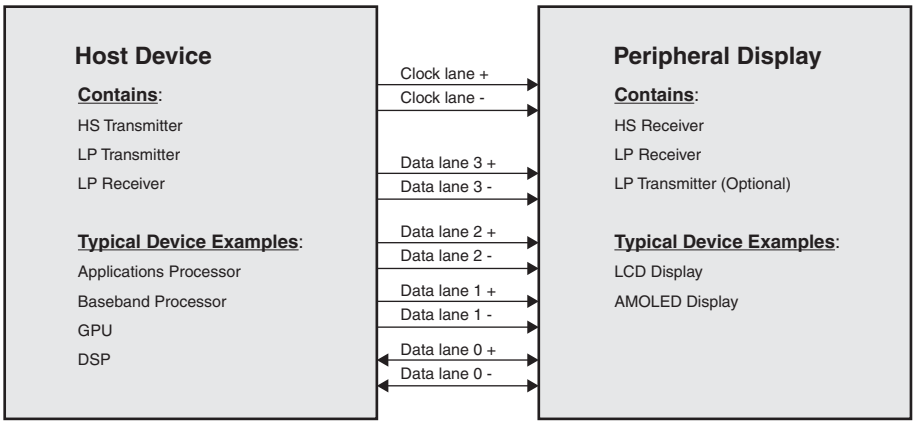
Parallel to MIPI DSI TX Bridge

January 2015 **Reference Design RD1184**

Introduction

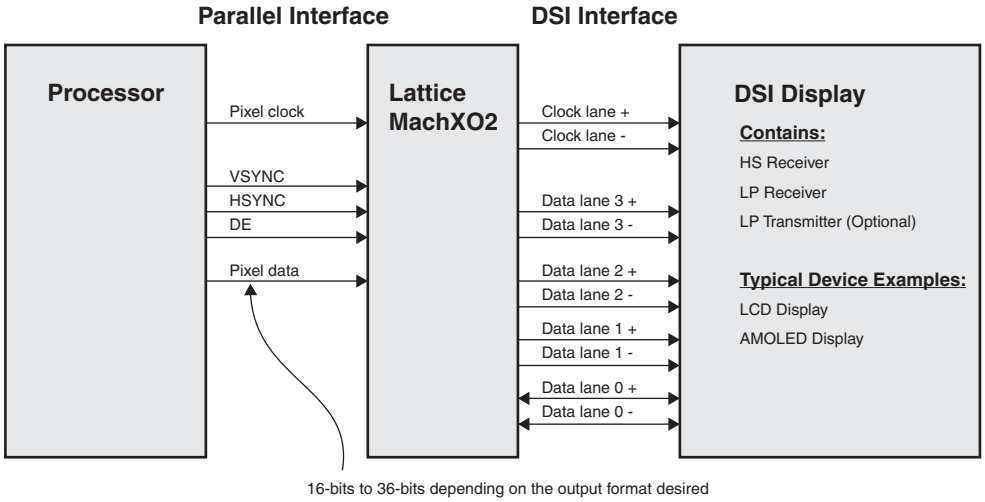
The Mobile Industry Processor Interface (MIPI) has become a specification standard for interfacing components in consumer mobile devices. The MIPI Display Serial Interface (DSI) specification provides a protocol layer interface definition, which is used to interface with displays. Normally an applications processor (AP) would directly connect to a display that has a DSI interface. See Figure 1.

Figure 1. DSI Interface



Because of the enormous volume of DSI displays that ship in smart phones & tablets, their cost is inexpensive. These attractively priced DSI screens are being adopted in wide variety of end markets. However, products that do not use an applications processor will be challenged to connect to a DSI screen. The Lattice Parallel to MIPI DSI Tx Bridge Reference Design allows a traditional processor to connect to a DSI screen. The Parallel to MIPI DSI TX Bridge Reference Design allows users to deliver data to a MIPI DSI compatible display from a standard parallel video interface.

Figure 2. Parallel to MIPI DSI TX Bridge



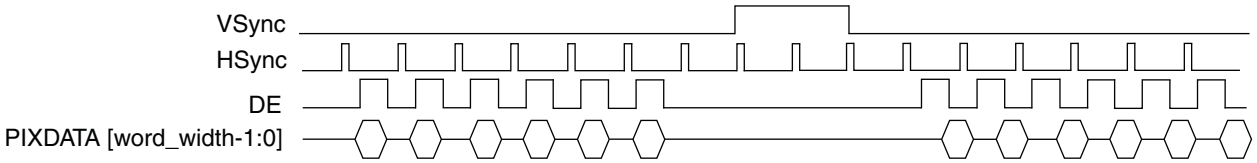
Key Features

- Interfaces to MIPI DSI Receiving Devices
- Supports Transmitting in HS (High Speed) Mode
- Supports Transmitting and Receiving in LP (Low Power) Mode
- Provides a DCS (Display Command Set) controller to program the display
- Serializes HS (High Speed) data up to four data lanes
- Supports all DSI compatible video formats (RGB, YCbCr and User Defined)

Functional Description

The Parallel to MIPI DSI TX Bridge Reference Design converts a standard parallel video interface into DSI byte packets. The input interface for the design consists of a data bus (PIXDATA), vertical and horizontal sync flags (VSYNC and HSYNC), a data enable and a clock (DE and PIXCLK). The parallel video interface consists of RGB or YCbCr data. This parallel bus is converted to the appropriate DSI output format. The DSI output serializes HS (High Speed) data and controls LP (Low Power) data and transfers them using RD1182, MIPI D-PHY Reference IP.

Figure 3. Parallel Input Bus Example Waveform



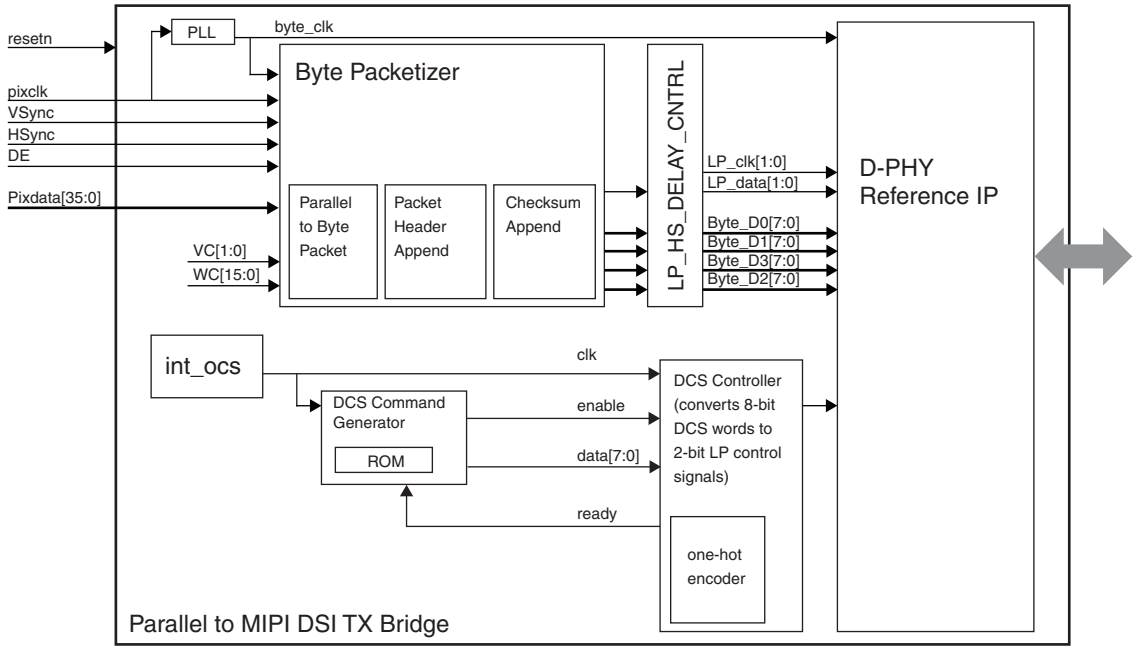
The output interface consists of HS and LP signals that must be connected together using an external resistor network, which is described in the External Resistor Network Implementation for D-PHY TX section of this document. Further information regarding this resistor network can also be found in the RD1182, MIPI D-PHY Reference IP.

Data Lane 0 can be used to configure the display with DCS (Display Command Set) commands. A module to assist in placing the DCS commands on the LP data lane is provided. HS and LP signals for the clock lane and data lanes are provided on DCK, D0, D1, D2, D3 and LPCLK, LP0, LP1, LP2, and LP3 signals respectively. Include parameters control the amount of data ports available for HS and LP modes at the top level depending on the number of data lanes used.

The top level design (top.v) consists of five main modules, a PLL module, and a test module:

- byte_packetizer.v - Converts parallel data to byte packets. Appends Packet Header, Checksum and EoTp (End of Transfer Packet).
- lp_hs_dly_ctrl.v - Controls time delay between clock and data lanes when entering and exiting HS mode. Controls time delay from when HS mode is entered to when data is placed on the data bus.
- dphy_tx_inst.v - Serializes byte data using iDDRx4 gearbox primitives. Controls high impedance and bi-directional states of HS and LP signals.
- dcs_encoder.v - one hot encodes 8-bit data bus input to a 2-bit LP data bus.
- dcs_rom.v - ROM controller containing DCS (Display Command Set) commands used to configure and initialize the display.

Figure 4. Top Level Block Diagram



To control the ports defined at the top level, `define compiler directives are used. These compiler directives can be found in compiler_directives.v

Table 1. Compiler Directives Defined in compiler_directives.v:

Directive	Description
`define HS_3	Generates IO for four HS data lanes.
`define HS_2	Generates IO for three HS data lanes. Overridden if HS_3 is defined.
`define HS_1	Generates IO for two HS data lanes Overridden if HS_3 or HS_2 is defined.
`define HS_0	Generates IO for one HS data lane Overridden if HS_3, HS_2, or HS_1 is defined.
`define LP_CLK	Generates IO for LP mode on clock lane
`define LP_0	Generates IO for LP mode on data lane 0
`define LP_1	Generates IO for LP mode on data lane 1
`define LP_2	Generates IO for LP mode on data lane 2
`define LP_3	Generates IO for LP mode on data lane 3

Design parameters control other features of the design. These design parameters are located at the top of the module declaration in top.v.

Table 2. Top Level Module Parameters

Parameter	Options	Description
VC	2-bit Virtual Channel value	Virtual Channel number appended to the Packet Header
WC	16-bit Word Count value	Word Count number appended to the Packet Header. Correlates to the number of bytes to be transferred in a Long Packet.
word_width	Up to 36 bits	Bus width of pixel data bus input
DT	6-bit Data Type Value	Data Type appended to Packet Header for Long Packet transfers
testmode	0 = off 1 = on	Adds colorbar pattern generator for testing purposes. Pattern generator utilizes reset_n and PIXCLK.
crc16	0 = off 1 = on	Appends checksum after Long Packet transfers. Turning off will reduce resource utilization and append 16'hFFFF in place of checksum.
EoTp	0 = off 1 = on	Bursts an End of Transfer packet after any HS data transfer.

Top level IO ports are defined as follows for top.v. The number of IO is dependent on the number of data lanes defined by compiler_directives.v.

Table 3. Top Level Design Port List

Signal	Direction	Description
reset_n	Input	Resets module (Active 'low')
DCK	Output	HS (High Speed) Clock
D0	Output	HS Data lane 0
D1	Output	HS Data lane 1
D2	Output	HS Data lane 2
D3	Output	HS Data lane 3
LPCLK [1:0]	Bidirectional	LP clock lane; LPCLK[1] = P wire, LPCLK[0] = N wire
LP0 [1:0]	Bidirectional	LP data lane 0; LP0[1] = P wire, LP0[0] = N wire
LP1 [1:0]	Bidirectional	LP data lane 1; LP1[1] = P wire, LP1[0] = N wire
LP2 [1:0]	Bidirectional	LP data lane 2; LP2[1] = P wire, LP2[0] = N wire
LP3 [1:0]	Bidirectional	LP data lane 3; LP3[1] = P wire, LP3[0] = N wire
PIXCLK	Input	Parallel Pixel Clock
VSYNC	Input	Parallel Vertical Sync Indicator
HSYNC	Input	Parallel Horizontal Sync Indicator
DE	Input	Parallel Data Enable Indicator
PIXDATA[*:0]	Input	Parallel Data Bus

The top level module instantiates and connects five main modules. In addition, a PLL module controls clocking for the entire design. The input of the PLL is pixel clock. The PLL outputs two high speed oDDRx4 gearbox clocks (0 degree and one with 90 degree phase shifts), the byte clock and the CRC clock.

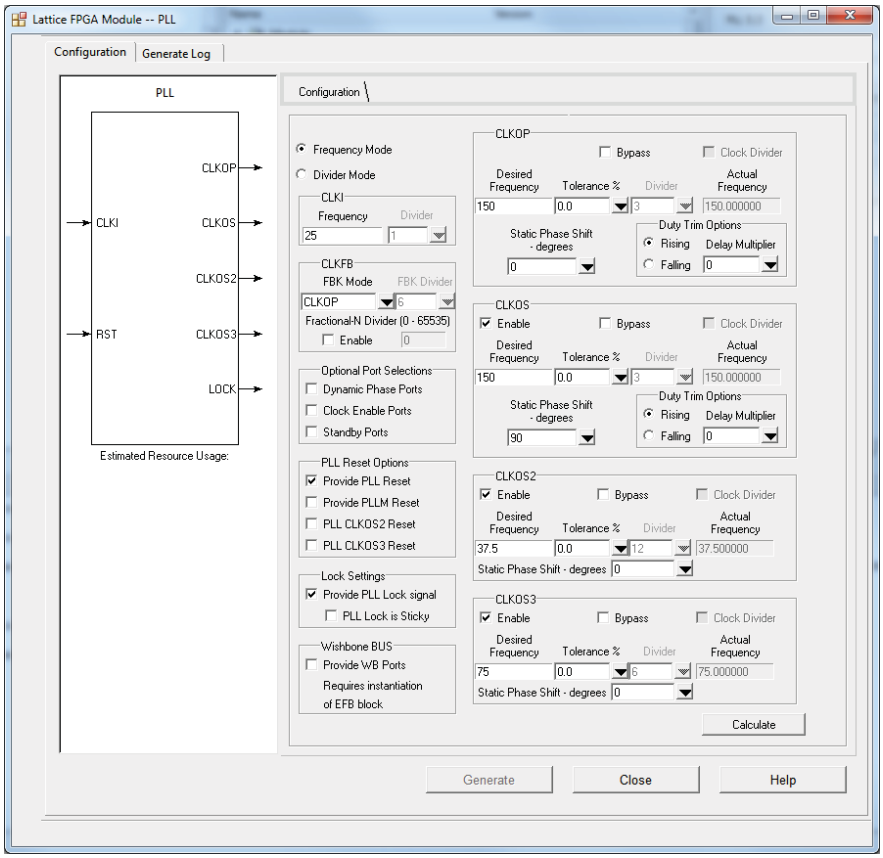
The clock equations for PLL output ports are shown in Table 4.

Table 4. Clocking for the PLL Output Ports

PLL Module Port Name	Clock description	Clock Equation
CLKI	PLL Input	CLKI
CLKOP	oDDRx4 gearbox Clock	CLKOP = CLKI * word_width / (8 * lane_width) *4
CLKOS	oDDRx4 gearbox Clock (90 degree shift)	Same as CLKOP, but with static phase shift of 90 degrees
CLKOS2	Byte Clock	CLKOS2 = CLKI * word_width / (8 * lane_width)

The PLL is configured using IPExpress in the Lattice Diamond Software. It can also be adjusted and reconfigured to individual design needs by double clicking on pll_pix2byte.ipx in the file list. An IPExpress configuration GUI will open to adjust the PLL.

Figure 5. IPExpress Configuration Page for pll_pix2byte.ipx:



BYTE_PACKETIZER Module Description

The Byte_Packetizer module converts pixels to one to four bytes depending on the number of MIPI data lanes defined. The input interface to the module is the pixel data bus. The format of the data on the pixel bus is dependent on the data type as described in Table 5.

Table 5. DSI Pixel Data Order

DATA TYPE	Cycle	FORMAT
RGB121212	Any	PIXELDATA[35:0]={R[11:0], G[11:0], B[11:0]}
RGB101010	Any	PIXELDATA[29:0]={R[9:0], G[9:0], B[9:0]}
RGB888	Any	PIXELDATA[23:0]={R[7:0], G[7:0], B[7:0]}
RGB666	Any	PIXELDATA[17:0]={R[5:0], G[5:0], B[5:0]}
RGB565	Any	PIXELDATA[15:0]={R[4:0], G[5:0], B[4:0]}
YCbCr 4:2:2 24-bit	Odd Pixels	PIXELDATA[23:0]={Cb[11:0], Y[11:0]}
	Even Pixels	PIXELDATA[23:0]={Cr[11:0],Y[11:0]}
YCbCr 4:2:2 20-bit	Odd Pixels	PIXELDATA[19:0]={Cb[9:0], Y[9:0]}
	Even Pixels	PIXELDATA[19:0]={Cr[9:0],Y[9:0]}
YCbCr 4:2:2 16-bit	Odd Pixels	PIXELDATA[15:0]={Cb[7:0], Y[7:0]}
	Even Pixels	PIXELDATA[15:0]={Cr[7:0],Y[7:0]}
YCbCr 4:2:0 12-bit	Odd Lines	PIXELDATA[11:0]={Cb1[7:0], Y1[7:4]}, {Y1[3:0],Y2[7:0]}
	Even Lines	PIXELDATA[11:0]={Cr1[7:0], Y1[7:4]}, {Y1[3:0],Y2[7:0]}

Additional, input ports include the byte clock, CRC clock, and EoTp enable. The virtual channel number and word count are also available as interface ports. These ports are clocked into a register on the rising edge and falling edges of VSYNC, HSYNC as well as the rising edge of DE and appended to the Packet Header. By default, the reference design controls the Virtual Channel and Word Count ports through VC, WC and EoTp parameters in top.v. However, the user can dynamically control VC, WC and EoTp if desired.

Parameters for the BYTE_PACKETIZER include word_width (bus width of the pixel bus), lane_width (number of byte lanes), dt (data type), crc16 enable. Different NGOs in the */NGO/* folder are called depending on the mode defined.

Within the module the pixel data is converted to bytes. If the data is going to be a long packet, identified by DE (Data Enable), the CRC checksum will be calculated over the data and appended to the end of the long packet. Also appended to the data stream in this module is the Packet Header for all packet types. The EoTp packet is consecutively transferred in a burst following any long or short packet if the feature is enabled.

The number of horizontal pixels the DE (Data Enable) is high should correlate to an integer multiple of the number of bytes used at the output. It is recommend that active lines be truncated or extended to meet this criteria. This will ensure proper readout of all pixels and a correct checksum calculation.

To ensure that the input pixel data is an integer multiple of the output byte data, the following equation can be used. The DE must be held for an integer number of byte clocks. If "number of byte clocks" does not calculate to an integer value, adjust the number of pixel clock cycles for which DE is active.

number of byte clocks = [(number of pixels) * (bits per pixel)] / [8 bits * (number of data lanes)]Output ports for the BYTE_PACKETIZER module include the 8-bit data buses for each lane and an enable signal. The hs_en signal goes active 'high' when any short packet or long packet is to be transmitted.

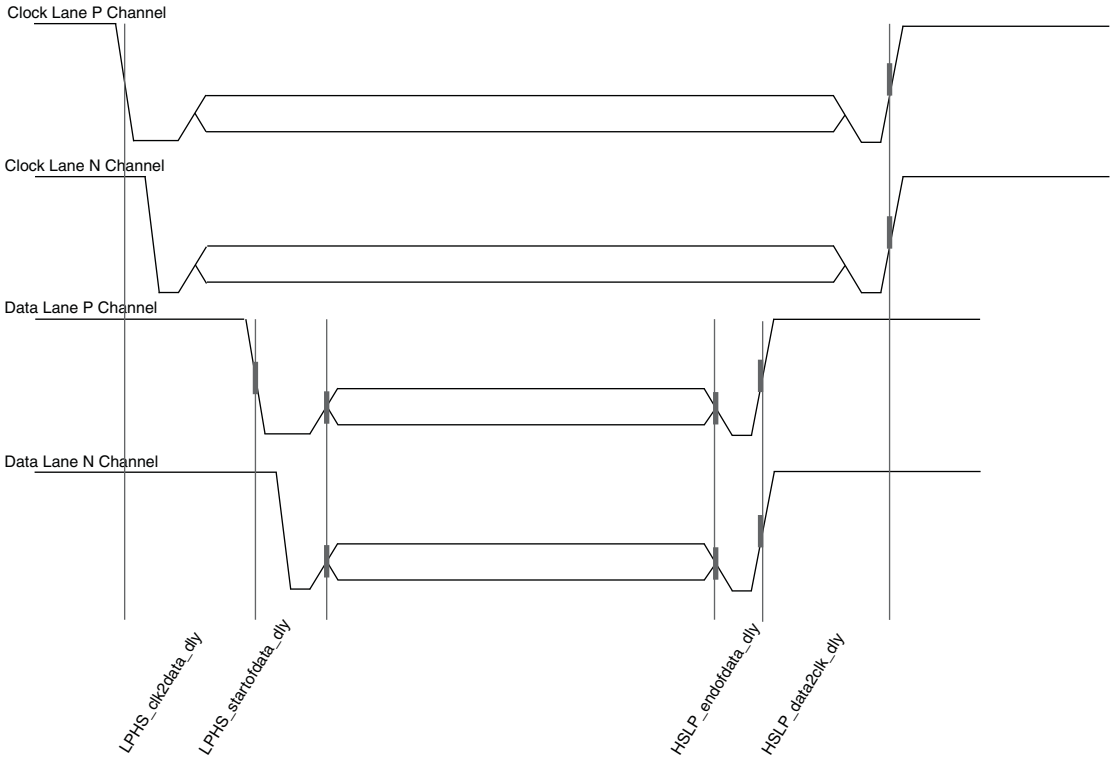
LP_HS_DELAY_CNTRL Module Description

The LP_HS_DELAY_CNTRL module uses the hs_en input from the BYTE_PACKETIZER and adds delays so that it is ready for transmission. There are controllable delay parameters available in the module header. These control the time delay between when the clock lane and data lanes transition from LP to HS mode as well as from HS mode to LP mode. It also controls when the data starts with respect to when it entered LP mode. LP11-LP01-LP00 transitions are also controlled with one byte clock between transitions. This module is open source and available for any user modifications desired in Lattice FPGA devices.

Table 6. LP_HS_DELAY_CNTRL Module Parameters:

Parameter	Description
LPHS_clk2data_dly	Number of clocks to delay between the MIPI clock lane and MIPI data lanes transitioning rom LP to HS mode
LPHS_startofdata_dly	Number of clocks to delay the MIPI data from the LP to HS mode transition
HSLP_data2clk_dly	Number of clocks to delay the HS to LP mode transition between the MIPI data lanes and MIPI clock lane
HSLP_endofdata_dly	Number of clocks to delay the MIPI data from the HS to LP mode transition
sizeofstartcntr	Size for the start timer counter. Number of bits to count LPHS_clk2data_dly+LPHS_startofdata_dly
sizeofendcntr	Size for the end timer counter. Number of bits to count HSLP_data2clk_dly+HSLP_endofdata_dly

Figure 6. Timing Diagram for LP_HS_DELAY_CNTRL Delay Parameters



DCS_Encoder Module Description

The DCS_Encoder module converts 8-bit DCS commands to a one-hot encoded 2-bit bus that can be transmitted over data lane 0 while in LP mode. The DCS commands are the mechanism for configuring the DSI screen registers. Along with a resetn, clk and data[7:0] input signals, data_en, escape_en, stop_en and ready signals allow you to control data transactions effectively.

Signal escape_en is used to initiate a DCS data transmission by placing the data lane into escape mode. When escape_en is clocked 'high', an LP data sequence will be initiated of LP11-LP10-LP00-LP01-LP00 on Lp and Ln output ports. During this transmission the ready output port will go 'low'. No additional data transfers shall be committed until ready is seen 'high' again. Once escape mode is entered, the data_en signal can be used similarly to clock in 8-bits of data at a time when the ready port is 'high'. This 8-bit data is one hot encoded and sent out of the Lp and Ln output ports on consecutive byte clock cycles. When a data transaction is complete, the stop enable signal can be used to return Lp and Ln output ports to the LP11 idle state. This module is open source and available for any user modifications desired in Lattice FPGA devices.

DCS_ROM Module Description

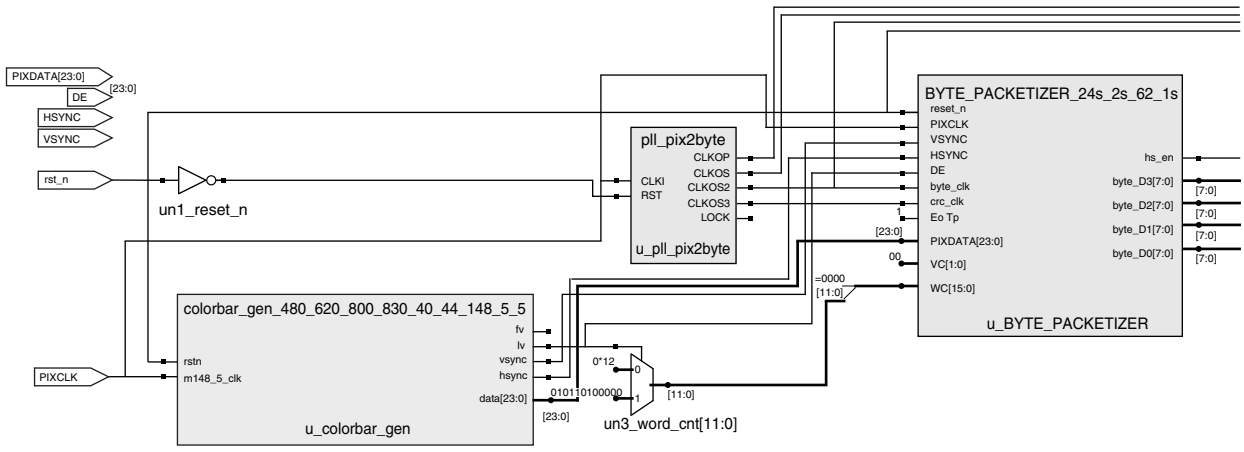
The DCS_ROM module gives the ability for users to store DCS commands in memory within the MachXO2™ device. The DCS_ROM module's controller is directly compatible with the DCS_Encoder module. By default, a set of DCS commands are initialized in ROM which are compatible with the Wintek WM-FL040V LCD Module, which comes with the Intrinsyc Snapdragon S4 development Platform's peripheral kit.

A parameter wait_time in the DCS_ROM module controls the time to wait between sets of data transfers. A DCS command of 8'h87 identifies a new data transfer to the controller. When this command is seen, the DCS_ROM controller waits wait_time before it delivers the data and toggles the appropriate escape_en, data_en, stop_en controls. This module is open source and available for any user modifications desired in Lattice FPGA devices.

Test Mode and colorbar_gen Module Description

This reference design also includes a colorbar pattern generator. This allows the user to initially control and drive a display panel with minimal external controls needed from the receiving side. The design is place in test mode setting the top level design parameter testmode = 1. When this is set an additional module colorbar_gen is instantiated at the top level and takes over the all input controls (VSYNC, HSYNC, DE and PIXDATA) with the exception of reset_n and PIXCLK.

Figure 7. RTL Block Diagram of colorbar_gen Instantiation When testmode=1:



Packaged Design

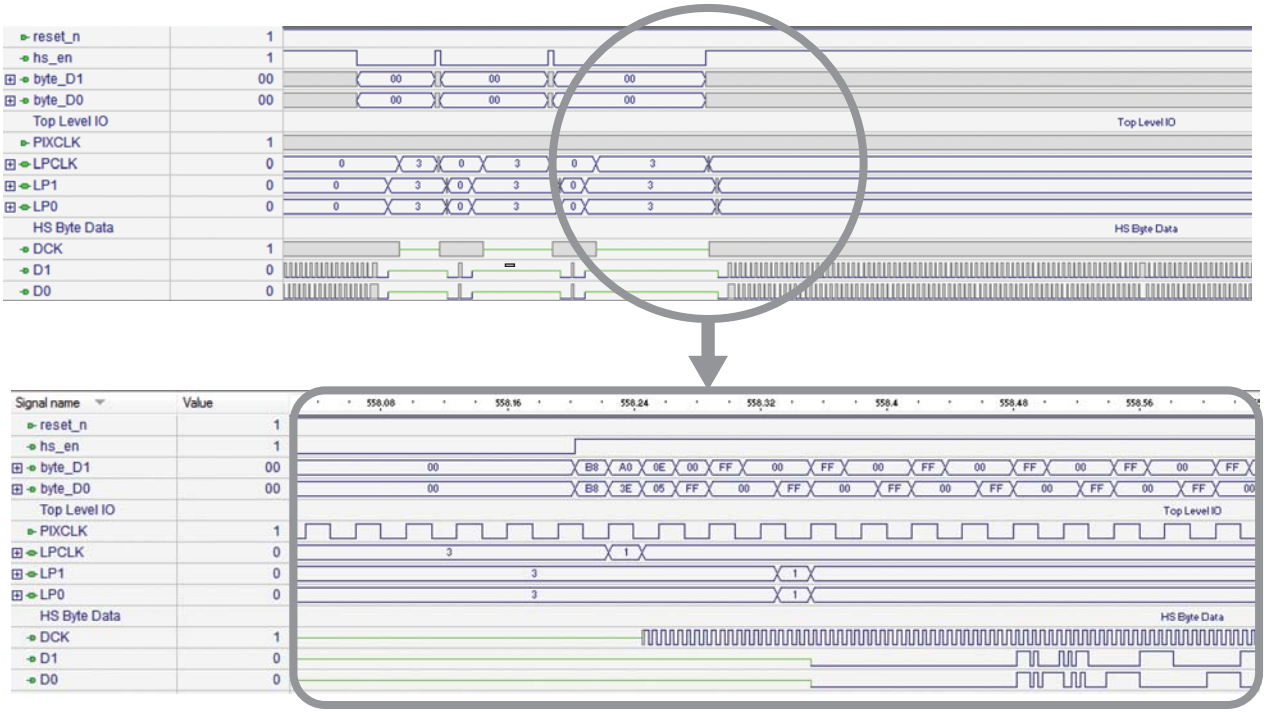
The Parallel to MIPI DSI TX Bridge Reference Design is available for Lattice MachXO2 devices. The reference design immediately available on latticesemi.com is configured for RGB888, 2-lane mode. Other designs are available through the bridge request form. The packaged design contains a Lattice Diamond project within the *\project\ folder configured for the MachXO2 and MachXO3™ (specifically the L version of the family) devices. Verilog source is contained within the *\source\ folder. The Verilog test bench is contained within the testbench folder. The simulation folder contains an Aldec Active-HDL project. It is recommended that users access the active HDL Simulation environment through the Lattice Diamond Software and the simulation setup script contained within the project. For details on how to access the design simulation environment see the Functional Simulation section of this document.

Figure 8. Packaged Design Directory Structure

Functional Simulation

The simulation environment and testbench Parallel2DSI_tb_*.v instantiates the top level design module. The top level design inputs are driven with a generated pattern from the colorbar_gen module. The DCS_Encoder module can be seen operating for the first 480us of simulation. After this point the done flag of the DCS_ROM will go 'high' and data from the pattern generator will run through the conversion pipeline to the output signals of the bridge design. It is recommended users run the simulation for at least 600 µs for the DCS command to complete and view the data output.

Figure 9. Simulation Waveforms

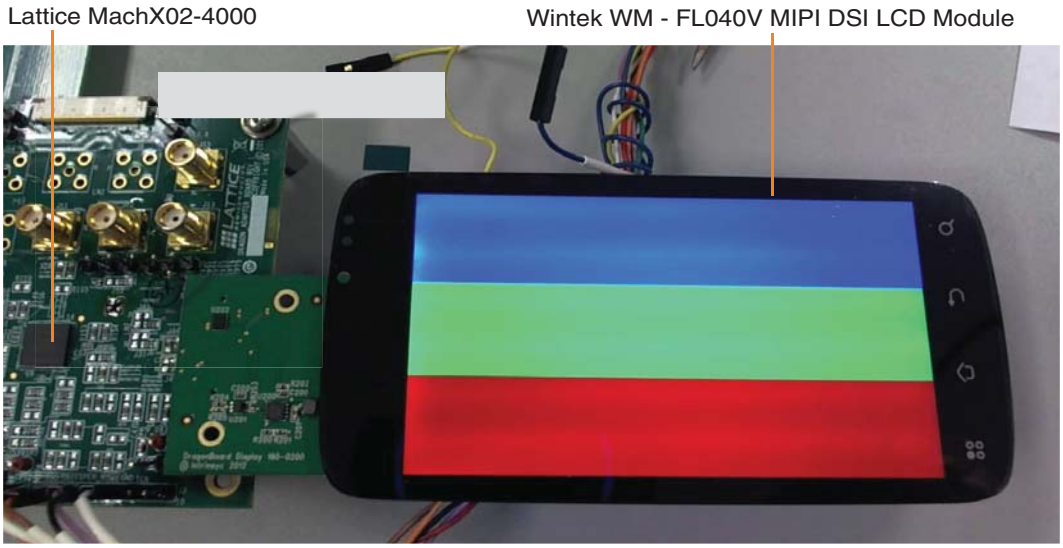


The simulation environment can be accessed by double clicking on the <name>.spf script file in Lattice Diamond from the file list. After clicking OK, Aldec ActiveHDL opens to the pop-up windows. Compile the project and initialize the simulation.

Design Testing and Demonstration

The Parallel to MIPI DSI TX Bridge Reference Design was tested with the design in test mode using a Wintek WM-FL040V MIPI DSI LCD Module and a Lattice MachXO2-4000 FPGA in cs132bga package. The resolution was set to match the display at 480x800p60 in RGB888, 2-lane mode.

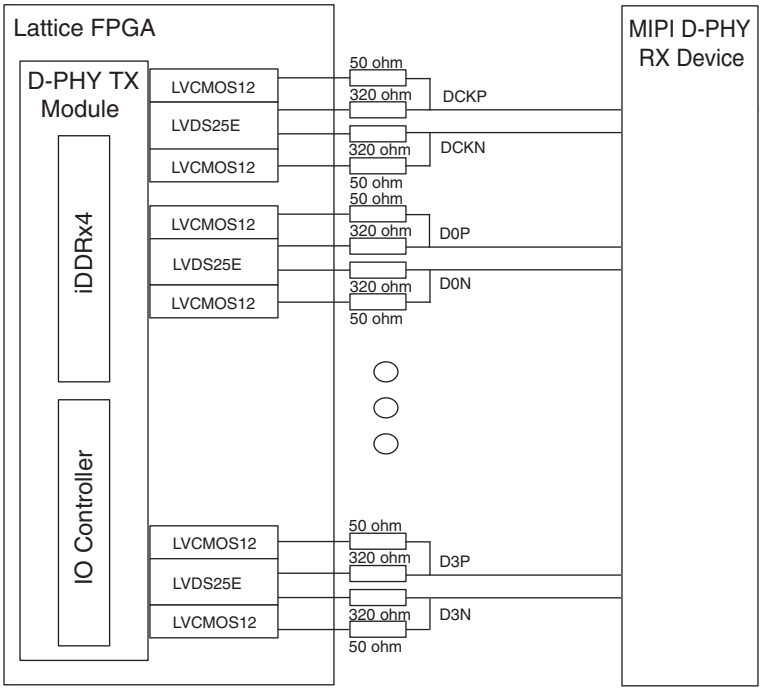
Figure 10. Hardware Development and Testing using a MIPI DSI Display:



External Resistor Network Implementation for D-PHY TX

As described in RD1182, MIPI D-PHY Reference IP, an external resistor network is needed to accommodate the LP and HS mode transitioning on the same signal pairs as well as the lower 200mV common mode voltage during HS clock and data transfers. The resistor network needed for MIPI TX implementations is provided below.

Figure 11. Unidirectional Transmit HS Mode and Bidirectional LP Mode Interface Implementation



Device Pinout and Bank Voltage Requirements

Choosing a proper pinout to interface with another D-PHY device is essential to meet functional and timing requirements.

The following are rules for choosing a proper pinout on MachXO2 devices:

- Bank 0 should be used for HS outputs (DCK, D0, D1, D2, D3) with the TX D-PHY IP since these pins utilize oDDRx4 gearbox primitives
- The VCCIO voltage for banks 0 should be 2.5V
- The HS input clock (DCK) for the RX D-PHY IP should use an edge clock on bank 2
- The HS data signals (D0, D1, D2, D3) for the RX and TX D-PHY IP's should only use A/B IO pairs
- LP signals (LPCLK, LP0, LP1, LP2, LP3) for RX and TX D-PHY IP's can use any other bank
- The VCCIO voltage for the bank containing LP signals (LPCLK, LP0, LP1, LP2, LP3) should be 1.2V
- When in doubt, run the pinout through Lattice Diamond software can check for errors

With the rules mentioned above a recommend pinout is provided for the most common packages chosen for this IP. For the MachXO2 the cs132bga is the most common package. The pinouts chosen below are pin compatible with MachXO2-1200, MachXO2-2000 and MachXO2-4000 devices.

Table 7. Recommended TX Pinout and Package

Signal	MachXO2 1200/2000/4000 cs132bga Package	
DCK_p	Bank 0	A7
DCK_n		B7
D0_p		B5
D0_n		C6
D1_p		A2
D1_n		B3
D2_p		A10
D2_n		C11
D3_p		C12
D3_n		A12
LPCLK [1]	Bank 1	E12
LPCLK [0]		E14
LP0 [1]		E13
LP0 [0]		F12
LP1 [1]		F13
LP1 [0]		F14
LP2 [1]		G12
LP2 [0]		G14
LP3 [1]		G13
LP3 [0]		H12

Table 8. TX IO Timing

Device Family	Speed Grade -4 @ 262Mhz		Speed Grade -5 @ 315Mhz		Speed Grade -6 @ 378Mhz	
MachXO2	Data Valid Before Clock (ps)	Data Valid After Clock (ps)	Data Valid Before Clock (ps)	Data Valid After Clock (ps)	Data Valid Before Clock (ps)	Data Valid After Clock (ps)
	710	710	570	570	455	455

Table 9. TX Maximum Operating Frequencies by Configuration[1]

Device Family	Configuration	Speed Grade -4 (MHz)		Speed Grade -5 (MHz)		Speed Grade -6 (MHz)		Speed Grade -7 (MHz)		Speed Grade -8 (MHz)	
		PIXCLK	byte_clk	PIXCLK	byte_clk	PIXCLK	byte_clk	PIXCLK	byte_clk	PIXCLK	byte_clk
ECP5™	RGB888, 2 Data Lanes (LP+HS) - LSE	—	—	—	—	209.074	111.161	258.532	115.580	286.451	149.499
	RGB888, 2 Data Lanes (LP+HS) - Syn	—	—	—	—	243.546	125.109	277.469	156.323	327.225	164.177
MachXO2	RGB888, 1 Data Lane (LP+HS)	150	63	164	70	183	85	—	—	—	—
	RGB888, 2 Data Lanes (LP+HS) - LSE	150.015	86.843	164.690	93.362	182.5	114.903	—	—	—	—
	RGB888, 2 Data Lanes (LP+HS) - Syn	150.015	70.676	164.690	77.803	182.5	88.285	—	—	—	—
	RGB888, 4 Data Lanes (LP+HS)	150	67	165	68	180	73	—	—	—	—
MachXO3L	RGB888, 2 Data Lanes (LP+HS) - LSE	—	—	164.690	80.103	182.5	85.012	—	—	—	—
	RGB888, 2 Data Lanes (LP+HS) - Syn	—	—	164.690	95.841	182.5	111.607	—	—	—	—

1. The maximum operating frequencies were obtained by post P&R timing analysis. They do not correlate to clocking ratios (obtained from PLL clock equations) used for proper design operation.

Resource Utilization

The resource utilization tables below represent the device usage in various configurations of the D-PHY IP. Resource utilization was performed on the IP in configurations of 1, 2 and 4 data lanes. For each of these configurations LP mode on the data lanes used was turned on. In addition, HS and LP clock signals were available for each configuration.

Table 10. TX Resource Utilization

Device Family	Configuration	Register	LUT	EBR	PLL	Gearbox	Clock Divider
ECP5 [3]	RGB888, 2 Data Lanes (LP+HS) - LSE	797	1828	5	1	3	1
	RGB888, 2 Data Lanes (LP+HS) - Syn	708	1210	5	1	3	1
MachXO2 [1]	RGB888, 1 Data Lane (LP+HS)	249	453	4	1	2	1
	RGB888, 2 Data Lanes (LP+HS) - LSE	639	1275	4	1	3	1
	RGB888, 2 Data Lanes (LP+HS) - Syn	348	758	3	1	3	1
	RGB888, 4 Data Lanes (LP+HS)	302	572	5	1	5	1
MachXO3L [2]	RGB888, 2 Data Lanes (LP+HS) - LSE	446	1349	3	1	3	1
	RGB888, 2 Data Lanes (LP+HS) - Syn	346	730	3	1	3	1

1. Performance and utilization characteristics are generated using LCMXO2-1200HC-6MG132C with Lattice Diamond 3.3 design software.
2. Performance and utilization characteristics are generated using LCMXO3L-4300C-6BG256C with Lattice Diamond 3.3 design software.
3. Performance and utilization characteristics are generated using LFE5UM-45F-8MG285C with Lattice Diamond 3.3 design software.

References

- MIPI Alliance Specification for Display Serial Interface (DSI) V1.1
- MIPI Alliance Specification for D-PHY V1.1

Technical Support Assistance

e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
January 2015	01.5	Added support for ECP5 device family.
		Updated DCS_ROM Module Description section.
		Updated Packaged Design section.
		Updated the Device Pinout and Bank Voltage Requirements section. Updated Table 9, TX Maximum Operating Frequencies by Configuration.
		Updated the Resource Utilization section. Updated Table 10, TX Resource Utilization. Added support for Lattice Diamond 3.1 design software.
April 2014	01.4	Added support for MachXO3L device family.
		Updated Functional Description section. Revised top level design (top.v) main modules.
		Updated Functional Simulation section. Revised .spf script file name.
		Updated Packaged Design section. — Updated introductory paragraph. — Updated Figure 8, Packaged Design Directory Structure.
		Updated Table 9, Performance and Resource Utilization and Updated Table 10, Performance and Resource Utilization. Added support for Lattice Diamond 3.1 design software.
March 2014	01.3	Updated Figure 6, Timing Diagram for LP_HS_DELAY_CNTRL Delay Parameters.
December 2013	01.2	Updated the BYTE_PACKETIZER Module Description section.
August 2013	01.1	Updated Table 8 title to TX Maximum Operating Frequencies by Configuration and added footnote.
	01.0	Initial release.