line Requirement

Video data is typically clocked at 1-pixel data per 1-pixel clock. The MIPI CSI-2 TX Controller Core includes a highly compressed 64-bit pixel data bus that offers the flexibility to clock multiple pixel data per 1-pixel clock depending on the data format (See [Table 14: Pixel Encoding](#) on page 11). This compression introduces some latencies for the internal logic to decompress the pixel data, converting them into byte clock domain, and transmitting out as byte data into MIPI I/O. As a result, minimum horizontal blanking per line needs to be fulfilled to prevent any data corruption during pixel data transmission.

The minimum horizontal blanking per line (in pixel clock) is defined as:

$$\text{HFP} + \text{HBP} + \text{HSA} > \text{byte_data_transfer_period} - \text{pixel_data_valid_period} + \text{LP-HS_timing}$$

Where,

$$\text{byte_data_transfer_period} = (\text{HACT} * \text{bit_per_pixel}) / 16 / \text{Lane_NUM} * \text{byte_clk_period}$$

$$\text{pixel_data_valid_period} = \text{HACT} / \text{pixels_per_clk} * \text{pixel_clk_period}$$

LP-HS_timing = DPHY timing between the lower-power and high-speed mode transition. For more details, download the [Titanium MIPI Utility.xlsm](#) from the example design portal.

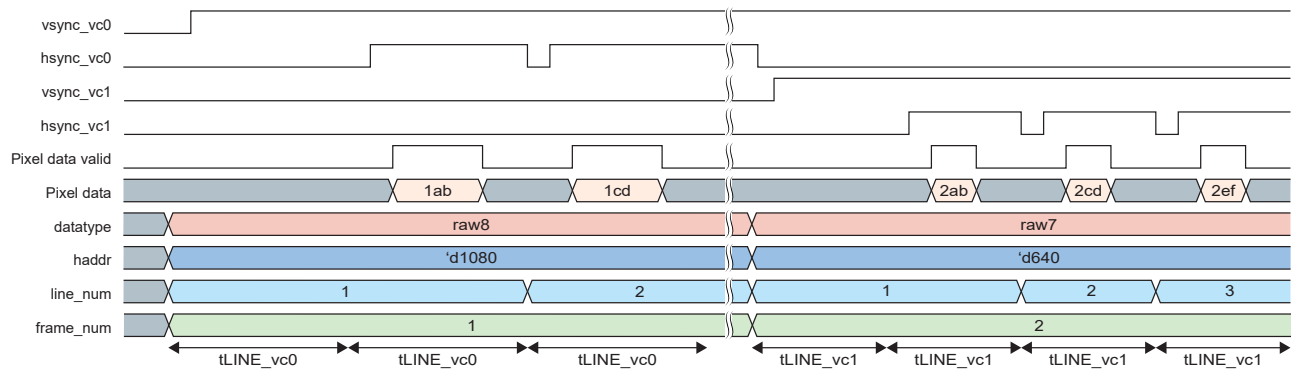
Interleaved Data Transmission with Virtual Channels

The MIPI CSI-2 TX Controller core supports interleaved data transmission of different images from different virtual channels to maximize bandwidth utilization. The interleaving data can be a per-frame or per-horizontal-line basis. The following figures illustrate examples of interleaved data inputs signaling at the pixel interface. With different interleaving methods, the total horizontal line period ($t_{LINE} = \text{horizontal active} + \text{blanking period}$) requirements need to be fulfilled to prevent data corruption during pixel-to-byte conversion and data transfer from one horizontal data line to another.

Example: Per-frame basis interleaving

Virtual channel 0: datatype = raw8, horizontal resolution = 1080
 Virtual channel 1: datatype = raw7, horizontal resolution = 640
 t_{LINE_vc0} and t_{LINE_vc1} can be different.

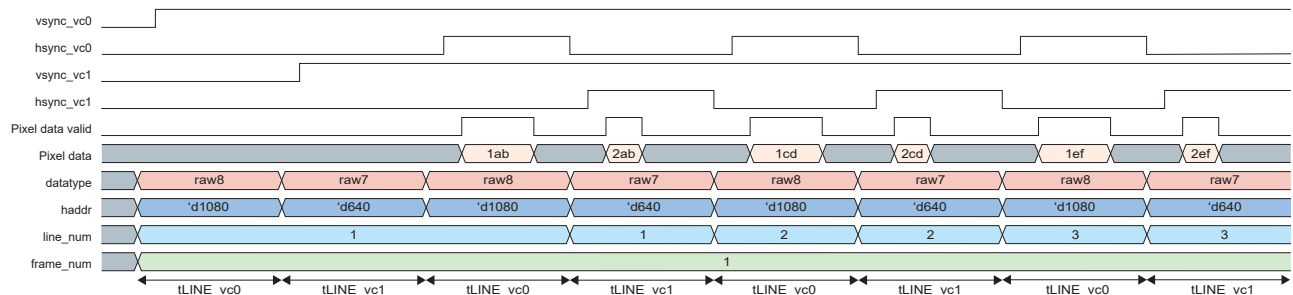
Figure 4: Interleaved Data Transmission Waveform (Per-Frame)



Example: Per-horizontal-line basis interleaving

Virtual channel 0 - datatype = raw8, horizontal resolution = 1080
 Virtual channel 1 - datatype = raw7, horizontal resolution = 640
 t_{LINE_vc0} and t_{LINE_vc1} need to be identical.

Figure 5: Interleaved Data Transmission Waveform (Per-Horizontal Line)

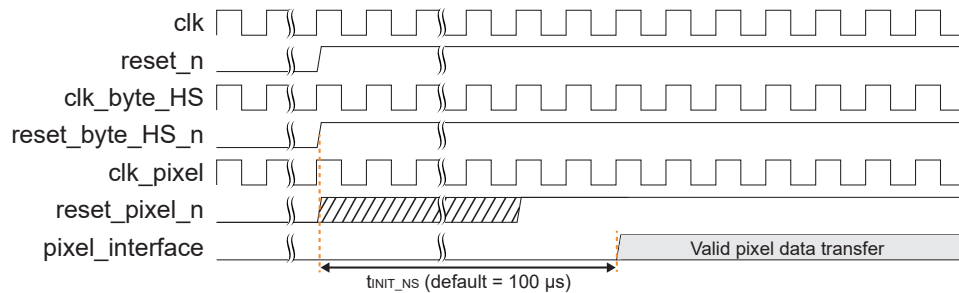


Reset Sequence and Initialization

During the initial power-up state, an initialization time, t_{INIT_NS} of $100\ \mu s$ is the minimum requirement for the MIPI D-PHY transmitter to function properly before an LP/HS data transfer. You must ensure that no traffic is sent to the pixel interface before the expiration of t_{INIT_NS} . For a valid data transfer to take place, there should be at least 2-pixel clocks after the `reset_pixel_n` is out from reset. For any reset assertion event during user mode, you must follow the reinitialization sequence for reset deassertion procedure.

The following figure describes the reset sequencing and initialization requirements.

Figure 6: Reset Sequence and Initialization



IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinity® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an Efinity development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.



Note: Not all Efinity IP cores include an example design or a testbench.

Generating the MIPI CSI-2 TX Controller Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **MIPI > MIPI CSI-2 TX Controller** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.



Note: You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the *Customizing the MIPI CSI-2 TX Controller* section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinity® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.



Note: You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:

- **<module name>_define.vh**—Contains the customized parameters.
- **<module name>_tpl.v**—Verilog HDL instantiation template.
- **<module name>_tpl.vhd**—VHDL instantiation template.
- **<module name>.v**—IP source code.
- **settings.json**—Configuration file.
- **<kit name>_devkit**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains generated RTL and testbench files.

Customizing the MIPI CSI-2 TX Controller

The core has parameters so you can customize its function. You set the parameters in the **General** tab of the core's IP Configuration window.

Table 35: MIPI CSI-2 TX Controller Core Parameter

Name	Option	Description
t _{LPX} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 50
t _{INIT} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100,000
Data Lanes	1, 2, 4	Number of data lanes Default: 4
MIPI Parallel Clock Frequency	10 - 187	MIPI parallel clock (clk_byte_HS) frequency in MHz to support data rate of 80 Mbps to 1,500 Mbps. Default: 187
IP Core Clock Frequency	40 - 100	IP core clock frequency in MHz Default: 100
DPHY Clock Mode	continuous, discontinuous	To enable discontinuous or continuous HS clock. Continuous refers to the HS byte clock continuously running during LP or HS mode. Discontinuous refers to the HS byte clock in stopped condition during LP mode and in running condition during HS mode. Default: Continuous
Pixel Data FIFO Depth Size	256 - 8192	FIFO depth size to store the pixel packet data (set to power of 2 value). Minimum FIFO depth required > horizontal_pixel (HACT) x bits_per_pixel / 64 Default: 1024
Image Frame Mode	GENERIC, ACCURATE	Selects frame mode. Generic mode: Frame format without accurate synchronization timing via Line Start and Line End. Accurate mode: Frame format with accurate synchronization timing via Line Start and Line End. Default: Generic
t _{LP_EXIT} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100
t _{CLK_ZERO} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 400
t _{CLK_TRAIL} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 80
t _{CLK_PRE} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 10

Name	Option	Description
t _{CLK_POST} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding 52 UI). Default: 455
t _{CLK_PREPARE} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 50
t _{WAKEUP} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 1000
t _{HS_ZERO} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). Default: 262
t _{HS_TRAIL} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). Actual = t _{HS_TRAIL_NS} + 4UI or 8UI (whichever bigger) Default: 60
t _{HS_EXIT} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100
t _{HS_PREPARE} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI).. Default: 40
Pack Type 40	Enable, Disable	Enables the controller to pack RAW10, RAW20, YUV_420_10, and YUV_422_10 data type. ⁽⁷⁾ Default: Enable
Pack Type 48	Enable, Disable	Enables the controller to pack RAW6, RAW12, RAW24, RGB888, and YUV_420_8_legacy data type. ⁽⁷⁾ Default: Enable
Pack Type 56	Enable, Disable	Enables the controller to pack RAW7, RAW14, and RAW28. ⁽⁷⁾ Default: Enable
Pack Type 64	Enable, Disable	Enables the controller to pack RAW8, RAW16, RGB444, RGB565, RGB555, YUV_422_8, YUV_420_8, generic long packet, user define 8-bit, and embedded 8-bit non image packet. ⁽⁷⁾ Default: Enable
Enable Extra Bits on Virtual Channel	Enable, Disable	Enable - 16 virtual channels are available. Disable - only 4 virtual channels are available. Default: Disable
MIPI_CSI2_TX_DEBUG	Enable, Disable	Enables debug ports for internal signal observation and monitoring. Default: Disable

⁽⁷⁾ Only enable the pack type that you are using to save logic resources.

MIPI CSI-2 TX Controller Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

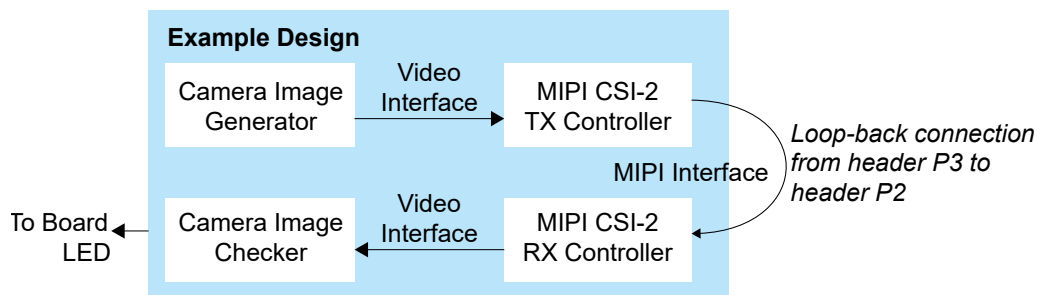


Important: Efinix tested the example design generated with the default parameter options only.

The example design targets the Titanium Ti60 F225 Development Board. The design instantiates both MIPI CSI-2 TX and RX Controller cores. This design requires a QTE header-compatible cable.

The design generates an image and sends the image data to the camera image checker through the MIPI CSI-2 TX Controller. The data is then sent through a hardware loopback on the board using a 4-lane MIPI interface to the MIPI CSI-2 RX Controller. The camera image checker compares the data received with the one created by the image generator, and outputs the results using the board LEDs.

Figure 7: MIPI CSI-2 TX Controller Core Example Design



After power-up and device programming is done, the following response can be observed:

- LED0_B—Blinks continuously indicating there are traffic being sent over to MIPI IP.
- LED0_G—Turns on to indicate that the hsync signal matches TX and RX.
- LED1_B—Turns on to indicate that the vsync signal matches TX and RX.
- LED1_G—Turns on to indicate that the pixel data signal matches TX and RX.
- LED1_R—turns on to indicate that the pixel data valid signal matches TX and RX.

The RX clock to RX data skew varies in different board, hence there is a possibility where the RX clock might not be able to capture the RX data correctly. In this case, both the LEDs do not turn on. You have to try increase the **Static Mode Delay Setting** of `mipi_dphy_rx_data` in the Interface Designer.



Note: You can use the **Titanium MIPI Utility-v<version>.xism** to check if your own design will work. Enter the related design information then verify whether your selections pass various tests. You can download the Titanium MIPI utility from the Design Support page in the Support Center.

Table 36: Example Design Implementation

FPGA	Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	f_{MAX} (MHz) ⁽⁸⁾				Efinity® Version ⁽⁹⁾
				clk1	clk2	clk3	clk4	
Ti60 F225 C4	6,329/60,800 (10.4%)	44/256 (17.2%)	0/160 (0%)	231	298	233	265	2024.2.294.4.12

- clk1—mipi_clk
- clk2—mipi_dphy_rx_clk_CLKOUT
- clk3—clk_pixel
- clk4—mipi_dphy_tx_SLOWCLK

⁽⁸⁾ Using default parameter settings.

⁽⁹⁾ Using Verilog HDL.

MIPI CSI-2 TX Controller Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.



Note: You must include all `.v` files generated in the `/testbench` directory in your simulation.



Important: Efinix tested the testbench generated with the default parameter options only.

Efinix provides a simulation script for you to run the testbench quickly using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` in a terminal application. You must have Modelsim installed on your computer to use this script.

The IP Manager generates different encrypted source code for you to simulate with different simulators.

Table 37: Testbench Files

Directory/File	Note
../Testbench/modelsim.do	Modelsim testbench script.
../Testbench/efx_<ip>_modelsim.sv	Encrypted source code for simulating loop-back example design in Modelsim testbench.
../Testbench/aldec	Contains the generated encrypted source code to simulate with the Aldec simulator.
../Testbench/modelsim	Contains the generated encrypted source code to simulate with the Modelsim simulator.
../Testbench/ncsim	Contains the generated encrypted source code to simulate with the NCSIM simulator.
../Testbench/synopsys	Contains the generated encrypted source code to simulate with the VCS simulator.

Tip: The default testbench is Modelsim which complies with CSI-2 TX IP loopback to CSI-2 RX IP. To simulate with other simulators, you can get the encrypted source code from `../Testbench/<simulation_tool>/` folder by generating the targeted CSI-2 IP and replacing it with `efx_csi2_tx_modelsim.sv` or `efx_csi2_rx_modelsim.sv`.

The simulation testbench simulates the example design. The design instantiates both MIPI CSI-2 TX and RX Controller cores. The loopback connection on the MIPI interface is done in the testbench file. This design generates an image and sends the image data to the camera image checker through the MIPI CSI-2 TX Controller and MIPI CSI-2 RX Controller. The camera image checker compares the data received with the one created by the image generator. After running the simulation successfully, the test prints the following message:

```
Correct RX data AA, received
Correct RX data AA, received
```

Revision History

Table 38: Revision History

Date	Document Version	IP Version	Description
July 2025	3.4	5.14	<p>Corrected on Interrupt Status Register in Control Status Registers R/W access attribute table. (DOC-2609)</p> <p>Added more details on the Video Timing Parameter Definition table in the Video Timing Parameters topic.</p> <p>Corrected the frame_num port description in Video Interface table. (DOC-2605)</p>
June 2025	3.3	5.12	<p>Correct description of static delay adjustment in example design.</p>
May 2025	3.2	5.12	<p>Updated example design. (SIP-891)</p> <p>Example Design IO bank update (HVIO 3.3V). (SIP-907)</p> <p>Updated default parameter value. (SIP-910)</p> <p>Updated Customizing the MIPI CSI-2 TX Controller.</p> <p>Added description on debug port for ready_to_xmit and init_skewcal_done in table Debug Interface.</p> <p>Updated observation after downloading bitstream and Example Design Implementation table in Example Design.</p> <p>Updated Reset Sequence and Initialization topic. (DOC-2485)</p>
March 2025	3.1	5.11	<p>Updated Interleaved Data Transmission Waveform (Per-Frame) and Interleaved Data Transmission Waveform (Per-Horizontal Line). (DOC-2442)</p>
January 2025	3.0	5.11	<p>Updated Figure Video Timing Waveform (Vertical), Interleaved Data Transmission Waveform (Per-Frame), and Interleaved Data Transmission Waveform (Per-Horizontal Line). (SIP-823)</p> <p>Added note in Table: Video Interface in Ports topic for user to fully utilize the pixel data bus. (SIP-819)</p> <p>Added description for DPHY Clock Mode and Enable Extra Bit on Virtual Channel in Table 35: MIPI CSI-2 TX Controller Core Parameter on page 20. (DOC-2307)</p> <p>Example design update to align with MIPI Utility change (DOC-1783). (SIP-792)</p>

Date	Document Version	IP Version	Description
December 2024	2.9	5.10	Added debug ports for internal signal observation and monitoring in Ports and Customizing the MIPI CSI-2 TX Controller. (SIP-580) Added section Interleaved Data Transmission with Virtual Channels. (SIP-784)
November 2024	2.8	5.9	Added Topaz in Features and Device Support. (DOC-2102) Added IP Version in Revision History. (DOC-2185) Soft DPHY 1.5Gbps performance improvement. (SIP-614)
September 2024	2.7	-	Updated Table 5: Video Interface on page 7. (DOC-2109)
September 2024	2.6	-	Added 8 lanes support in Features and Table 35: MIPI CSI-2 TX Controller Core Parameter on page 20. (SIP-677) Updated pixel data[63:0] in Figure 2: Video Timing Waveform (Horizontal) on page 15. Removed data type RGB666 from Table 14: Pixel Encoding on page 11. (DOC-2068) Updated t_{CLK_PRE} and t_{CLK_POST} in Table 35: MIPI CSI-2 TX Controller Core Parameter on page 20. (DOC-2078)
July 2024	2.5	-	Fixed typo in Table 6: AXI4-Lite Interface on page 7. (DOC-2004)
June 2024	2.4	-	Updated Pixel FIFO depth requirement in table MIPI CSI-2 TX Controller Core Parameter. (SIP-570) Revised supported lane number in Features and in table MIPI CSI-2 RX Controller Core Parameter. (SIP-578) Added Reset Sequence and Initialization sub-section.
March 2024	2.3	-	Added important note in Testbench regarding using default parameters options only. (DOC-1781) Added testbench file for Modelsim and Aldec simulation model support. (DOC-1782) Updated content and topic title Minimum Horizontal Blanking Per Line to Minimum Horizontal Blanking Per Line Requirement.
October 2023	2.2	-	Updated MIPI video data format tables to include RGB information. (DOC-1474)
September 2023	2.1	-	Updated Minimum Horizontal Blanking Per Line formula and example.
August 2023	2.0	-	Updated Video Timing Waveform (Horizontal) figure and added Minimum Horizontal Blanking Per Line section. (DOC-1414)

Date	Document Version	IP Version	Description
July 2023	1.9	-	Added more description for Accurate and Generic image frame modes. (DOC-1343)
June 2023	1.8	-	Corrected hsync_vcx and vsync_vcx signal directions. (DOC-1341) Added Device Support and release notes sections. (DOC-1234) Updated supported data rate. (DOC-1217) Updated port descriptions. Added RAW16, RAW20, RAW24, and RAW28 format support. Updated MIPI Parallel Clock Frequency, IP Core Clock Frequency, Pixel Data FIFO Depth Size, Pack Type40, Pack Type48, Pack Type56, Pack Type64 parameters. Improved Interrupt Enable Register Definition descriptions. Editorial changes.
February 2023	1.7	-	Added note about the resource and performance values in the resource and utilization table are for guidance only.
August 2022	1.6	-	Updated Control Status Register note. (DOC-898)
August 2022	1.5	-	Added MIPI RX Video Data Formats. Added video parameters waveform, and port clock domains. (DOC-819)
January 2022	1.4	-	Improved description about CSR is accessed through AXI4-Lite interface. (DOC-690) Corrected interrupt status register width and improved D-PHY stop state status description. (DOC-697) Updated resource utilization table. (DOC-700)
December 2021	1.3	-	Added simulation testbench. Added new IP manager parameters. Added new ports.
November 2021	1.2	-	Added support for 8 data lanes. (DOC-604)
October 2021	1.1	-	Added note to state that the f_{MAX} in Resource Utilization and Performance, and Example Design Implementation tables were based on default parameter settings. Updated design example target board to production Titanium Ti60 F225 Development Board and updated Resource Utilization and Performance, and Example Design Implementation tables. (DOC-553)
June 2021	1.0	-	Initial release.