易灵思

MIPI CSI-2 TX Controller Core User Guide

UG-CORE-MIPI-CSI2-TX-v2.2
October 2023
www.elitestek.com

efinity®

Contents

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 易灵思® 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 to 187 MHz (80 to 1,500 Mbps data rate)
- 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

Device Support

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

FPGA Family	Supported Device
Trion	–
钛金系列	All

[1] Source: MIPI Alliance.

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 change depending on the device resource utilization, design congestion, and user design.

Table 2: 钛金系列 Resource Utilization and Performance

MIPI CSI-2 TX Controller with 4 data lanes.

FPGA	Logic and Adders	Flip-flops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)[2]				Efinity® Version[3]
					clk	axi_clk	clk_byte_HS	clk_pixel	
Ti60 F225 C4	4,462	1,610	7	1	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 is available in the Efinity Downloads page under each Efinity software release version.

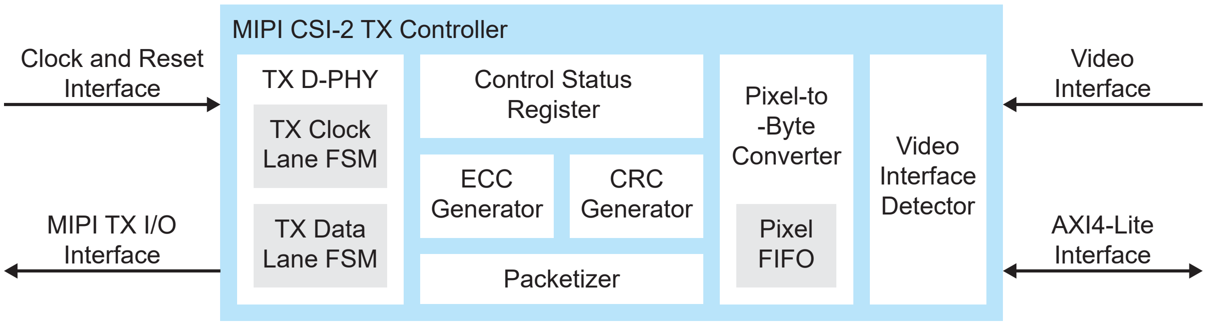
Note: You must be logged in to the Support Portal to view the IP Core Release Notes.

(2) Using default parameter settings.
(3) 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.

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 = lane 0 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
hsync_vcx	Input	Active-high horizontal sync for virtual channel. x = virtual lane 0 to 15
vsync_vcx	Input	Active-high vertical sync for virtual channel. x = virtual lane 0 to 15
datatype [5:0]	Input	Data type of the long packet. Sampled at Hsync rising edge.
pixel_data [63:0]	Input	Video Data. Sampled when pixel_data_valid is high. The actual width is dependent on pixel type. Refer to the pixel encoding table.
pixel_data_valid	Input	Active-high pixel data enable.
haddr [15:0]	Input	16 bit horizontal number of pixels. Sampled at Hsync rising edge.
line_num[15:0]	Input	Line number to use. Sampled at Hsync rising edge.
frame_num[15:0]	Input	Frame number to use. Sampled at Hsync rising edge.

Table 6: AXI4-Lite Interface

Interface to access **Table 7: Control Status Registers** on page 7.
All signals are clocked with axi_clk and axi_reset_n.

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

Pixel Clock Calculation

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

PIX_CLK_MHZ <= (DATARATE_MBPS * NUM_DATA_LANE) / 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 9

Control Status Registers

Table 7: Control Status Registers

Word Address Offset	Name	R/W	Width (bits)
0x00	Interrupt Status Register	R/W1C[4]	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	Reserved	–	–
0x14	Reserved	–	–
0x18	D-PHY ULPS control signal	R/W	9

[4] Read register. Write 1 to clear the register.

Table 8: 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 tInit timing parameter is met. TX controller is ready to send MIPI transaction.

Table 9: 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 10: 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 11: 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 12: 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
0x23	RGB666	3	18	54
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 13: 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 14: 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 15: 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 16: 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 17: 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 18: 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 19: RAW16 (4 Pixels per Clock)

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

Table 20: RAW20 (2 Pixels per Clock)

63 40	39 20	19 0
0	Pixel 2	Pixel 1

Table 21: RAW24 (2 Pixels per Clock)

63 48	47 24	23 0
0	Pixel 2	Pixel 1

Table 22: RAW28 (2 Pixels per Clock)

63 56	55 28	27 0
0	Pixel 2	Pixel 1

Table 23: RGB444 (4 Pixels per Clock)

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

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

Table 24: 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] Red	[54:50] Green	[49:45] Blue	[44:40] Red	[39:35] Green	[34:30] Blue	[29:25] Red	[24:20] Green	[19:15] Blue	[14:10] Red	[9:5] Green	[4:0] Blue

Table 25: RGB565 (4 Pixels per Clock)

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

Table 26: RGB888 (2 Pixels per Clock)

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

Table 27: 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 Y8	Pixel 7 Y7	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1
Even Lines							
Pixel 4	Pixel 3			Pixel 2	Pixel 1		
Y4	V3	Y3	U3	Y2	V1	Y1	U1

Table 28: 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 29: 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 30: 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 31: 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)

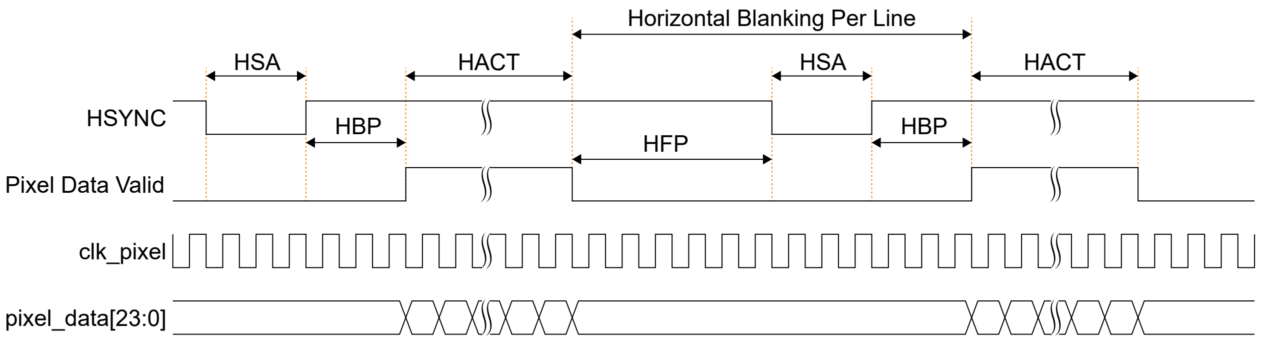


Figure 3: Video Timing Waveform (Vertical)

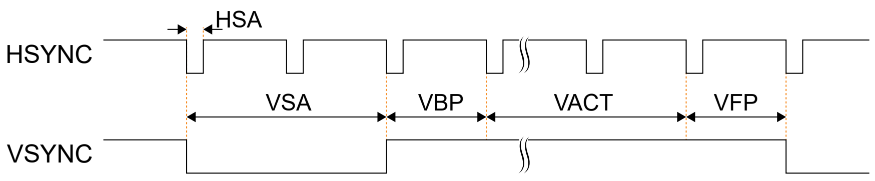


Table 32: Video Timing Parameter Definitions

MIPI Video Timing Parameters	Definition
HACT	Total number of pixel per line
VACT	Total number of line per frame
HSA	HSYNC pulse width
HBP	Horizontal back porch
HFP	Horizontal front porch
VSA	VSYNC pulse width
VBP	Vertical back porch
VFP	Vertical front porch
Pixel Clock	Video stream pixel clock frequency in MHz
MIPI Speed	CSI-2 TX MIPI speed in Mbps
No. data lane	Number of MIPI data lane

Minimum Horizontal Blanking Per Line

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 12: Pixel Encoding** on page 9). This compression introduces some latencies for the internal logic to decompress the pixel data, convert them into byte clock domain, and transmit out as byte data into MIPI I/O. As a result, minimum horizontal blanking per line needs to be fulfilled in order to prevent any data corruption during pixel data transmission.

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

HFP+HBP+HSA>=((Total Byte Clock per PPI * 4 / Number of Lanes) + 80) * Pixel Clock / Byte Clock
Where,
Total Byte Clock per PPI = (Horizontal Resolution * Pixel Data Bits per Pixel Clock) / (Pixel per Clock * Data Width)
Byte Clock = Data Rate / Data Width

Example: Horizontal Blanking Per Line Calculation

Given values:

- Data format = RAW10
- Pixel per Clock = 4 (See **Table 12: Pixel Encoding** on page 9)
- Pixel Data Bits per Pixel Clock = 40 (See **Table 12: Pixel Encoding** on page 9)
- Pixel Clock = 50 MHz (See **Pixel Clock Calculation** on page 7)
- Horizontal Resolution = 4000 pixel
- Data Width = 8
- Data Rate = 1500 Mbps
- Number of Lanes = 4

Total Byte Clock per PPI = (4000 * 40) / (4 * 8) = 5000

Byte Clock = 1500 / 8 = 187.5

Minimum horizontal blanking per line >= ((5000 * 4 / 4) + 80) * 50 / 187.5 = 1355 pixel clocks

You can freely distribute the 1355 clocks between HFP, HBP, or HSA parameters.

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate 易灵思® 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 易灵思 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 易灵思 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 易灵思® 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>_tmpl.v**—Verilog HDL instantiation template.
- **<module name>_tmpl.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 33: MIPI CSI-2 TX Controller Core Parameter

Name	Option	Description
tLPX (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 50
tINIT (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100000
Data Lanes	1, 2, 4, 8	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 1500 Mbps. Default: 100
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 Default: Continuous
Pixel Data FIFO Depth Size	256 - 8192	FIFO depth size to store the pixel packet data. Set to the power of 2 value that is bigger than the 2 * (max horizontal pixel / 8). For example, when maximum horizontal pixel is 1280, set this parameter to 512. 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
tLP_EXIT (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100
tCLK_ZERO (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 400
tCLK_TRAIL (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 80
tCLK_PRE (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns.(value before adding UI52) Default: 10
tCLK_POST (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns.(value before adding UI52) Default: 455
tCLK_PREPARE (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 50
tWAKEUP (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 1000
tHS_ZERO (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 262
tHS_TRAIL (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns.(value before adding 4UI) Actual THS TRAIL = tHS_TRAIL_NS + 4UI or 8UI (whichever bigger) Default: 1220

Name	Option	Description
tHS_EXIT (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100
tHS_PREPARE_NS	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 50
Pack Type 40	Enable, Disable	Enables the controller to pack RAW10, RAW20, YUV_420_10, and YUV_422_10 data type.(5) Default: Enable
Pack Type 48	Enable, Disable	Enables the controller to pack RAW6, RAW12, RAW24, RGB888, and YUV_420_8_legacy data type.(5) Default: Enable
Pack Type 56	Enable, Disable	Enables the controller to pack RAW7, RAW14, and RAW28.(5) 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.(5) Default: Enable
Enable Extra Bits on Virtual Channel	Enable, Disable	Enables 16 virtual channel support. Default: Disable

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

MIPI CSI-2 TX Controller Example Design

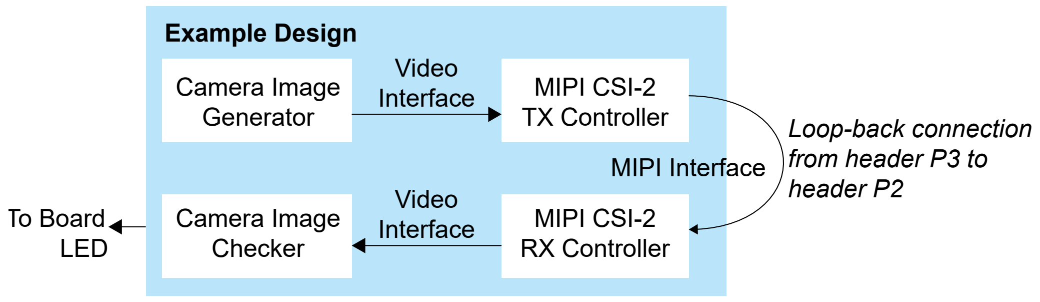
You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

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

The example design targets the 钛金系列 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 4: MIPI CSI-2 TX Controller Core Example Design



After power-up, the LED D19 blinks continuously and LED D18 turns on if the received image matches the generated image.

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_clk` in the Interface Designer.

Note: You can use the **钛金系列 MIPI Utility-v<version>.xlsm** 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 钛金系列 MIPI utility from the Design Support page in the Support Center.

Table 34: Example Design Implementation

FPGA	Logic and Adders	Flip-flops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)[6]				Efinity® Version[7]
					clk1	clk2	clk3	clk4	
Ti60 F225 C4	5,362	2,912	145	0	311	308	198	279	2021.2

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

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.

易灵思 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 35: Testbench Files

Directory/File	Note
../Testbench/modelsim.do	Modelsim testbench script.
../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.

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
```

[6] Using default parameter settings.
[7] Using Verilog HDL.

Revision History

Table 36: Revision History

Date	Version	Description
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)
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 钛金系列 Ti60 F225 Development Board and updated Resource Utilization and Performance, and Example Design Implementation tables. (DOC-553)
June 2021	1.0	Initial release.