# CMOS to MIPI D-PHY Interface Bridge Soft IP

**Supporting MIPI CSI-2 and MIPI DSI for Image Sensor Interface and Displays**

# User Guide

FPGA-IPUG-02007-1.4

April 2019

# Contents

# Figures



# Tables

# 1. Introduction

The Mobile Industry Processor Interface (MIPI®) D-PHY was developed primarily to support camera and display interconnections in mobile devices, and it has become the industry's primary high-speed PHY solution for these applications in smartphones today. It is typically used in conjunction with MIPI Camera Serial Interface-2 (CSI-2) and MIPI Display Serial Interface (DSI) protocol specifications. It meets the demanding requirements of low-power, low noise generation, and high noise immunity that mobile phone designs demand.

MIPI D-PHY is a practical PHY for typical camera and display applications. It is designed to replace traditional parallel bus based on LVCMOS or LVDS. However, many processors and displays/cameras still use an RGB, CMOS, or MIPI Display Pixel Interface (DPI) as interface.

A bridge is often required to connect a processor with an RGB interface to a display with a MIPI DSI interface or a camera with a CMOS interface to a processor with CSI-2 interface. The Lattice Semiconductor CMOS to MIPI D-PHY Interface Bridge IP provides this conversion for Lattice Semiconductor CrossLink™ devices. This is useful for wearable, tablet, human machine interfacing, medical equipment and many other applications.

This user guide is for CMOS to MIPI D-PHY Interface Bridge Soft IP design version 1.x.

## 1.1. Quick Facts

Table 1.1 provides quick facts about the CMOS to MIPI D-PHY Interface Bridge Soft IP for CrossLink device.

**Table 1.1. CMOS to MIPI D-PHY Interface Bridge Soft IP Quick Facts**

| | | CMOS to MIPI D-PHY Interface Bridge Soft IP Configuration | | |
|---|---|---|---|---|
| | | **4-Lane RGB888 Configuration** | **2-Lane RGB888 Configuration** | **1-Lane RGB888 Configuration** |
| **IP Requirements** | FPGA Families Supported | CrossLink | | |
| **Resource Utilization** | Target Device | LIF-MD6000-6MG81I | | |
| | Data Path Width | 24-bit input, serial output | | |
| | LUTs | 1719 | 3085 | 1924 |
| | sysMEM™ EBRs | 8 | 4 | 2 |
| | Registers | 1017 | 679 | 590 |
| | HW MIPI Block | 1 | 1 | 1 |
| | Programmable I/O | 30 | 30 | 30 |
| **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 CMOS to MIPI D-PHY Interface Bridge IP are:

- Compliant with MIPI DSI v1.1, MIPI CSI-2 v1.1 and MIPI D-PHY v1.1 specifications
- Supports MIPI DSI and MIPI CSI interfacing up to 6 Gb/s
- Supports 1, 2, or 4 MIPI D-PHY data lanes
- MIPI Display Command Set (DCS) controller to program the display for DSI Interface in either HS or HSLP mode
- Supports Non-Burst Mode with Sync Events data transaction
- Supports RGB888, RGB666, RAW8, RAW10, RAW12, YUV420/YUV422 8/10-bit video formats

## 1.3. Conventions

### 1.3.1. Nomenclature

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

### 1.3.2. Data Ordering and Data Types

The most significant bit within the pixel data is the highest index. D-PHY outputs the least significant bit first.

### 1.3.3. Signal Names

Signal names that end with:

- *_n* are active low
- *_i* are input signals

  Some signals are declared as bidirectional (I/O) but are only used as input. Hence, *_i* identifier is used.
- *_o* are output signals

  Some signals are declared as bidirectional (I/O) but are only used as output. Hence, *_o* identifier is used.
- *_io* are bidirectional signals

# 2. Functional Description

The CMOS to MIPI D-PHY Interface Bridge Soft IP converts a standard parallel video interface into either DSI or CSI-2 byte packets. The input interface for the design consists of a pixel bus (RGB888, RGB666), vertical and horizontal sync flags, a data enable and a clock for DSI and pixel bus (RGB888, RAW8, RAW10, RAW12, and YUV420/YUV422 8/10-bit), frame and line valid flags and a clock for CSI-2.

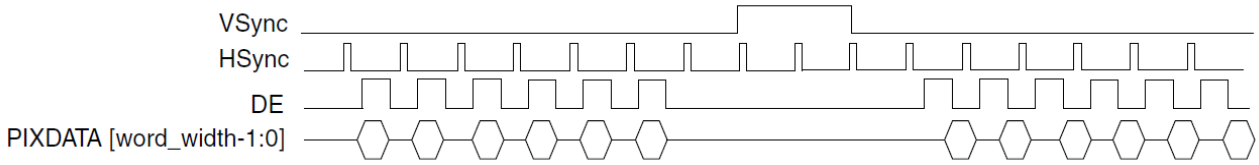


**Figure 2.1. Display Parallel Input Bus Waveform**

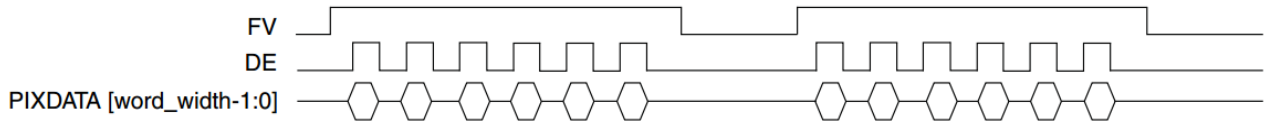


**Figure 2.2. Camera Sensor Parallel Input Bus Waveform**

This parallel bus in Figure 2.1 and Figure 2.2 is converted to the appropriate DSI or CSI-2 output format. The DSI/CSI-2 output serializes HS (High Speed) data and controls LP (Low Power) data and transfers them through MIPI D-PHY IP. MIPI D-PHY also has a maximum of 5 lanes per channel. It consists of 1 clock lane and 4 data lanes. The maximum D-PHY data rate per lane is 1.5 Gb/s. Serialized MIPI D-PHY data are transmitted to the connected display via Display Serial Interface protocol.

Data Lane 0 can be used to configure the display with DCS (Display Command Set) commands when DCS is in HSLP mode. When DCS is in HS mode, commands are distributed in all the data lanes depending on the number of lanes selected with data lane 0 having the 1st byte command. A module (dcs_rom.v or dcs_rom_hs.v) to assist in placing the DCS commands on the LP data lane is provided. If you want to change the DCS commands to be compatible with the display to be used, DCS command must be provided to include the proper display commands.

Figure 2.3 shows the CMOS to MIPI D-PHY Interface Bridge IP top level block diagram.

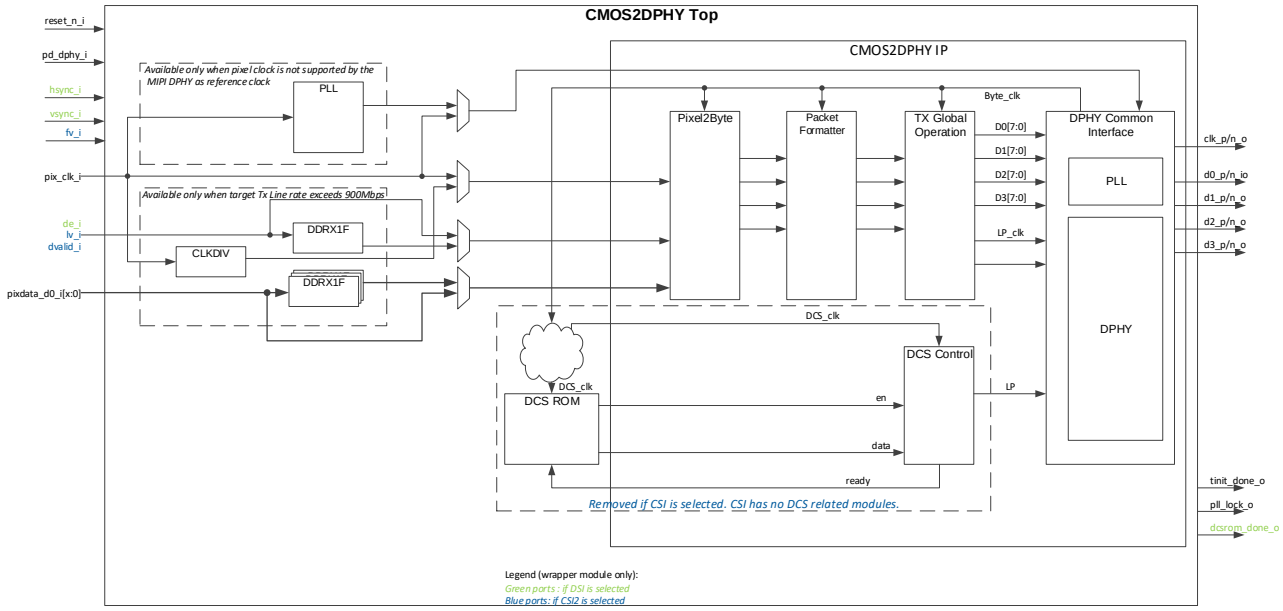


**Figure 2.3. Top Level Block Diagram**

**Table 2.1. Top Level Ports**

| Pin Name | Direction | Function Description |
|---|---|---|
| **Common Interface** | | |
| pix_clk_i | Input | Input pixel clock |
| reset_n_i | Input | Active low asynchronous reset input to the system |
| pd_dphy_i | Input | Power down input signal for D-PHY. When high, all D-PHY blocks are powered down. Tie low if not used. |
| **For Display Interface** | | |
| vsync_i | Input | Vertical Sync for parallel interface |
| hsync_i | Input | Horizontal Sync for parallel interface |
| de_i | Input | Data Enable for parallel interface |
| pixdata_d0_i [L-1:0] | Input | Pixel data bus for parallel interface<br>L:<br>24 — RGB888<br>18 — RGB666 |
| **For Camera Interface** | | |
| fv_i | Input | Frame Valid signal for parallel interface |
| lv_i | Input | Line Valid signal for parallel interface |
| dvalid_i | Input | Pixel valid signal for parallel interface. Tie to lv_i. |
| pixdata_d0_i[L-1:0] | Input | Pixel data bus for parallel CSI interface:<br>L:<br>24 — RGB888<br>12 — RAW12<br>10 — RAW10, YUV420_10, YUV422_10<br>8 — RAW8, YUV420_8, YUV422_8 |
| **D-PHY Interface** | | |
| d0_p_io | Bidirectional | D-PHY data lane 0 Positive Data |
| d0_n_io | Bidirectional | D-PHY data lane 0 Negative Data |
| d1_p_o | Output | D-PHY data lane 1 Positive Data |
| d1_n_o | Output | D-PHY data lane 1 Negative Data |
| d2_p_o | Output | D-PHY data lane 2 Positive Data |
| d2_n_o | Output | D-PHY data lane 2 Negative Data |
| d3_p_o | Output | D-PHY data lane 3 Positive Data |
| d3_n_o | Output | D-PHY data lane 3 Negative Data |
| clk_p_o | Output | D-PHY clock lane Positive end |
| clk_n_o | Output | D-PHY clock lane Negative end |
| **Miscellaneous** | | |
| tinit_done_o* | Output | tINIT done signal generated from IP |
| pll_lock_o* | Output | D-PHY PLL lock signal |
| dcsrom_done_o* | Output | Indicates that DCS ROM command has been completely sent to display or AP. Only available for DSI. |

***Note**: Can be turned-on if selected in the user interface.

## 2.1. Design and Module Description

The top level module instantiates *cmos_2_dphy_ip.v* module which contains the IPs and DCS ROMs. The DCS ROMs contain the DCS commands for display when DSI is selected. The *cmos_2_dphy_ip.v* module instantiates the *pixel2byte.v*, *pkt_formatter.v*, *tx_global_operation.v*, *dci_wrapper.v*, and *dcs_control.v* modules.

### 2.1.1. Initialization and Reset

#### 2.1.1.1. Initialization

After power-up, the transmitting D-PHY is required to drive a Stop State (LP-11) for a period longer than tINIT.
The D-PHY forces the lane module into transmit mode and generate stop state after system reset.
The Slave PHY is initialized when the Master PHY drives a Stop State (LP-11). The first Stop state that is longer than the specified tINIT is called the Initialization period. tINIT is estimated to be minimum 100 µs.

#### 2.1.1.2. Reset

An asynchronous reset pin (active high) is used for resetting the entire FPGA. Internal reset logic is implemented to guarantee synchronous de-assertion throughout different clock domains for both hard and soft IPs. Unless specified by MIPI IP or Soft IP requirement, no special reset sequence is needed for CrossLink. However, there is a wait time requirement before the Application Processor can send valid data to the bridge. When set in DSI, until DCS ROM is done sending the DCS commands to the display, valid data from the Application Processor might be lost. Likewise, for CSI-2, until PLL is locked, valid data from the Application Processor might be lost.

### 2.1.2. Pixel2byte

The pixel2byte.v module converts pixel data to MIPI byte data based on number of bits per pixel, the data type (DT) and number of D-PHY lanes. The input to this system is the pixel data (pixdata[L:0], vsync, hsync) and a data enable for display interfaces and pixel data (pixdata[L:0], FV, LV) for camera interfaces. The output of this module provides a byte enable indicating data payload is available and payload itself. The data type (DT) output provides the packet info indicator at the output of Pixel2byte.

### 2.1.3. Packet Formatter

The pkt_formatter.v module wraps the packet header and packet footer modules. The Packet header module generates and appends the packet header and footer to the data payload. The long_en input and the byte_data bus are used together to identify when payload is available. The header field and payload size is configurable by setting VC, WC, and DT parameters. End of Transmission packet (EoTp) is supported by this module. The Packet footer module computes a CRC16 checksum based on incoming data and data enable. The data bus input is maximum 24-bits as 24-bit RGB is the maximum parallel input width.

### 2.1.4. Tx Global Operations Controller

The tx_global_operation.v module controls HS request path and timing using parameters. Currently LP-request, escape mode and turnaround path are not supported. This block follows the requirements described D-PHY Specification version1.2 section 6 — operating modes for control and high-speed data transmission.
DSI data goes into Lower-Power mode during vertical and horizontal blanking depending on the Soft IP design.

Example: HS-0/1 -> LP11(Stop) -> LP01(LP-Rqst) -> LP00(Bridge) -> HS-0/1

This module controls the timing entering HS and coming from HS entering to LP following the MIPI D-PHY Specification 1.1 Table 16. The delay parameters can be adjusted inside this module by changing its localparam.

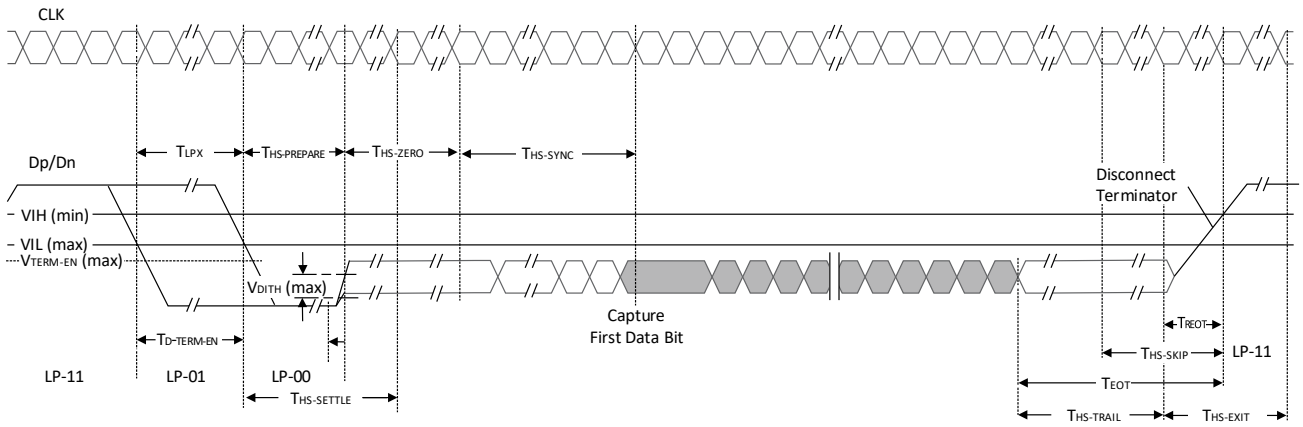


**Figure 2.4. High-Speed Data Transmission Timing Diagram**

Figure 2.4 shows that prior to the HS mode data transfer all clock and data lanes are in the LP11 state (1.2 V on the P channel and 1.2 V on the N channel). The clock lane then goes to the LP01 state (0 V on the P channel and 1.2 V on the N channel) then the LP00 state (0 V on the P channel and 0V on the N channel). After that, the clock lane goes into HS mode with SLVS200 signaling (Vcm=200 mV, Vdiff=±100 mV) and holds an HS0 state (differential 0 state of P channel = 100 mV and N channel = 300 mV when termination of the receiver is turned on). Then the clock starts shortly after. Once the HS clock is running the data lanes follow a similar procedure going from LP11 to LP01, LP00, and HS0 states. Then the HS-SYNC sequence is driven on the line followed by the packet header and data payload.
At the end of the transfer the data lanes first go back into LP mode by going to LP00 then LP11 states. The clock lane follows shortly after.

### 2.1.5. D-PHY Common Interface Wrapper

The dci_wrapper.v is used as the wrapper of MIPI D-PHY IP to make a connection between the PHY hard IP and higher protocol layers. The DCI wrapper serializes the incoming byte data and transmits it to D-PHY receiver.
Based on the Tx global operation state, it determines how to enable HS or LS mode for data transfer.

### 2.1.6. DCS ROM

The dcs_rom.v/dcs_rom_hs.v is used to store the Display Command Set for the interfaced DSI display. This module is used to place the DCS commands on the data lane provided. See Appendix B for details.

### 2.1.7. DCS HS Controller

The DCS_hs.v is used to control the flow of DCS data when DCS is transmitted in HS mode and append proper trail bits.

### 2.1.8. DCS LP Controller

The dcs_control.v provides method to send MIPI DCS configuration commands through data lane 0. DCS is only used with MIPI DSI, not CSI-2. The controller provides a multiplexed path to data lane 0, where inputs to the controller are the 8-bit MIPI DCS words and the output is a one-hot encoded 2-channel signal to be transferred on the P and N channel of data lane 0. A ROM is used to hard code configuration DCS command for display at start-up.

# 3. Parameter Settings

Table 3.1 lists the parameters used to generate the CMOS to MIPI D-PHY Interface Bridge Soft IP. All parameters are either set automatically or input in the user interface during the CMOS to MIPI D-PHY Interface Bridge Soft IP generation.

**Table 3.1. CMOS to MIPI D-PHY Interface Bridge Soft IP Parameters**

| Parameter | Attribute | Options | Description |
|---|---|---|---|
| Number of Tx Channels | Read-Only | 1 | Number of CMOS to MIPI D-PHY Interface Bridge Soft IP instance. Always set to 1. |
| Number of Tx Lanes | User-Input | 1, 2, 4 | Generates I/O up to four HS Tx data lane |
| Tx Interface | User-Input | DSI, CSI-2 | Set the Tx interface |
| Tx Gear | Read-Only | 8, 16 | Always set to 8. Gear16 is not supported in this soft IP |
| Rx Line Rate | Read-Only | <Value> | Rx line rate; Automatically computed based on target TX line rate |
| Tx Line Rate | Read-Only | <Value> | Target Tx line. Expected to be the same with Rx line rate. |
| Tx Line Rate (per lane) | User-Input | <Value> | Target Tx line rate per lane. |
| D-PHY Clock Frequency | Read-Only | <Value> | D-PHY clock; Automatically computed based on target Tx line rate |
| D-PHY Clock Mode | User-Input | 0, 1 | Specify the Tx clock transmission mode.<br>1– High Speed<br>0 – Low Power |
| Byte Clock Frequency | Read-Only | <Value> | Byte clock; Automatically computed based on target Tx line rate |
| Pixel Clock Frequency | Read-Only | <Value> | Pixel clock; Automatically computed based on target Tx line rate |
| DCS Clock Frequency | Read-Only | <Value> | DCS clock; Automatically computed based on target Tx line rate. Used when DCS is transmitted in Low Speed mode. |
| DCS ROM Wait Time | User-Input | <Value> | Specify delay time for sending DCS commands. This is clocked by byte clock if DCS is in High-Speed mode and DCS clock when DCS is in Low Speed mode. This needs to be adjusted if DCS is not meeting protocol timing requirements. Normal value is 1200 based on 111.375 MHz byte clock. |
| tINIT_SLAVE Value | User-Input | <Value> | Specify delay time for Tx D-PHY tINIT requirement. Must satisfy Tx D-PHY tINIT minimum requirement of 100 µs and is clocked by byte clock. |
| Data Type | User-Input | RGB888,<br>RGB666,<br>RAW8,<br>RAW10,<br>RAW12,<br>YUV420 8-bit,<br>YUV422 8-bit,<br>YUV420 10-bit,<br>YUV422 10-bit | Specify the data type depending on the Data Format selected |
| Virtual Channel | User-Input | 0, 1, 2, 3 | Specify the Virtual Channel |
| Word Count | User-Input | 1 – 65536 | Specify the total number of bytes for active pixels in a line.<br>Word Count = (Number of active pixel per line)·(Bits per pixel) / 8<br>For example RGB888, 1920x1080p60 resolution<br>Word Count = (1920)·(24) / 8 = 5760 |
| Number of DCS words | User-Input | <Value> | Specify the number of words for the DCS commands |
| DCS Mode | User-Input | 0, 1 | Specify the DCS transmission mode.<br>1 – High Speed<br>0—Low Power |
| DCS rom initialization file | User-Input | <file> | Specify the DCS commands. See Appendix B for details. |

| Parameter | Attribute | Value | Description |
|---|---|---|---|
| Data HS-Prepare | User-Input | <Value> | Specify the time that the D-PHY Tx drives the Data Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission. Timing parameter is in number of byte clock cycles. Value is initially automatically computed but can be customized. |
| Data HS-Zero | User-Input | <Value> | Specify the time that the D-PHY Tx drives the HS-0 state prior to transmitting the Sync sequence. Timing parameter is in number of byte clock cycles. Value is initially automatically computed but can be customized. |
| Clock Pre | User-Input | <Value> | Specify the time that the HS clock shall be driven by the D-PHY Tx prior to any associated Data Lane beginning the transition from LP to HS mode. Timing parameter is in number of byte clock cycles. Value is initially automatically computed but can be customized. |
| Clock Post | User-Input | <Value> | Specify the time that the D-PHY Tx 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 THS-TRAIL to the beginning of TCLK-TRAIL. Timing parameter is in number of byte clock cycles. Value is initially automatically computed but can be customized. |

# 4. IP Generation and Evaluation

This section provides information on how to generate the Lattice CMOS to MIPI D-PHY Interface Bridge Soft 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 CMOS to MIPI D-PHY Interface Bridge Soft IP in a complete, top level design. The CMOS to MIPI D-PHY Interface Bridge Soft IP is available free of charge.

Request your license by going to the link http://www.latticesemi.com/en/Support/Licensing and request the free Lattice Diamond license. In this form, select the desired CrossLink IP for your design.

You may download and generate the CMOS to MIPI D-PHY Interface Bridge Soft IP and fully evaluate the IP through functional simulation and implementation (synthesis, map, place and route) without an IP license. The CMOS to MIPI D-PHY Interface Bridge Soft 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 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 CMOS to MIPI D-PHY Interface Bridge Soft 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 Designer user interface as shown in Figure 4.1.

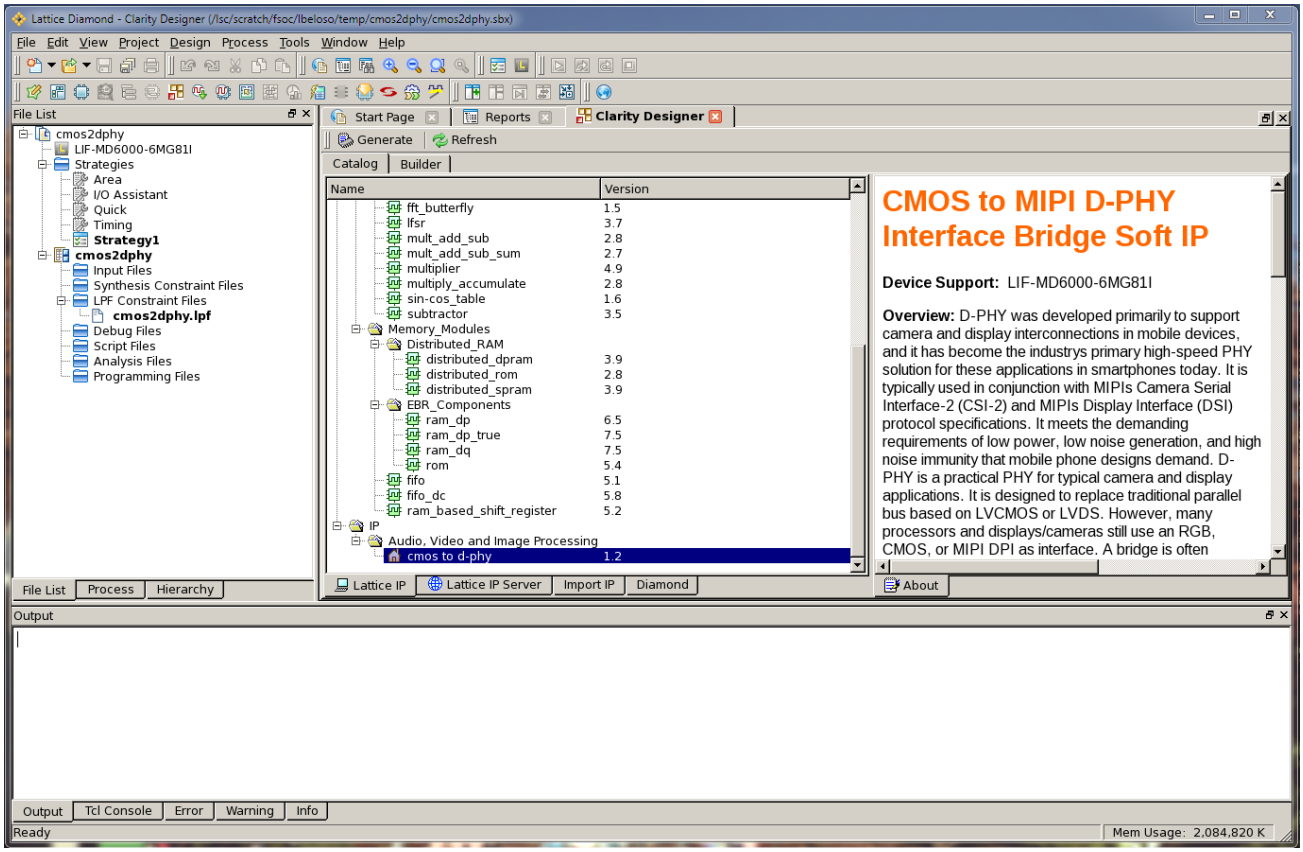


**Figure 4.1. Clarity Designer Window**

## 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 CMOS to MIPI D-PHY Interface Bridge Soft IP in Clarity Designer.

Clarity Designer is 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 CMOS to MIPI D-PHY Interface Bridge Soft IP is 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.

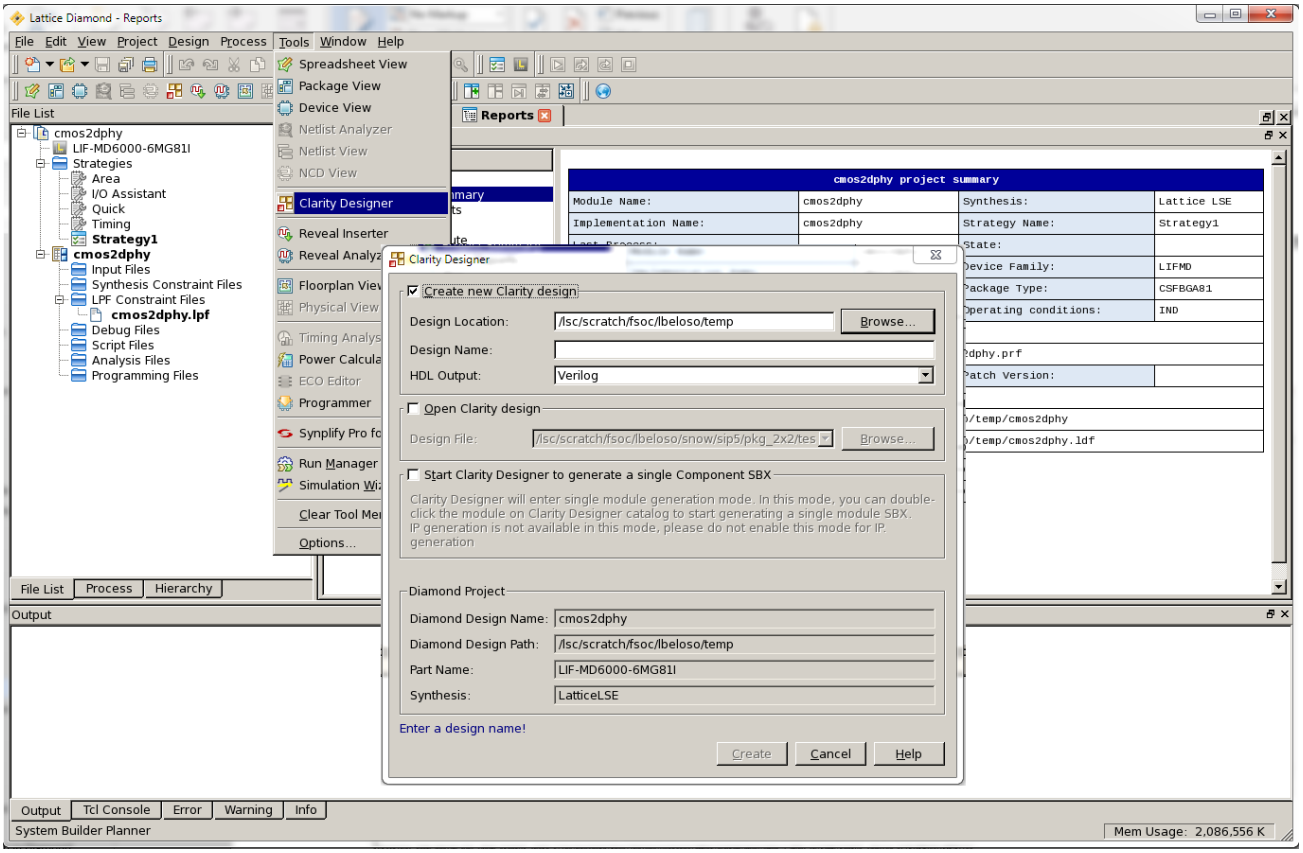


**Figure 4.2. Starting Clarity Designer from Diamond Design Environment**

To configure CMOS to MIPI D-PHY Interface Bridge Soft IP in Clarity Designer:

1. Double-click **CMOS to DPHY** in the IP list of the System Catalog view. The **cmos_to_dphy** dialog box is displayed as shown in Figure 4.3.

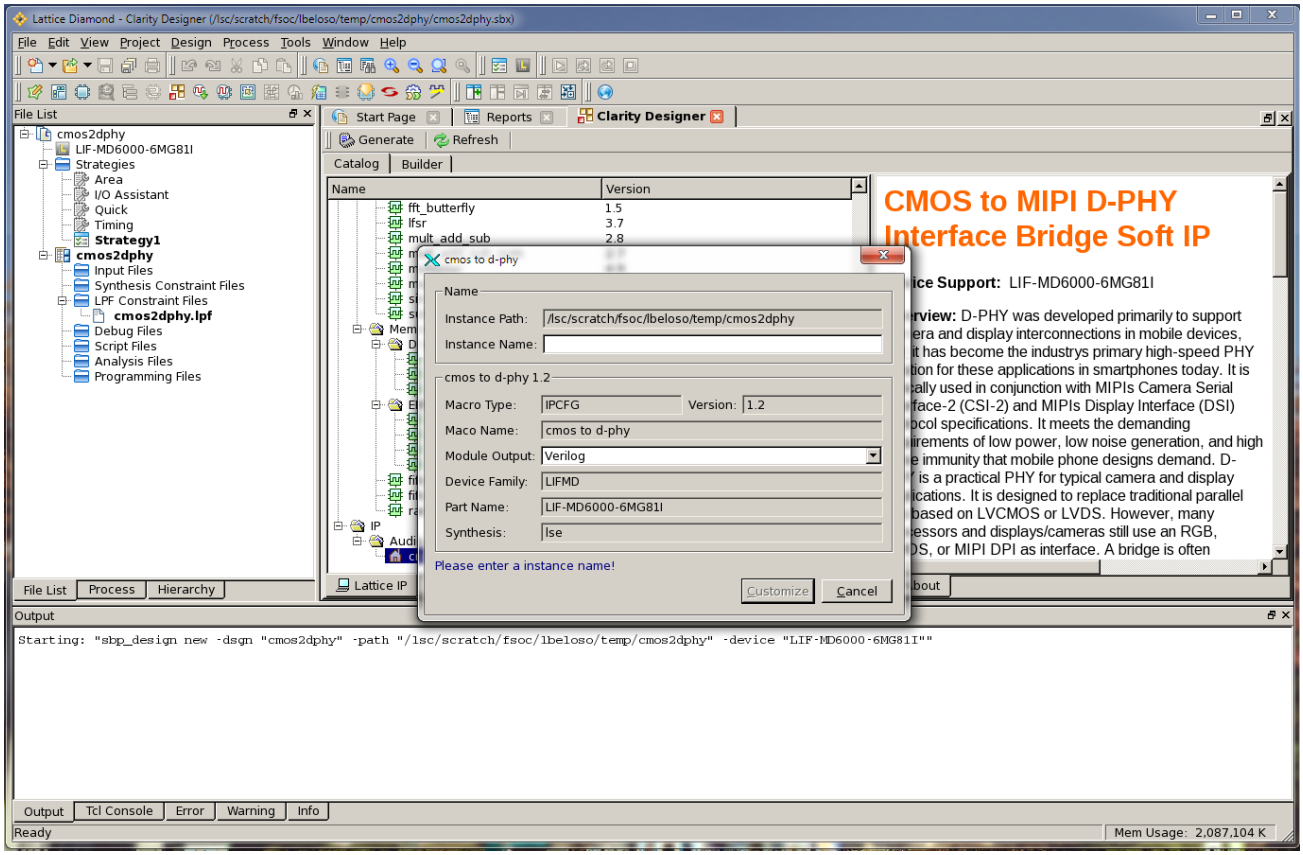


**Figure 4.3. Configuring CMOS to MIPI D-PHY Interface Bridge Soft IP in Clarity Designer**

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

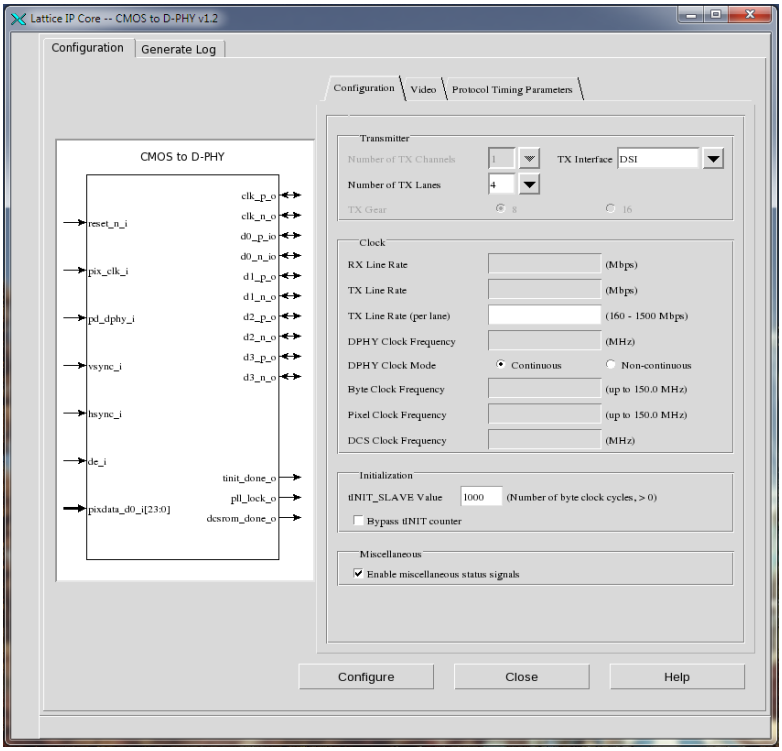


**Figure 4.4. Configuration Tab in IP User Interface**

4. Input valid values in the required fields in the **Configuration** tab shown in Figure 4.4.
5. Go to **Video** tab shown in Figure 4.5 and input valid values in the required fields. DCS Parameters Frame is disabled when Tx Interface is set to CSI2 as shown in Figure 4.6.

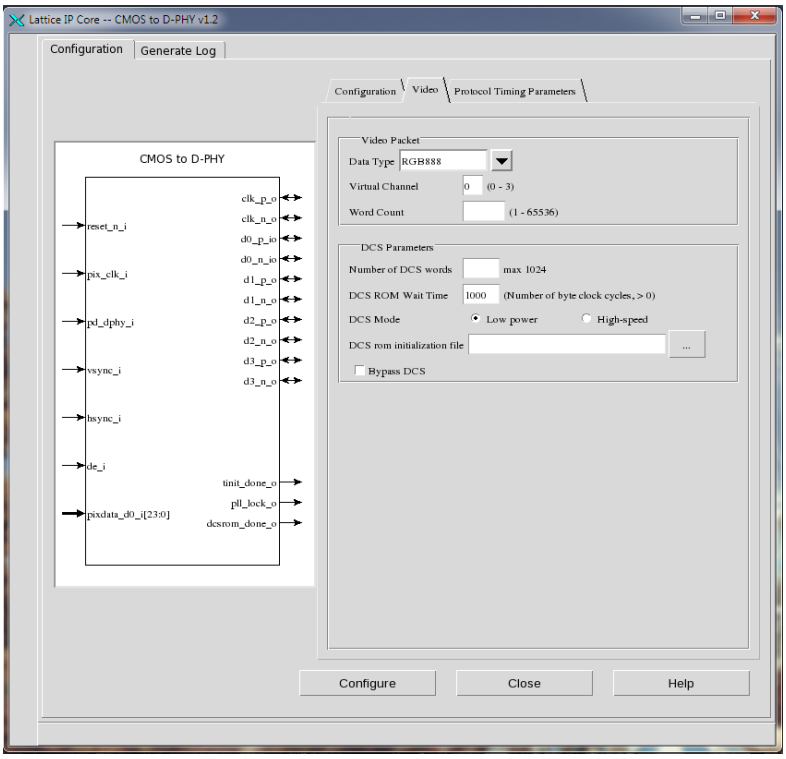


**Figure 4.5. Video Tab – DSI in IP User Interface**

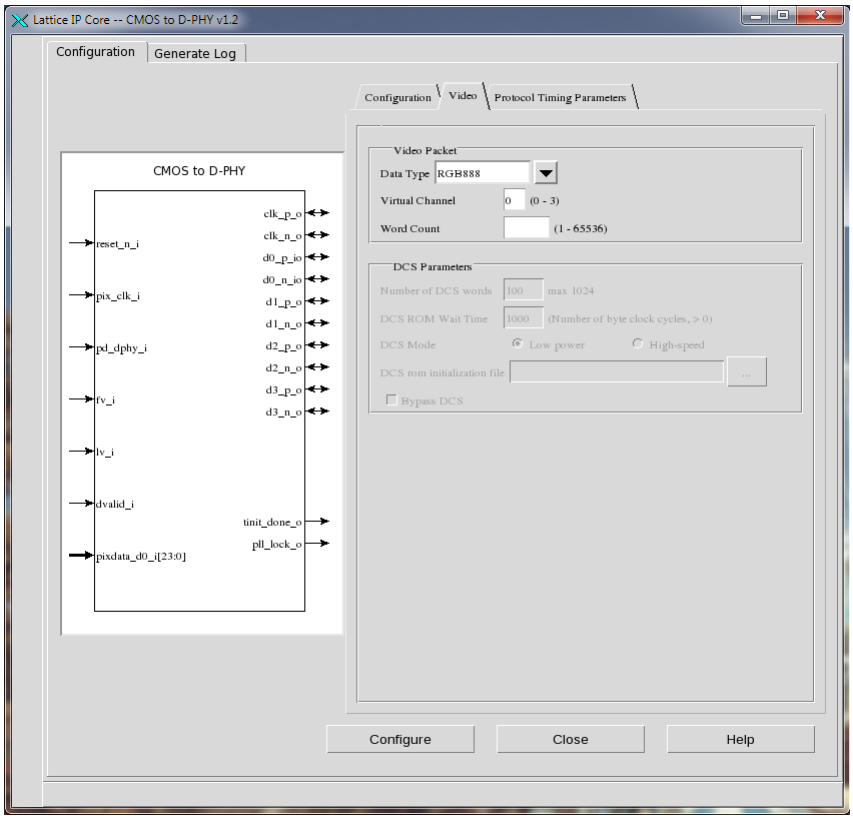


**Figure 4.6. Video Tab – CSI-2 in IP User Interface**

6. Go to **Protocol Timing Parameters** tab shown in Figure 4.7. Values are initially automatically computed. Input valid values in the required fields if customization is desired.
7. Select the required parameters, and click the **Configure** button.
8. Click **Close**.
9. 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.

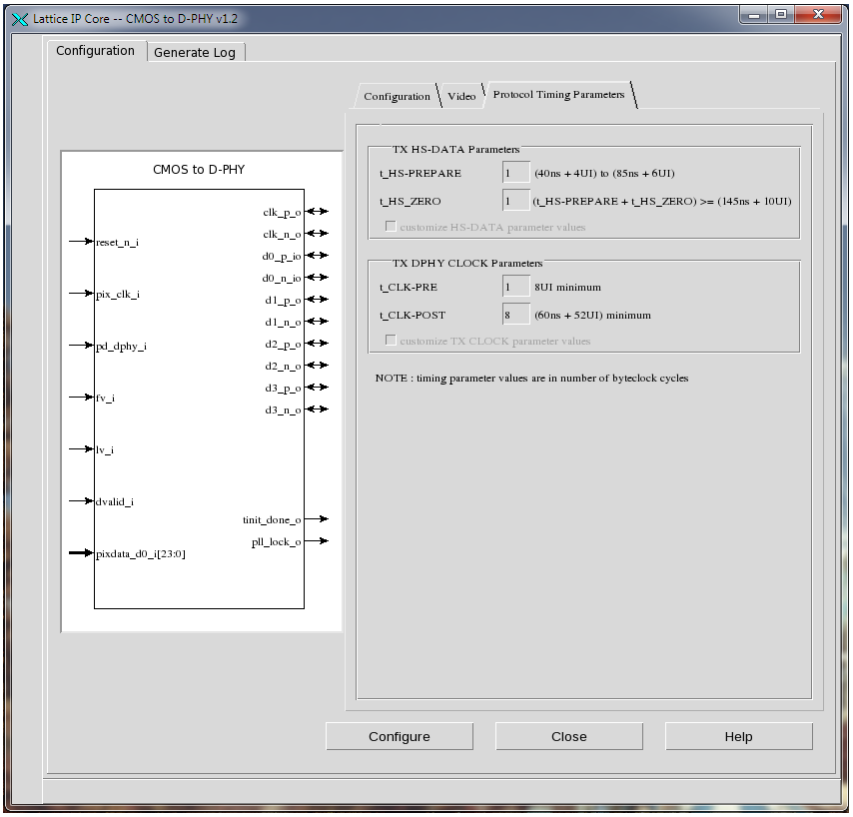


**Figure 4.7. Protocol Timing Parameters Tab in IP User Interface**

## 4.4. Generated IP Directory Structure and Files

The directory structure of the generated IP and supporting files is shown in Figure 4.8.

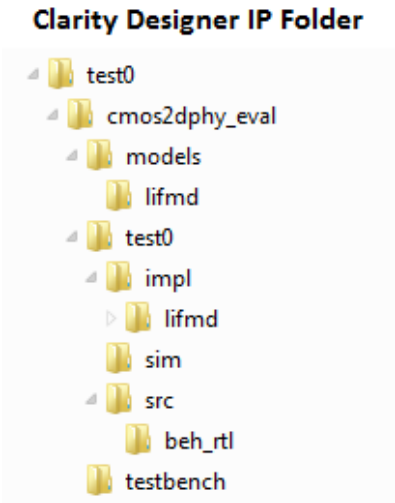


**Figure 4.8. CMOS to MIPI D-PHY Interface Bridge Soft IP Directory Structure**

The design flow for the IP created with Clarity Designer uses a post-synthesized module (NGO) for synthesis and uses protected model for simulation. The post-synthesized module and the protected model are customized when you configure the IP and created automatically when the IP is generated.

Table 4.1 provides a list of key files and directories created by Clarity Designer and how they are used. The post-synthesized module (NGO), the protected simulation model, and all other files are also generated based on your configuration and provided as examples to use or evaluate the IP.

**Table 4.1. Files Generated in Clarity Designer**

| File | Description |
|---|---|
| *<instance_name>*.v | Verilog top-level module of CMOS to MIPI D-PHY Interface Bridge Soft 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>*_*_beh.v | Protected Verilog models for simulation. |
| *<instance_name>* _*_bb.v | Verilog black box modules for synthesis. |
| *<instance_name>*_*.ngo | User interface 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 user interface in Clarity Designer when it is being reconfigured. |
| *<instance_name>*_inst.v/vhd | Template for instantiating the generated soft IP top-level in another user-created top module. |

Besides the files listed in the tables, most of the files required to evaluate the CMOS to MIPI D-PHY Interface Bridge Soft IP reside under the directory *\<cmos2dphy_eval>*. This includes the simulation model. A Lattice Diamond project file is also included under the folder at *\<cmos2dphy _eval>\<instance_name>\impl\lifmd\<synthesis_tool>*.

The *\<instance_name>* folder (test0 in Figure 4.8) contains files/folders with content specific to the *<instance_name>* configuration. This directory is created by Clarity Designer each time the IP is generated and regenerated with the same file name. A separate *\<instance_name>* directory is generated for IPs with different names, such as *\<my_IP_0>,\<my_IP_1>*, and others.

Aside from the *folder\<instance_name>*, the *\cmos2dphy_eval* and subdirectories provide files supporting CMOS to MIPI D-PHY Interface Bridge Soft IP evaluation that includes files/folders with content that is constant for all configurations of the CMOS to MIPI D-PHY Interface Bridge Soft IP. The *\cmos2dphy_eval* directory is created by Clarity Designer the first time the IP is generated, when multiple CMOS to MIPI D-PHY Interface Bridge Soft IPs are generated in the same root directory and updated each time the IP is regenerated.

The prebuilt Diamond projects are available at *\<project_root>\cmos2dphy_eval\<instance_name>\impl\lifmd\<synthesis_tool>* to evaluate the implementation (synthesis, map, place and route) of the IP in Lattice Diamond tool. The *src* directory contains the behavioral models of the black-boxed modules and the `models` directory provides library elements.

## 4.5. Instantiating the IP

The core modules of CMOS to MIPI D-PHY Interface Bridge Soft IP are synthesized and provided in NGO format with black box Verilog source files for synthesis. A Verilog source file named *<instance_name>_cmos_2_dphy_ip.v* instantiates the black box of core modules. The top-level file *<instance_name>.v* instantiates *<instance_name>_cmos_2_dphy_ip.v* and DCS components. A Verilog instance template *<instance_name>_inst.v* or VHDL instance template *<instance_name>_inst.vhd* is also provided as a guide if the design is to be included in another top level module.

You do not need to instantiate the IP instances 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.

## 4.6. Running Functional Simulation

To run the simulation in Active-HDL (Windows only):

1. Modify the ***.do** file located in **\<project_dir>\cmos2dphy_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 is 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 for the list of valid testbench compiler directives.
3. Under the **Tools** menu, select **Active-HDL**.
4. In Active-HDL window, on the **Tools** tab, click **Execute Macro**.
5. Select the ***.do** file **\<project_dir>\cmos2dphy_eval\<instance_name>\sim\aldec*.do**:
6. Click **OK**.
7. Wait for the simulation to finish.

Testbench parameters and directives can be modified by setting the define in the vlog command in the **.do* file.

Example:

```
vlog \
+define+NUM_FRAMES=60 \
+define+NUM_LINES=1080 \
....
```

**Table 4.2 Testbench Compiler Directives**

| Compiler Directive | Description |
|---|---|
| NUM_PIXELS | Number of pixels per line |
| NUM_LINES | Number of lines per frame |
| NUM_FRAMES | Number of frames to be transmitted |
| PIX_CLK | Clock period in ps; used as reference clock |
| **CSI-2 Optional Compiler Directives** | |
| INIT_DRIVE_DELAY | Delay from reset deassertion or tinit_done assertion before start of transaction |
| FV_H_TO_LV_H | Used to inject delay (in terms of clock cycle) from fv assertion to lv assertion |
| LV_L_TO_LV_H | Used to inject delay (in terms of clock cycle) from lv negation to lv assertion |
| LV_L_TO_FV_L | Used to inject delay (in terms of clock cycle) from lv negation to fv negation |
| FV_L_TO_FV_H | Used to inject delay (in terms of clock cycle) from fv negation to fv assertion |
| **DSI Optional Compiler Directives** | |
| INIT_DRIVE_DELAY | Delay from reset deassertion or tinit_done assertion before start of transaction |
| HFRONT_PORCH | Used to set duration of HFRONT_PORCH in terms of clock cycles |
| HSYNC_PULSE | Used to set duration of HSYNC_PULSE in terms of clock cycles |
| HBACK_PORCH | Used to set duration of HBACK_PORCH in terms of clock cycles |
| VFRONT_PORCH | Used to set duration of VFRONT_PORCH in terms of hsync pulses |
| VSYNC_PULSE | Used to set duration of VSYNC_PULSE in terms of hsync pulses |
| VBACK_PORCH | Used to set duration of VBACK_PORCH in terms of hsync pulses |

## 4.7. Simulation Strategies

This section describes the simulation environment which demonstrates basic CMOS2DPHY functionality. Figure 4.9 shows the block diagram of simulation environment.

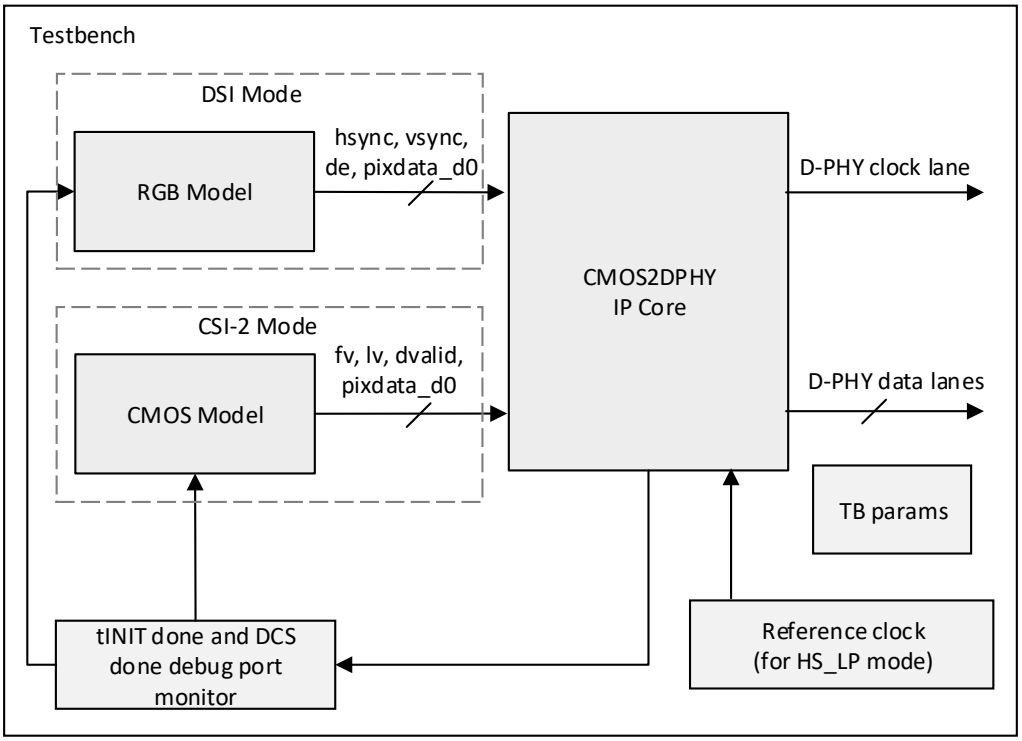


**Figure 4.9. Simulation Environment Block Diagram**

## 4.8. Simulation Environment

The simulation environment is made up of either RGB model instance or CMOS model instance connected to the input of CMOS2DPHY IP instance in the testbench depending on the configuration. The models are configured based on the CMOS2DPHY IP configurations and testbench configurations. You can adjust testbench parameters to modify parameters such as data type to be transmitted, timing in between packets, pixels per line, lines per frame, and so on. The testbench provides reference clock to the CMOS2DPHY IP. The testbench also monitors assertion of tinit_done or DCS ROM done signal before transmitting data to CMOS2DPHY IP core.

Figure 4.10 shows an example simulation of DSI transaction. The testbench waits for DCS completion by detecting *dcsrom_done_o* signal assertion before transmitting the data to the DUT input. In this example, one frame is transmitted with two lines.

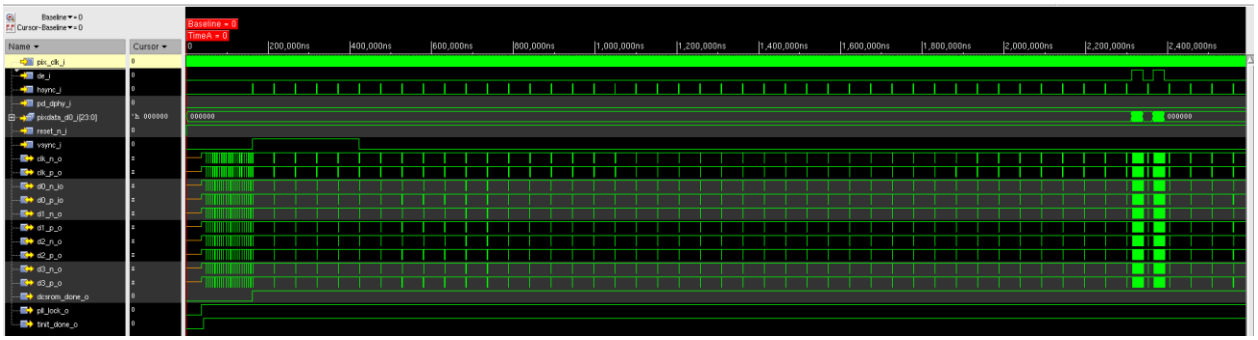

**Figure 4.10. DSI Transaction**

Figure 4.11 shows an example simulation of CSI-2 transaction.

In this example, miscellaneous signals such as *tinit_done_o* and *pll_lock* are disabled. Testbench monitors *tinit_done_o* signal before transmitting the data. However, if these signals are not available, the testbench uses a configurable delay to ensure that DUT has properly initialized before transmitting the data to DUT input. See the Running Functional Simulation section for details on how to set testbench parameters.

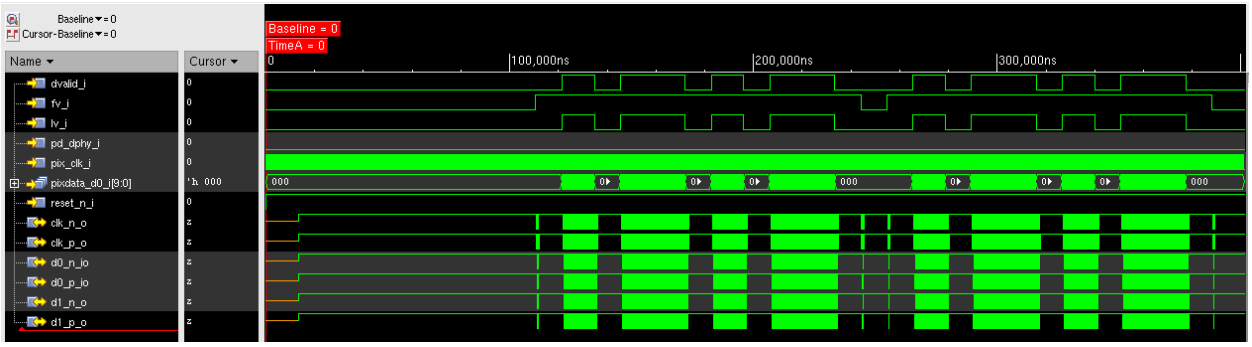


**Figure 4.11. CSI-2 Transaction**

## 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 all IPs are 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>\cmos2dphy_eval\<instance_name>\impl\lifmd\lse* or *\<project_dir>\cmos2dphy_eval\<instance_name>\impl\lifmd\synplify* directories. All required files are invoked automatically. You can directly synthesize, map and place/par the design in the Diamond design environment after the cores are generated.

Push-button implementation of this top-level design with either Synplify or Lattice Synthesis Engine is supported via the Diamond project files *<instance_name>_top.ldf* located in *\<project_dir>\cmos2dphy_eval\<instance_name>\impl\lifmd\<synthesis_tool>* directory.

To use the project files in Diamond:

1. Choose **File > Open > Project**.
2. In the **Open Project** dialog box browse to **\<project_dir>\cmos2dphy_eval\<instance_name>\impl\lifmd\<synthesis_tool>**.
3. Select and open **<instance_name>_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 user interface window.
5. Implement the complete design via the standard Diamond user interface flow.

## 4.10. Hardware Evaluation

The CMOS to MIPI D-PHY Interface Bridge Soft 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) with- out requiring the request of an IP license. It may also be used to evaluate the IP in hardware in user-defined designs.

### 4.10.1. Enabling Hardware Evaluation in Diamond

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 interface allows you to update the local IPs from the Lattice IP server. You can use the updated IP to regenerate the IP 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 4.12.
2. The IP Configuration user interface is displayed. Change the parameters as required and click the **Configure** button.
3. Click 

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

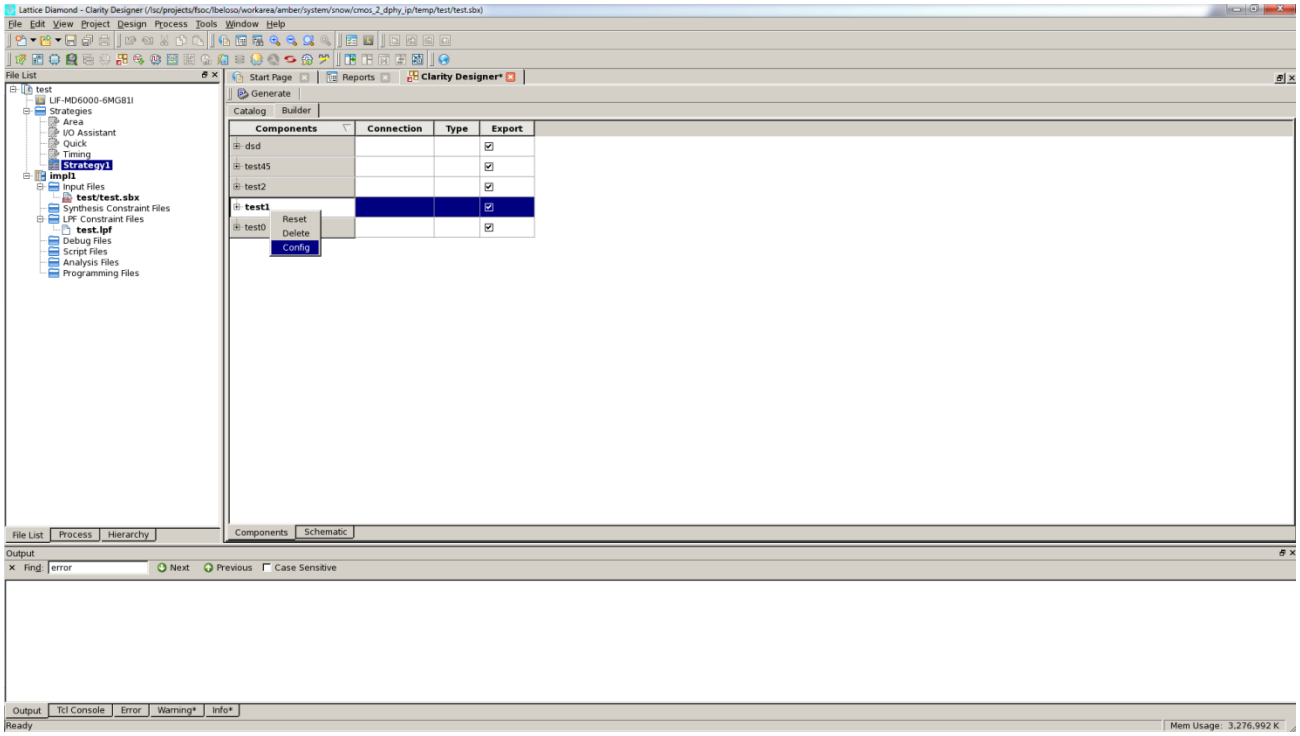


**Figure 4.12. IP Regeneration in Clarity Designer**

# References

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

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 (DSI), version 1.1, November 22, 2011, www.mipi.org
- MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2) version 1.1, July 18, 2012, www.mipi.org

# Technical Support Assistance

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

# Appendix A. Resource Utilization

Table A.1 provides resource utilization information for Lattice CrossLink FPGAs using the CMOS to MIPI D-PHY Interface Bridge Soft IP.

The Clarity Designer is a Lattice IP configuration utility, and is included as a standard feature of the Diamond tool. Details regarding the usage of Clarity Designer are available in the Clarity Designer and Diamond help system.

For more information on the Diamond design tools, visit the Lattice web site at www.latticesemi.com/Products/DesignSoftwareAndIP.

**Table A.1. Resource Utilization[1]**

| IP User-Configurable Parameters | Slices | LUTs | Registers | sysMEM EBRs | Actual $f_{MAX}$ (MHz)[2] | Target $f_{MAX}$ (MHz)[2] |
|---|---|---|---|---|---|---|
| DSI, RGB888, NUM_TX_LANE = 4 | 1216 | 1719 | 1017 | 8 | 94.625 | 93.750 |
| DSI, RGB888, NUM_TX_LANE = 2 | 1741 | 3085 | 679 | 4 | 112.070 | 93.750 |
| DSI, RGB888, NUM_TX_LANE = 1 | 1141 | 1924 | 590 | 2 | 129.938 | 93.750 |

**Notes**:

1. Performance and utilization data target an LIF-MD6000-6MG81I device using Lattice Diamond 3.8 and Lattice Synthesis Engine software. Performance may vary when using a different software version or targeting a different device density or speed grade within the Lattice CrossLink family. This does not show all possible configurations of the CMOS to MIPI D-PHY Interface Bridge Soft IP.
2. The $f_{MAX}$ values are based on byte clock.

# Appendix B. 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 user interface.

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 B.1 shows the sample entries.

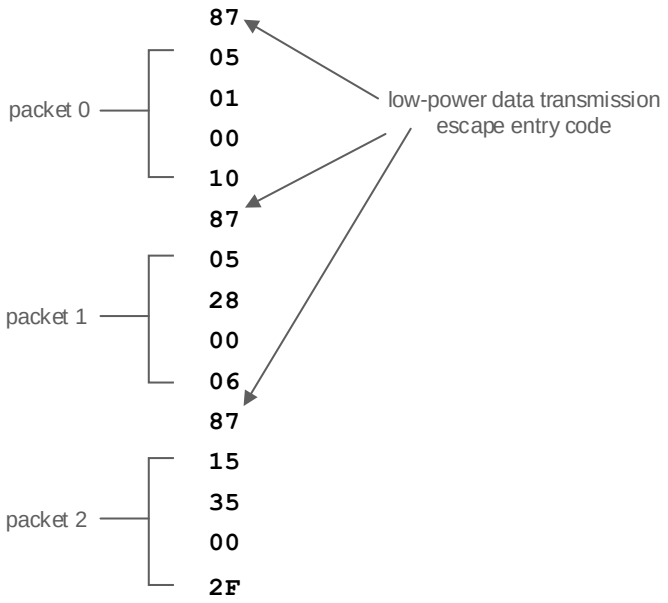


**Figure B.1. Sample DCS ROM for DCS Low-Power Mode**

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 8

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

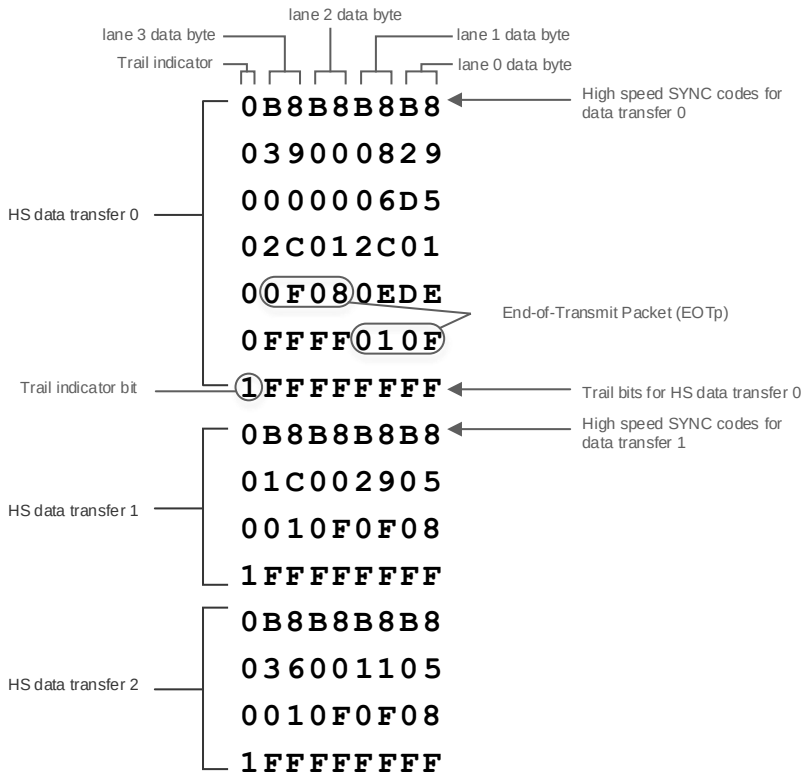


**Figure B.2. Sample DCS ROM for x4 Gear 8 DCS High-Speed Mode**

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 ROM files are also available under *./<IPinstallation_path>/samples* .

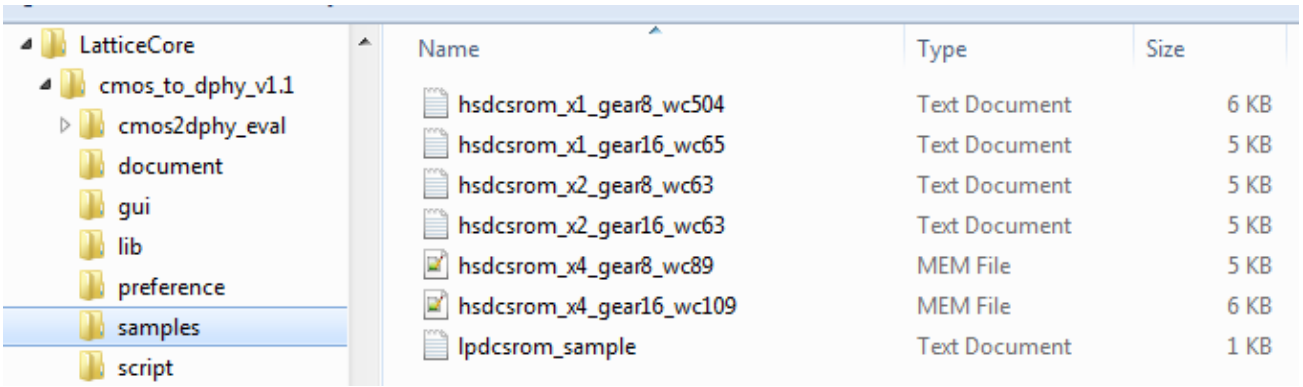


**Figure B.3. Directory Containing the Sample DCS ROM Initialization Files**

# Appendix C. What is Not Supported

The IP does not support configuration through registers.

The IP has the following limitations:

- The maximum frequency limit for LVCMOS I/O has not been updated yet in Lattice Diamond version 3.9. Thus, when setting a frequency constraint for LVCMOS I/O that is higher than 250 MHz, a timing violation is reported by the Static Timing Analysis tool. This violation can be ignored, and all the internal timings are still closed correctly by the tool. This violation is eliminated in later releases of Lattice Diamond.
- Word count limitation, see Table C.1.

**Table C.1. Word Count Limitation**

| Tx Gear | Number of Tx Lanes | Word Count Divisibility |
|---|---|---|
| 16 | 4 | 8 |
| 16 | 2 | 4 |
| 16 | 1 | 2 |
| 8 | 4 | 4 |
| 8 | 2 | 2 |
| 8 | 1 | 1 |

# Revision History

### Revision 1.4, IP Version 1.2, April 2019

| Section | Change Summary |
|---|---|
| Introduction | Specified that this user guide can be used for IP design version 1.x. |
| IP Generation and Evaluation | In Licensing the IP, modified the instructions for requesting free license. |
| Revision History | Updated revision history table to new template. |
| All | Minor adjustments in style and formatting. |

### Revision 1.3, IP Version 1.2, January 2017

| Section | Change Summary |
|---|---|
| Introduction | Updated number of LUTs, sysMEM EBRs, and Registers in Table 1.1. CMOS to MIPI D-PHY Interface Bridge Soft IP Quick Facts in Quick Facts section. |
| Parameter Settings | Added D-PHY Clock Mode parameter in Table 3.1. CMOS to MIPI D-PHY Interface Bridge Soft IP Parameters. |
| IP Generation and Evaluation | Updated figures in Getting Started and Creating IP in Clarity Designer sections. |
| Appendix C. What is Not Supported | Updated content. |

### Revision 1.2, November 2016

| Section | Change Summary |
|---|---|
| Parameter Settings | Updated Word Count description in Table 3.1. CMOS to MIPI D-PHY Interface Bridge Soft IP Parameters. |
| IP Generation and Evaluation | • Updated Licensing the IP section – Added email address lic_admn@latticesemi.com for requesting free license.<br>• Added step 9 in Creating IP in Clarity Designer section. |

### Revision 1.1, July 2016

| Section | Change Summary |
|---|---|
| All | Updated document number, the previous document number was IPUG126. |
| Introduction | Updated Table 1.1. CMOS to MIPI D-PHY Interface Bridge Soft IP Quick Facts, and added simulation in Quick Facts section. |
| Parameter Settings | Added parameters Data HS-PREPARE, Data HS-Zero, Clock Pre, Clock Post to Table 3.1. CMOS to MIPI D-PHY Interface Bridge Soft IP Parameters. |
| IP Generation and Evaluation | • Updated Generated IP Directory Structure and Files section.<br>• Added new Running Functional Simulation, Simulation Strategies, Simulation Environment sections. |
| Appendix A. Resource Utilization | Updated values of Slices, LUTs, Registers, Actual $f_{MAX}$ in Table A.1. Resource Utilization. |
| Appendix B. Initializing the DCS ROM | Updated content. |

### Revision 1.0, IP Version 1.0, May 2016

| Section | Change Summary |
|---|---|
| All | Initial release |

www.latticesemi.com