1:2 and 1:1 MIPI DSI Display Interface Bridge Soft IP

User Guide

FPGA-IPUG-02001 Version 1.1

July 2016

Contents

Tables

Figures

1. Introduction

The Mobile Industry Processor Interface (MIPI®) provides specifications for standardization in consumer mobile devices. MIPI Display Serial Interface (DSI) and MIPI D-PHY specifications were developed to create a standardized interface for all displays used in the mobile industry. As the industry evolves, bandwidth requirements have exceeded what display manufacturers are capable of manufacturing, while application processor vendors can provide very fast interfacing capabilities. For a cost effective solution, displays can later be replaced with newer display, with the processor retained. Also, multiple displays have gained popularity and extending the output to two display interfaces from a single source becomes a requirement to support these applications.

The Lattice Semiconductor MIPI DSI to DSI Display Interface Bridge IP allows users to resolve these interfacing problems with the Lattice Semiconductor CrossLink™ programmable device.

1.1. Quick Facts

Table 1.1 provides quick facts about the MIPI DSI to DSI Display Interface Bridge IP for Crosslink device.

Table 1.1. MIPI DSI to DSI Display Interface Bridge IP Quick Facts

		DSI to DSI IP Configuration			
		4-Lane Gear 16 Continuous Rx Clock	**4-Lane Gear 8 Continuous Rx Clock**	**4-Lane Gear 16 Non-continuous Rx Clock**	**4-Lane Gear 8 Non-continuous Rx Clock**
IP Requirements	FPGA Families Supported	Crosslink			
Resource Utilization	Targeted Device	LIF-MD6000-6MG81I			
	LUTs	5202	5790	5135	5811
	EBRs	20	8	20	8
	Registers	1983	1591	1991	1600
	Programmable IO	13	13	14	14
Design Tool Support	Lattice Implementation	Lattice Diamond® 3.8			
	Synthesis	Lattice Synthesis Engine			
		Synopsys® Synplify Pro® L-2016.03L			
	Simulation	Aldec® Active HDL™ 10.3 Lattice Edition			

1.2. Features

The key features of the MIPI DSI to DSI Display Interface Bridge IP are:

- Interfaces one or two MIPI DSI compliant receivers to a MIPI DSI transmitter
- Supports up to 5.76 Gb/s per MIPI DSI interface.
- Supports 1, 2, 3 or 4 data lanes and one clock lane per MIPI DSI interface
- Allows users to store and program a new set of device DCS (Display Command Set)
- Supports all MIPI DSI compatible video formats (RGB, YCbCr and User Defined)
- Compliant with MIPI D-PHY v1.1 and MIPI DSI v1.1 specifications

1.3. Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL. This includes radix indications and logical operators.

1.3.2. Data Ordering and Data Types

The highest bit within a data bus is the most significant bit.

Single-bit data stream from each MIPI D-PHY data lane is deserialized into 8-bit or 16-bit parallel data where bit 0 is the first received bit. The size of the parallel data bus depends on the Rx gear setting.

When gear setting is 16, the byte in the lower 8 bits of the 16-bit parallel data is the first byte.

Byte arrangement within data words is illustrated in Figure 1.1.

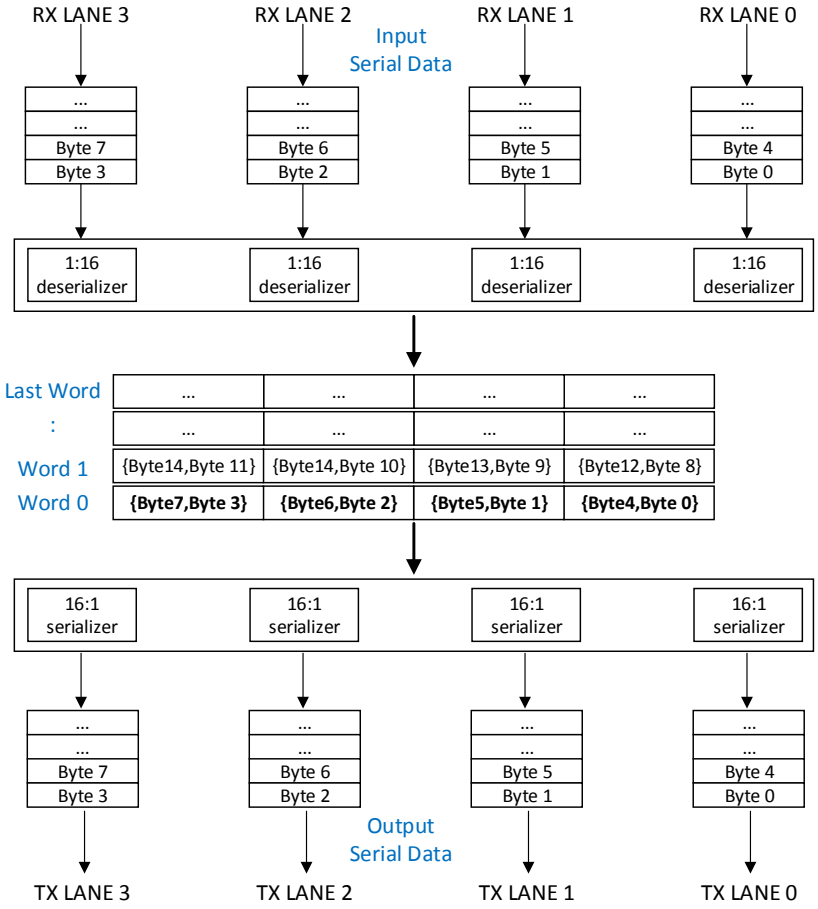

Figure 1.1. Data Ordering for a Gear 16 x4 Configuration

1.3.3. Signal Names

Signal names that end with:

- "_i" are input pins.
- "_o" are output pins.
- "_io" are bi-directional pins.
- "_n_i" are active low input signals.

2. Functional Description

Figure 2.1 shows the top level diagram of the 2:1 MIPI DSI Display Interface Bridge.

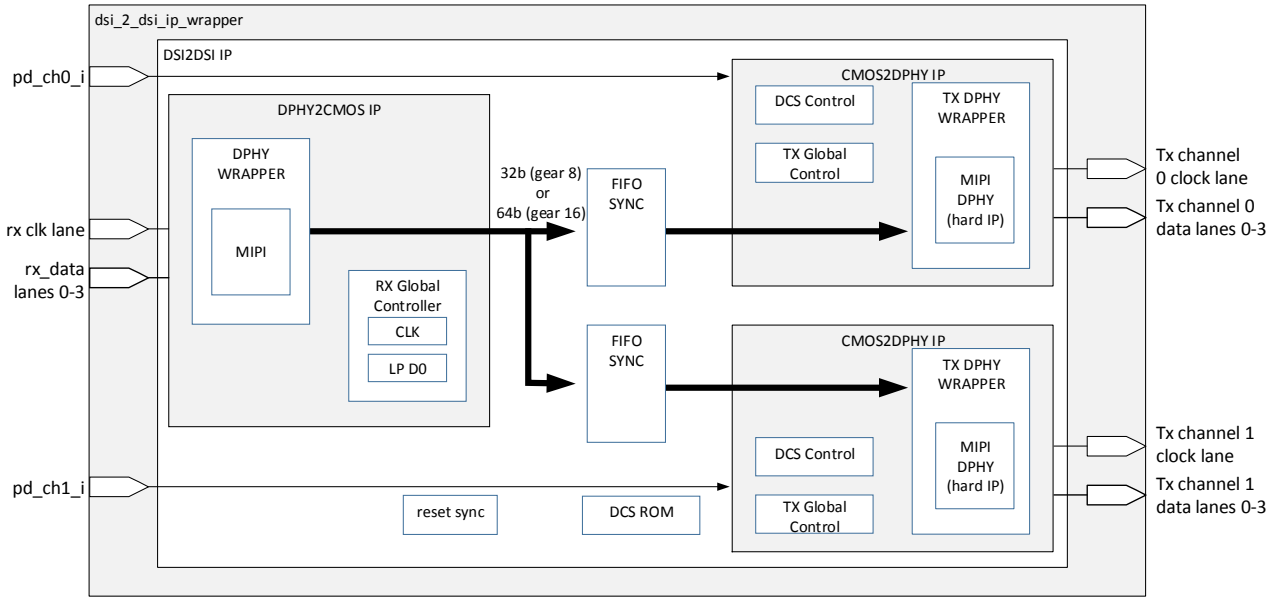

Figure 2.1. DSI to DSI Block Diagram

Table 2.1 describes the ports of the 1:2 MIPI DSI Display Interface Bridge IP.

Table 2.1. Top Level Ports

Signal	Direction	Functional Description
clk_p_i	I	Positive differential Rx D-PHY input clock
clk_n_i	I	Negative differential Rx D-PHY input clock
d0_p_io	IO	Positive differential Rx D-PHY input data 0
d0_n_io	IO	Negative differential Rx D-PHY input data 0
d1_p_i	I	Positive differential Rx D-PHY input data 1
d1_n_i	I	Negative differential Rx D-PHY input data 1
d2_p_i	I	Positive differential Rx D-PHY input data 2
d2_n_i	I	Negative differential Rx D-PHY input data 2
d3_p_i	I	Positive differential Rx D-PHY input data 3
d3_n_i	I	Negative differential Rx D-PHY input data 3
clk_ch0_p_o	O	Positive differential Tx D-PHY output clock, channel 0
clk_ch0_n_o	O	Negative differential Tx D-PHY output clock, channel 0
d0_ch0_p_io	IO	Positive differential Tx D-PHY output data 0, channel 0
d0_ch0_n_io	IO	Negative differential Tx D-PHY output data 0, channel 0
d1_ch0_p_o	O	Positive differential Tx D-PHY output data 1, channel 0
d1_ch0_n_o	O	Negative differential Tx D-PHY output data 1, channel 0
d2_ch0_p_o	O	Positive differential Tx D-PHY output data 2, channel 0
d2_ch0_n_o	O	Negative differential Tx D-PHY output data 2, channel 0
d3_ch0_p_o	O	Positive differential Tx D-PHY output data 3, channel 0
d3_ch0_n_o	O	Negative differential Tx D-PHY output data 3, channel 0
clk_ch1_p_o	O	Positive differential Tx D-PHY output clock, channel 1
clk_ch1_n_o	O	Negative differential Tx D-PHY output clock, channel 1
d0_ch1_p_io	IO	Positive differential Tx D-PHY output data 0, channel 1

Table 2.1 Top Level Ports *(Continued)*

Signal	Direction	Functional Description
d0_ch1_n_io	IO	Negative differential Tx D-PHY output data 0, channel 1
d1_ch1_p_o	O	Positive differential Tx D-PHY output data 1, channel 1
d1_ch1_n_o	O	Negative differential Tx D-PHY output data 1, channel 1
d2_ch1_p_o	O	Positive differential Tx D-PHY output data 2, channel 1
d2_ch1_n_o	O	Negative differential Tx D-PHY output data 2, channel 1
d3_ch1_p_o	O	Positive differential Tx D-PHY output data 3, channel 1
d3_ch1_n_o	O	Negative differential Tx D-PHY output data 3, channel 1
ref_clk_i	I	Input reference clock in Non-continuous Rx clock mode
reset_n_i	I	Asynchronous active low system reset
pd_ch0_i	I	Power down pin of Tx channel 0
pd_ch1_i	I	Power down pin of Tx channel 1
Miscellaneous Debug Ports		
tx0_dcsrom_done	O	Indicates the DCS initialization of Tx channel 0 is done
tx0_tinit_done	O	Indicates the Initialization delay counter from Tx channel 0 is done
tx0_pll_lock	O	PLL lock indicator for Tx channel 0
tx0_byteclock	O	Tx channel 0 output byte clock
tx0_lp_clk_en	O	Low-power clock enable of Tx channel 0
fifo0_empty	O	Indicates that synchronizing FIFO for Tx channel 0 is empty
tx1_dcsrom_done	O	Indicates the DCS initialization of Tx channel 1 is done
tx1_tinit_done	O	Indicates the Initialization delay counter from Tx channel 1 is done
tx1_pll_lock	O	PLL lock indicator for Tx channel 1
tx1_byteclock	O	Tx channel 1 output byte clock
tx1_lp_clk_en	O	Low-power clock enable of Tx channel 1
fifo1_empty	O	Indicates that synchronizing FIFO for Tx channel 1 is empty

2.1. Design and Module Description

Within the dsi2dsi_ip block are instances of:

- dphy2cmos.v
  - Instantiates the MIPI D-PHY Rx soft IP wrapper. The D-PHY wrapper uses Snow DDR I/O and fabric to receive MIPI serial data. This converts the incoming serial data from the D-PHY data lanes to 32-bit (gear8) or 64-bit (gear 16) words.
  - Instantiates the Rx global controller which contains finite state machines (FSMs) that detects the state transitions of the clock and data lanes.
- fifo_sync.v
  - The design uses two instances of 32-word deep FIFO used to pass the Rx data to the Tx clock domains, one for each Tx channel.
  - The purpose of these FIFOs are for clock domain synchronization only and not for data buffering. The MIPI D-PHY specification does not allow for flow control, therefore, the Rx and the Tx byte clock must equal.
- cmos2dphy.v
  - There are two instances of this IP block, one for each Tx channel. Each instance contains the hard MIPI D-PHY Tx wrapper that serializes the packet data. Each wrapper contains its own Tx PLL.
  - This also contains the Tx Global Control module that controls the transitions of the Tx clock and data lanes.
  - This also have the instance of the file containing the Display Command Set. To change the DCS of the DSI peripheral, the user must modify the DCS data array in the dcs_rom.v.
- dcs_rom.v
  - the DCS ROM contains the Display Command Set for the DSI slave.

3. Parameter Settings

Table 3.1 lists the user parameters used to generate the design.

Table 3.1. 2-bit DSI to DSI Parameter Settings

Parameter	Attribute	Options	Description
Number of Rx Lanes	User - configurable	1, 2, 3 or 4	This selects the number of MIPI D-PHY data lanes. The number of Tx data lanes is the same as the number of Rx lanes.
Rx Gear	User - configurable	Gear8 or Gear16	Input serial bits are converted into 8-bit or 16-bit data bus. Gear8 is automatically selected for line rates lower than 250 Mb/s, and Gear16 is automatically selected for line rates above 900 Mb/s.
Rx D-PHY IP	User - configurable	Hard D-PHY or Soft D-PHY	The Hard IP option is only available for single DSI to single DSI configuration. Selecting Hard IP makes use of one of the hardened D-PHY modules as the Rx D-PHY. Selecting the Soft D-PHY IP option generates a Soft D-PHY Rx logic. The maximum line rate when Soft IP is selected is 1.2 Gb/s.
Number of Tx Channels	Read Only	1 or 2	Rx D-PHY IP should be set to Soft D-PHY to enable this option. This option configures the bridge to have two transmission channels. The two channels are asynchronous with each other.
Rx Line Rate	User - configurable	192–1440	Data rate per lane in Mb/s. It must be noted that T_{LPX}, the D-PHY low-power state period, must be twice the byteclock period.
D-PHY Clock Frequency	Read Only	96–720	The D-PHY Clock Frequency is half of the line rate, in MHz.
D-PHY Clock Mode	User - configurable	Continuous or Non-Continuous	In continuous mode, the input D-PHY clock lanes are always in high-speed mode. The DSI to DSI bridge utilizes this clock to generate the byteclock for internal logic. In non-continuous mode, the input D-PHY clock lanes go to low-power states. Therefore, an external clock, refclk_i, is needed.
Byte Clock Frequency	Read Only	(Line rate) / (Rx Gear)	This is the frequency that the internal logic operates at.
Reference Clock Frequency	Read Only	(Line rate) / (Rx Gear)	The frequency of the reference clock for the Tx PLL. This clock is also used to clock the Rx byte clock domain if the D-PHY Clock Mode is non-continuous.
Enable Miscellaneous status signals	User - configurable	ON or OFF	Enabling the miscellaneous signals ports out some internal signals for debug purposes.
tINIT_SLAVE value	User - configurable	16-bit non-zero decimal value	This parameter, in addition to the DCS ROM Wait Time parameter, sets the period needed to meet the required initialization time of the DSI slave. The value is in terms of byteclock cycles. The D-PHY specification places a minimum period of 100us, but this parameter may be increased depending on the receiver requirement. During this period, all incoming data is ignored by the bridge.
Number of DCS words	User - configurable	10-bit non-zero decimal value	This defines the number of valid words in the DCS ROM initialization file, including the sync pattern and the trail bytes.

Table 3.1. 2-bit DSI to DSI Parameter Settings *(Continued)*

Parameter	Attribute	Options	Description
DCS ROM Wait Time	User - configurable	1–4096	This parameter sets the interval between DCS packets in terms of number of byteclock cycles. This applies to both high-speed and low-power DCS timing mode.
DCS Mode	User - configurable	Low power or High Speed	DCS initialization of the DSI slave may be performed in D-PHY low-power timing mode, or in high-speed mode. High-Speed option is only available for 1-, 2- or 4-lane configuration only.
DCS ROM initialization file	User - configurable	Text file	This must be a text file that contains the display command set data packets. See Appendix C. *Initializing the DCS ROM* for the format of entries.
HS-SKIP parameter*	User - configurable	0–20	This sets the $T_{HS-SKIP}$ in terms of number of byteclock cycles. This is the time interval during which the design neglects the high-speed data lane transitions before data lane0 goes to LP-11 state. It is suggested that the minimum input trail be at least 3 byteclock cycles to give the design enough allowance to trim the glitches and still have enough trail bytes.
t_HS-PREPARE	User - configurable	1–99	This sets the $T_{HS-PREPARE}$ counter in terms of number of byte clock cycles. It is the time the transmitter drives the Data Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.
t_HS-ZERO	User - configurable	1–99	This sets the $T_{HS-ZERO}$ counter in terms of number of byte clock cycles. It is the time the transmitter drives the HS-0 state prior to transmitting the Sync sequence. In gear 8, the actual $T_{HS-ZERO}$ has ~2.5 cycles more than the specified value due to the register delays when converting data from parallel to serial. In gear 16, the actual value has ~3.5 cycles more.
t_CLK-PRE	User - configurable	1–99	This sets the $T_{CLK-PRE}$ counter in terms of number of byte clock cycles. This is the time that the transmitter drives the HS clock prior to any associated Data Lane beginning the transition from LP to HS mode. The actual $T_{CLK-PRE}$ has 2 more additional byteclock cycles due to register delays.
t_CLK-POST	User - configurable	1–99	This sets the $T_{CLK-POST}$ counter in terms of number of byte clock cycles. This is the time that the transmitter continues to send HS clock after the last associated Data Lane has transitioned to LP Mode. Interval is defined as the period from the end of $T_{HS-TRAIL}$ to the beginning of $T_{CLK-TRAIL}$. The actual $T_{CLK-POST}$ has one more additional byteclock cycle due to register delays.

***Note**: EoT processing is not performed in the design. Thus, one or more additional bytes from data lanes may be present after the trail. Refer to *MIPI D-PHY Specification, Appendix A.3 High Speed Receive at the Slave Side.*

To clean up these unwanted bytes, it is recommended to extend the input trail and increase the HS-SKIP value.

4. IP Generation and Evaluation

This section provides information on how to generate MIPI DSI to DSI Display Interface Bridge IP using the Diamond Clarity Designer, and how to run simulation, synthesis and hardware evaluation.

4.1. Licensing the IP

An IP-specific license is required to enable full, unrestricted use of the MIPI DSI to DSI Display Interface Bridge IP in a complete, top-level design. The MIPI DSI to DSI Display Interface Bridge IP license is available free of charge. Please request your free IP license at:

http://www.latticesemi.com/licenseprocessing/ipcore_lic_req.cfm?api=true

You may download and generate the MIPI DSI to DSI Display Interface Bridge IP and fully evaluate through functional simulation and implementation (synthesis, map, place and route) without an IP license. The DSI to DSI Bridge IP also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP that operate in hardware for a limited time (approximately four hours) without requiring an IP license. See the Hardware Evaluation section on page 20 for further details. However, the IP license is required to enable timing simulation, to open the design in Diamond EPIC tool, or to generate bitstreams that do not include the hardware evaluation timeout limitation.

4.2. Getting Started

The MIPI DSI to DSI Display Interface Bridge IP is available for download from the Lattice IP Server using the Clarity Designer tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP has been installed, the IP is available in the Clarity Design GUI as shown in Figure 4.1.

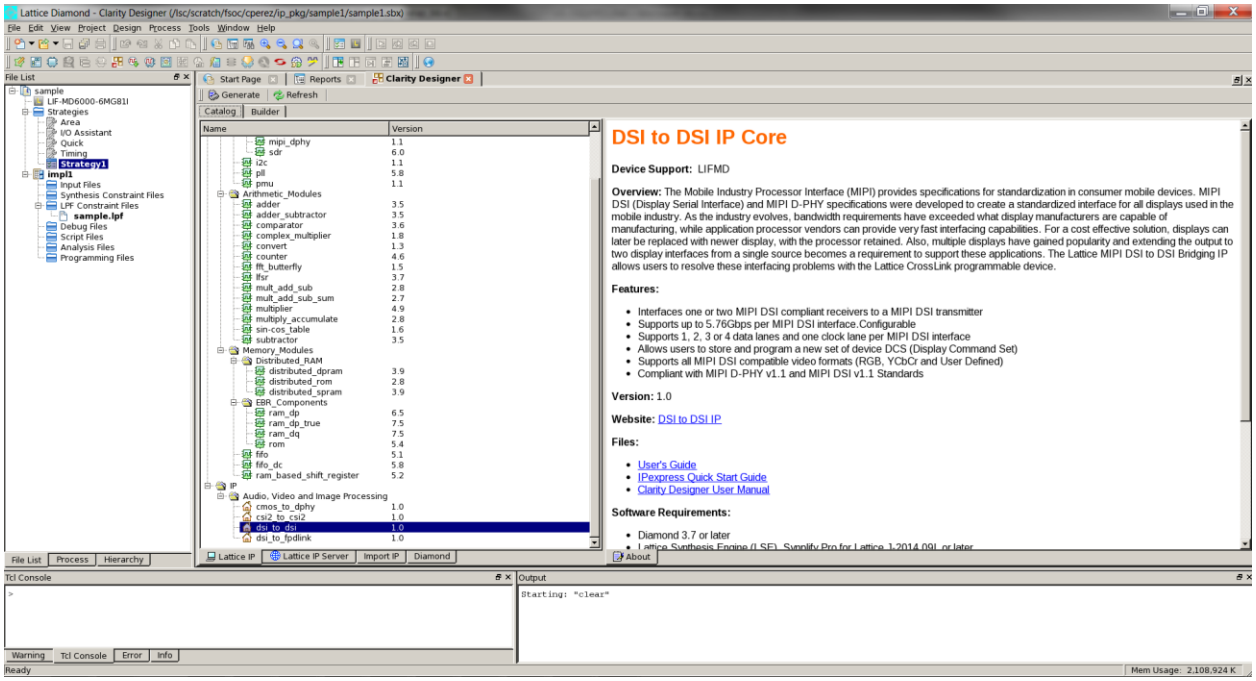

Figure 4.1. Clarity Designer Window

4.3. Generating IP in Clarity Designer

The Clarity Designer tool is used to customize modules and IPs and place them into the device's architecture. Besides configuration and generation of modules and IPs, Clarity Designer can also create a top module template in which all generated modules and IPs are instantiated.

The following describes the procedure for generating MIPI DSI to DSI Display Interface Bridge IP in Clarity Designer.

Clarity Designer can be started from the Diamond design environment.

To start Clarity Designer:

1. Create a new empty Diamond project for CrossLink family devices.
2. From the Diamond main window, choose **Tools** > **Clarity Designer**, or click  in Diamond toolbox. The Clarity Designer project dialog box is displayed.
3. Select and or fill out the following items as shown in Figure 4.2:
   - **Create new Clarity design** - Choose to create a new Clarity Design project directory in which the MIPI DSI to DSI Display Interface Bridge IP will be generated.
   - **Design Location** - Clarity Design project directory path.
   - **Design Name** - Clarity Design project name.
   - **HDL Output** - Hardware Description Language Output Format (Verilog).

   The Clarity Designer project dialog box also allows you to open an existing Clarity Designer project by selecting the following:
   - **Open Clarity design** - Open an existing Clarity Design project.
   - **Design File** - Name of existing Clarity Design project file with .sbx extension.
4. Click the **Create** button. A new Clarity Designer project is created.

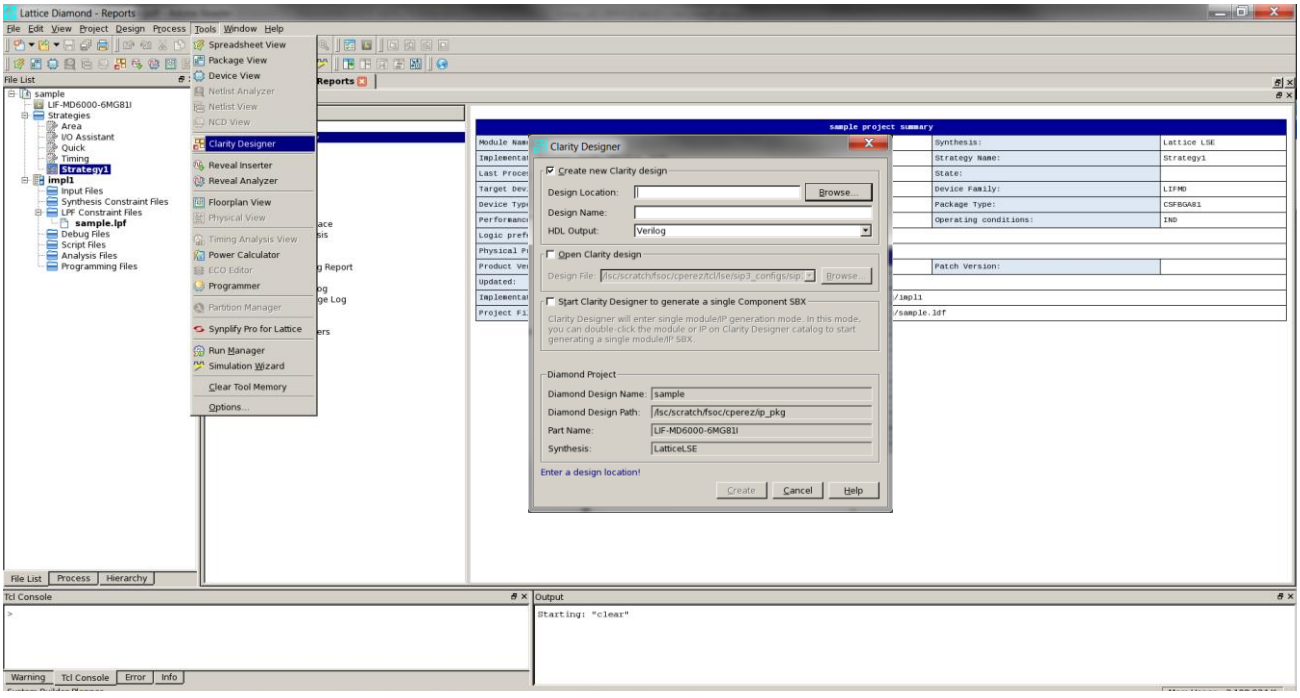

Figure 4.2. Starting Clarity Designer from Diamond Design Environment

To configure the MIPI DSI to DSI Display Interface Bridge IP in Clarity Designer:

1. Double-click **dsi_to_dsi** in the IP list of the Catalog view. The **dsi_to_dsi** dialog box is displayed as shown in Figure 4.3.

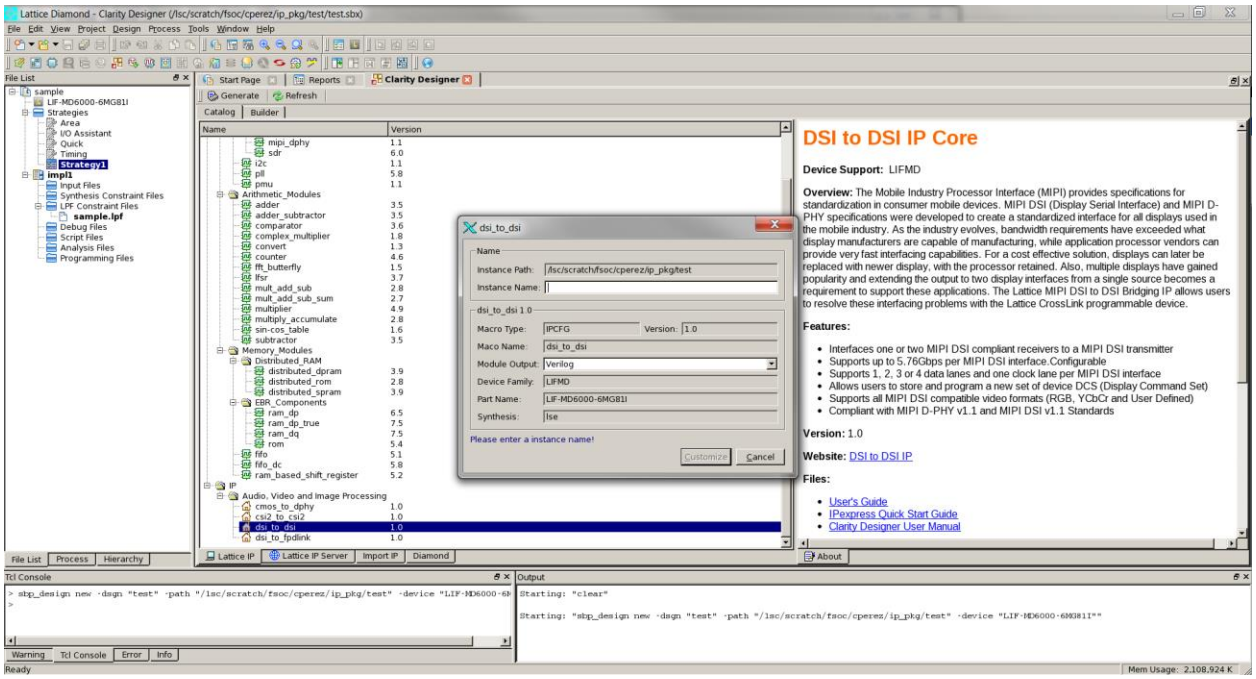

Figure 4.3. Configuring DSI to DSI Bridge IP in Clarity Designer

2. Enter the Instance Name.
3. Click the **Customize** button. An IP configuration interface is displayed as shown in Figure 4.4 – Figure 4.6. From this dialog box, you can select the IP parameter options specific to your application.

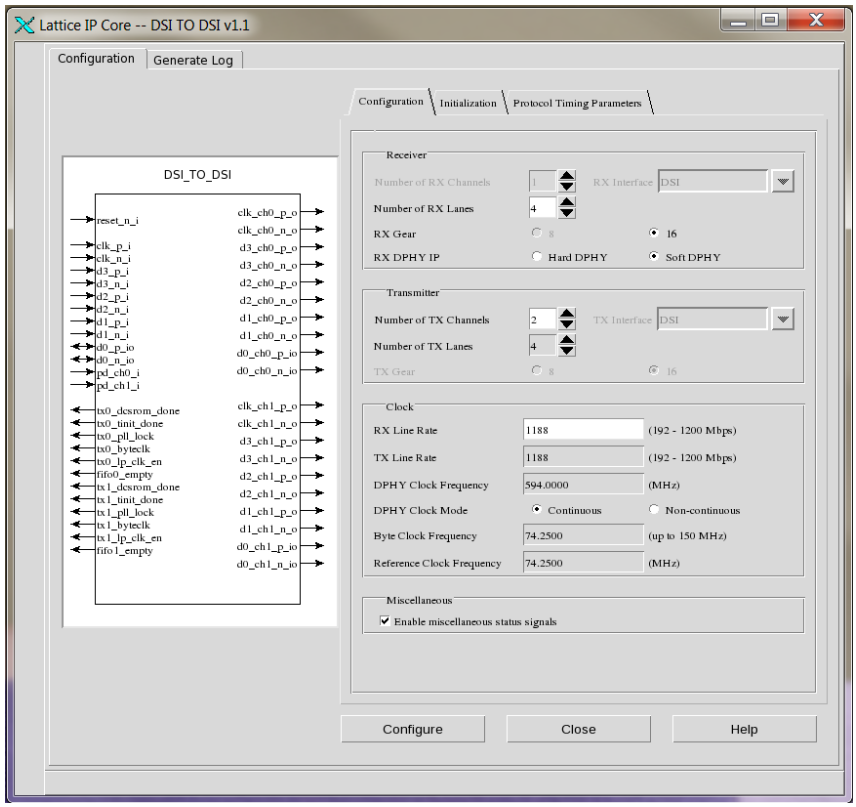

Figure 4.4. Configuration Tab in IP GUI

4. To configure Initialization parameters, click the **Initialization** tab as shown in Figure 4.5.

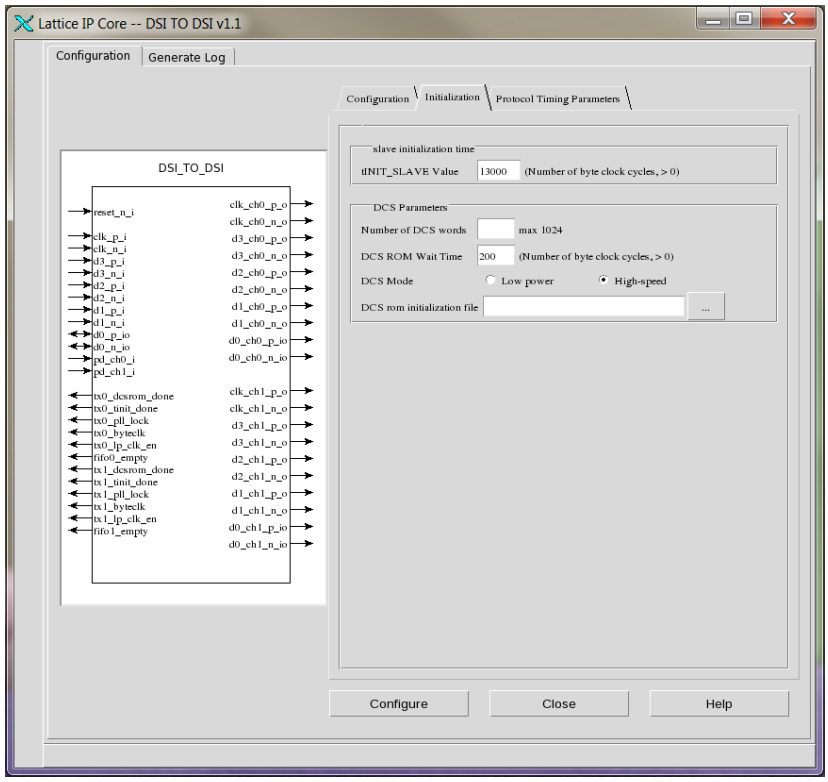

Figure 4.5. Initialization Tab in IP GUI

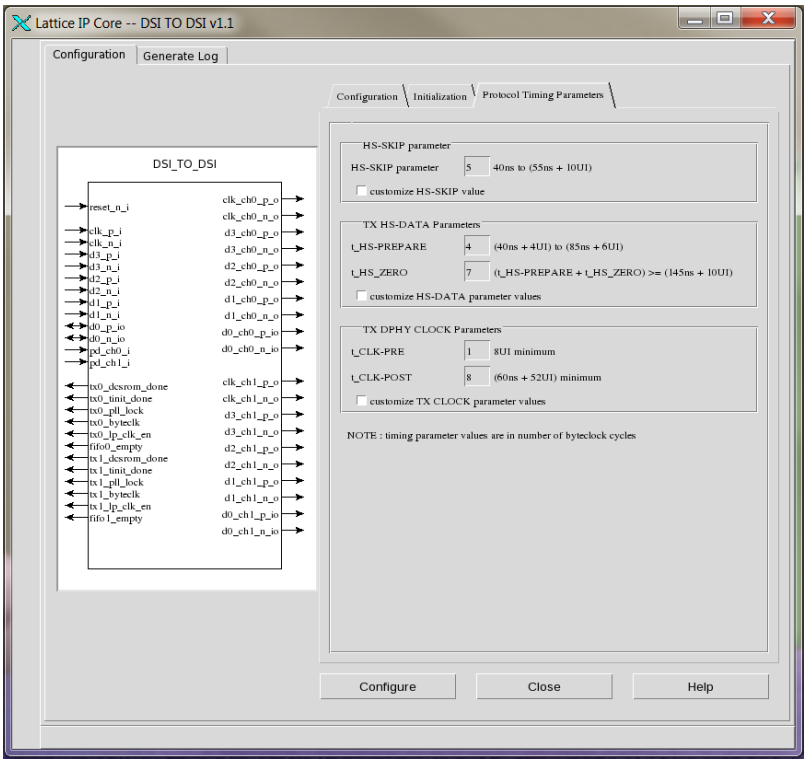

Figure 4.6. Protocol Timing Parameters Tab in IP GUI

5. Select the required parameters, and click the **Configure** button.
6. Click **Close**.
7. Click Generate in the toolbox. Clarity Designer generates all the IPs and modules, and creates a top module to wrap them.

For detailed instructions on how to use the Clarity Designer, refer to the Lattice Diamond software user guide.

4.4. Generated IP Directory Structure and Files

The directory structure of generated IP files is shown in Figure 4.7.

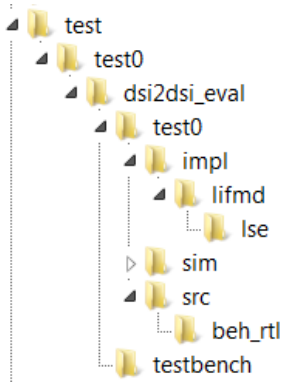

Figure 4.7. DSI to DSI Bridge IP Directory Structure

The design flow for the IP created with Clarity Designer uses post-synthesized modules (NGO) of IP core modules for synthesis and uses protected models for simulation. The post-synthesized modules are customized when you configure the IP and created automatically when the IP is generated. The protected models are common to all configurations. Other files are also provided to enable functional simulation and implementation.

Table 4.1 provides a list of key files and directories created by Clarity Designer with details on where they are located and how they are used.

Table 4.1. List of Generated Files

File	Description
<instance_name>.v	Verilog top-level module of MIP D-PHY to CMOS IP used for both synthesis and simulation
<instance_name>_*.v	Verilog submodules for simulation. Files that do not have equivalent black box modules are also used for synthesis.
<instance_name>_inst.v/vhd	Template for instantiating the design in another user-created top module
<instance_name>_*_beh.v	Protected Verilog models for simulation
<instance_name>_*_bb.v	Verilog black box modules for synthesis
<instance_name>_*.ngo	GUI configured and synthesized modules for synthesis
<instance_name>_params.v	Verilog parameters file which contains required compiler directives to successfully configure IP during synthesis and simulation
<instance_name>.lpc	Lattice Parameters Configuration file. This file records all the IP configuration options set through Clarity Designer. It is used by IP generation script to generate configuration-specific IP. It is also used to reload parameter settings in the IP GUI in Clarity Designer when it is being reconfigured.

All IP files are generated inside `\<project_dir>` directory (test in Figure 4.7). The `\<project_dir>` is `<design_location>\<design_name>\<instance_name>`, see the Generating IP in Clarity Designer section on page 11. A separate `\<project_dir>` is created each time DSI to DSI Bridge IP is created with a different IP instance name.

The \dsi2dsi_eval and subdirectories provide files supporting push-button IP evaluation through functional simulations, design implementation (synthesis, map, place and route) and hardware evaluation. Inside \dsi2dsi_eval is \<instance_name> folder (test0 in Figure 4.7) which contains protected behavioral files in \<instance_name>\src\beh_rtl and a pre-built Diamond project in \<instance_name>\impl\lifmd\<synthesis_tool>. The <instance_name> is the IP instance name specified by the user in Clarity Designer. The simulation part of user evaluation provides testbench and test cases supporting RTL simulation for Active-HDL simulator under \<project_dir>\testbench. Separate directories located at \<project_dir>\dsi2dsi_eval\<instance_name>\sim\aldec are provided and contain specific pre-built simulation script files. See the Running Functional Simulation section below for details.

4.5. Running Functional Simulation

To run simulations using Active HDL follow these steps:

1. Modify the "do" file located in \<project_dir>\dsi2dsi_eval\<instance_name>\sim\aldec\
   a. Specify working directory (sim_working_folder), for example
      `set sim_working_folder "C:/my_design"`
   b. Specify workspace name that will be created in working directory, for example
      `set workspace_name "design_space"`
   c. Specify design name, for example
      `set design_name "DesignA"`
   d. Specify design path where the IP Core generated using Clarity Designer is located, for example
      `set design_path "C:/my_designs/DesignA"`
   e. Specify design instance name (same as the instance name specified in Clarity Designer), for example
      `set design_inst "DesignA_inst"`
   f. Specify Lattice Diamond Primitive path (diamond_dir) to where it is installed, for example
      `set diamond_dir "C:/lscc/diamond/3.8_x64"`
2. Update testbench parameters to customize data size, clock and/or other settings. See Table 4.2 and Table 4.3 for the list of valid testbench compiler directives.
3. From the **Tools** menu, select **Active-HDL**.
4. In the Active-HDL window, under the **Tools** tab, select **Execute Macro**.
5. Select the "do" file <\project_dir>\dsi2dsi_eval\<instance_name>\sim\aldec\dsi2dsi.do
6. Click **OK**.
7. Wait for the simulation to finish.

Table 4.2 is a list of testbench directives which can be modified by setting the define in the vlog command in the dsi_2_dsi.do file.

Example:
vlog \
+define+NUM_FRAMES=60 \
+define+NUM_LINES=1080 \

Table 4.2. Testbench Directives

Directive	Description
PLL_DCS_DURATION	Used when miscellaneous signals are off (for example debug output ports for PLL lock and DCS ROM done are not included in the generated design). This directive is used to set the duration (in ps) of PLL and DCS ROM done before the D-PHY model in the testbench transmits input data to the design. For example +define+PLL_DCS_DURATION=140000000
PD_CH0	Used to control the power-down pin of Tx channel 0 0 – Channel 0 is enabled 1 – Channel 0 is enabled
PD_CH1	Used to control the power-down pin of Tx channel 1 0 – Channel 1 is enabled 1 – Channel 1 is enabled
NUM_FRAMES	Used to set the number of video frames
NUM_LINES	Used to set the number of lines per frame
FRAME_LPM_DELAY	Used to set the low-power mode delay between frames (in ps)
VIDEO_DATA_TYPE	Video data type, in decimal value. For example, for RGB888 (0x3E), +define+VIDEO_DATA_TYPE=62
VACT_PAYLOAD	Number of bytes of active pixels per line
HSA_PAYLOAD	Number of bytes of Horizontal Sync Active Payload (used for Non-burst sync pulse)
BLLP_PAYLOAD	Number of bytes of BLLP Payload (used for HS data blanking)
HBP_PAYLOAD	Number of bytes of Horizontal Back Porch Payload (used for HS data blanking, and in LP blanking for Non-burst sync pulse mode)
HFP_PAYLOAD	Number of bytes of Horizontal Front Porch Payload (used for HS data blanking, and in LP blanking for Non-burst sync pulse mode)
VSA_LINES	Number of Vertical Sync Active Lines
VBP_LINES	Number of Vertical Back Porch Lines
VFP_LINES	Number of Vertical Front Porch Lines
EOTP_ENABLE	Used to enable/disable transmission of End-of-Transmit packet 0 – EOTP packet is disabled 1 – EOTP packet is enabled
LPS_BLLP_DURATION	Used to set the duration (in ps) for BLLP low-power state (used for LP blanking)
LPS_HBP_DURATION	Used to set the duration (in ps) for Horizontal Back Porch low-power state (used for LP blanking in Non-burst sync events and Burst mode)
LPS_HFP_DURATION	Used to set the duration (in ps) for Horizontal Front Porch low-power state (used for LP blanking in Non-burst sync events and Burst mode)
VIRTUAL_CHANNEL	Used to set the virtual channel number
NON_BURST_SYNC_EVENTS BURST_MODE NON_BURST_SYNC_PULSE	Video Mode Types. One of the following video mode types must be defined. The default mode used by the testbench is Non-burst sync pulse. For example add +define+BURST_MODE in vlog command to enable Burst Mode
TRAIL_GLITCH_ENABLE	User can enable transmitting of glitches at the end of the HS trail to model tREOT by defining TRAIL_GLITCH_ENABLE (+define+TRAIL_GLITCH_ENABLE)
TRAIL_GLITCH_INTERVAL	Used to set interval of the trail glitches if TRAIL_GLITCH_ENABLE is defined
DPHY_DEBUG_ON	Used to enable or disable debug messages 0 – Debug messages are disabled 1 – Debug messages are enabled
FRAME_LPM_DELAY	Used to set the low-power mode delay between frames (in ps)

Table 4.2. Testbench Directives *(Continued)*

Directive	Description
DPHY_CLK_PERIOD	By default, the testbench automatically calculates the D-PHY clock period, but the user can change/override the clock period by defining the directive in vlog (in ps). For example +define+DPHY_CLK_PERIOD=1684
REFCLK_PERIOD	By default, the testbench automatically calculates the reference clock period for Non-Continuous Rx Clock Mode, but the user can change/override the clock period by defining the directive in vlog (in ps). For example +define+REFCLK_PERIOD=6736
HS_BLANKING	By default, low-power blanking is used during HS_LP mode. To use HS data blanking, HS_BLANKING may be added in the list of defines (+define+HS_BLANKING)
LP_BLANKING	By default, HS data blanking is used during HS_ONLY mode. To use low-power blanking, LP_BLANKING may be added in the list of defines (+define+LP_BLANKING)

The testbench has default setting for the D-PHY timing parameters listed in Table 4.3.
If the user would like to modify the D-PHY timing parameters, the user can set the following directives.

Table 4.3. Testbench Directives for D-PHY Timing Parameters

Directive	Description
DPHY_LPX	Used to set T-LPX (in ps)
DPHY_CLK_PREPARE	Used to set T-CLK-PREPARE (in ps)
DPHY_CLK_ZERO	Used to set T-CLK-ZERO (in ps)
DPHY_CLK_PRE	Used to set T-CLK-PRE (in ps)
DPHY_CLK_POST	Used to set T-CLK-POST (in ps)
DPHY_CLK_TRAIL	Used to set T-CLK-TRAIL (in ps)
DPHY_HS_PREPARE	Used to set T-HS-PREPARE (in ps)
DPHY_HS_ZERO	Used to set T-HS-ZERO (in ps)
DPHY_HS_TRAIL	Used to set T-HS-TRAIL (in ps)

Refer to *MIPI D-PHY Specification version 1.1, Table 14* for information regarding D-PHY timing requirements.

4.6. Simulation Strategies

This section describes the simulation environment which demonstrates basic DSI to DSI Bridge IP functionality. Figure 4.8 shows a block diagram of the simulation environment.

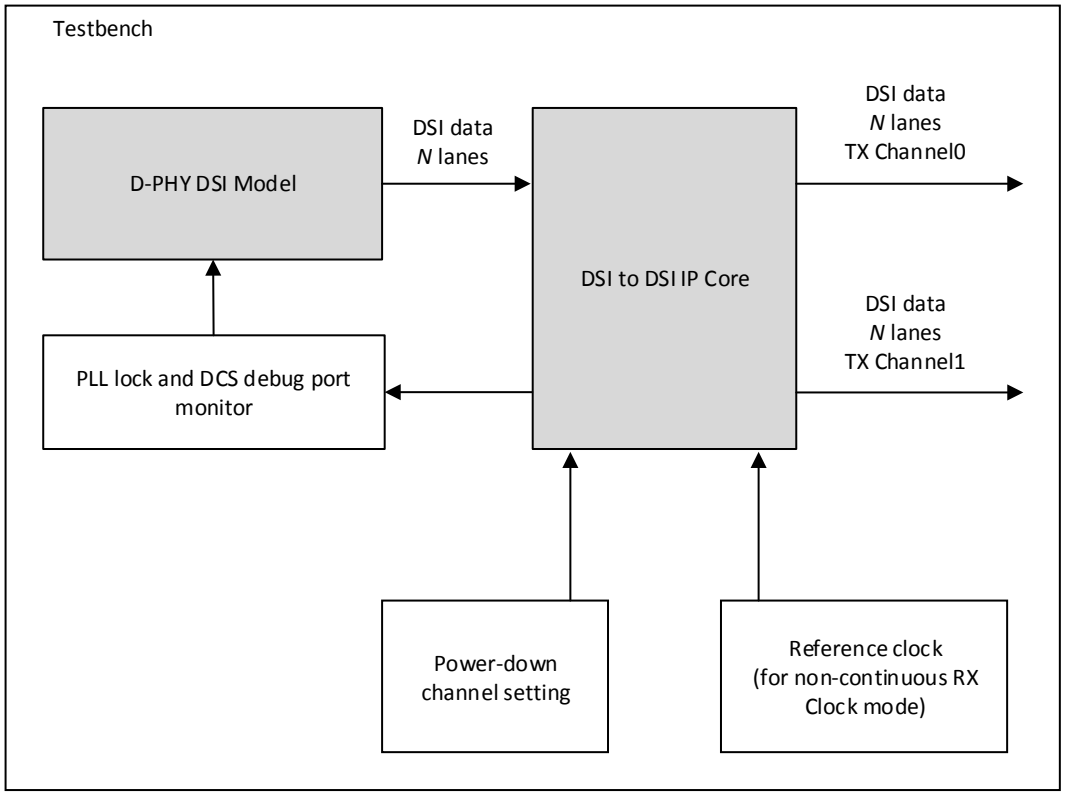

Figure 4.8. Simulation Environment Block Diagram

4.7. Simulation Environment

The simulation environment is made up of the D-PHY DSI model instance connected to the IP core instance in the testbench. The D-PHY DSI model is configured based on the IP core configurations and testbench configurations. The testbench can be configured to set the power-down channel setting for the IP core Tx channels. The testbench also transmits reference clock to the IP core if the clock mode is non-continuous. If miscellaneous signals such as PLL lock and DCS done debug ports are included in the IP core, the testbench monitors assertion of these signals before sending the DSI video data to the IP core. Figure 4.9 shows an example simulation where PLL lock and DCS done debug ports are included.

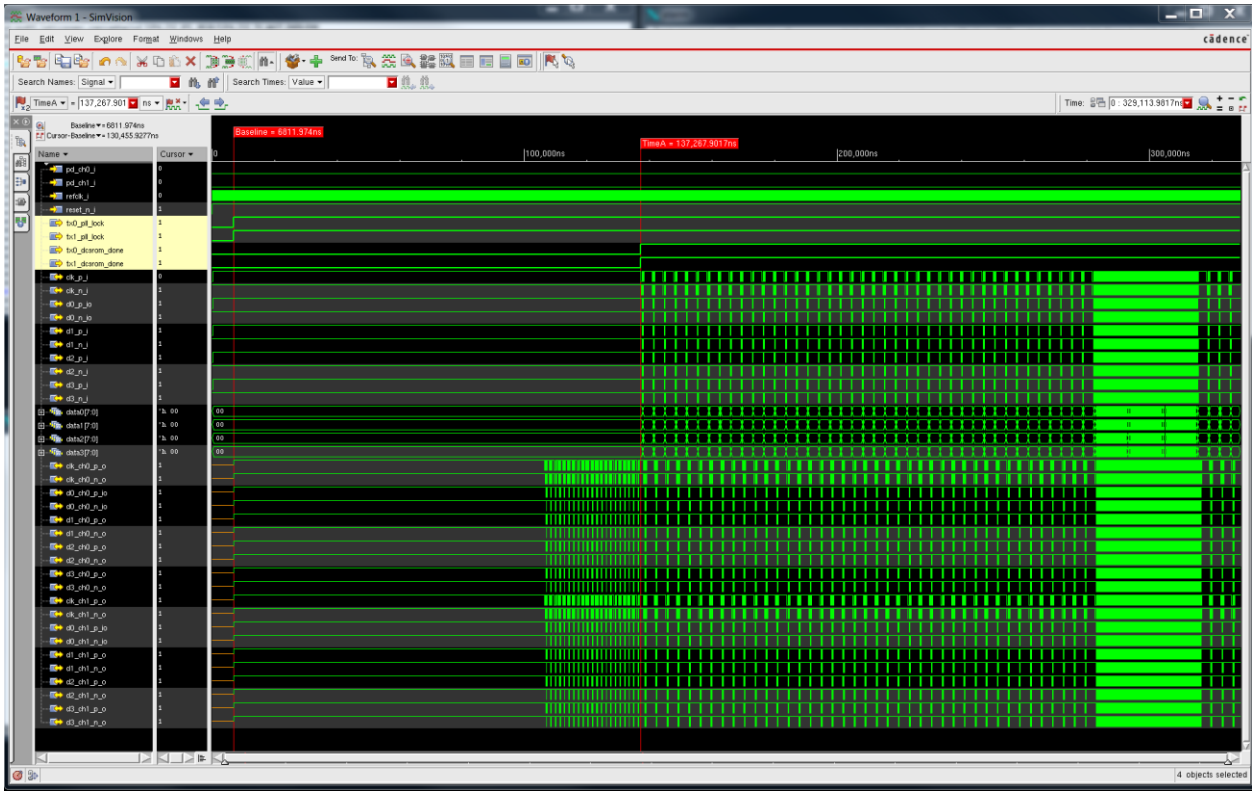

Figure 4.9. PLL Lock and DCS done Miscellaneous Signals

The video data transmitted by the D-PHY DSI model can viewed in the waveform, see Figure 4.10:

- tb.dphy_ch0.data0 – refers to the data bytes transmitted in D-PHY data lane 0
- tb.dphy_ch0.data1 – refers to the data bytes transmitted in D-PHY data lane 1
- tb.dphy_ch0.data2 – refers to the data bytes transmitted in D-PHY data lane 2
- tb.dphy_ch0.data3 – refers to the data bytes transmitted in D-PHY data lane 3

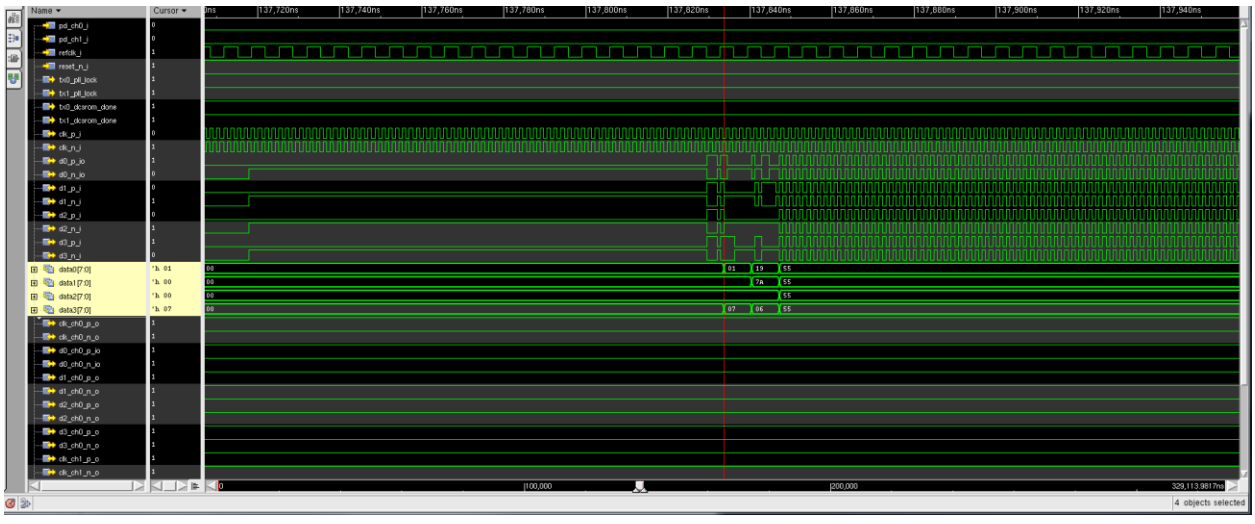

Figure 4.10. D-PHY DSI Model Video Data

4.8. Instantiating the IP

The core modules of the 1:2 MIPI DSI Display Interface Bridge IP are synthesized and provided in NGO format with black box Verilog source files for synthesis. A Verilog source file named `<instance_name>_dsi_2_dsi_ip.v` instantiates the black box of core modules. The top-level file `<instance_name>.v` instantiates `<instance_name>_dsi_2_dsi_ip.v`.

The IP instances do not need to be instantiated one by one manually. The top-level file and the other Verilog source files are provided in `\<project_dir>`. These files are refreshed each time the IP is regenerated.

The MIPI DSI Display Interface Bridge Soft IP is intended as a complete standalone solution. However, a Verilog instance template `<instancename>_inst.v` or VHDL instance template `<instancename>_inst.vhd` is also generated as a guide if the design is to be included in another top level module.

4.9. Synthesizing and Implementing the IP

In Clarity Designer, the Clarity Designer project file (.sbx) is added to Lattice Diamond as a source file after IP is generated. All required Verilog source files for implementation are invoked automatically. The IP can be directly synthesized, mapped and placed/routed in the Diamond design environment after the IP is generated. Note that default Diamond strategy (.sty) and default Diamond preference file (.lpf) are used. When using the .sbx approach, import the recommended strategy and preferences from `\<project_dir>\dsi2dsi_eval\<instancename>\impl\lifmd\lse` or `\<project_dir>\dsi2dsi_eval\<instancename>\impl\lifmd\synplify` directories. All required files are invoked automatically. The design can be directly synthesized, mapped, placed and routed (PAR) in the Diamond design environment after the cores are generated.

Push-button implementation of this top-level design with either Lattice Synthesis Engine (LSE) or Synopsys Synplify Pro RTL synthesis is supported via the Diamond project file `<instancename>_top.ldf` located in `\<project_dir>\dsi2dsi_eval\<instancename>\impl\lifmd\<synthesis_tool>` directory.

To use the pre-built Diamond project file:

1. Choose **File** > **Open** > **Project**.
2. In the **Open Project** dialog box browse to `\<project_dir>\dsi2dsi_eval\<instancename>\impl\lifmd\<synthesis_tool>`
3. Select and open `<instancename>_top.ldf`. At this point, all of the files needed to support top-level synthesis and implementation are imported to the project.
4. Select the **Process** tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI flow.

4.10. Hardware Evaluation

The MIPI DSI to DSI Display Interface Bridge IP supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP that operate in hardware for a limited period of time (approximately four hours) without requiring the request of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

4.10.1. Enabling Hardware Evaluation in Diamond

If using LSE, choose **Project > Active Strategy > LSE Settings**. If using Synplify Pro, choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled or disabled in the **Strategy** dialog box. It is enabled by default.

4.11. Updating/Regenerating the IP

The Clarity Designer allows you to update the local IPs from the Lattice IP server. The updated IP can be used to regenerate the IP instance in the design. To change the parameters of the IP used in the design, the IP must also be regenerated.

4.11.1. Regenerating an IP in Clarity Designer

To regenerate IP in Clarity Designer:

1. In the **Builder** tab, right-click the IP instance to be regenerated and select **Config** in the menu as shown in Figure 4.11.

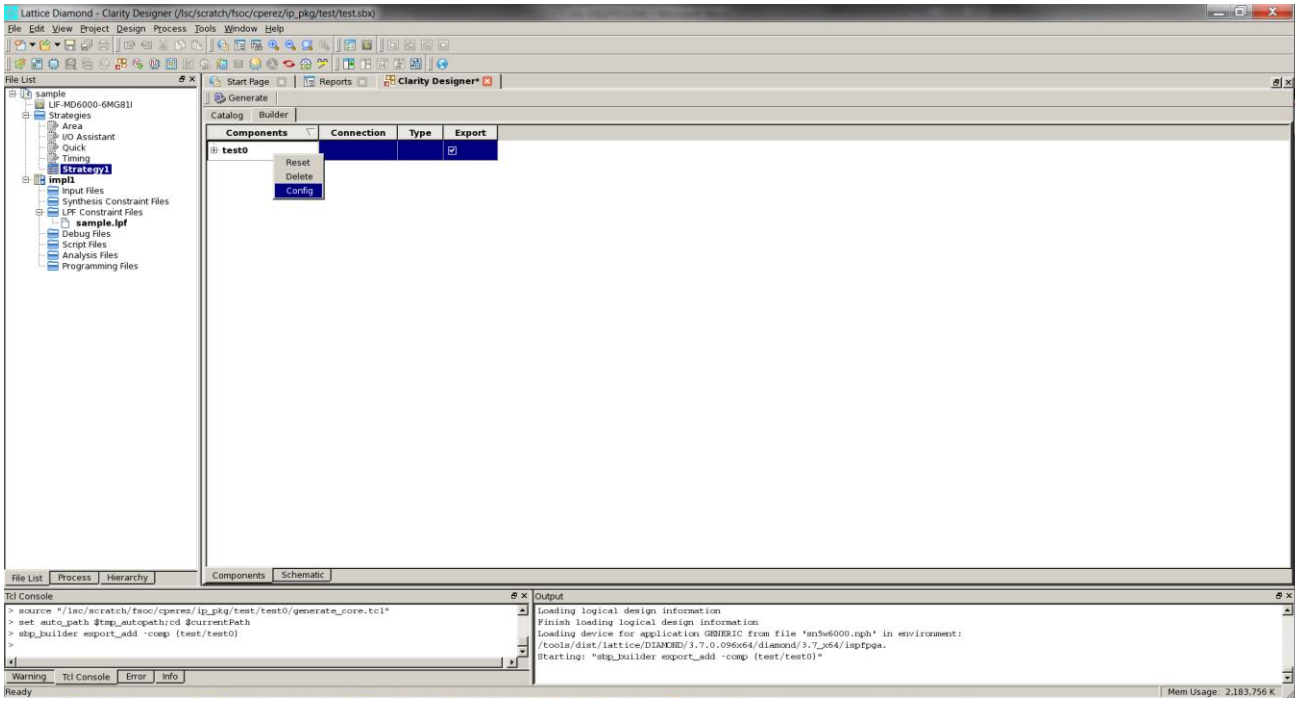

Figure 4.11. Regenerating IP in Clarity Designer

2. The IP Configuration GUI is displayed. Change the parameters as required and click the **Configure** button.
3. Click 

in the toolbox. Clarity Designer regenerates all the IP instances which are reconfigured.

References

For more information about CrossLink devices, refer to FPGA-DS-02007, CrossLink Family Data Sheet.

For further information on interface standards refer to:

- MIPI Alliance Specification for D-PHY, version 1.1, November 7, 2011, www.mipi.org
- MIPI Alliance Specification for Display Serial Interface, version 1.1, November 22, 2011, www.mipi.org

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Appendix A. Resource Utilization

Table A.1 lists resource utilization for Lattice CrossLink FPGAs using the 1:2 MIPI DSI Display Interface Bridge IP. The performance and utilization data target an LIF-MD6000-6MG81I device with –6 speed grade using Lattice Diamond 3.8 and Lattice Synthesis Engine. Performance may vary when using a different software version or targeting a different device density or speed grade within the CrossLink family. The values of f_{MAX} shown are based on byte clock. The Target f_{MAX} column shows target byte clock frequency for each configuration.

Table A.1. Resource Utilization

IP User-Configurable Parameters	Slices	LUTs	Registers	sysMEM EBRs	Actual f_{MAX} (MHz)	Target f_{MAX} (MHz)
Continuous Rx Clock Mode, 4-lane, gear 16	2967	5202	1983	20	94.455	74.25
Continuous Rx Clock Mode, 4-lane, gear 8	2967	5790	1591	8	153.492	148.50
Non-continuous Rx Clock Mode, 4-lane, gear 16	2967	5135	1991	20	98.377	74.25
Non-continuous Rx Clock Mode, 4-lane, gear 8	2967	5811	1600	8	156.470	148.50

Appendix B. What is Not Supported

The IP does not support the following features:

- Cycling Redundancy Check (CRC) and Error Correction Code (ECC) checking and generation
- Bidirectional Communication
- Low-Level Protocol Error reporting
- Protocol Watchdog Timers
- End of Transmit (EoT) processing

The 2:1 MIPI DSI Display Bridge IP has the following design limitations:

- Minimum duration of MIPI D-PHY low-power states (tLPX) should be at least two times the byte clock period
- Maximum fabric speed is 150 MHz
- Maximum byte clock frequency is 112 MHz, lower than maximum fabric speed due to heavy logic inside core modules
- Video VSYNC and HSYNC outputs solely depend on MIPI DSI VSYNC/HSYNC start and end short packets. For displays that require strict timing, the design needs to be modified to have additional control
- The PLL which is used by the MIPI D-PHY Tx to generate the D-PHY clocks has to be programmed such that the frequency after the input divider (PLL_N) ranges from 24 MHz to 30 MHz (that is $24 \leq (CLKREF/PLL\_N) \leq 30$), otherwise, the D-PHY PLL will not lock. The CLKREF corresponds to the input reference clock or the byteclock. Due to this requirement, there are frequency holes in the frequency range supported by the design.

Appendix C. Initializing the DCS ROM

Display Command Set (DCS) initialization is used to configure the command registers of a DSI-compliant display. The bridge has an option to perform this in high-speed or in low-power mode.

In either DCS mode, the number of entries must correspond to the Number of DCS Words indicated in the GUI.

There should be no empty lines within the text file. Comments within the file are not supported.

Low-Power Mode

To initialize the DCS ROM in low-power mode, the input file must contain one byte of data in each line, in hex format. Figure C.1. shows the sample entries.

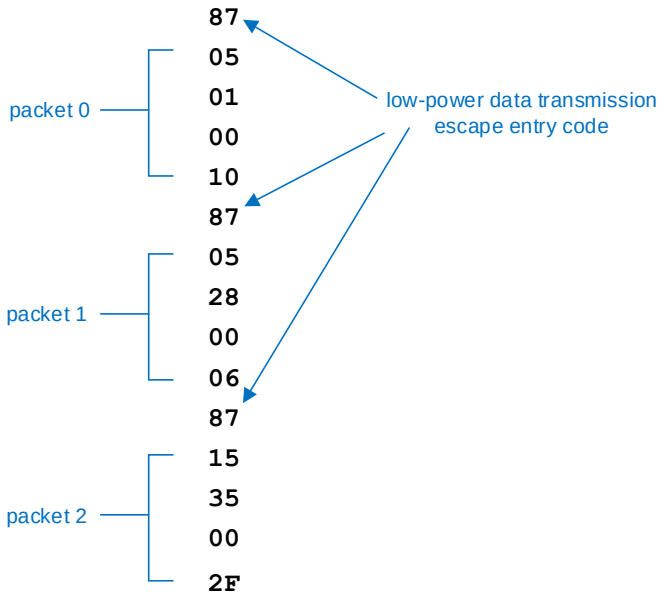

Figure C.1. DCS ROM for DCS Low-Power Mode

The 8'h87 byte indicates the start of a new packet. In this example, the DCS Controller breaks down the DCS words into 3 packets. The last entry should be the last valid byte. DCS Word Count in this example is 15.

High-Speed Mode

When the DCS ROM initialization is in high-speed mode, the interval between high-speed transmissions may be set through the DCS ROM Wait Time parameter. Multiple packets may be concatenated to reduce overhead of frequent switching between low-power state and high-speed mode.

The entries within the input file should be in the following format:

```
<trail bit indicator><DCS byte lane3><DCS byte lane2><DCS byte lane1><DCS byte lane0>
```

For each high-speed transmission, each lane must start with the SoT pattern 8'hB8, and the last word should be made up of complete trail bytes with the trail indicator bit set to 1. The design checks this trail bit indicator to determine the end of the high-speed transmission.

Sample DCS for Gear 8Figure C.2 shows the sample entries for the DCS initialization file of a 4-lane, gear 8 configuration.

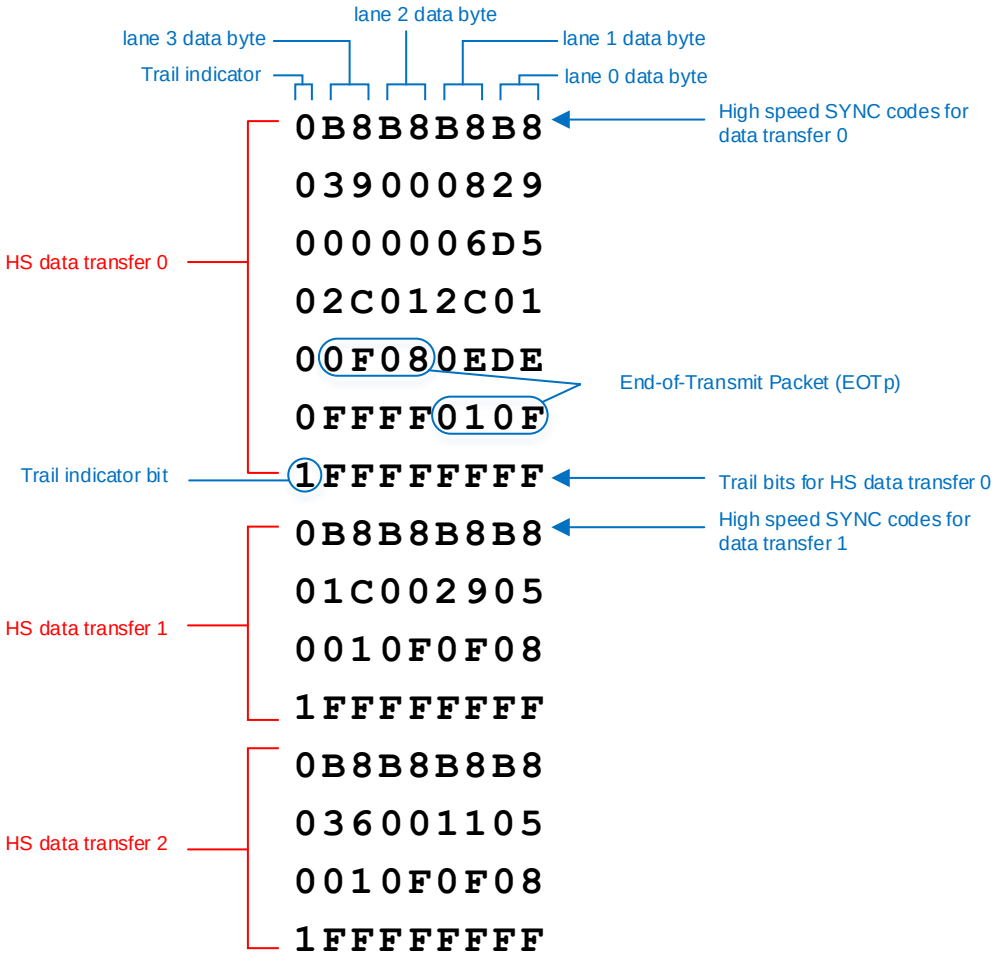

Figure C.2. Sample DCS ROM for x4 Gear 8 DCS High-Speed Mode

In this example, an End-of-Transmit packet is sent after each DCS packets. The last word after each high-speed transmission must be made up completely of trail bits. The DCS Word Count in this example is 15.

Sample DCS for Gear 16

The byte order of DCS words for gear16 configuration should follow the order described in Figure 1.1 on page 5.

Figure C.3 shows the sample entries for 4-lane gear 16 configuration:

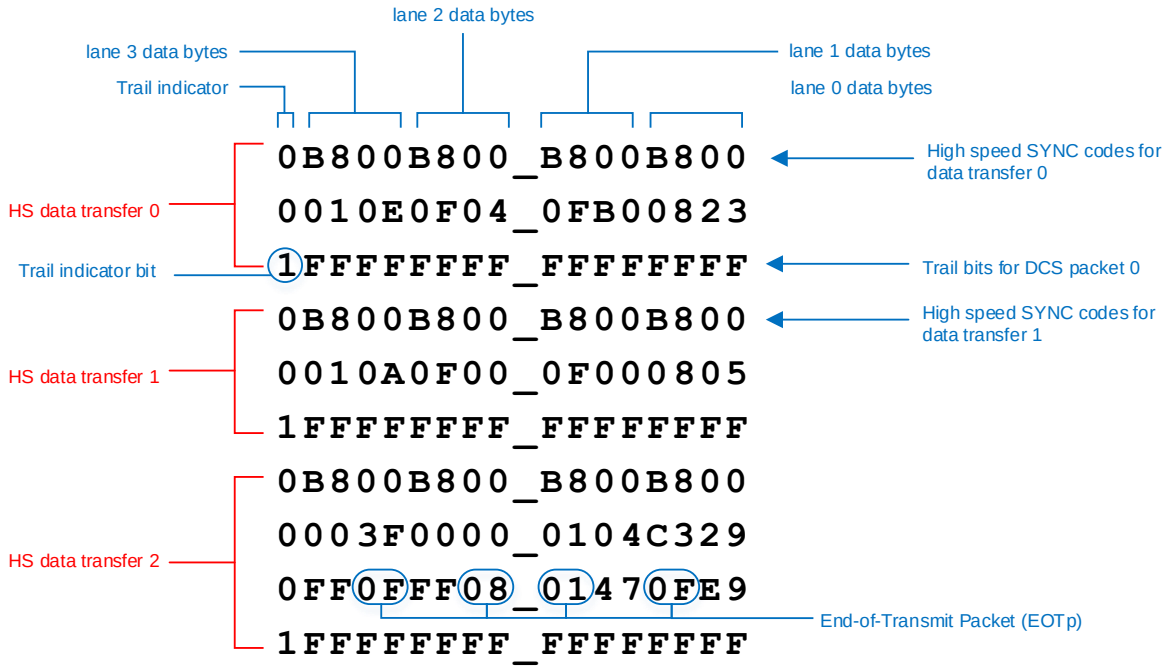

Figure C.3. Sample DCS ROM for x4 Gear 16 DCS High-Speed Mode

This example contains three DCS packets concatenated with EoTp at the end of each. The first word for each lane contains the MIPI D-PHY high-speed synchronization sequence 8'hB8, padded with zeros at the start. The zero padding is used for alignment purposes only. These may be removed, but the data bytes should be adjusted accordingly. The DCS Word Count in this example is 10.

Sample DCS ROM files are also available in the *dsi_to_dsi_v1.1/samples* folder in the IP installation directory.

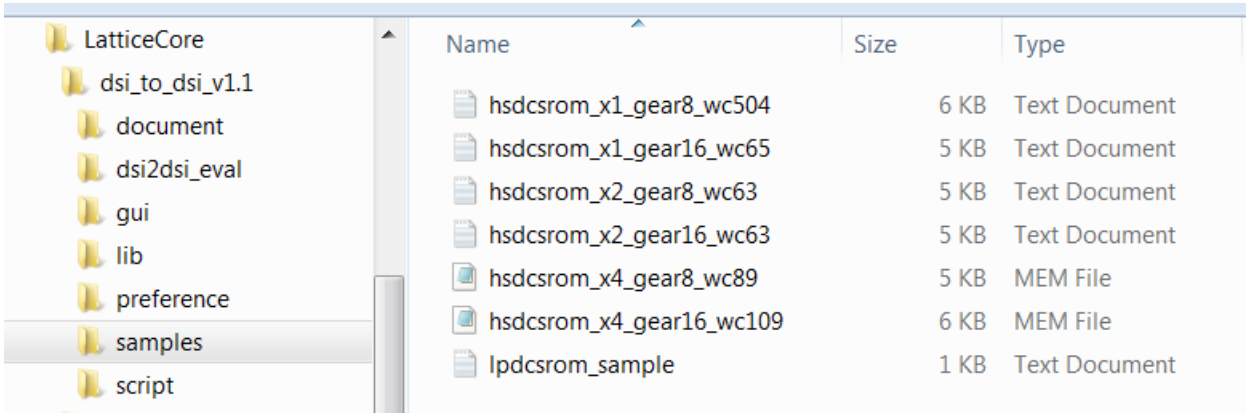

Figure C.4. Directory Containing the Sample DCS ROM Initialization Files

Revision History

Date	Version	Change Summary
July 2016	1.1	• Updated document numbers • Added configurability of Tx D-PHY parameters • Updated Synplify Pro version in Table 1.1. MIPI DSI to DSI Display Interface Bridge IP Quick Facts • Updated Figure 1.1. Data Ordering for a Gear 16 x4 Configuration • Added parameters t_HS-PREPARE, t_HS-ZERO, t_CLK-PRE, and t_CLK-POST to Table 3.1. 2-bit DSI to DSI Parameter Settings • Updated Figure 4.4. Configuration Tab in IP GUI, Figure 4.5. Initialization Tab in IP GUI, and Figure 4.6. Protocol Timing Parameters Tab in IP GUI • Updated Running Functional Simulation section • Updated Appendix C Initializing the DCS ROM
May 2016	1.0	Initial release.

7th Floor, 111 SW 5th Avenue
Portland, OR 97204, USA
T 503.268.8000
www.latticesemi.com