MIPI DSI Transmitter Subsystem v1.0

Product Guide

Vivado Design Suite

PG238 April 6, 2016

Table of Contents

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, etc.) in accordance to the DSI protocol and user programmed options. The packet framed is sent over an MIPI DPHY 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 80 to 1500 Mb/s
- Supports all mandatory data types with fixed Virtual Channel Identifier (VC) of 0
- Programmable EoTp generation support
- ECC generation for packet header
- CRC generation for data bytes (optional)
- 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]	UltraScale+™ Families Zynq® UltraScale+ MPSoC
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
Test Bench	Not Provided
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver[2]	Standalone
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	
Provided by Xilinx at the 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 SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

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.

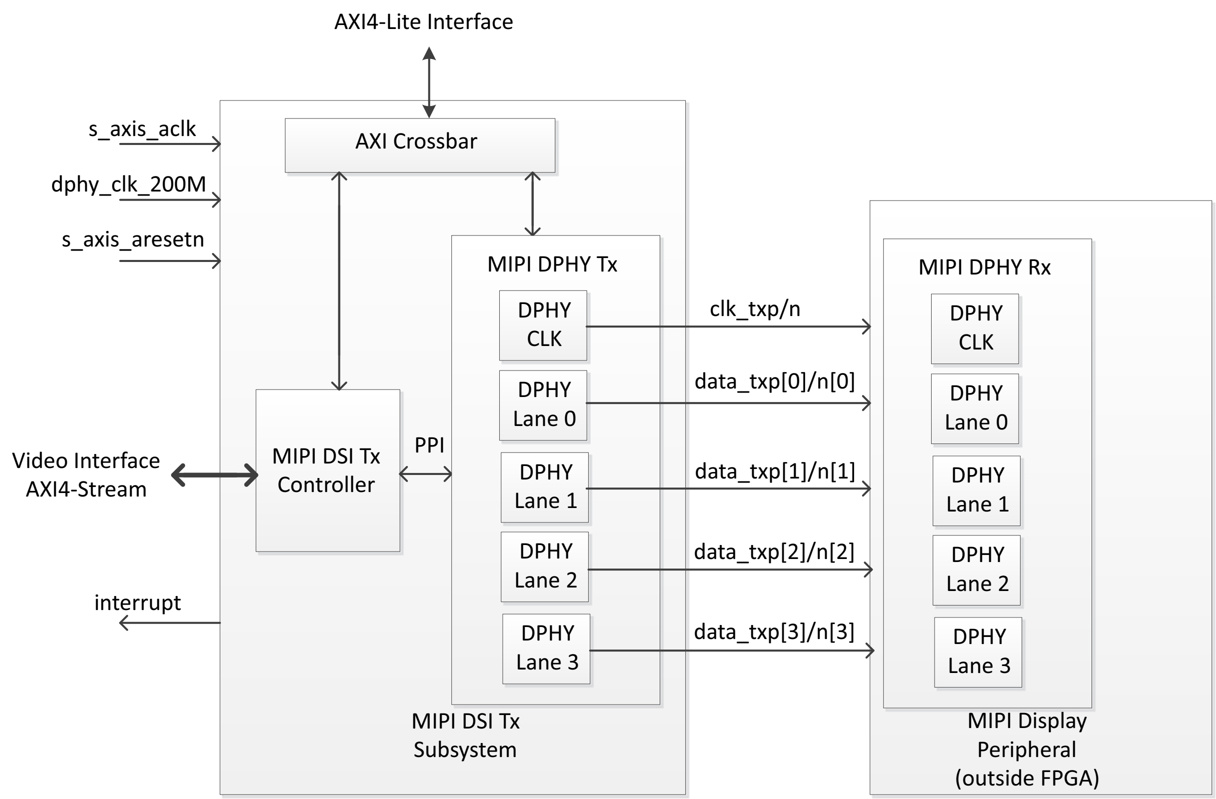

Figure 1-1: **Subsystem Architecture**

The subsystem consists of the following sub-blocks:

- MIPI D-PHY
- MIPI DSI TX Controller
- AXI Crossbar

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. 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 command packets during BLLP periods of video frames. Sub-block details of MIPI DSI TX Controller are shown in Figure 1-2.

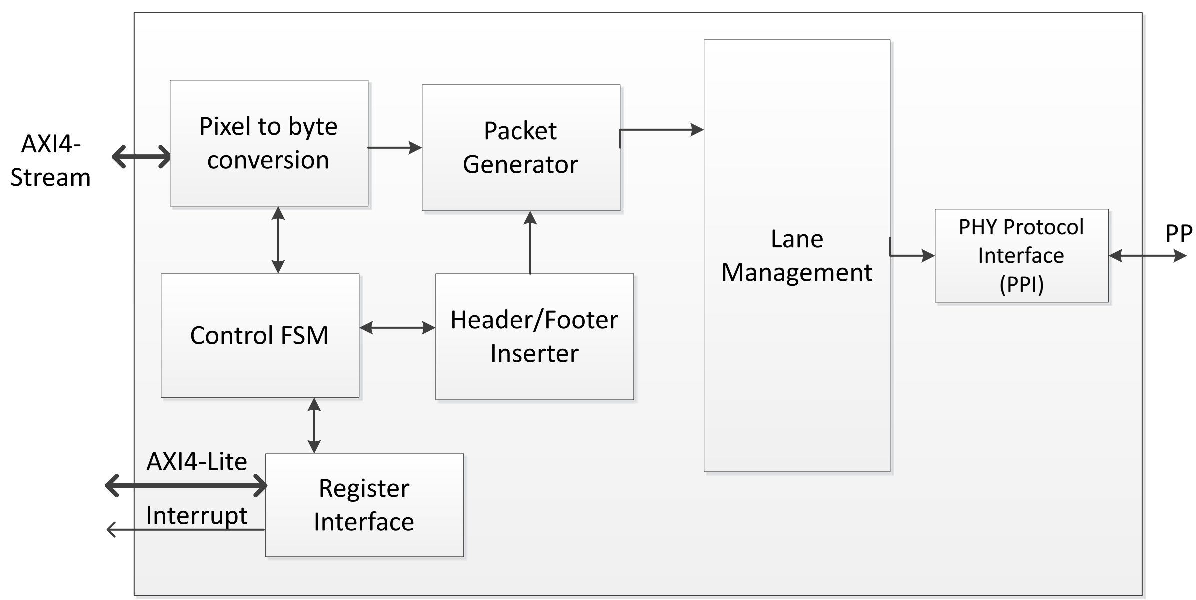

Figure 1-2: **Sub-blocks**

The features of this core include:

- 1 to 4 Lane support with a data rate of 1500Mbps per lane: 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 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 the controller finds enough time-slot available during the video blanking periods, these commands are sent over DSI link.
- 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.

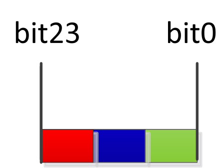

Figure 1-3: **Single Pixel RGB888**

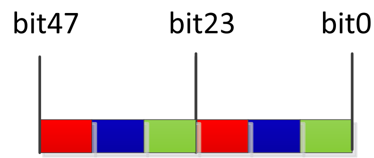

Figure 1-4: **Dual Pixels RGB888**

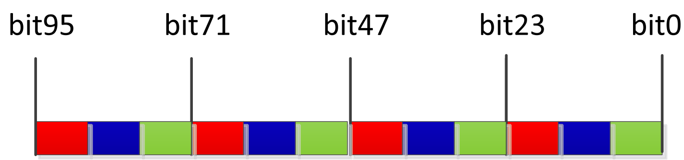

Figure 1-5: **Quad Pixels RGB888**

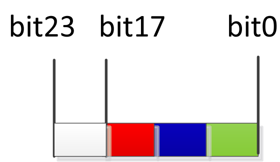

Figure 1-6: **Single Pixel RGB666 (Loosely, Packed)**

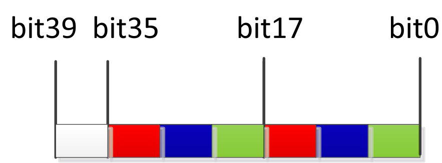

Figure 1-7: **Dual Pixels RGB666 (Loosely, Packed)**

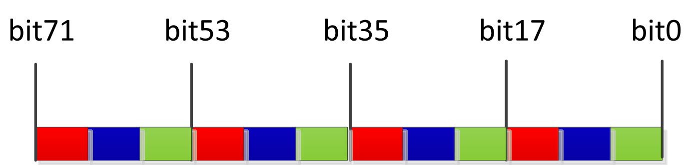

Figure 1-8: **Quad Pixels RGB666 (Loosely, Packed)**

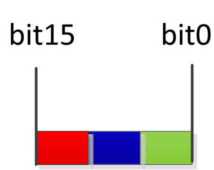

Figure 1-9: **Single Pixel RGB565**

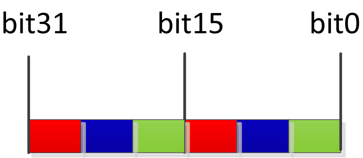

Figure 1-10: **Dual Pixels RGB565**

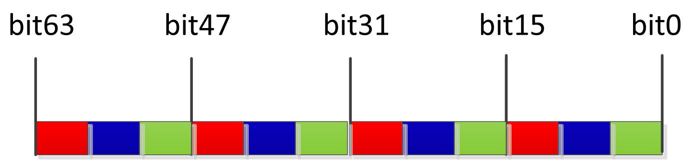

Figure 1-11: **Quad Pixels RGB565**

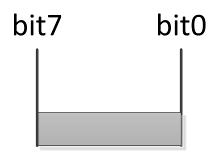

Figure 1-12: **Single Pixel Compressed**

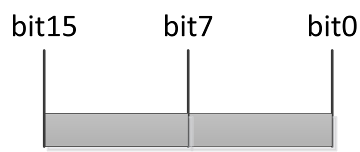

Figure 1-13: **Dual Pixel Compressed**

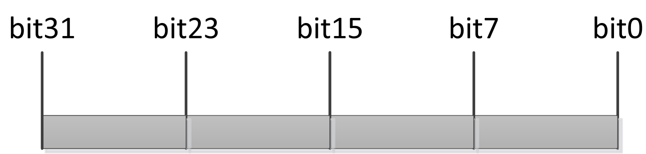

Figure 1-14: **Quad Pixel Compressed**

- 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

- Command mode (which needs Bi-directional MIPI I/O support).
- Optional features of spec (Sub-links) are not supported.
- Bus Turn Around (BTA) not supported.

Licensing and Ordering Information

License Checkers

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following tools:

- Vivado synthesis
- Vivado implementation
- write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

License Type

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. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

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.

Port Descriptions

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

Table 2-1: **Port Descriptions**

Signal Name	Interface	I/O	Description
clk_txp/n	PPI	O	DPHY Clock lane serial lines
data_txp[N-1:0]/n[N-1:0]	PPI	O	DPHY Data lane serial lines N is based on number of lanes configured during core generation.
dphy_clk_200M	System	I	Fixed 200Mhz clock required for MIPI DPHY. The same clock is used by s_axi interface of the subsystem.
s_axis_aclk	System	I	AXI4-Stream Video clock
s_axis_aresetn	System	I	AXI reset. Active Low(Same reset for lite & stream interface).

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

Signal Name	Interface	I/O	Description
s_axi_*	S_AXI	-	AXI4-Lite Interface
s_axis_tready	S_AXIS	O	AXI4-Stream Interface
s_axis_tvalid	S_AXIS	I	AXI4-Stream Interface
s_axis_tlast	S_AXIS	I	AXI4-Stream Interface
s_axis_tdata	S_AXIS	I	AXI4-Stream Interface. Width of this port is dependent on pixel type and no.of pixels per beat
s_axis_tkeep	S_AXIS	I	AXI4-Stream Interface.
s_axis_tuser	S_AXIS	I	AXI4-Stream Interface. TUSER[0] is used to map the Fsync signal of the AXI4-Stream Video Interface. The core do not use this signal, but generates FSYNC packets based on timing registers programming.
System Interface			
Interrupt	System	O	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.

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	Core configuration options
0x04	Protocol Configuration	Protocol configuration options
0x08	Reserved	
0x0C	Reserved	
0x10	Reserved	
0x14	Reserved	
0x18	Reserved	
0x1C	Reserved	
0x20	Global Interrupt Enable	Global interrupt enable registers
0x24	Interrupt status	Interrupt status register
0x28	Interrupt enable	Interrupt enable register
0x2C	Reserved	
0x30	Command Queue Packet	Packet Entry to command Queue.
0x34	Reserved	
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	Blanking packet payload size in bytes(WC) available during VSA,VBP,VFP lines
0x68	Reserved	
0x6C	Reserved	
0x70	Reserved	
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.

Core Configuration Register (0x00)

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
31:2	Reserved	NA	NA	Reserved
2	Controller ready	0x1	R	Controller is ready for processing. During soft-reset or core disable, user can rely on this status that the core stopped all its activity. 0: Controller is not ready 1: Controller is ready
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 FIFO's and registers. Once enabled, controller start from VSS packet.(ie new video frame)

Protocol Configuration Register (0x04)

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 0x0E – Packed RGB565 0x1E- packed RGB666 0x2E – Loosely packed RGB666 0x3E- Packed RGB888 0x0B- Compressed Pixel Stream

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

Bits	Name	Reset Value	Access	Description
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)
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 (0x20)

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 (0x24)

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.
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, user has 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 (0x28)

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 (0x2C)

This register captures different status conditions of the core.

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

Bits	Name	Reset Value	Access	Description
31:6	Reserved	NA	NA	Reserved
5:0	Command Queue Vacancy	0x20	R	Number of command queue entries can be safely written to Command queue FIFO, before it goes full.

Command Queue Packet (0x30)

Only short packets are supported.

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

Bits	Name	Reset Value	Access	Description
31:24	Reserved	NA	NA	Reserved Not used by the core. Recommended to write 0
23:16	Byte-1	0x0	R/W	Byte 1 of short packet(1)
15:8	Byte-0	0x0	R/W	Byte 0 of short packet(1)
7:6	VC	0x0	R/W	VC value of short packet
5:0	Data type	0x0	R/W	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 2-16 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

Table 2-11: **Command Queue Packet Data Types *(Cont'd)***

Data Type	Description
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).

Timing Register-1 (0x50)

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.*

Table 2-12: **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)
15:0	BLLP Burst Mode	0x0	R/W	BLLP duration of VACT region packet payload size in bytes(WC). Applicable only Burst mode

Timing Register-2 (0x54)

Table 2-13: **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 (0x58)

Table 2-14: **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)
15:0	HFP	0x0	R/W	Horizontal front porch blanking packet payload size in bytes(WC)

Timing Register-4 (0x5C)

Table 2-15: **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
15:8	VBP	0x0	R/W	Vertical back porch lines
7:0	VFP	0x0	R/W	Vertical front porch lines

Line Time (0x60)

Total line time is calculated by the core (in bytes) based on timing parameters programmed and non-burst/burst mode selection, shown in Table 2-16.

Table 2-16: **Video Timing (Line Time)**

Bits	Name	Reset Value	Access	Description
31:0	Line Time	0x0	R	Total line size in bytes

BLLP Time (0x64)

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 2-17.

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

Bits	Name	Reset Value	Access	Description
31:0	BLLP Time	0x0	R	BLLP duration in bytes.

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.

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.

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

The values of the timing registers must arrive in terms of DSI byte 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.

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

There is a level of hierarchy called <component_name>_support. Figure 3-1 and Figure 3-2 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 DSI TX Subsystem. The shared logic comprises an MMCM, a PLL and some BUFGs (maximum of 4).

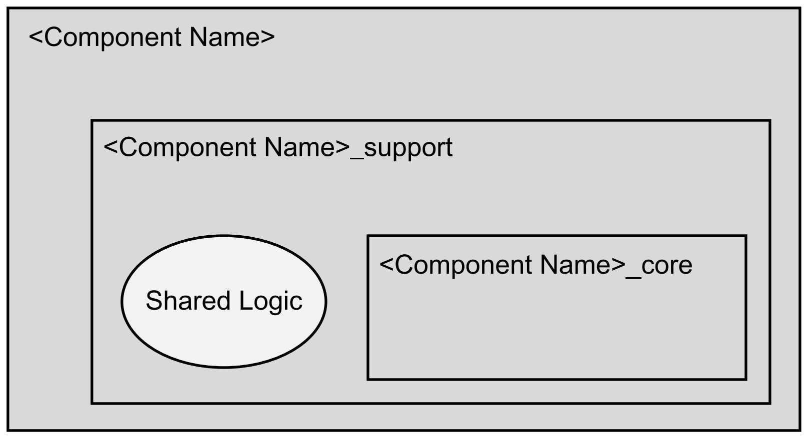

Figure 3-1: **Shared Logic Included in the Subsystem**

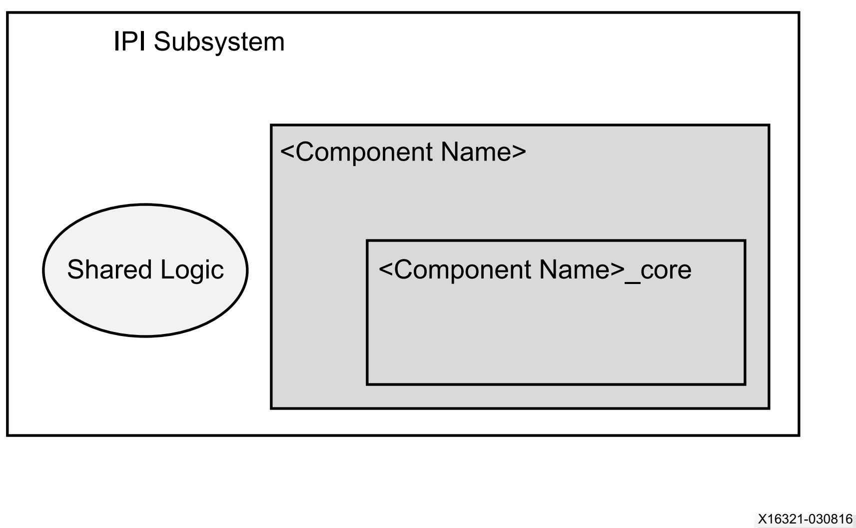

Figure 3-2: **Shared Logic Outside the Subsystem**

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 3-3 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.*

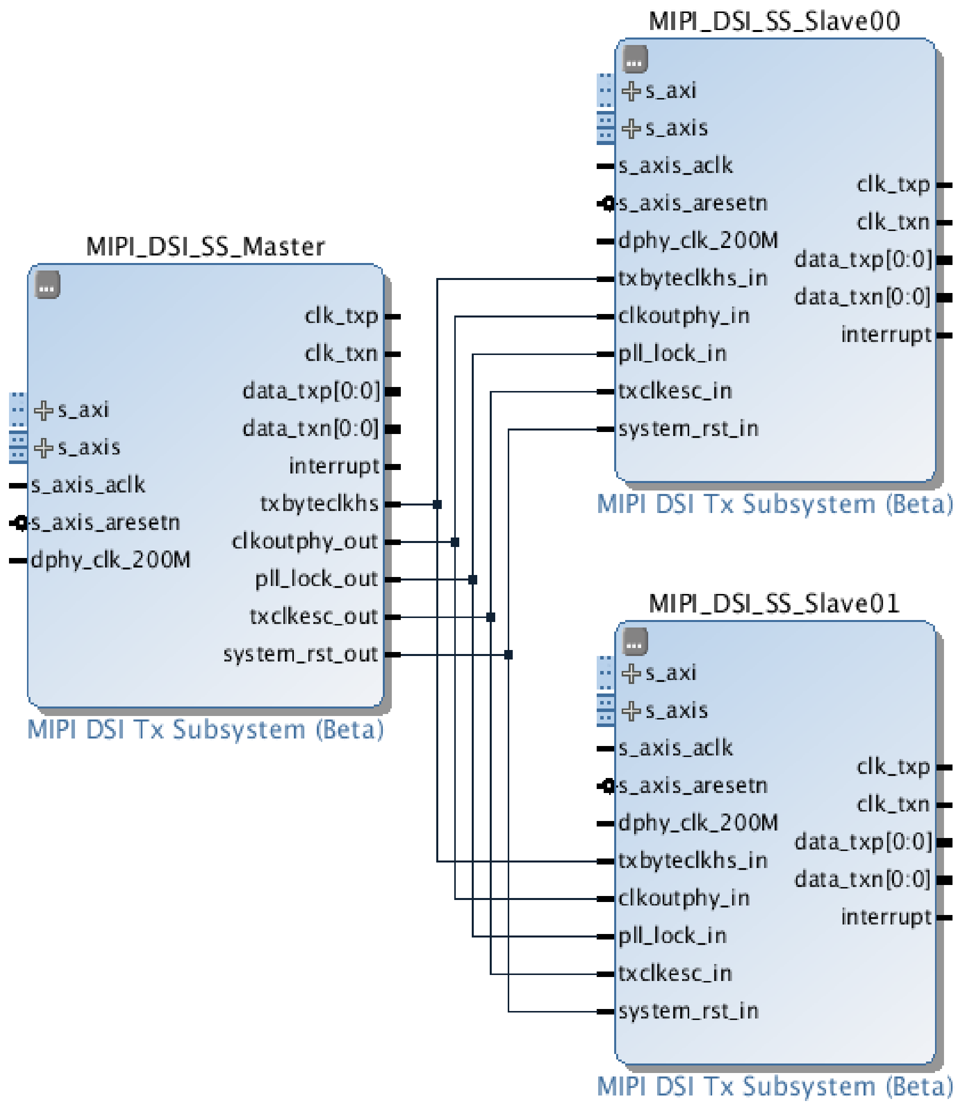

Figure 3-3: **Shared Logic in the Example Design**

Clocking

The subsystem clocks are described in Table 3-1. 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 3-1: **Subsystem Clocks**

Clock Mame	Description
s_axis_aclk	Clock used by the subsystem to receive pixel stream on AXI4-Stream Interface.(1)
dphy_clk_200M	See the *MIPI D-PHY LogiCORE IP Product Guide* (PG202) [Ref 4] for information on this clock. The same 200Mhz 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@60Hz,8bits per pixel, the minimum required pixel frequency is 148.5Mhz. Therefore the s_axis_aclk should be minimum of 148.5Mhz or higher.

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 20 `dphy_clk_200M cycles` to propagate the reset throughout the system.

The reset sequence is shown in Figure 3-4.

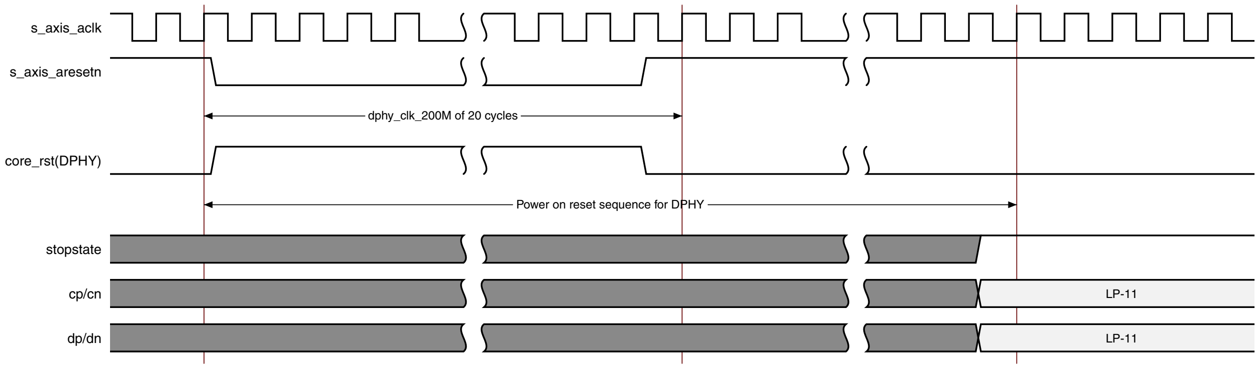

Figure 3-4: **Reset Sequence**

Table 3-2 summarizes all resets available to the MIPI DSI TX Subsystem and the components affected by them.

Table 3-2: **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 3-3 shows an example based on a subsystem base address of 0x44A0_0000 (32 bits) where MIPI D-PHY register interface is enabled.

Table 3-3: **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 3-5 and Figure 3-6 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.

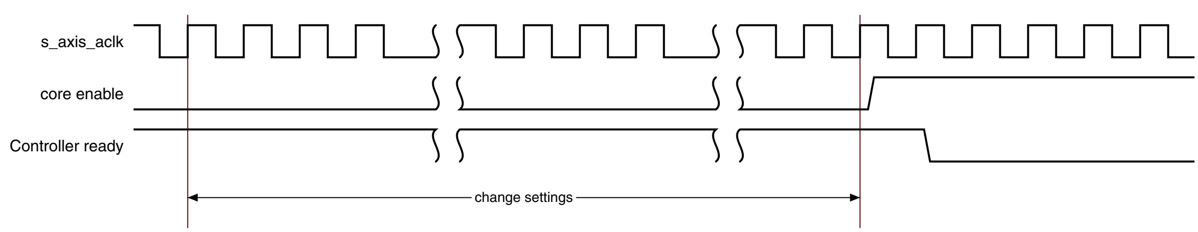

Figure 3-5: **Core Programming Sequence - 1**

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. Program the next set of values any time during the core operation
3. The new set will be taken into account at the frame boundary by the core automatically.

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.

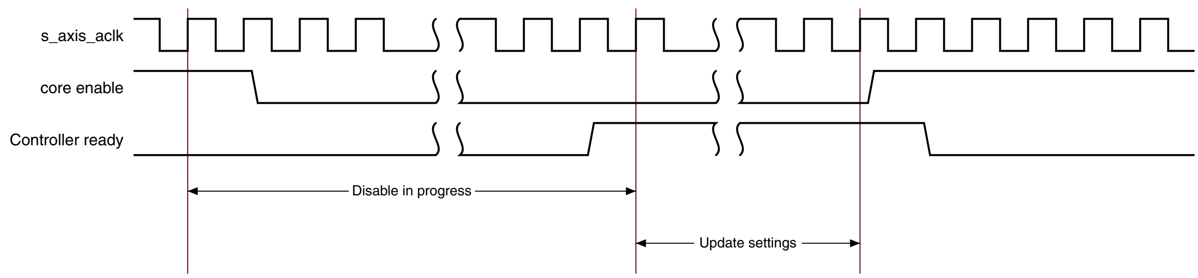

Figure 3-6: **Core Programming Sequence - 2**

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 8]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 10]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 11]

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 8] 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 9] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 10].

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.

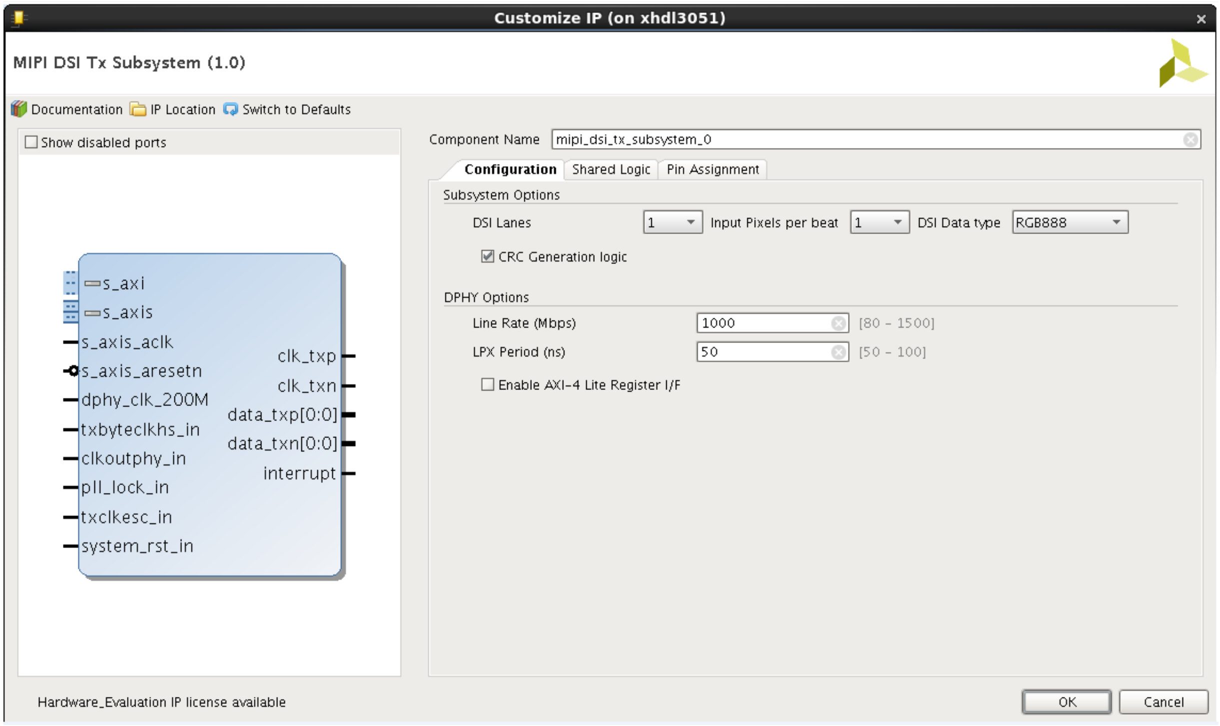

Figure 4-1: **Customization Screen**

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.

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).

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

Line Rate (Mbps): Selects the line rate for the MIPI D-PHY core.

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.

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
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 (Mbps)	1000	80 – 1500 Mbps
DPHY_LPX_PERIOD	LPX Period (ns)	50	50-100 (ns)
DPHY_EN_REGIF	Enable AXI-4 Lite Register I/F	0	0: Disable register interface for DPHY 1: Enable register interface for DPHY

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

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. The LOC and I/O standards must be specified in the XDC file for all input and output ports of the design.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 11].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

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 interleaving cases (data type and virtual channel)
- All possible output pixels per clock, pixel type combinations.
- Recovery from error conditions
- Register read and write access

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 License Checkers in Chapter 1 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 13].

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.
- Ensure underrun condition does not get reported during normal operation of the core. Ensure line buffer full condition is not set in the MIPI DSI TX Controller Interrupt Status register.

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.

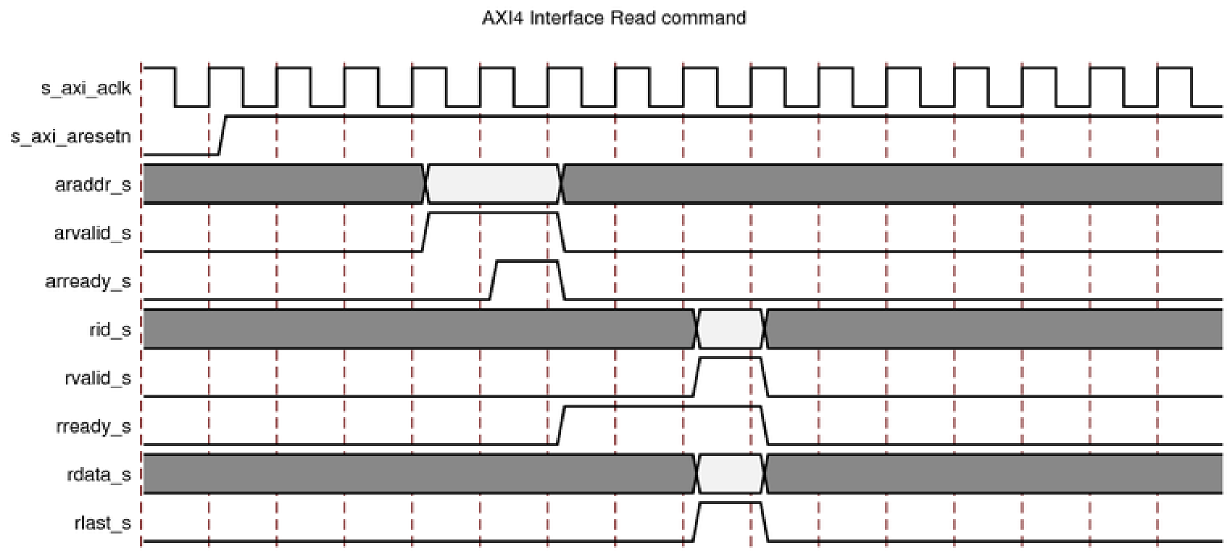

Figure B-1: **AXI4-Lite Timing**

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.
- Ensure “Stream line buffer full” condition not getting reported in subsystem Interrupt Status register

Application Software Development

Software driver information is not currently available.

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

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

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. *AXI Interconnect LogiCORE IP* Product Guide (PG059)
6. *AXI IIC Bus Interface LogiCORE IP Product Guide* (PG090)
7. *Vivado Design Suite: AXI Reference Guide* (UG1037)
8. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
9. *Vivado Design Suite User Guide: Designing with IP* (UG896)
10. *Vivado Design Suite User Guide: Getting Started* (UG910)
11. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
12. *ISE to Vivado Design Suite Migration Guide* (UG911)
13. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
14. *Vivado Design Suite User Guide: Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/06/2016	1.0	Initial Xilinx release.

Please Read: Important Legal Notices