MIPI DSI Transmitter Subsystem v2.2

Product Guide

Vivado Design Suite

PG238 April 26, 2022

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

AMD
XILINX

AMD
XILINX

Table of Contents

Appendix B: Debugging

Appendix C: Application Software Development

Appendix D: Additional Resources and Legal Notices

Introduction

The Mobile Industry Processor Interface (MIPI) Display Serial Interface (DSI) Transmitter Subsystem implements a DSI transmit interface in adherence to the MIPI DSI standard v1.3 (Type 4 architecture) [Ref 1]. The subsystem receives a pixel stream from an AXI4-Stream interface and inserts the required markers (such as hsync start, hsync end) in accordance to the DSI protocol and user programmed options. The packet framed is sent over an MIPI DPHY (Compliant to MIPI Alliance Standard for D-PHY Specification, version 1.2.) transmitter based on the number of lanes selected. The subsystem allows fast selection of the top-level parameters and automates most of the lower level parameterization. The AXI4-Stream interface allows a seamless interface to other AXI4-Stream-based subsystems.

Features

- 1-4 Lane Support
- Line rates ranging from:
  - 260 to 2500 Mb/s for Versal® ACAPs
  - 80 to 2500 Mb/s, refer to the data sheet of your device
- Supports different data types with fixed Virtual Channel Identifier (VC) of 0. For list of data types refer to "DSI Data type" GUI parameter
- Programmable EoTp generation support
- ECC generation for packet header
- CRC generation for data bytes (optional)
- Optional DCS Command mode support to send command packets long/short
- Pixel-to-byte conversion based on data format
- AXI4-Lite interface to access core registers
- Compliant with *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 3] for input video stream
- Interrupt generation to indicate subsystem status information

IP Facts Table	
Subsystem Specifics	
Supported Device Family(1)	Versal® ACAP, UltraScale+™, Zynq® UltraScale+ MPSoC, Zynq®-7000 SoC, 7 series FPGAs
Supported User Interfaces	AXI4-Lite, AXI4-Stream
Resources	Performance and Resource Utilization web page
Provided with Subsystem	
Design Files	Encrypted RTL
Example Design	Not Provided(4)
Test Bench	Not Provided
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver(2)	Standalone and Linux
Tested Design Flows(3)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 66769
All Vivado IP Changes Logs	Master Vivado IP Changes Logs: 72775
Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the Vitis directory (<install_directory>/Vitis/<release>/data/embeddedsw/doc/xilinx_drivers.htm).
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
4. *MIPI CSI-2 Receiver Subsystem Product Guide (PG232)* [Ref 5] uses this IP in example design.

Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **Hardware, IP, and Platform Development**: Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado™ timing, resource and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - Register Space
  - Clocking
  - Resets
  - Customizing and Generating the Subsystem

Core Overview

MIPI DSI TX subsystem allows you to quickly create systems based on the MIPI protocol. It interfaces between the Video Processing Subsystem and MIPI-based displays. An internal high-speed physical layer design, D-PHY, is provided to allow direct connection to display peripherals. The top-level customization parameters select the required hardware blocks needed to build the subsystem. Figure 1-1 shows the subsystem architecture.

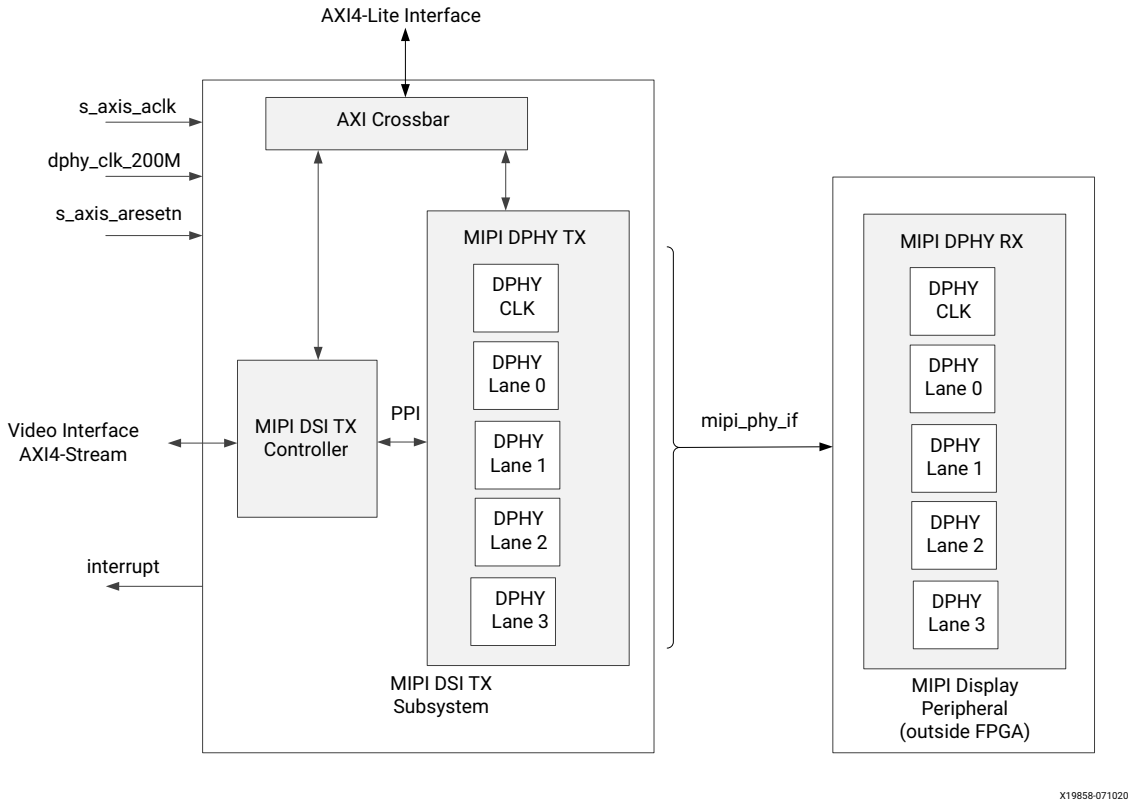

Figure 1-1: **Subsystem Architecture**

The subsystem consists of the following sub-blocks:

- MIPI D-PHY
- MIPI DSI TX Controller

Sub-core Details

MIPI-DPHY

The MIPI D-PHY IP core implements a D-PHY TX interface and provides PHY protocol layer support compatible with the DSI TX interface.The MIPI D-PHY IP core supports initial skew calibration for line rates >1500 Mb/s. See the *MIPI D-PHY LogiCORE IP Product Guide* (PG202) [Ref 4] for more information.

MIPI DSI TX Controller

The MIPI DSI TX Controller core consists of multiple layers defined in the MIPI DSI TX 1.3 specification, such as the lane management layer, low level protocol, and pixel-to-byte conversion.

The DSI TX Controller core receives stream of image data through an input stream interface. Based on the targeted display peripheral supported resolution and timing requirements, the controller must be programmed with required timing values. The controller then generates packets fulfilling the required video timing markers based on different video transmit mode sequences. In addition, the core supports sending short command packets during BLLP periods of video frames and also supports to send long or short command packets when configured in command mode. Sub-block details of MIPI DSI TX Controller are shown in Figure 1-2.

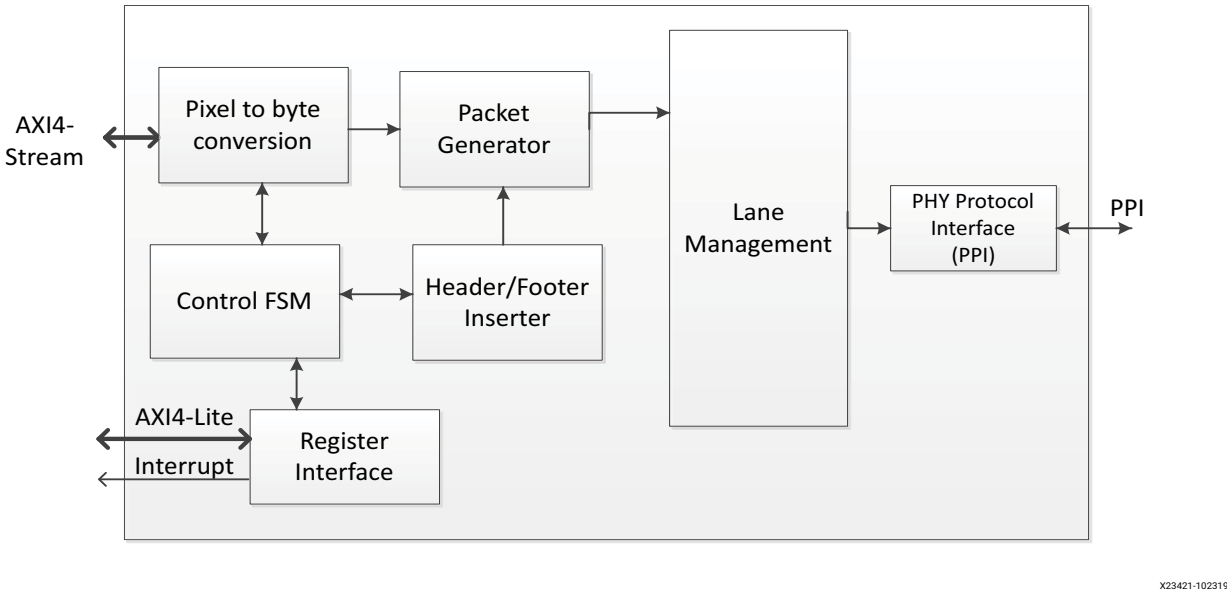

Figure 1-2: **Sub-blocks**

The features of this core include:

- 1 to 4 Lane support with a data rate of 2500 Mb/s per lane, for UltraScale+ only: Allows more bandwidth than that provided by one lane. If you are trying to avoid high clock rates, the subsystem can expand the data path to multiple lanes and obtain approximately linear increases in peak bus bandwidth.
- Generates PPI transfers towards DPHY with continuous clock.
- ECC and CRC calculation based on algorithm specified in DSI Specification: The correct interpretation of the data identifier and word count values is vital to the packet structure. ECC is calculated over packet header.

  To detect possible errors in transmission, a checksum is calculated over each data packet. The checksum is realized as 16-bit CRC. The generator polynomial is x16+x12+x5+x0.

  The CRC is computed only for the pixel bytes. The CRC fields for all other long packets are filled with 0x0000.

- Command Queue, data queue and command generation logic for non-video packets: To send non-video packets to display peripheral, a command queue is implemented to store the required command packets to be sent (Ex: Color mode on-off, Shutdown peripheral command, etc). When in video mode, the controller finds enough time-slot available during the video blanking periods, these short commands are sent over DSI link. When in command mode, the controller sends only the commands long/short programmed through register. Video data is not sent in this mode.
- All three video modes supported (Non-burst with sync pulses, Non-burst with sync events, Burst mode)
- Pixel to byte Conversion: The input video stream is expected to be compliant with *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 3] recommendations. Based on data type the incoming pixel stream is converted to byte stream to match with the DSI requirements detailed in sec 8.8 of the MIPI Alliance Standard for DSI specification [Ref 1].

  RGB component ordering, packed, unpacked mechanisms differ between *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 3] and DSI Specification. Refer to *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 3] and DSI specifications for better understanding on component ordering, packed, unpacked styles, etc.

  Figure 1-3 through Figure 1-14 illustrate the incoming pixel stream ordering on an AXI4-Stream video interface for different data types and pixels per clock combinations.

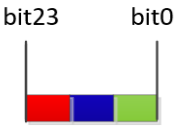

Figure 1-3: **Single Pixel RGB888**

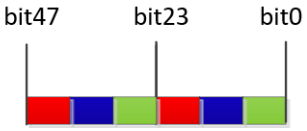

Figure 1-4: **Dual Pixels RGB888**

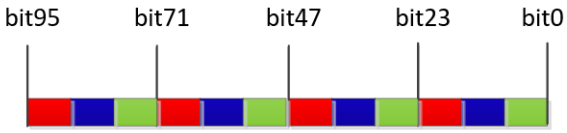

Figure 1-5: **Quad Pixels RGB888**

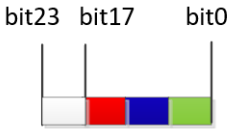

Figure 1-6: **Single Pixel RGB666 (Loosely, Packed)**

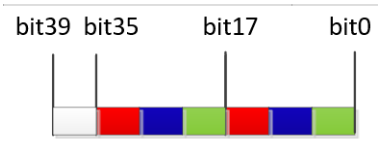

Figure 1-7: **Dual Pixels RGB666 (Loosely, Packed)**

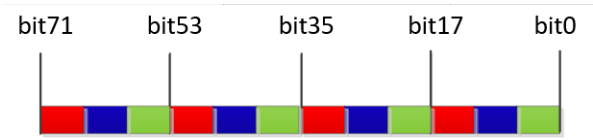

Figure 1-8: **Quad Pixels RGB666 (Loosely, Packed)**

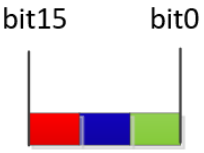

Figure 1-9: **Single Pixel RGB565**

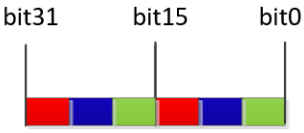

Figure 1-10: **Dual Pixels RGB565**

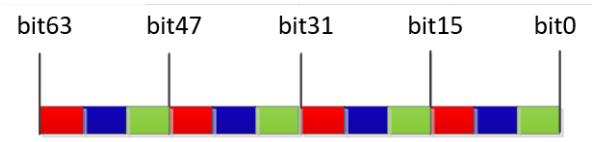

Figure 1-11: **Quad Pixels RGB565**

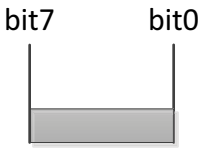

Figure 1-12: **Single Pixel Compressed**

Figure 1-13: **Dual Pixel Compressed**

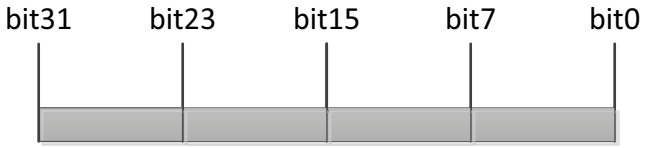

Figure 1-14: **Quad Pixel Compressed**

- Register programmable EoTp generation support.
- Interrupt generation to indicate detection of under run condition during pixel transfer and unsupported data types detected in command queue.
- One horizontal scanline of active pixels are transferred as one single DSI packet.
- All mandatory uncompressed pixel formats 16 bpp (RGB565), 18 bpp (RGB666 packed), 18 bpp (RGB666 loosely packed), 24 bpp (RGB888) are supported.
- Core accepts compressed data type from GUI selection, where the user is expected to pump in the compressed data. Core passes those stream of data without any conversion.

Applications

The MIPI DSI specification defines a high-speed serial interface between a host processor and a peripheral, typically a small form factor display such as LCD. The interface uses MIPI D-PHY physical layer. It was defined to enable displays for mobile platforms such mobile phones but the economics of scale driven by the success of mobile platforms is finding DSI display in other applications with small form factor displays such as tablets, portable monitors.

Unsupported Features

- DCS Commands which need bi-directional MIPI I/O support are not supported.
- Optional features of spec (Sub-links) are not supported.
- Bus Turn Around (BTA) not supported.
- Non-continuous clock mode not supported.
- Dynamic linerate is not supported.
- Periodic deskew is not supported.
- LP mode data transmission (Escape mode) is not supported.

Licensing and Ordering

This Xilinx module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado® Design Suite. The core does not need any additional license.

For more information, visit the MIPI DSI TX Subsystem product web page.

Information about other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

- MIPI Alliance Standard for Display Serial Interface DSI v1.3 [Ref 1]
- Output Pixel Interface: see the *AXI4-Stream Video IP and System Design Guide* (UG934) [Ref 3]
- MIPI Alliance Standard for Physical Layer D-PHY [Ref 2]

Resource Utilization

For full details about performance and resource utilization, visit the Performance and Resource Utilization web page.

Note: The MIPI DSI TX subsystem requires at least 7.5 block RAMs (BRAMs) to be available in the device. For example, device xc7s6csga225-1Q of Spartan-7 family has only 5 BRAMs. The subsystem does not work in such devices.

Port Descriptions

The MIPI DSI TX Subsystem I/O signals are described in Table 2-1.

Table 2-1: **Port Descriptions**

Signal Name	Direction	Description
UltraScale+ Shared Logic Outside Subsystem		
mipi_phy_if	Output	DPHY interface.
txbyteclkhs_in	Input	High-speed transmit byte clock.
clkoutphy_in	Input	DPHY serial clock.
pll_lock_in	Input	PLL lock indication.
txclkesc_in	Input	Clock for escape mode operations.

Table 2-1: **Port Descriptions *(Cont'd)***

Signal Name	Direction	Description
system_rst_in	Input	An active-high system reset output to be used by the example design level logic.
UltraScale+ Shared Logic in the Subsystem		
mipi_phy_if	Output	DPHY interface.
txbyteclkhs	Output	High-speed transmit byte clock.
clkoutphy_out	Output	DPHY serial clock.
pll_lock_out	Output	PLL lock indication.
txclkesc_out	Output	Clock for escape mode operations.
system_rst_out	Output	An active-high system reset output to be used by the example design level logic.
7 Series Shared Logic Outside Subsystem		
mipi_phy_if	Output	DPHY interface.
txbyteclkhs_in	Input	Input to DPHY and used to transmit high-speed data.
oserdes_clk_in	Input	Used to connect the CLK pin of TX clock lane OSERDES.
oserdes_clk90_in	Input	Used to connect the CLK pin of TX data lane OSERDES and should have 90 phase shift with oserdes_clk_in.
oserdes_clkdiv_in	Input	Used to connect the CLKDIV pin of TX clock lane OSERDES and should be generated from oserdes_clk_in as source.
txclkesc_in	Input	Clock used for escape mode operations.
system_rst_in	Input	System level reset.
7 Series Shared Logic in Subsystem		
mipi_phy_if	Output	DPHY interface.
txbyteclkhs	Output	Clock used to transmit high speed data.
oserdes_clk_out	Output	Used to connect the CLK pin of TX clock lane OSERDES.
oserdes_clk90_out	Output	Used to connect the CLK pin of TX data lane OSERDES. It has 90 phase shift relationship with oserdes_clk_out.
oserdes_clkdiv_out	Output	Used to connect the CLKDIV pin of TX clock lane OSERDES and generated from oserdes_clk_out as source.
mmcm_lock_out	Output	MMCM lock indication.
txclkesc_out	Output	Clock for escape mode operations.
Other Ports		
system_rst_out	Output	An active-High system reset output to be used by the example design level logic.
dphy_clk_200M	Input	Fixed 200 MHz clock required for MIPI DPHY. The same clock is used by s_axi interface of the subsystem.
s_axis_aclk	Input	AXI4-Stream Video clock

Table 2-1: **Port Descriptions *(Cont'd)***

Signal Name	Direction	Description
s_axis_aresetn	Input	AXI reset. Active-Low (Same reset for lite & stream interface).
s_axi_*	-	AXI4-Lite Interface
s_axis_tready	Output	AXI4-Stream Interface
s_axis_tvalid	Input	AXI4-Stream Interface
s_axis_tlast	Input	AXI4-Stream Interface
s_axis_tdata	Input	AXI4-Stream Interface. Width of this port is dependent on pixel type and no.of pixels per beat
s_axis_tkeep	Input	AXI4-Stream Interface.
s_axis_tuser	Input	AXI4-Stream Interface. TUSER[0] is used to map the Fsync signal of the AXI4-Stream Video Interface. The core does not use this signal, but generates FSYNC packets based on timing registers programming.
System Interface		
Interrupt	Output	System Interrupt output

Register Space

This section details registers available in the MIPI DSI TX Subsystem. The address map is split into following regions:

- MIPI DSI TX Controller core
- MIPI D-PHY core

Each IP core is given an address space of 64K. Example offset addresses from the system base address when the MIPI D-PHY registers are enabled are shown in Table 2-2. Registers are provided for informational and debug purposes. For more details, refer provided driver API for the supported control of the MIPI DSI TX Subsystem.

Table 2-2: **Sub-Core Address Offsets**

IP Cores	Offset
MIPI DSI TX Controller	0x0_0000
MIPI D-PHY	0x1_0000

MIPI DSI TX Controller Core Registers

Table 2-3 specifies the name, address, and description of each firmware addressable register within the MIPI DSI TX controller core.

Table 2-3: **MIPI DSI TX Controller Core Registers**

Address Offset	Register name	Description
0x00	Core Configuration Register	Core configuration options
0x04	Protocol Configuration Register	Protocol configuration options
0x08	Reserved	
0x0C	Reserved	
0x10	Reserved	
0x14	Reserved	
0x18	Reserved	
0x1C	Reserved	
0x20	Global Interrupt Enable Register	Global interrupt enable registers
0x24	Interrupt Status Register	Interrupt status register
0x28	Interrupt Enable Register	Interrupt enable register
0x2C	Status Register	Status register
0x30	Command Queue Packet	Packet Entry to command Queue.
0x34	Data FIFO Register	Data FIFO register
0x38	Reserved	
0x3C	Reserved	
0x40	Reserved	
0x44	Reserved	
0x48	Reserved	
0x4C	Reserved	
0x50	Timing Register-1	Video timing(5)
0x54	Timing Register-2	Video timing(5)
0x58	Timing Register-3	Video timing(5)
0x5C	Timing Register-4	Video timing(5)
0x60	Line Time	Total Line time
0x64	BLLP Time	Blanking packet payload size in bytes (WC) available during VSA,VBP,VFP lines
0x68	D-PHY LP HS offset	D-PHY LP HS offset register
0x6C	Reserved	
0x70	Reserved	

Table 2-3: **MIPI DSI TX Controller Core Registers *(Cont'd)***

Address Offset	Register name	Description
0x74	Reserved	
0x78	Reserved	
0x7C	Reserved	

Notes:

1. Access type and reset value for all the reserved bits in the registers is read only with value 0.
2. All register access should be word aligned and no support for write strobe. WSTRB is not used internally.
3. Only the lower 7-bits (for example, 6:0) of read and write address of AXI are decoded, which means accessing address 0x00 and 0x80 results in reading the same address of 0x00.
4. Read and write access to address outside of above range does not return any error response.
5. All video timing registers need to appropriately programmed for the successful transfer of video data. For more details, refer to the provided driver API for programming details.

Core Configuration Register

Allows you configure core for enabling/disabling the core and soft during operation.

Table 2-4: **Core Configuration Register (0x00)**

Bits	Name	Reset Value	Access	Description
31:6	Reserved	NA	NA	Reserved
5	Command FIFO Reset	0x0	WO	Write 1 to reset the Command FIFO. It is a self clearing register. Reading this bit will only return 0.
4	Data FIFO Reset	0x0	WO	Write 1 to reset the Data FIFO. It is a self clearing register. Reading this bit will only return 0.
3	Command Mode	0x0	R/W	0: Video mode 1: Command mode
2	Controller ready	0x0	R	Controller is ready for processing. During core disable, you can rely on this status that the core stopped all its activity. Controller ready also depends on init_done from DPHY. Only when init_done is 1, Controller ready can be 1. 0: Controller is not ready 1: Controller is ready. Enable the Controller Core (Bit 0 of 0x00) when controller is ready.

Table 2-4: **Core Configuration Register (0x00) *(Cont'd)***

Bits	Name	Reset Value	Access	Description
1	Soft Reset	0x0	R/W	Soft reset to core. Writing 1 to this bit resets the ISR bits ONLY. Writing 0 takes the core out of soft reset. Once soft reset is released, core starts capturing new status information to ISR.
0	Core Enable	0x0	R/W	Enable/Disable the core 0: Stop generating packets 1: Start generating packets Controller ends the current transfer by resetting all internal FIFOs and registers. Once enabled, controller start from VSS packet.(that is new video frame)

Protocol Configuration Register

Allows you to configure protocol specific options such as the number of lanes to be used.

Table 2-5: **Protocol Configuration Register (0x04)**

Bits	Name	Reset Value	Access	Description
31:14	Reserved	NA	NA	Reserved
13	EoTp	0x1	R/W	0: Disable EoTp Generation. 1: Enable EoTp Generation.
12:7	Pixel Format	0x3E	R	Data type for pixel format. Data type of pixel format selected through the GUI. 0x0E – Packed RGB565 0x1E- packed RGB666 0x2E – Loosely packed RGB666 0x3E- Packed RGB888 0x0B- Compressed Pixel Stream ***Note:*** The default data type selected in the GUI is RGB888. Hence, the default value of the register is mentioned as 0x3E. It varies with the datatype selected in the GUI.
6	BLLP Mode	0x0	R/W	BLLP selection 0: Send blanking packets during BLLP periods 1: Use LP mode for BLLP periods
5	Blanking packet type	0x0	R/W	Blanking packet type for BLLP region 0: Blanking packet(0x19) 1: Null packet(0x09)

Table 2-5: **Protocol Configuration Register (0x04) *(Cont'd)***

Bits	Name	Reset Value	Access	Description
4:3	Video Mode	0x0	R/W	Video mode transmission sequence 0x0- Non-burst mode with sync pulses 0x1 – Non-burst mode with Sync Events 0x2- Burst mode
2	Reserved	NA		Reserved for future lane extension support.
1:0	Active Lanes	Configured Lanes during core generation	R	Configured lanes in the core 0x0 - 1 Lane 0x1 - 2 Lanes 0x2 - 3 Lanes 0x3 - 4 Lanes

Global Interrupt Enable Register

Table 2-6: **Global Interrupt Enable register (0x20)**

Bits	Name	Reset Value	Access	Description
31:1	Reserved	NA	NA	Reserved
0	Global Interrupt enable	0x0	R/W	Master enable for the device interrupt output to the system 1: Enabled: Corresponding IER bits are used to generate interrupt. 0: Disabled: Interrupt generation blocked irrespective of IER bits.

Interrupt Status Register

Captures different error/status information of the core.

Table 2-7: **Interrupt Status Register (0x24)**

Bits	Name	Reset Value	Access	Description
31:3	Reserved	NA	NA	Reserved
2	Command Queue FIFO Full	0x0	R/W1C[(1)]	Asserted when command queue FIFO full condition detected.
1	Unsupported/ Reserved Data type	0x0	R/W1C[(1)]	Asserted when unsupported/reserved data types seen in command queue.

Table 2-7: **Interrupt Status Register (0x24) *(Cont'd)***

Bits	Name	Reset Value	Access	Description
0	Pixel Data underrun	0x0	R/W1C[(1)]	Byte stream FIFO starves for Pixel during HACT transmission.[(2)]

Notes:
1. W1C – Write 1 to Clear (to clear register bit, you have to write 1 to corresponding bits).
2. Pixel Data underrun is not expected during normal core operation, which implies that the rate of incoming data is insufficient to keep up with the outgoing data rate.

Interrupt Enable Register

This register allows you to selectively enable each error/status bits in Interrupt Status register to generate a interrupt at output port. An IER bit set to '0' does not inhibit an interrupt condition for being captured, but reported in the status register.

Table 2-8: **Interrupt Enable Register (0x28)**

Bits	Name	Reset Value	Access	Description
31	Reserved	NA	NA	Reserved
2	Command Queue FIFO Full	0x0	R/W	Generate interrupt on command queue FIFO full condition.
1	Unsupported/ Reserved Data type	0x0	R/W	Generate interrupt on "Unsupported/ Reserved" data type detection.
0	Pixel data underrun	0x0	R/W	Generate interrupt on "Pixel data underrun" condition

Status Register

This register captures different status conditions of the core.

Table 2-9: **Status Register (0x2C)**

Bits	Name	Reset Value	Access	Description
31:13	Reserved	NA	NA	Reserved
12	Current command type under processing	0	R	0 - Short command when bit 11 is 1 1- long command when bit 11 is 1
11	Command execution in progress	0	R	Status of command execution in command mode. When 0: No commands are being processed. Can shift to Video mode.

Table 2-9: **Status Register (0x2C) *(Cont'd)***

Bits	Name	Reset Value	Access	Description
10	Waiting for data	0	R	Waiting for data to process long command. There is no timeout for this state in the core and it keeps waiting forever and until the data is received or data FIFO is reset.
9	Data FIFO Empty	0	R	Data FIFO Empty condition. ***Note:*** Reset Value is 1 when Command mode is enabled.
8	Data FIFO Full	0	R	Data FIFO Full condition
7	Ready for packet	0	R	1: The core is ready to accept a long command. If this bit is 0, any long command written to the command register is ignored
6	Ready for short packet	0	R	1: The core is ready to accept a short command. If this bit is 0, any command written to the command register is ignored
5:0	Command Queue Vacancy	0x20	R	This number gives an estimate of number of command queue entries that can safely be written to the Queue before it gets Full. Command FIFO Depth is 31. Command FIFO will be Full when this count is 1. Number of further commands you can write = Vacancy count - 1

Command Queue Packet

Table 2-10: **Command Queue Packet (0x30)**

Bits	Name	Reset Value	Access	Description
31:24	Reserved	NA	NA	Reserved
23:16	Data-1	0x0	R/W	Data byte 1 of short packet(1). Ignored in case of long packets.
15:8	Data 0/ WC	0x0	R/W	Data byte 0 of short packet(1). WC in case of Long packet data type. Supports a maximum WC of 251.
7:6	VC	0x0	R/W	VC value of command packet.

Table 2-10: **Command Queue Packet (0x30) *(Cont'd)***

Bits	Name	Reset Value	Access	Description
5:0	Data type	0x0	R/W	Long/short packet data type.

Notes:
1. The controller passes the payload content as-is and no checks are performed over the payload content. For example, the second byte of Generic Short WRITE with 1 parameter must be 0x00.

Table 2-11 describes the short packet data types that are supported. Command queue writes with any other data type value is ignored and indicated as Unsupported Data type in ISR.

Table 2-11: **Command Queue Packet Data Types**

Data Type	Description
0x07	Compression Mode Command
0x02	Color Mode (CM) Off Command
0x12	Color Mode (CM) On Command
0x22	Shut Down Peripheral Command
0x32	Turn On Peripheral Command
0x03	Generic Short WRITE, no parameters
0x13	Generic Short WRITE, 1 parameter
0x23	Generic Short WRITE, 2 parameters
0x05	DCS Short WRITE, no parameters
0x15	DCS Short WRITE, 1 parameter
0x16	Execute Queue (1)
0x37	Set Maximum Return Packet Size

Notes:
1. After the execute command is detected by the core, no further command queue packets are sent until a VSS and a frame end are seen as described in the DSI specification (Sec 8.7.2).

Data FIFO Register

Table 2-12: **Data FIFO Register (0x34)**

Bits	Name	Reset Value	Access	Description
31:0	Payload data	0x0	WO	DCS Long command data is written through this register

Timing Register-1

During burst mode of operation, due to time compression of video data, there is BLLP duration available during active region of horizontal lines. This value of "BLLP Burst mode" must be programmed when operating in burst mode.

IMPORTANT: *The controller must be programmed with required timing values for video data transfer. For sample examples, refer Example Timing Register Calculations.*

Table 2-13: **Timing Register-1 (0x50)**

Bits	Name	Reset Value	Access	Description
31:16	HSA	0x0	R/W	Horizontal Sync active width blanking packet payload size in bytes (WC)(1)
15:0	BLLP Burst Mode	0x0	R/W	BLLP duration of VACT region packet payload size in bytes (WC). Applicable only Burst mode

Notes:
1. The values of HSA, HBP, and HFP must be greater than 4 bytes. The core functionality is not guaranteed otherwise.

Timing Register-2

Table 2-14: **Timing Register-2 (0x54)**

Bits	Name	Reset Value	Access	Description
31:16	HACT	0x0	R/W	Active per video line payload size in bytes (WC)
15:0	VACT	0x0	R/W	Vertical active region lines

Timing Register-3

Table 2-15: **Timing Register-3 (0x58)**

Bits	Name	Reset Value	Access	Description
31:16	HBP	0x0	R/W	Horizontal back porch blanking packet payload size in bytes (WC)(1)
15:0	HFP	0x0	R/W	Horizontal front porch blanking packet payload size in bytes (WC)(1)

Notes:
1. The values of HSA, HBP, and HFP must be greater than 4 bytes. The core functionality is not guaranteed otherwise.

Timing Register-4

Table 2-16: **Timing Register-4 (0x5C)**

Bits	Name	Reset Value	Access	Description
31:24	Reserved	NA	NA	Reserved
23:16	VSA	0x0	R/W	Vertical sync active lines

Table 2-16: **Timing Register-4 (0x5C) *(Cont'd)***

Bits	Name	Reset Value	Access	Description
15:8	VBP	0x0	R/W	Vertical back porch lines
7:0	VFP	0x0	R/W	Vertical front porch lines

Line Time

Total line time is calculated by the core (in bytes) based on timing parameters programmed and non-burst/burst mode selection, shown in Table 2-17.

Table 2-17: **Video Timing (Line Time)**

Bits	Name	Reset Value	Access	Description
31:0	Line Time	0x0	R	Total line size in bytes (Value is valid only when all the required timing registers are programmed)

BLLP Time

Total BLLP time is calculated by the core (in bytes) based on timing parameters programmed and non-burst/burst mode selection. This durations refers to BLLP regions defined in VSA, VBP, VACT, VFP lines of video timing.

This BLLP duration is used byte the core to accommodate command queue packets, shown in Table 2-18.

Table 2-18: **Video Timing (BLLP Time)**

Bits	Name	Reset Value	Access	Description
31:0	BLLP Time	0x0	R	BLLP duration in bytes (Value is valid only when all the required timing registers are programmed)

D-PHY LP HS offset

The core does not consider the LP to HS delay in the video timing calculations by default. Hence, the user sees slightly less Frames Per Second (FPS) than intended. In situations where the user needs to consider the D-PHY LP to HS latency, the user is required to program the register given below.

The D-PHY LP to HS (more specifically the time from PPI signal txrequesths to txreadyhs) delay varies based on Line Rate (Mb/s), LPX Period (ns), internal design logic, and CDC stages.

User Inputs:

P1 = UI in ns

P2 = LPX Period in ns

P3 = INT(PPI Byteclk in ns) is calculation to converts the specified value into an integer number.

P4 = PPI Byteclk in ns (with decimal places)

Table 2-19: **Internal parameters L1**

Linerate	L1
<150	INT(P1)
150 to <300	INT(P1*2)
301 to <600	INT(P1*4)
601 and above	P3

L2 = P2 + 25

L3 = 85 + 6*P1

L4 = ((L3-3*L1)/5)-1

L5 = (L4*5 + L2)/P4

L6 = INT((105+6*P1)/P3) + 4

LP-HS delay = L5 + L6

Example LP-HS delay for certain line rates are given below:

Table 2-20: **Example LP-HS delay for certain line rates**

Line rate	P1	P2	P3	L1	P4	L2	L3	L4	L5	L6	LP-HS Delay
80	12.5	50	100	12	100	75	160	23	1	5	6
150	6.666667	50	53	13	53.33333333	75	125	16	2	6	8
550	1.818182	50	14	7	14.54545455	75	95.90909091	13	9	12	21
1000	1	50	8	8	8	75	91	12	16	17	33
1188	0.841751	50	6	6	6.734006734	75	90.05050505	13	20	22	42
1440	0.694444	50	5	5	5.555555556	75	89.16666667	13	25	25	50

Due to CDC stages and decimal numbers of byteclk period, the user may need to program a slightly higher or lower value obtained in the above table to get the closer FPS on the hardware.

The final value to program is “LP-HS Delay” * DSI Lanes.

Table 2-21: **Video Timing (D-PHY LP HS offset)**

Bits	Name	Reset Value	Access	Description
31:16	Reserved	NA	NA	Reserved
15:0	D-PHY LP HS offset	0x0	R/W	To offset D-PHY LP to HS transition delay in the video timing calculations

Designing with the Subsystem

This chapter includes guidelines and additional information to facilitate designing with the subsystem.

General Design Guidelines

The subsystem is designed to fit into a video pipe transmit path. The input to the subsystem must be connected to a AXI4-Stream source which generates the pixel data. The output of the subsystem is a MIPI complaint serial data. Based on the throughput requirement, the output PPI interface can be tuned using customization parameters available for the subsystem, for example, Number of lanes.

IMPORTANT: *Initialize all MIPI interfaces in the same HP IO Bank at the same time. For example, multiple DSI or D-PHY instances in a system. For more information on implementing multiple interfaces in the same HP IO Bank, see UltraScale Architecture SelectIO Resources (UG571) [Ref 8].*

Because the MIPI protocol does not allow throttling on the output interface, the module connected to the input of this subsystem should have sufficient bandwidth to pump the pixel data at the required rate to prevent underflow or overflow. For this purpose, it is always recommended to use Frame buffer / VDMA to drive data to MIPI DSI Tx Subsystem.

All the MIPI specific video timing is taken care by the MIPI DSI Tx Subsystem.

All horizontal timing parameters should be in terms of word count (WC). For the same resolution, these may vary based on the pixel type selected (for example, RGB888 versus RGB666). The WC value should adhere to the word count restriction defined by the DSI specification. For example, the RGB888 word count should be a multiple of three.

The values of the timing registers must arrive in terms of the DSI word count (WC) such that these DSI bytes would approximately take the same amount of time as the video event that took in the pixel clock domain.

All the vertical timing parameters should be in terms of the number of lines. MIPI DSI TX Subsystem do not define any video timing parameters on its own. It is the sync device (DSI Panel) timing parameters that are the starting point to arrive at the timing parameters of the source device (MIPI DSI TX Subsystem).

If used properly the driver API is created to control the programming of all required registers. Please see the driver API documentation or example design provided with PG232 for more details. Table 3-1 shows the applicable timing parameters based on Video mode:

Table 3-1: **Applicable Parameters Based on Video Mode**

Timing Register	Description	Sync Events	Sync Pulses	Burst Mode
Timing Register-1 (0x50)	HSA (WC)	No	Yes	No
Timing Register-1 (0x50)	BLLP (WC)	No	No	Yes
Timing Register-2 (0x54)	HACT (WC)	Yes	Yes	Yes
Timing Register-2 (0x54)	VACT (Lines)	Yes	Yes	Yes
Timing Register-3 (0x58)	HBP (WC)	Yes	Yes	Yes
Timing Register-3 (0x58)	HFP (WC)	Yes	Yes	Yes
Timing Register-4 (0x5C)	VSA (Lines)	Yes	Yes	Yes
Timing Register-4 (0x5C)	VBP (Lines)	Yes	Yes	Yes
Timing Register-4 (0x5C)	VFP (Lines)	Yes	Yes	Yes

Example Timing Register Calculations

The calculated values are used to set the timing registers.

Example Configuration 1

Consider an exemplary MIPI DSI Panel with the following specifications:

Table 3-2: **MIPI DSI Panel Parameters**

Parameter	Value
MIPI Parameters	
Line Rate	1000 Mb/s
Data Type	RGB888 3 bytes or 24 bits
Video Mode	Sync Events
Lanes	4
Timing Parameters	
Horizontal Active	1920 pixels
Horizontal Blanking	120 pixels
Vertical Active	1200 lines
Vertical Blanking	12 lines
Frame Rate	60 fps
Clock Frequency	148.5 MHz

To generate the values and set timing registers, based on the above example configuration, do the following:

1. Get HACT(WC) and VACT

```
HACT(WC)= Active pixels per line * Bits per pixel/8
= 1920*24/8
= 5760 (decimal) -> 0x1680
VACT = Active lines per frame
= 1200(decimal) -> 0x04B0
```

2. Get total line-time in pixel clock.

```
Pixel Frequency = 148.5Mhz
Pixel clock period = 1000/148.5 = 6.73ns
Total pixel in one line = 1920+120 = 2040 (Active Pixels + Blanking Pixels)
Total line time = 2040*6.73 = 13730ns
```

3. Calculate blanking time.

```
Byte clock (PPI) frequency = Line-rate/8 = 1000/8 = 125Mhz
Byte clock period = 1000/125 = 8ns
Active pixels per line (HACT) Duration (ns) = Pixels* (Bytes per pixel) * Byte clk
Period/ Lanes
= (1920 * 3 * 8) /4
= 11520 ns
Blanking time = Line-time - HACT Duration
= 13730 - 11520
= 2210 ns
Word Count (WC) in Bytes to meet "Blanking time" of 2210 ns
= Blanking time * Lanes / (Byte Period)
= 2210 * 4 / 8
= 1105
```

4. Get timing parameter based on Video mode.

```
Video mode: Sync Events
One line is composed of HSS + HBP + HACT + HFP
Horizontal Sync Start (HSS) -> Short packet ->4 bytes
HBP/HACT/HFP -> Long packet -> 4 bytes header + Payload + 2 bytes CRC
Total of 4 + 3*6 = 22 bytes are covered in header and footer.
Available blanking WC = 1105- 22 = 1083
```

5. Divide the total available WC across available blanking parameters HBP and HFP. Considering a ratio of 5:1 gives: (The values for HBP and HFP are the distribution of horizontal blanking words across these parameters. In the examples, Xilinx tries to split these based on a ratio. The MIPI specification does not define the ratio. However, some DSI displays may have specific requirements. Hence, please consult the data sheet for your display.)

```
Horizontal Back Porch (HBP) = 902
Horizontal Front Porch (HFP) = 1083 - 902
= 181
```

6. Set applicable horizontal timing registers with above calculated values.

```
HBP = 902(decimal) -> 0x386
HFP = 181(decimal) -> 0x0B5
```

7. Divide the total available vertical blanking lines between VSA, VBP, and VFP.

   Considering a ratio of 1:1:1 gives: (The values for VSA, VBP, and VFP are the distribution of the vertical blanking lines across these parameters. In the examples, Xilinx tries to split these based on a ratio. The MIPI specification does not define the ratio. However, some DSI displays may have specific requirements. Hence, please consult the data sheet for your display.)

```
VSA =  4 (decimal) -> 0x04
VBP =  4 (decimal) -> 0x04
VFP = 4 (decimal) -> 0x04
```

8. Final consolidate set of horizontal and vertical timing parameters for a DSI panel with 1920x1200 @60 fps, 4-Lane, 1000 Mb/s, RGB888, Video clock of 148.5 MHz are as shown below:

```
HACT = 0x1680
HBP = 0x386
HFP = 0x0B5

VACT= 0x04B0
VSA = 0x04
VBP = 0x04
VFP = 0x04
```

Example Configuration 2

Consider an exemplary MIPI DSI Panel with the following specifications:

Table 3-3: **MIPI DSI Panel Parameters**

Parameter	Value
MIPI Parameters	
Line Rate	500 Mb/s
Data Type	Compressed data 1-byte or 8 bits
Video Mode	Sync Pulses
Lanes	2
Timing Parameters	
Horizontal Active	640 pixels
Horizontal Blanking	100 pixels
Vertical Active	480 lines
Vertical Blanking	15 lines
Frame Rate	60 fps
Clock Frequency	50 MHz

To generate the values and set timing registers, based on the above example configuration, do the following:

1. HACT(WC) = Active pixels per line * Bits per pixel/8

```
= 640*8/8
= 640 (decimal) -> 0x280
    VACT = Active lines per frame
         = 480(decimal) -> 0x1E0
```

2. Get total line-time in pixel clock.

```
Pixel Frequency = 50Mhz
Pixel clock period = 1000/50 = 20ns
Total pixel in one line = 640+100 = 740
Total line time = 740*20 = 14800 ns
```

3. Calculate blanking time.

```
Byte clock(PPI) frequency = Line-rate/8 = 500/8  = 62.5Mhz
Byte clock period = 1000/62.5 = 16 ns
HACT Duration = Pixels* (Bytes per pixel) * Byte clk Period/ Lanes
= (640 * 1 * 16) /2
= 5120 ns
Blanking time = Line-time - HACT Duration
= 14800 - 5120
= 9680 ns
WC(Bytes) to meet "Blanking time" of 9680 ns
= Blanking time * Lanes / (Byte Period)
= 9680 * 2 / 16
= 1210
```

4. Get timing parameter based on Video mode

```
Video mode: Sync Pulses
One line is composed of HSS +HSA + HSE + HBP + HACT + HFP
HSS/HSE -> Short packet -> 4 bytes
HSA/HBP/HACT/HFP -> Long packet -> 4 bytes header + Payload + 2 bytes CRC
Total of 2*4 + 4*6 = 32 bytes are covered in header and footer
Available blanking WC = 1210- 32 = 1178
```

5. Divide the total available WC across available blanking parameters HSA, HBP and HFP. Considering a ratio of 2:2:2 gives: (The values for HBP and HFP are the distribution of the horizontal blanking words across these parameters. In the examples, Xilinx tries to split these based on a ratio. The MIPI specification does not define the ratio. However, some DSI displays may have specific requirements. Hence, please consult the data sheet for your display.)

```
Horizontal Sync Active (HSA) = 2*1178/6 = 393
Horizontal Back Porch (HBP) = 2*1178/6 = 393
Horizontal Front Porch (HFP) = 1178 - 393 - 393 =
= 393
```

6. Set applicable horizontal timing registers with above calculated values.

```
HSA = 393(decimal) ' 0x189
HBP = 393(decimal) ' 0x189
HFP = 392(decimal) ' 0x188
```

7. Divide the total available vertical blanking lines between VSA, VBP, and VFP. Considering a ratio of 1:1:1 gives: (The values for VSA, VBP, and VFP are the distribution of the vertical blanking lines across these parameters. In the examples, Xilinx tries to split these based on a ratio. The MIPI specification does not define the ratio. However, some DSI displays may have specific requirements. Hence, please consult the data sheet for your display.)

```
VSA =  5 (decimal) -> 0x05
VBP =  5 (decimal) -> 0x05
VFP = 5 (decimal) -> 0x05
```

8. Final consolidate set of horizontal and vertical timing parameters for a DSI panel with 640x480@60 fps, 2-Lane, 500 Mb/s, Compressed data of 8 bit width, Video clock of 50 MHz are as specified below:

```
HACT = 0x280
HSA = 0x189
HBP = 0x189
HFP = 0x189

VACT= 0x1E0
VSA = 0x05
VBP = 0x05
VFP = 0x05
```

Example Configuration 3

Consider an exemplary MIPI DSI Panel with the following specifications:

Table 3-4: **MIPI DSI Panel Parameters**

Parameter	Value
MIPI Parameters	
Line Rate	500 Mb/s
Data Type	Compressed 1-byte or 8 bits
Video Mode	Burst Mode
Lanes	2
Timing Parameters	
Horizontal Active	640 pixels (Compressed to 500 pixels)
Horizontal Blanking	100 pixels
BLLP during Active region of Horizontal lines	140 pixels
Vertical Active	480 lines
Vertical Blanking	15 lines

Table 3-4: **MIPI DSI Panel Parameters *(Cont'd)***

Parameter	Value
Frame Rate	60 fps
Clock Frequency	50 MHz

Note: For compressed data type, you are expected to pump in the compressed data. Core passes through those stream of data without any conversion. In the current example, it is expected that the 640 pixels of input stream is compressed to 500 pixels.

To generate the values and set timing registers based on the above example configuration, perform the following:

```
HACT(WC) = Active pixels per line * Bits per pixel/8
= 500*8/8
= 500 (decimal) -> 0x1F4
BLLP(WC) = BLLP Pixels per line * Bits per pixel/8
                  = 140*8/8
                 = 140(decimal) -> 0x8C
    VACT = Active lines per frame
             = 480(decimal) -> 0x1E0
```

1. Get total line-time in pixel clock.

```
Pixel Frequency = 50Mhz
Pixel clock period = 1000/50 = 20ns
Total pixel in one line = 500+140+100 = 740
Total line time = 740*20 = 14800 ns
```

2. Calculate blanking time.

```
Byte clock(PPI) frequency = Line-rate/8 = 500/8 = 62.5Mhz
Byte clock period = 1000/62.5 = 16 ns
HACT Duration = Pixels* (Bytes per pixel) * Byte clk Period/ Lanes
= (500 * 1 * 16) /2
= 4000 ns
BLLP Duration = Pixels* (Bytes per pixel) * Byte clk Period/ Lanes
= (140 * 1 * 16) /2
= 1120 ns
Blanking time(BLLP+Horizontal Blanking) = Line-time - HACT Duration - BLLP Duration
= 14800 - 4000 -1120
= 9680 ns
WC(Bytes) to meet "Blanking time" of 9680 ns
= Blanking time * Lanes / (Byte Period)
= 9680 * 2 / 16
= 1210
```

3. Get timing parameter based on Video mode

```
Video mode: Burst Mode
One line is composed of HSS + HBP + HACT + BLLP + HFP
Horizontal Sync Start (HSS) -> Short packet ->4 bytes
HBP/HACT/BLLP/HFP -> Long packet -> 4 bytes header + Payload + 2 bytes CRC
Total of 4 + 4*6 = 28 bytes are covered in header and footer.
Available blanking WC = 1210- 28 = 1182
```

4. Divide the total available WC across available blanking parameters HBP and HFP. Considering a ratio of 1:2 gives: (The values for HBP and HFP are the distribution of the

horizontal blanking words across these parameters. In the examples, Xilinx tries to split these based on a ratio. The MIPI specification does not define the ratio. However, some DSI displays may have specific requirements. Hence, please consult the data sheet for your display.)

```
Horizontal Back Porch (HBP) = 394
Horizontal Front Porch (HFP) = 1182 - 394
= 788
```

5. Set applicable horizontal timing registers with above calculated values.

```
HBP = 394(decimal) -> 0x18A
HFP = 788(decimal) -> 0x314
```

6. Divide the total available vertical blanking lines between VSA, VBP, and VFP. Considering a ratio of 1:1:1 gives: (The values for HBP and HFP are the distribution of the horizontal blanking words across these parameters. In the examples, Xilinx tries to split these based on a ratio. The MIPI specification does not define the ratio. However, some DSI displays may have specific requirements. Hence, please consult the data sheet for your display.)

```
VSA =  5 (decimal) -> 0x05
VBP =  5 (decimal) -> 0x05
VFP = 5 (decimal) -> 0x05
```

7. Final consolidate set of horizontal and vertical timing parameters for a DSI panel with 640x480@60 fps, 2-Lane, 500 Mb/s, Compressed data type, Video clock of 50 MHz are as specified below:

```
HACT = 0x1F4
BLLP(0x50[15:0] = 0x8C
HBP = 0x18A
HFP = 0x314

VACT= 0x1E0
VSA = 0x05
VBP = 0x05
VFP = 0x05
```

Implementing More Than 4-Lane DSI-TX Design

The existing MIPI DSI TX Subsystem allows a maximum of 4 lanes. Guidelines to achieve DSI designs with higher lane requirements (for example, an eight lane design) are listed below:

1. After the stream source, you need a splitter module which splits the incoming video stream into two streams; the left-half and the right-half image.
2. Each splitter output then feeds to one DSI 4 lanes instance.
3. The DSI 8-lane Receiver should be following reverse to combine the images.
4. You need to program each DSI-TX 4 lanes instance timing parameters based on half image rather than full image timing parameters.

5. In DSI-RX, one DSI instance reconstructs left half of the image and the other DSI instance reconstructs the right half of the image.

The 8-lane implementation using two 4-lane MIPI DSI TX instances is shown in Figure 3-1.

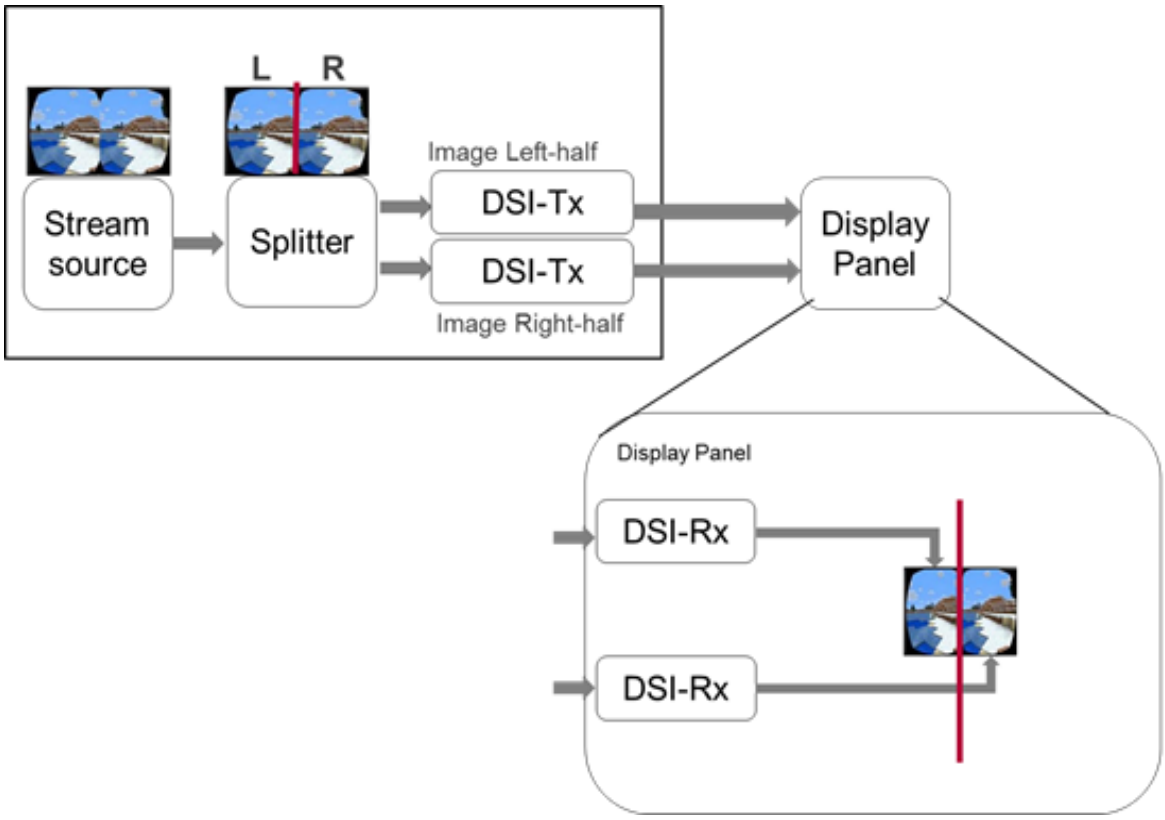

Figure 3-1: **Eight-Lane DSI Implementation**

Shared Logic

Shared Logic provides a flexible architecture that works both as a stand-alone subsystem and as part of a larger design with one of more subsystem instances. This minimizes the amount of HDL modifications required, but at the same time retains the flexibility of the subsystem.

Shared logic in the MIPI DSI TX Subsystem allows you to share MMCMs and PLLs with multiple instances of the MIPI DSI TX Subsystem within the same I/O bank.

There is a level of hierarchy called `<component_name>_support`. Figure 3-2 and Figure 3-3 show two hierarchies where the shared logic is either contained in the subsystem or in the example design. In these figures, `<component_name>` is the name of the generated subsystem. The difference between the two hierarchies is the boundary of the subsystem. It is controlled using the Shared Logic option in the Vivado IDE Shared Logic tab

for the MIPI MIPI DSI TX Subsystem. The shared logic comprises an MMCM, a PLL and some BUFGs (maximum of 4).

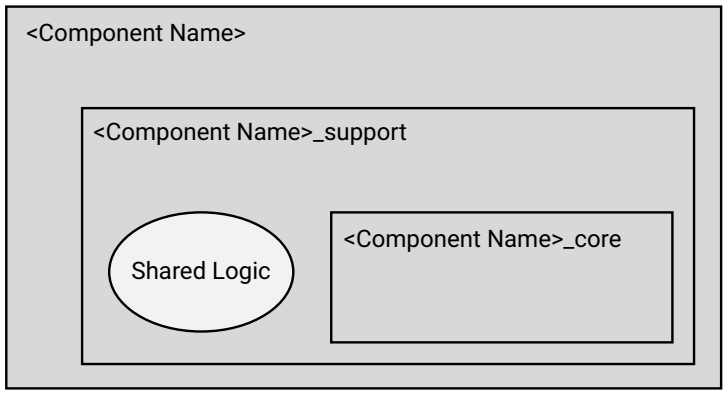

Figure 3-2: **Shared Logic Included in the Subsystem**

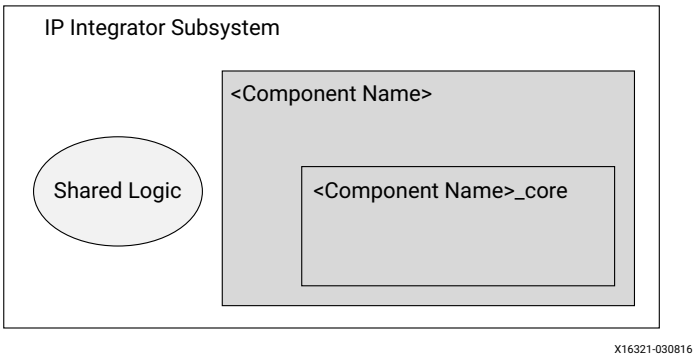

Figure 3-3: **Shared Logic Outside the Subsystem**

Shared Logic in the Subsystem

Selecting **Shared Logic in the Core** implements the subsystem with the MMCM and PLL inside the subsystem to generate all the clocking requirement of the PHY layer.

Select **Include Shared Logic in Core** if:

- You do not require direct control over the MMCM and PLL generated clocks.
- You want to manage multiple customizations of the subsystem for multi-subsystem designs.
- This is the first MIPI DSI TX Subsystem in a multi-subsystem system.

These components are included in the subsystem, and their output ports are also provided as subsystem outputs.

Shared Logic Outside the Subsystem

The MMCMs and PLLs are outside this subsystem instance.

Select **Include Shared Logic in example design** if:

- This is the second MIPI DSI TX Subsystem instance in a multi-subsystem design.
- You only want to manage one customization of the MIPI DSI TX Subsystem in your design.
- You want direct access to the input clocks.

To fully utilize the MMCM and PLL, customize one MIPI DSI TX Subsystem with shared logic in the subsystem and one with shared logic in the example design. You can connect the MMCM/PLL outputs from the first MIPI DSI TX Subsystem to the second subsystem. If you want fine control you can select **Include Shared Logic in example design** and base your own logic on the shared logic produced in the example design.

Figure 3-4 shows the sharable resource connections from the MIPI DSI TX Subsystem with shared logic included (MIPI_DSI_SS_Master) to the instance of another MIPI DSI TX Subsystem without shared logic (MIPI_DSI_SS_Slave00 and MIPI_DSI_SS_Slave01).

A total of 24 MIPI DSI TX subsystems can be implemented in a single HP I/O bank assuming one TX clock lane and one TX data lane are configured per core.

IMPORTANT: *The master and slave cores should be configured with the same line rate when sharing clkoutphy.*

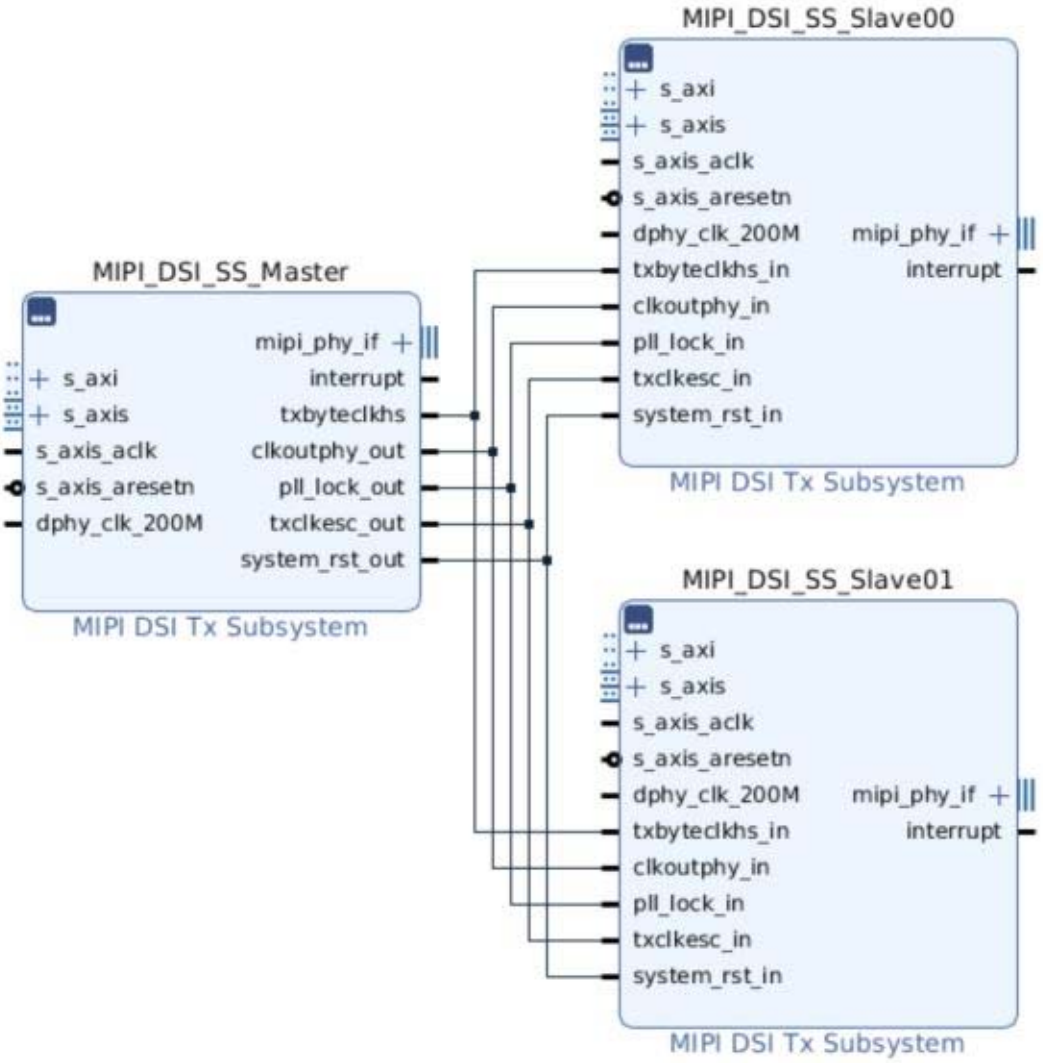

Figure 3-4: **Shared Logic in the Example Design**

I/O Planning for Versal ACAPs

The MIPI DSI TX Subsystem GUI do not have I/O Assignment tab for Versal® ACAPs. Instead you need to use consolidated I/O planning in the main Vivado IDE Planning that is nibble planner. You can select any I/O for the clock and data lanes for the selected XPIO bank.

Detailed steps on how to perform the Vivado IDE planning is detailed under section "I/O Planning for Versal Advanced IO Wizard" in *Advanced I/O Wizard LogiCORE IP Product Guide* (PG320) [Ref 18]. While selecting IOs in a bank across nibbles, you need to ensure the Inter-nibble, Inter-byte clock guidelines are followed. Refer to the "Clocking" section in *Versal ACAP SelectIO Resources Architecture Manual* (AM010) [Ref 19].

Clocking

The subsystem clocks are described in Table 3-5. Clock frequencies should be selected to match the data rate selected on PPI interface. As PPI interface does not allow any throttling, the input video stream should have enough bandwidth to provide the pixel data.

Table 3-5: **Subsystem Clocks**

Clock Mame	Description
s_axis_aclk(1)(2)	Clock used by the subsystem to receive pixel stream on AXI4-Stream Interface.
dphy_clk_200M	See the *MIPI D-PHY LogiCORE IP Product Guide* (PG202) [Ref 4] for information on this clock. The same 200 MHz clock is used by register interface (s_axi) of the subsystem to access registers of its sub-cores.

Notes:

1. s_axis_aclk: The frequency of this clock should be greater than or equal to the minimum required frequency based on the resolution. For example for 1080p@60 Hz, 8 bits per pixel, the minimum required pixel frequency is 148.5 MHz. Therefore the s_axis_aclk should be minimum of 148.5 MHz or higher.
2. Maximum video clock is 250 MHz for UltraScale+ devices and 175 MHz for 7 series devices. If required, a higher throughput can be achieved by increasing the Pixels per clock option from Single to Dual or Quad.

Resets

DSI Transmitter Controller has one hard reset (**s_axis_aresetn**) and one register based reset (soft reset).

- **s_axis_aresetn**: All the core logic blocks reset to power-on conditions including registers.
- The soft reset resets the Interrupt Status register (ISR) of DSI TX Controller and does not affect the core processing.

The subsystem has one external reset port:

- **s_axis_aresetn**: Active-Low reset for the subsystem blocks

The duration of **s_axis_aresetn** should be a minimum of 40 **dphy_clk_200M cycles** to propagate the reset throughout the system.

The reset sequence is shown in Figure 3-5.

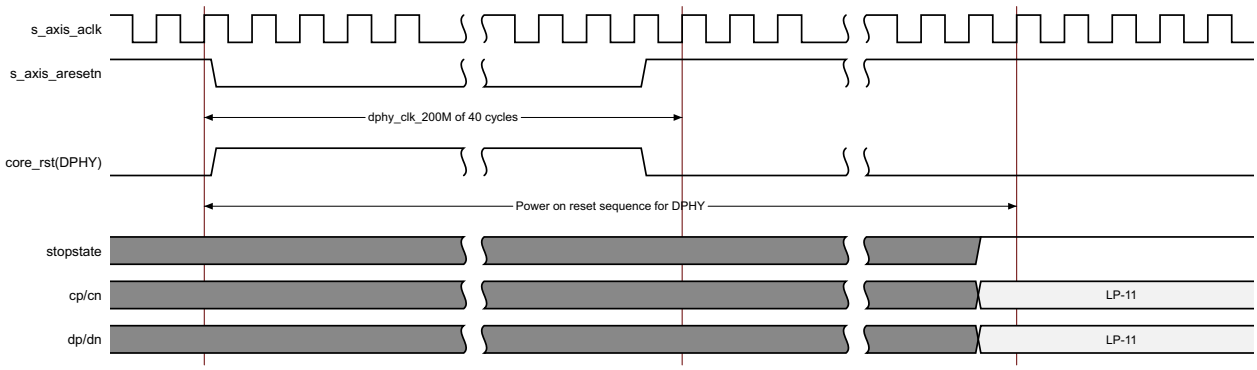

Figure 3-5: **Reset Sequence**

Table 3-6 summarizes all resets available to the MIPI DSI TX Subsystem and the components affected by them.

Table 3-6: **Subsystem Components**

Sub-core	s_axis_aresetn
MIPI DSI TX Controller	Connected to s_axi_aresetn core port
MIPI DPHY	Inverted signal connected to core_rst port
AXI Crossbar	Connected to aresetn port

Note: The effect of each reset (`s_axis_aresetn`) is determined by the ports of the sub-cores to which they are connected. See the individual sub-core product guides for the effect of each reset signal.

Protocol Description

This section contains the programming sequence for the subsystem. Program and enable the components of subsystem in the following order:

1. MIPI DSI TX Controller
2. MIPI D-PHY (if register interface is enabled)

Address Map Example

Table 3-7 shows an example based on a subsystem base address of 0x44A0_0000 (32 bits) where MIPI D-PHY register interface is enabled.

Table 3-7: **Address Map**

Core	Base address
MIPI DSI TX Controller	0x44A0_0000
MIPI DPHY	0x44A1_0000

MIPI DSI TX Controller Core Programming

The MIPI DSI TX Controller programming sequence is described in Programming Sequence. Figure 3-6 and Figure 3-7 show a graphical representation of the sequence.

Programming Sequence

This sequence describes the general steps to transfer a video data received on input stream interface with required video marking packets embedded.

Case 1: Program Timing Values Set and Enable the Core

1. Read `core_config` register to ensure that "control ready" bit is set to '1' before enabling the core any time (for example, after reset or after disabling the core).
2. Select the required settings for Video Mode, EoTp, etc. in the Protocol configuration register.
3. Based on peripheral resolution and timing requirements, arrive the word count values for all the different packets to be sent in video frame (HBP, HFP, HSA, HACT, etc.).
4. Enable the core and send video stream on input interface.
5. The core starts adding the required markers and then consumes the input video stream when the internal timing reaches the active portion of the video.
6. All along this sequence either continuously poll or wait for external interrupt (if enabled) and read Interrupt status register for any errors/status reported.

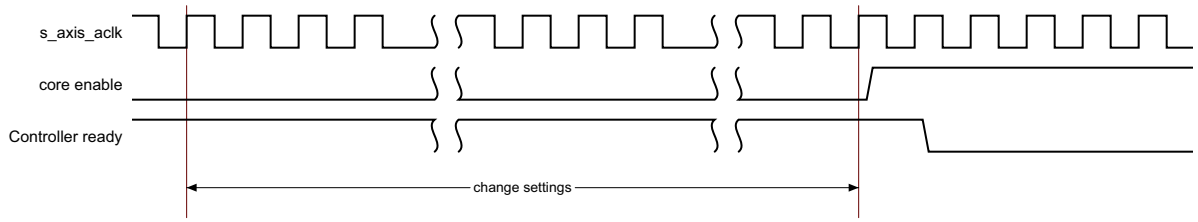

Figure 3-6: **Core Programming Sequence - 1**

Case 2: Program a Different Timing Values Set

1. Follow sequence in Case 1: Program Timing Values Set and Enable the Core for first set of values.
2. In order to program the next set of timing values, the core has to be disabled and then enabled back with a new set of timing values programmed.

Case 3: Disabling/Enabling the Core

1. Any time during the core operation, the core can be disabled using the **core_config** register.
2. After the core is disabled, you must wait/poll until the control ready bit is set in the **core_config** register.
3. Then you can re-enable the core after programming new settings.

Note: Any changes to `bllp_mode` and blanking packet type values during core operation will take effect during the next BLLP period.

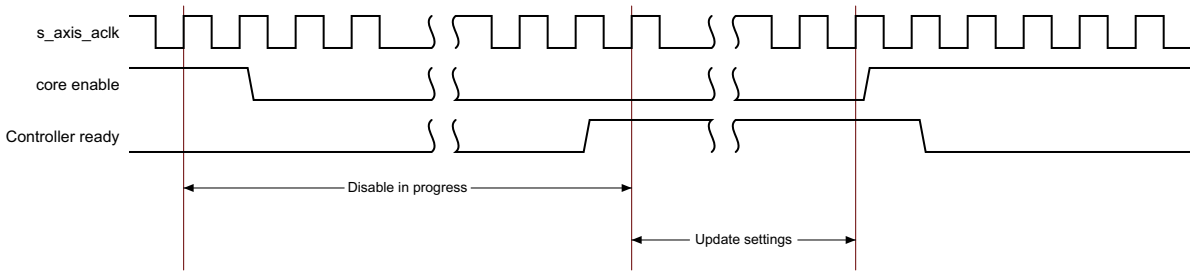

Figure 3-7: **Core Programming Sequence - 2**

Case 4: Enabling the Core in Video/Command Mode

1. Command Mode:
   - If **core_en** = 0: Enable bits 3 and 0 in Core Configuration Register (0x0)
   - If **core_en** = 1 and **command_mode** = 0
     - Disable the core, make core enable 0
     - Enable command mode
     - Enable Core
2. Video Mode:
   - If **core_en** = 0:
     - Program required Timing Registers
     - Make **core_en** = 1 and command mode bit = 0
   - If **core_en** = 1 and **command_mode** = 1
     - Wait for command execution in progress (bit 11 of 0x2C) to be 0
     - Ensure timing registers are programmed
     - Make command mode = 0

Writing Short Command:

- Check for ready for short packet to be 1 (Bit 6 in 0x2C).
- Then Write the required short packet command in 0x30 register.

Note: Short Commands written in command mode are executed within command mode. Short commands given in video mode are executed in Blanking periods.

Writing Long Command:

Long commands are to be given only in command mode, 0x29 and 0x39 are the valid data types.

- Check for ready for long packet bit to be 1 (Bit 7 in 0x2C register).
- Write datatype and word count (WC) fields in 0x30 Data FIFO register. Word count is written to bits (15:8) of 0x30.
- Write 4 bytes at a time of payload data to 0x34 Data FIFO register.
  - For example, if you have 3 bytes as WC. Write 0x00P3P2P1 to the 0x34 Data FIFO register. Append the extra MSB bits with 0s.
  - Similarly, if you have 6 bytes of valid data program WC as 6, then write 0xP4P3P2P1 to 0x34 register. Next write 0x0000P6P5 to the same 0x34 Data FIFO register

MIPI D-PHY IP Core Programming

See the *MIPI D-PHY LogiCORE IP Product Guide* (PG202) [Ref 4] for MIPI D-PHY IP core programming details.

Design Flow Steps

This chapter describes customizing and generating the subsystem, constraining the subsystem, and the simulation, synthesis and implementation steps that are specific to this subsystem. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 10]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 12]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 13]

Customizing and Generating the Subsystem

This section includes information about using Xilinx tools to customize and generate the subsystem in the Vivado Design Suite.

If you are customizing and generating the subsystem in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 10] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the **`validate_bd_design`** command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the subsystem using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the **Customize IP** command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 12].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). The layout depicted here might vary from the current version.

The subsystem configuration screen is shown in Figure 4-1.

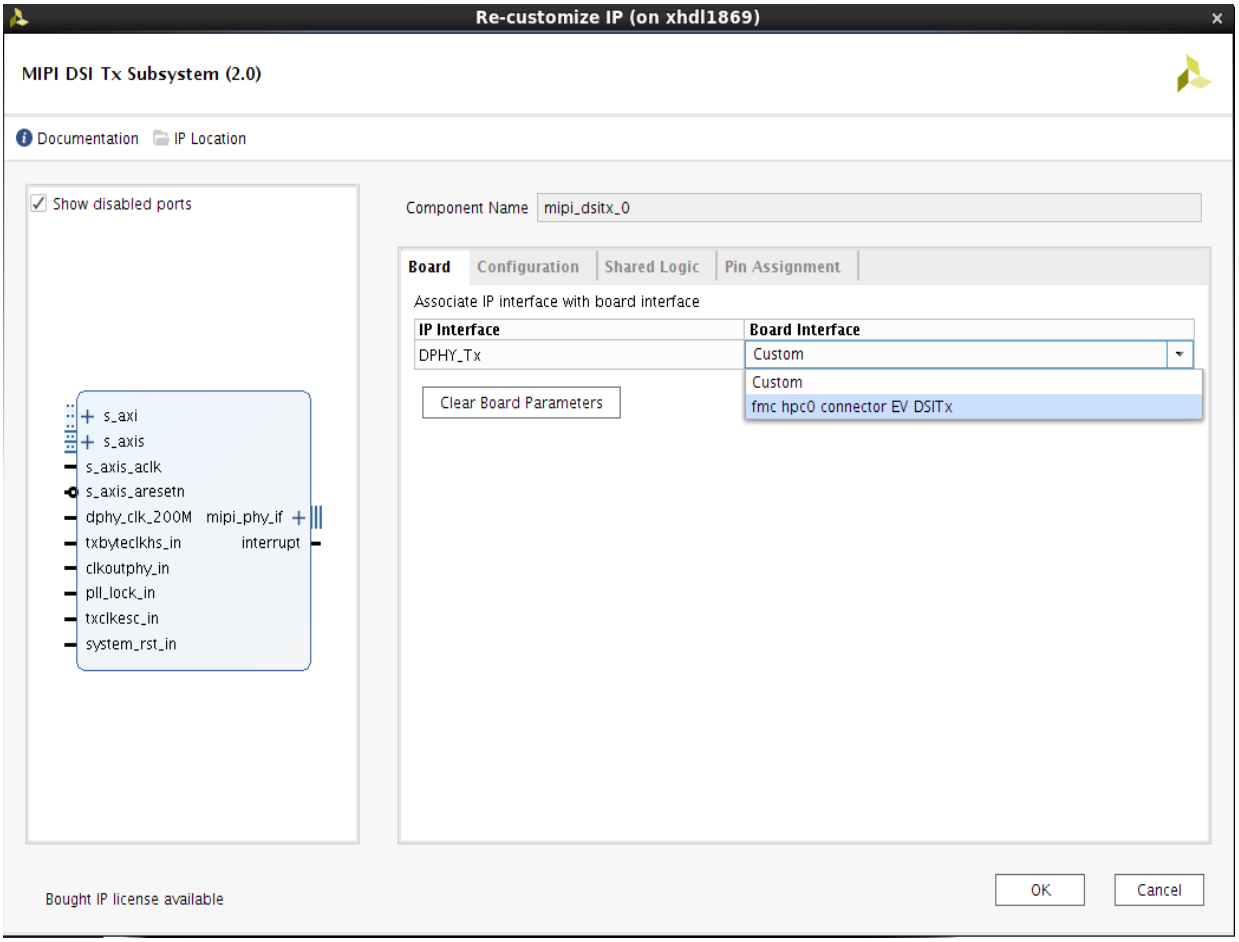

Figure 4-1: **Customization Screen - Board**

Component Name: The Component Name is used as the name of the top-level wrapper file for the subsystem. The underlying netlist still retains its original name. Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9, and "_". The default is **`mipi_dsi_tx_subsystem_0`**.

Board Tab

The Board tab page provides board automation related parameters. The subsystem board configuration screen is shown in Figure 4-1.

Board Interface: Select the board parameters.

- **Custom:** Allows user to configure custom values (no board automation support).
- **fmc_hpc0_connector_EV_DSITx**: Applicable only for ZCU102 board with FMC_HPC0 connector selected as LI-IMX274MIPI-FMC V1.0 during board selection.
  This selection automatically configures the MIPI DSI TX Subsystem to support AUO display which can be connected to EV FMC card. For more information, see the LI-IMX274MIPI-FMC product page [Ref 20].

Note: Board automation is valid for ZCU102 board with FMC_HPC0 slot only.

Configuration Tab

The Configuration tab page provides core related configuration parameters. The subsystem configuration screen is shown in Figure 4-2.

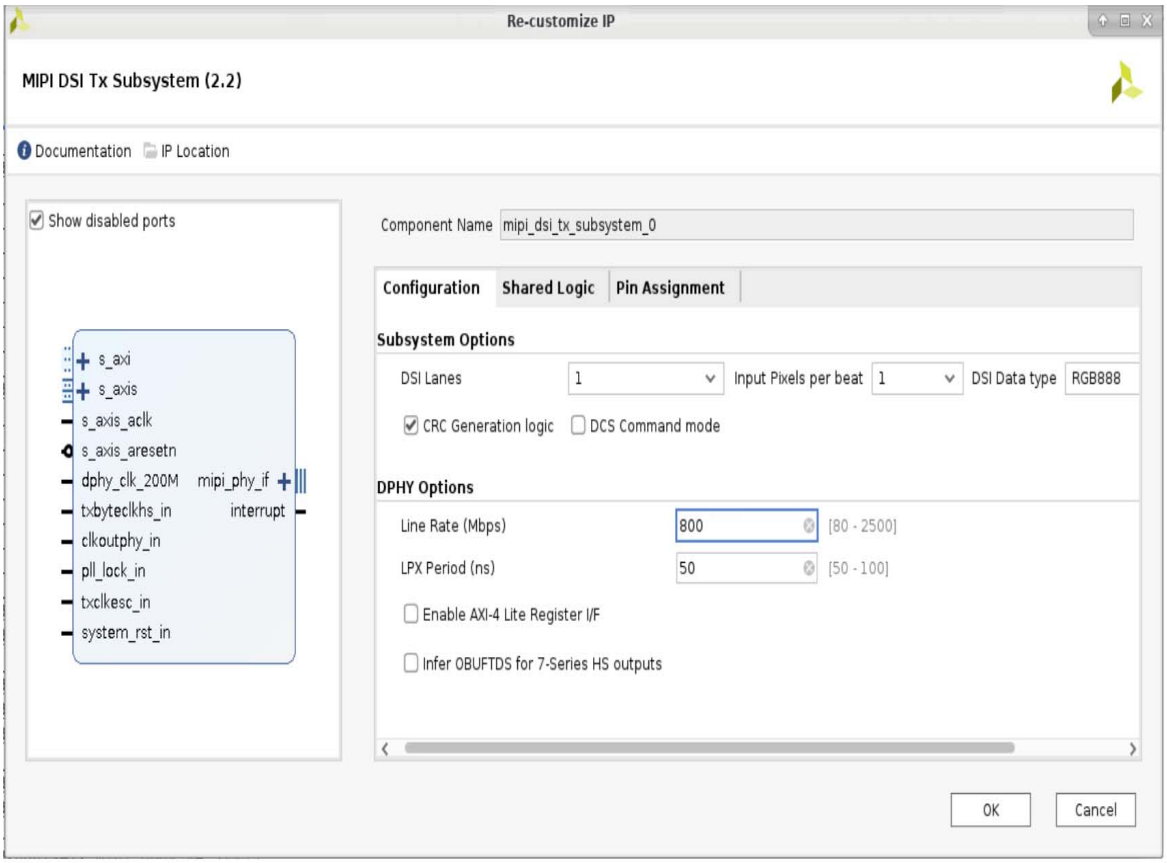

Figure 4-2: **Customization Screen - Configuration**

DSI Lanes: Specifies the number of D-PHY lanes for this subsystem.

Input Pixels per beat: Specifies the number of pixels per clock received on AXI-4 Stream Video interface.

DSI Data type: Specifies the Data Type (Pixel Format) as per DSI protocol (RGB888, RGB565, RGB666_L, RGB666_P, Compressed).

DCS Command Mode: Includes DCS command mode logic in controller sub core.

CRC Generation logic: Includes CRC generation logic for long packets.

Enable Initial Deskew Transmission: When set, the core generates initial skew calibration packets.

Note: Periodic Deskew is not supported.

Line Rate (Mb/s): Selects the line rate for the MIPI D-PHY core. Vivado® IDE automatically limits the line rates based on the selected device. For details about family or device specific line rate support, refer to the data sheet for your device.

For 7-Series devices, the internal IP timing is tested and validated up to maximum value of 1152Mb/s (For example, On ZC702). Depending on setup, meeting timing at higher line rates is possible. GUI provides option to configure higher line rate.

Note: Internal timing limitations are different than physical timing limitations which are based on the PCB configuration and may be lower. The variables include external PHY, resistor network, and termination options. Ensure to validate the line requirement pf design with full system handling.

LPX Period (ns): Transmitted length of any low-power state period.

Enable AXI-4 Lite Register I/F: Select to enable the register interface for the MIPI D-PHY core.

Guarantees the clock Rising Edge Alignment to first payload bit on serial lines: This option is available for Versal® TX configuration when selected the first payload bit will be aligned to rising edge of the serial clock.

Infer OBUFTDS for 7 series HS outputs: Select this option to infer OBUFTDS for HS outputs.

Note: This option is available only for 7 series D-PHY TX configuration. It is recommended to use this option for D-PHY compatible solution based on resistive circuit. For details, see *D-PHY Solutions* (XAPP894) [Ref 17].

Shared Logic Tab

The Shared Logic tab page provides shared logic inclusion parameters. The subsystem shared logic configuration screen is shown in Figure 4-3.

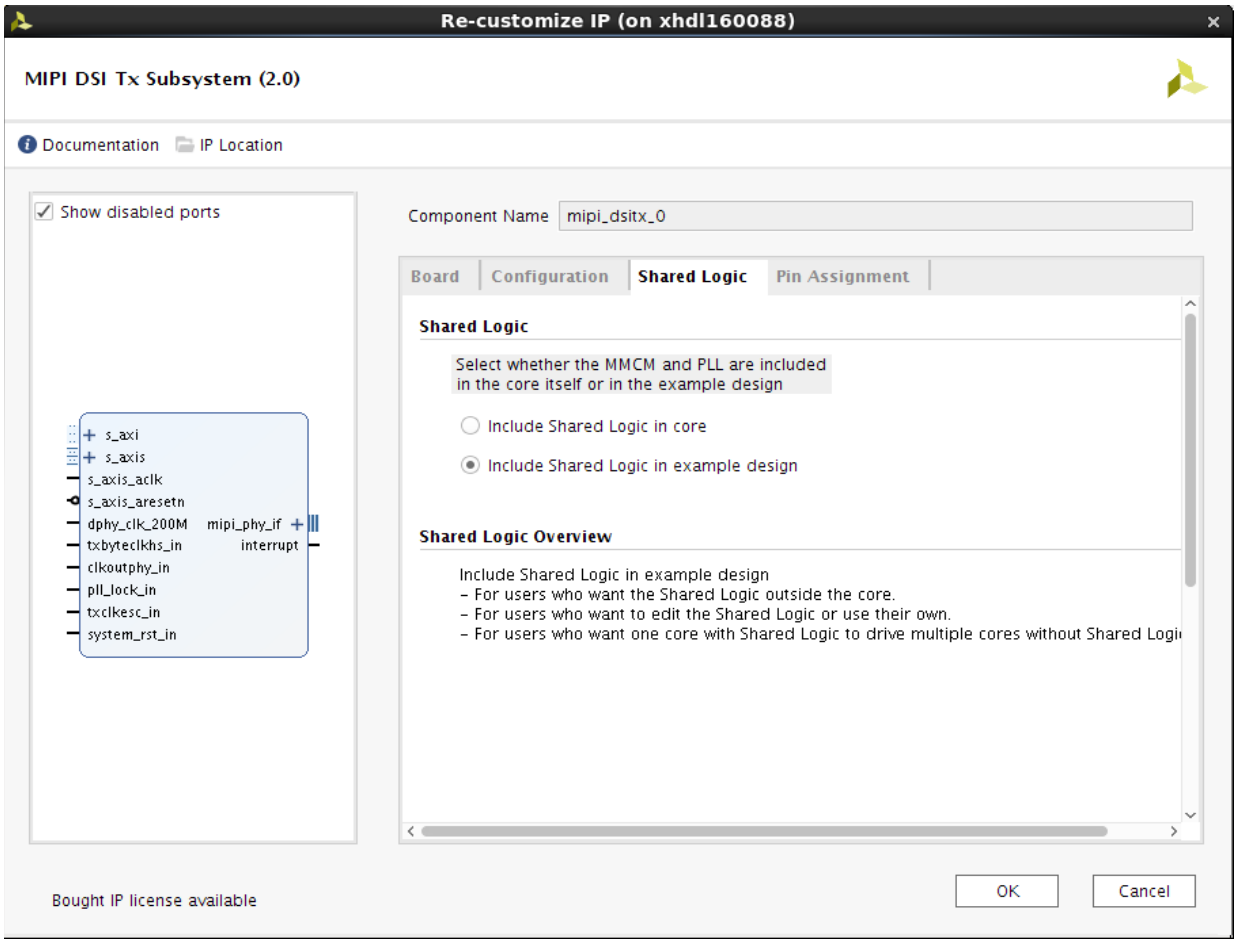

Figure 4-3: **Customization Screen - Shared Logic**

Shared Logic: Select whether the MMCM and PLL are included in the core or in the example design. Values are:

- Include Shared Logic in core
- Include Shared Logic in example design

Pin Assignment Tab for UltraScale+

The Pin Assignment tab page allows to select pins. The subsystem pin assignment configuration screen is shown in Figure 4-4.

Note: This tab is not available for 7 series device configurations.

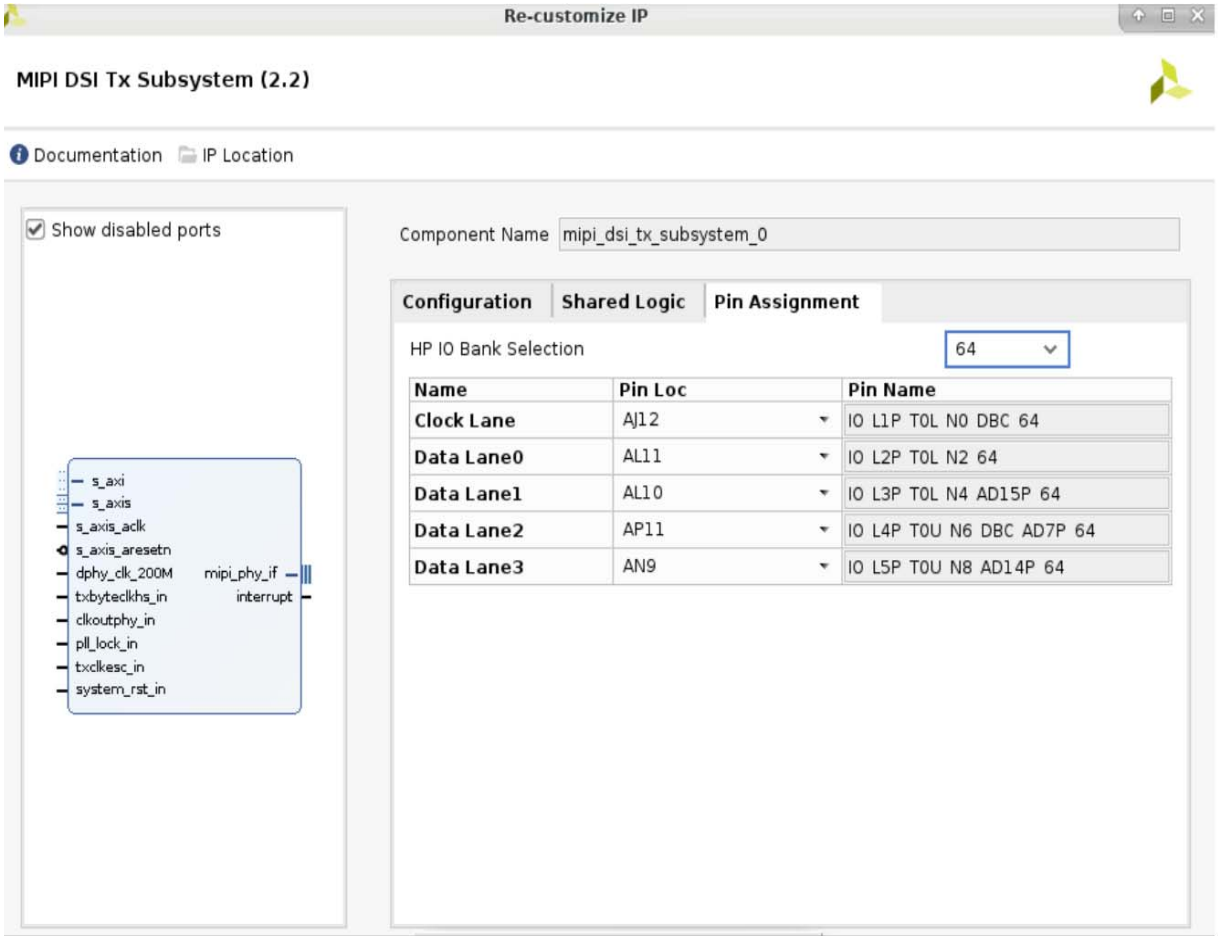

Figure 4-4: **Customization Screen - Pin Assignment**

HP IO Bank Selection: Select the HP I/O bank for clock lane and data lane implementation.

Clock Lane: Select the LOC for clock lane. This selection determines the I/O byte group within the selected HP I/O bank.

Data Lane 0/1/2/3: Displays the Data lanes 0, 1, 2, and 3 LOC based on the clock lane selection.

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

User Parameter	Vivado IDE Parameter	Default Value	Allowable Value
DSI_LANES	DSI Lanes	1 to 4	Maximum of 4 lanes
DSI_DATATYPE	DSI Data type	RGB888	RGB666 (Loosely, Packed), RGB565, RGB888, Compressed Pixel stream. (Only formats listed in sec 10.2.1 of DSI Specification are supported.)
DSI_CRC_GEN	CRC Generation logic	1	0: No CRC calculated for long packets, fixed to 0x0000 1: CRC calculated for long packets
C_DSI_XMIT_INITIAL_DESKEW	Enable Initial Deskew Transmission	0	0: No initial skew calibration packets sent. 1: the core generates initial skew calibration packets.
C_INCLUDE_DCS_CMD_MODE	DCS Command Mode,	0	0: The subsystem doesn't support command only mode and long command packets. 1: Supports long and short commands in command mode.
DSI_PIXELS	Input Pixels per beat	1	Pixels per beat received on input stream interface Single pixel per beat Dual pixels per beat Quad pixels per beat
DHY_LINERATE	Line Rate (Mb/s)	800	Versal ACAPs: 260–2500 Mb/s UltraScale+/7 series: 80–2500 Mb/s
DPHY_LPX_PERIOD	LPX Period (ns)	50	50–100 (ns)
C_EN_CTS_TX	Guarantees the rising edge clock alignment to first payload data bit	False	True: Guarantees the rising edge clock alignment to first payload data bit. False: Do not guarantee the rising edge clock alignment to first payload data bit. ***Note:*** Applicable only for Versal devices.
DPHY_EN_REGIF	Enable AXI-4 Lite Register I/F	0	0: Disable register interface for DPHY 1: Enable register interface for DPHY
SupportLevel	Shared Logic	0	
HP_IO_BANK_SELECTION	HP IO Bank Selection	Value based on part selected.	
CLK_LANE_IO_LOC	Clock Lane	Value based on part selected.	
DATA_LANE0_IO_LOC	Data Lane0	Value based on part selected.	
DATA_LANE1_IO_LOC	Data Lane1	Value based on part selected.	
DATA_LANE2_IO_LOC	Data Lane2	Value based on part selected.	

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

User Parameter	Vivado IDE Parameter	Default Value	Allowable Value
DATA_LANE3_IO_LOC	Data Lane 3	Value based on part selected.	
C_EN_HS_OBUFTDS	Infer OBUFTDS for 7Series HS outputs	0	Enable OBUFTDS for 7Series devices

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11].

Constraining the Subsystem

This section contains information about constraining the subsystem in the Vivado Design Suite.

Required Constraints

This section is not applicable for this subsystem.

Device, Package, and Speed Grade Selections

This section is not applicable for this subsystem.

Clock Frequencies

See Clocking.

Clock Management

The MIPI DSI TX Subsystem sub-core MIPI D-PHY uses an MMCM to generate the general interconnect clocks, and the PLL is used to generate the serial clock and parallel clocks for the PHY. The input to the MMCM is constrained as shown in *Clock Frequencies* section of *MIPI D-PHY LogiCORE IP Product Guide* (PG202) [Ref 4]. No additional constraints are required for the clock management.

Clock Placement

This section is not applicable for this subsystem.

Banking

The MIPI DSI TX Subsystem provides the Pin Assignment Tab option to select the HP I/O bank. Clock lane and data lane(s) are implemented on the selected I/O bank BITSLICE(s).

Transceiver Placement

This section is not applicable for this subsystem.

I/O Standard and Placement

The MIPI standard serial I/O ports should use MIPI_DPHY_DCI for the I/O standard in the XDC file for UltraScale+ family. The LOC and I/O standards must be specified in the XDC file for all input and output ports of the design. The MIPI DSI TX Subsystem MIPI D-PHY sub-core generates the I/O pin LOC for the pins that are selected during IP customization for UltraScale+ designs. No I/O pin LOC are provided for 7 series MIPI D-PHY IP designs. You have to manually select the clock capable I/O for 7 series TX clock lane and restrict the I/O selection within the I/O bank for MIPI D-PHY TX.

It is recommended to select the I/O bank with VRP pin connected for UltraScale+ MIPI D-PHY TX IP core. If VRP pin is present in other I/O bank in the same I/O column of the device the following DCI_CASCADE XDC constraint should be used. For example, I/O bank 65 has a VPR pin and the D-PHY TX IP is using the I/O bank 66.

```
set_property DCI_CASCADE {66} [get_iobanks 65]
```

Simulation

There is no simulation supported example design available for MIPI DSI TX Subsystem.

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 11].

Example Design

MIPI DSI TX Subsystem does not support the application example design generation. For MIPI DSI TX example design, refer to MIPI CSI-2 RX Subsystem Application example design

which uses MIPI DSI TX Subsystem for display option. This example design also allows you to test the MIPI DSI only path using Test Pattern Generator.

For more information, see *MIPI CSI-2 Receiver Subsystem Product Guide* (PG232) [Ref 5].

AMD XILINX

Appendix A

Verification, Compliance, and Interoperability

The MIPI DSI TX Subsystem has been verified using both simulation and hardware testing. A highly parameterizable transaction-based simulation test suite has been used to verify the core. The tests include:

- Different lane combinations and line rates
- High-Speed Data reception with short/long packets, different pixel formats and video modes.
- All possible output pixels per clock, pixel type combinations.
- Recovery from error conditions
- Register read and write access

Hardware Validation

The MIPI DSI TX Subsystem is tested for functionality, performance, and reliability using Xilinx® evaluation platforms. The MIPI DSI TX Subsystem verification test suites for all possible modules are continuously being updated to increase test coverage across the range of possible parameters for each individual module.

A series of MIPI DSI TX Subsystem test scenarios are validated using the Xilinx development boards listed in Table A-1. These boards permit the prototyping of system designs where the MIPI DSI TX Subsystem processes different short/long packets received on serial lines.

Table A-1: **Xilinx Development Board**

Target Family	Evaluation Board	Characterization Board
Zynq® UltraScale+™ MPSoC	ZCU102	N/A

7 series devices do not have a native MIPI IOB support. You will have to target the HP bank and/or HR Bank I/O for MIPI IP implementation. For more information on MIPI IOB compliant solution and guidance, refer *D-PHY Solutions* (XAPP894) [Ref 17].

Table A-2 shows the Interoperability Testing:

Table A-2: **Interoperability Testing**

MIPI Display	Board/Device	Tested Configuration	Resolution
B101UAN01.7	ZCU102/ xczu9eg-ffvb1156-2-e	1000 Mb/s 4 Lanes RGB888 Sync Events	1920x1200 @60fps
B101UAN01.7	VCK190	1000 Mb/s 4 Lanes RGB888 Sync Events	1920x1200 @60fps

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

TIP: *If the IP generation halts with an error, there might be a license issue. See Licensing and Ordering for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the MIPI DSI Transmitter Subsystem, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the MIPI DSI Transmitter Subsystem. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this subsystem can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the MIPI DSI Transmitter Subsystem

AR: 66769

Technical Support

Xilinx provides technical support at the Xilinx Support web page for this IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address MIPI DSI Transmitter Subsystem design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 15].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

General Checks

- Ensure MIPI DPHY and MIPI DSI TX Controller cores are in the enable state by reading the registers.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. See Figure B-1 for a read timing diagram. Output **`s_axi_arready`** asserts when the read address is valid, and output **`s_axi_rvalid`** asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The **`lite_aclk`** inputs are connected and toggling.
- The interface is not being held in reset, and **`lite_aresetn`** is an active-Low reset.
- The main subsystem clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a debug feature capture that the waveform is correct for accessing the AXI4-Lite interface.

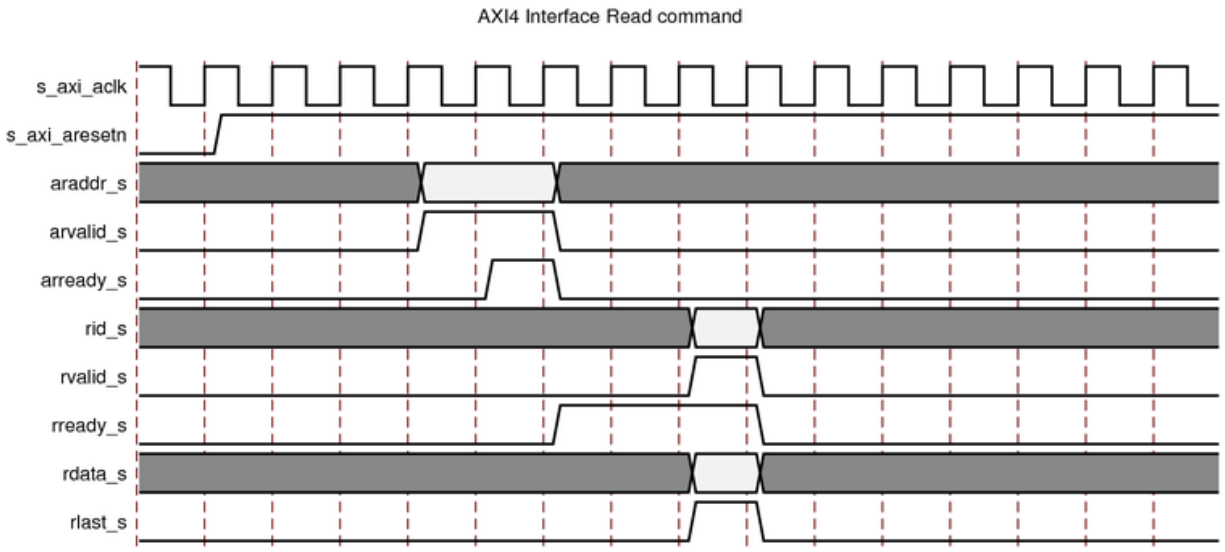

Figure B-1: **AXI4-Lite Timing**

AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions:

- If transmit **<interface_name>_tready** is stuck Low following the **<interface_name>_tvalid** input being asserted, the subsystem cannot send data.
- If the receive **<interface_name>_tvalid** is stuck Low, the subsystem is not receiving data.
- Check that the **video_aclk** and **dphy_clk_200M** inputs are connected and toggling.
- Check subsystem configuration.

Application Software Development

The MIPI **dsitxss** driver is provided with the installation of the Xilinx tool set. It is also available in Xilinx GitHub. Additionally, link to a GUI version of the API is available in DocNav's Video Design Hub. For an example of the driver API, see the examples section of *MIPI CSI-2 Receiver Subsystem Product Guide* (PG232) [Ref 5].

AMD XILINX

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter **`docnav`**.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. MIPI Alliance Standard for Display Serial Interface DSI: mipi.org/specifications/display-interface
2. MIPI Alliance Standard for Physical Layer D-PHY: mipi.org/specifications/physical-layer
3. *AXI4-Stream Video IP and System Design Guide* (UG934)
4. *MIPI D-PHY LogiCORE IP Product Guide* (PG202)
5. *MIPI CSI-2 Receiver Subsystem Product Guide* (PG232)
6. *AXI Interconnect LogiCORE IP* Product Guide (PG059)
7. *AXI IIC Bus Interface LogiCORE IP Product Guide* (PG090)
8. *UltraScale Architecture SelectIO Resources User Guide* (UG571)
9. *Vivado Design Suite: AXI Reference Guide* (UG1037)
10. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
11. *Vivado Design Suite User Guide: Designing with IP* (UG896)
12. *Vivado Design Suite User Guide: Getting Started* (UG910)
13. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
14. *ISE to Vivado Design Suite Migration Guide* (UG911)
15. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
16. *Vivado Design Suite User Guide: Implementation* (UG904)
17. *D-PHY Solutions* (XAPP894)
18. *Advanced I/O Wizard LogiCORE IP Product Guide* (PG320)
19. *Versal ACAP SelectIO Resources Architecture Manual* (AM010)
20. LI-IMX274MIPI-FMC product page: LI-IMX274MIPI-FMC

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/26/2022	2.2	Added Register 0x68 to consider D-PHY LP to HS Switching latency.
07/15/2021	2.2	Editorial update.
07/14/2021	2.2	• Updated Features section. • Updated Configuration Tab with a new parameter. • Updated Table 4-1 with C_EN_CTS_TX parameter.
02/04/2021	2.2	• Updated Configuration Tab screen for V2.2. • Added information for Line Rate (Mb/s). • Updated Default Line Rate (Mb/s) value for DHY_LINERATE to 800. • Added Versal (VCK190) information in Interoperability. • Updated Licensing and Ordering. • Updated Application Software Development.
07/14/2020	2.1	Added support for Versal devices.
06/26/2020	2.1	• New clocking architecture for line rates above 1500 Mb/s, which removes need for ctrl_clk. • Relaxed restriction on input pixels per clock and data types for line rates above 1500 Mb/s.
10/30/2019	2.0	• Extended line rate support to 2500 Mb/s • Added DCS Long Packet Support
11/14/2018	2.0	• Added Spartan 7 series support • Updated Unsupported Features section • Added an important note in the Shared Logic Outside the Subsystem section • Updated Simulation section
10/04/2017	2.0	• MIPI D-PHY serial pins grouped as interface • Board automation support added for FMC:LI-IMX274MIPI-FMC V1.0 which can be placed on ZCU102 FMC HPC0 slot. This FMC can interface MIPI AUO display
04/05/2017	1.1	• MIPI D-PHY 3.1 changes integrated
10/05/2016	1.1	• MIPI D-PHY 3.0 changes integrated • 7 Series support • Details on Timing Register(s) calculation procedure and more than 4 Lane implementation added
04/06/2016	1.0	Initial Xilinx release

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.