SubLVDS Image Sensor Receiver Submodule IP

User Guide

FPGA-IPUG-02023-1.3

March 2020

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

1. Introduction

The Lattice Semiconductor SubLVDS Image Sensor Receiver Submodule IP converts double data rate interface to pixel clock domain.

The subLVDS interface is primarily used in image sensors. It has one clock pair and more than one data pairs. The number of data pairs varies, depending on bandwidth requirement.

Compared to LVDS interface, SubLVDS features:

- Lower common mode of 0.9 V. The common mode for LVDS is 1.25 V. SubLVDS is typically powered by 1.8 V supply, while LVDS uses 2.5 V supply.
- Lower differential swing of ±150 mV. The differential swing for LVDS is ±175 mV.
- Source synchronous interface with the clock pair running at the same rate as the data. This is not a 7:1 interface.
- Clock that center-aligned with the data.

This user guide is for SubLVDS Image Sensor Receiver Submodule IP design version 1.3.

1.1. Quick Facts

Table 1.1 provides quick facts about the SubLVDS Image Sensor Receiver submodule IP for CrossLink™ and CrossLinkPlus™ devices.

Table 1.1. SubLVDS Image Sensor Receiver Submodule IP Quick Facts

		SubLVDS Receiver Submodule IP Configuration
		10-Lane Configuration
IP Requirements	FPGA Families Supported	CrossLink/CrossLinkPlus
Resource Utilization	Targeted Device	LIF-MD6000-6MG81I
	Data Path Width	10-bit (RAW10) or 12-bit (RAW12) input data
	LUTs	1362
	sysMEM™ EBRs	0
	Registers	1037
	HW MIPI Block	0
Design Tool Support	Lattice Implementation	Lattice Diamond® 3.11 SP1
	Synthesis	Lattice Synthesis Engine
		Synplify Pro® N-2018.03L-SP1-1
	Simulation	Aldec® Active HDL™ 10.5 Lattice Edition

1.2. Features

The key features of the SubLVDS Receiver Interface IP include:

- Supports 4, 6, 8, or 10 data lanes from an image sensor
- Supports 10-bit (RAW10) or 12-bit (RAW12) pixel widths
- Can generate XVS and XHS for image sensors operating in slave mode

Table 1.2. SubLVDS Receiver Submodule IP Features Summary

IP Configuration	Options
Number of Channels	1
Data type	RAW10, RAW12
Number of Rx Lanes	4, 6, 8, 10
Gearing	8, 16*

***Note**: Gear 16 option is only for 4-lane configuration.

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 most significant bit within the pixel data is the highest index.

1.3.3. Signal Names

Signal names that end with:

- _*n* are active low
- _*i* are input signals
- _*o* are output signals
- _*io* are bidirectional signals

2. Functional Descriptions

The SubLVDS Receiver submodule IP converts double data rate interface into pixel clock domain. The input interface of the design consists of a data bus and a clock in subLVDS interface format. The output interface consists of a 10-bit or 12-bit multi-pixel data, frame valid, line valid, data valid, and a pixel clock with a gearing of 1:8 or 1:16.

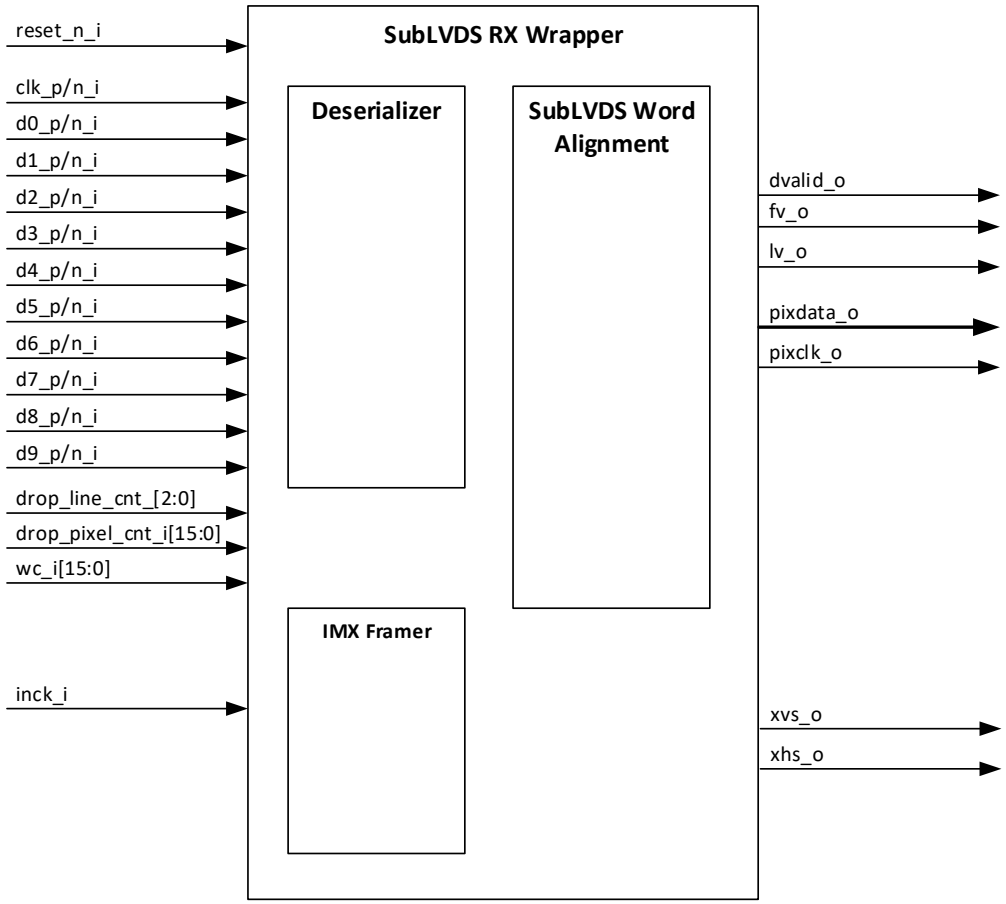


Figure 2.1. SubLVDS Receiver Submodule Top Level Block Diagram

Table 2.1. SubLVDS Receiver Submodule IP Pin Function Description

Port Name	Width	Dir	Type	Description
Deserializer				
reset_n_i	1	I	LSR	System active low asynchronous reset
clk_p_i	1	I	Clock	Positive subLVDS Input clock to subLVDS Rx Wrapper
clk_n_i	1	I	Clock	Negative subLVDS Input clock to subLVDS Rx Wrapper
d0_p_i	1	I	Data +	Positive subLVDS Input data lane 0 to subLVDS Rx Wrapper
d0_n_i	1	I	Data -	Negative subLVDS Input data lane 0 to subLVDS Rx Wrapper, complement of d0_p_i
d1_p_i	1	I	Data +	Positive subLVDS Input data lane 1 to subLVDS Rx Wrapper
d1_n_i	1	I	Data -	Negative subLVDS Input data lane 1 to subLVDS Rx Wrapper, complement of d1_p_i
d2_p_i	1	I	Data +	Positive subLVDS Input data lane 2 to subLVDS Rx Wrapper
d2_n_i	1	I	Data -	Negative subLVDS Input data lane 2 to subLVDS Rx Wrapper, complement of d2_p_i
d3_p_i	1	I	Data +	Positive subLVDS Input data lane 3 to subLVDS Rx Wrapper

Port Name	Width	Dir	Type	Description
d3_n_i	1	I	Data -	Negative subLVDS Input data lane 3 to subLVDS Rx Wrapper, complement of d3_p_i
d4_p_i	1	I	Data +	Positive subLVDS Input data lane 4 to subLVDS Rx Wrapper
d4_n_i	1	I	Data -	Negative subLVDS Input data lane 4 to subLVDS Rx Wrapper, complement of d4_p_i
d5_p_i	1	I	Data +	Positive subLVDS Input data lane 5 to subLVDS Rx Wrapper
d5_n_i	1	I	Data -	Negative subLVDS Input data lane 5 to subLVDS Rx Wrapper, complement of d5_p_i
d6_p_i	1	I	Data +	Positive subLVDS Input data lane 6 to subLVDS Rx Wrapper
d6_n_i	1	I	Data -	Negative subLVDS Input data lane 6 to subLVDS Rx Wrapper, complement of d6_p_i
d7_p_i	1	I	Data +	Positive subLVDS Input data lane 7 to subLVDS Rx Wrapper
d7_n_i	1	I	Data -	Negative subLVDS Input data lane 7 to subLVDS Rx Wrapper, complement of d7_p_i
d8_p_i	1	I	Data +	Positive subLVDS Input data lane 8 to subLVDS Rx Wrapper
d8_n_i	1	I	Data -	Negative subLVDS Input data lane 8 to subLVDS Rx Wrapper, complement of d8_p_i
d9_p_i	1	I	Data +	Positive subLVDS Input data lane 9 to subLVDS Rx Wrapper
d9_n_i	1	I	Data -	Negative subLVDS Input data lane 9 to subLVDS Rx Wrapper, complement of d9_p_i
pixclk_o	1	O	Clock	Pixel clock
SubLVDS Word Alignment				
dvalid_o	1	O	Flag	Data Valid detection signal, indicates valid pixel data
fv_o	1	O	Flag	Frame Valid detection signal, indicates valid frame
lv_o	1	O	Flag	Line Valid detection signal, indicates a valid line
pixdata_o	BUS_WIDTH* LANE_WIDTH	O	Data	Pixel data coming from parser module. Multi-pixel data bus
drop_line_cnt_i	3	I	Control	Number of dropped lines
drop_pixel_cnt_i	16	I	Control	Number of dropped pixels
wc_i	16	I	Control	Word count
IMX Framer				
inck_i	1	I	Clock	IMX Framer input clock. This clock is shared with the Sony Image Sensor.
xvs_o	1	O	Control	Sony slave readout vertical control signal
xhs_o	1	O	Control	Sony slave readout horizontal control signal

2.1. Interface and Timing Diagram

Figure 2.2 shows the timing of SubLVDS input interface. It shows the sync signal and data output timing during 10-bit length serial received from the image sensor.

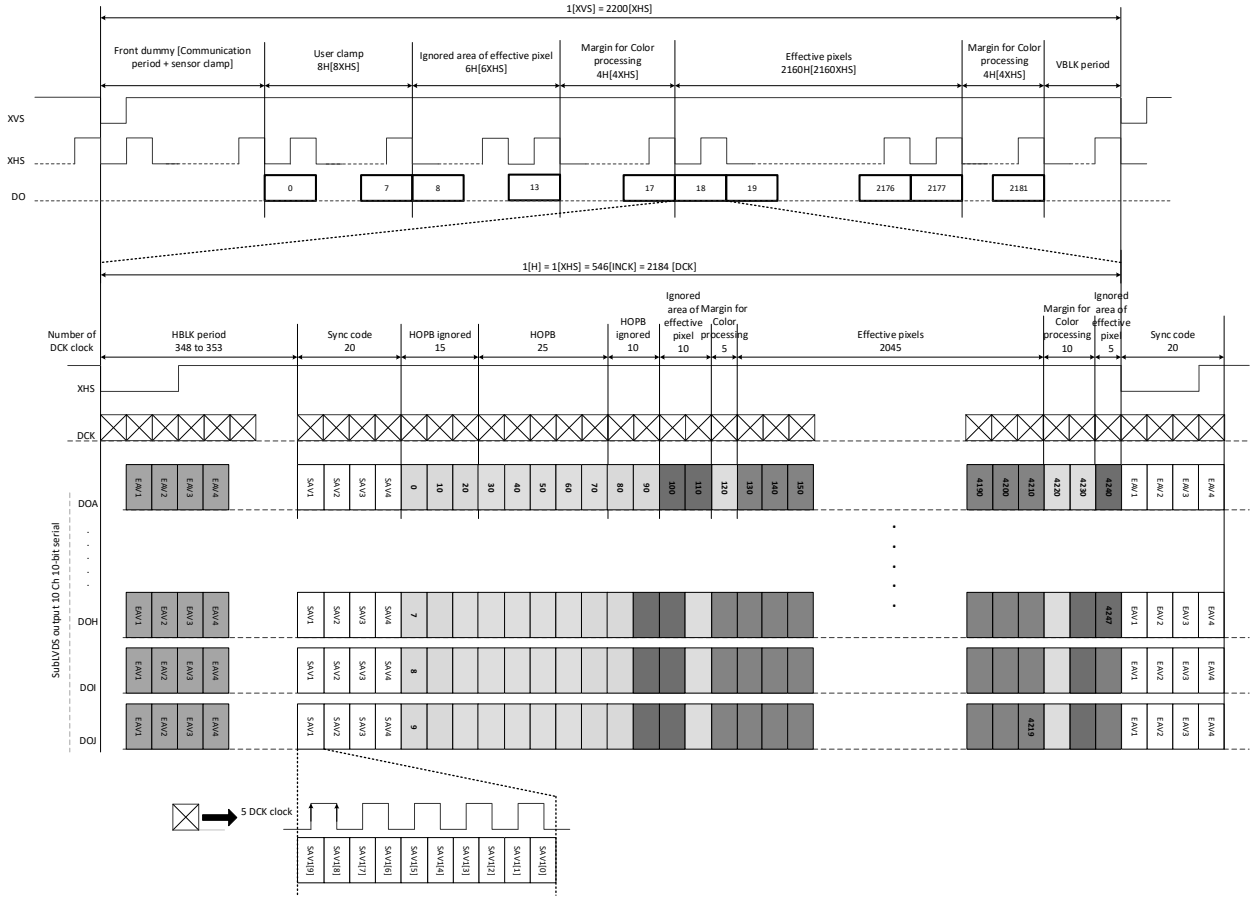


Figure 2.2. SubLVDS Input Bus Waveform

The horizontal and vertical timing of the received data are controlled by the XVS and XHS sync signals. The sync code is added before and after the pixel data. Table 2.2 lists the sync code details.

Table 2.2. Sync Code Details

SubLVDS Output Bit No.		Sync Code				
12-bit Output	10-bit Output	1st Word	2nd Word	3rd Word	4th Word	
11	9	1	0	0	1	
10	8	1	0	0	0	
9	7	1	0	0	V	1: Blanking line 0: Except blanking line
8	6	1	0	0	H	1: End sync code 2: Start sync code
7	5	1	0	0	P3	Protection bits
6	4	1	0	0	P2	
5	3	1	0	0	P1	
4	2	1	0	0	P0	
3	1	1	0	0	0	
2	0	1	0	0	0	
1	—	1	0	0	0	
0	—	1	0	0	0	

		Protection Bits			
V	H	P3	P2	P1	P0
0	0	0	0	0	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	0

2.2. Clock, Reset, and Initialization

2.2.1. Reset and Initialization

Active low reset is used in the design with synchronous release. Resets for each clock domain are synced to their respective clock domains.

The system reset (reset_n_i) is synchronized to the pixel clock domain, and it serves as a reset source for the SubLVDS Word Alignment module.

No special reset sequence is required in this IP.

2.2.2. Clock Domains and Clock Domain Crossing

The Rx clock input (clk_p_i) is from an external source (image sensor) and should be connected to a dedicated SubLVDS edge clock pin. The Deserializer block generates a pixel clock (pixclk_o) with a gearing of 1:8 or 1:16 for the pixel data.

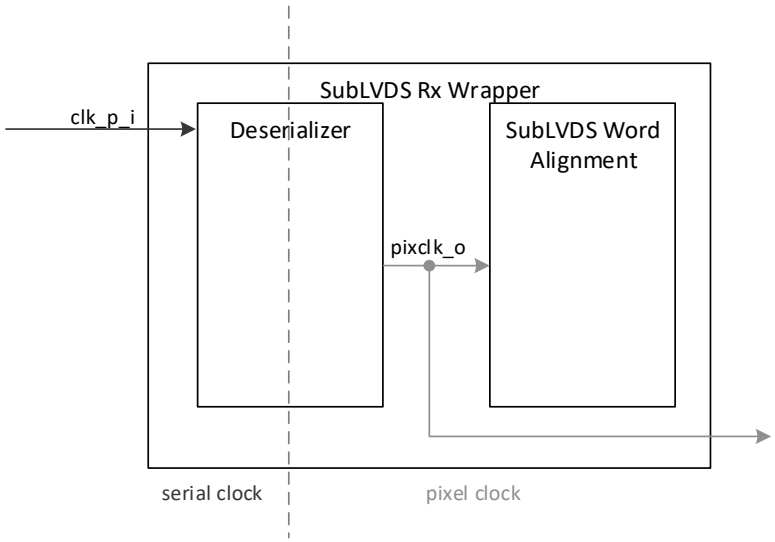


Figure 2.3. Clock Domain Crossing Block Diagram

Table 2.3. Clock Domain Crossing

Clock Domain Crossing	Handling Approach
SubLVDS Serial Clock to Pixel Clock	1:8/1:16 gearbox DDR Hard IP

The general formula for computing the required clocks of the system:

Rx line rate (total) $= total\ pixels(active + blanking) * frame\ rate * bits\ per\ pixel$

Rx line rate (per lane) $= \frac{RX\ line\ rate\ (total)}{no.\ of\ Rx\ lane}$

Rx input clock $= \frac{RX\ line\ rate\ (per\ lane)}{2}$

Pixel clock $= \frac{Rx\ input\ clock}{gearing}$

Note: gearing = 4 if 1:8 gearing; 8 if 1:16 gearing

2.3. Design and Module Description

Figure 2.4 shows the detailed block diagram of SubLVDS receiver wrapper.

The Deserializer block converts each double data rate lane (d*_p_i signals) to a single data rate 8-bit or 16-bit at a slower operating speed within a system.

The word alignment module receives the 8-bit (1:8 gearing) or 16-bit (1:16 gearing) deserialized data (deser_q_o signal) and converts it to 10-bit or 12-bit pixel data according to the set configuration of data type (RAW10 or RAW12). The output of the module is a multi-pixel bus (pixdata_o), pixel clock (pixclk_o), a dvalid_o, fv_o, and lv_o control signals.

The IMX Framer module is used for Image Sensors that operate in slave mode. For master mode image sensors, this module is not needed.

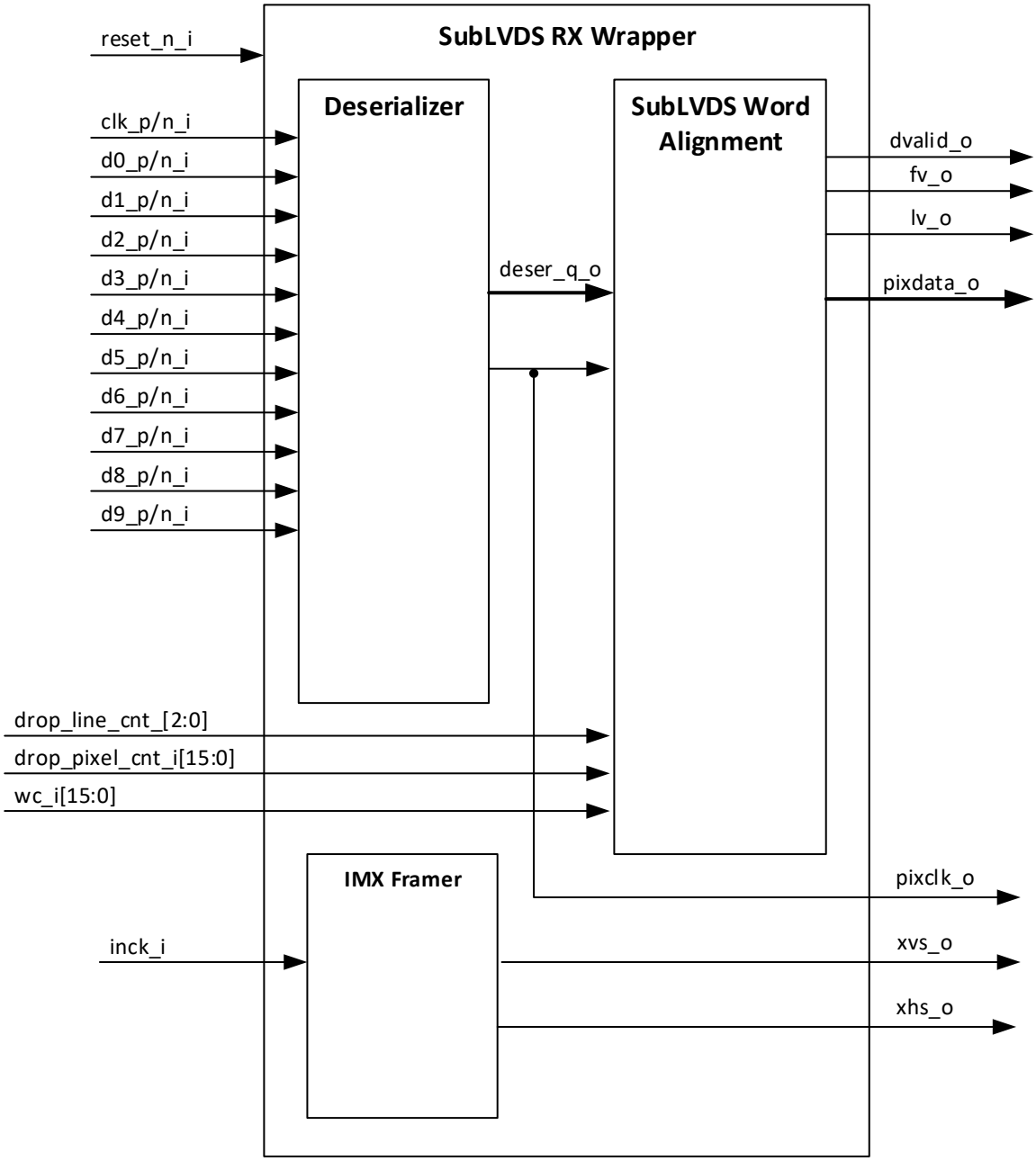


Figure 2.4. SubLVDS Rx Wrapper detailed Block Diagram

Table 2.4. Deserializer and Word Alignment Module Pin List

Port Name	Width	Dir	Type	Description
Deserializer				
reset_n_i	1	I	LSR	System active low asynchronous reset
clk_p_i	1	I	Clock	Positive subLVDS Input clock to subLVDS Rx Wrapper
clk_n_i	1	I	Clock	Negative subLVDS Input clock to subLVDS Rx Wrapper
d0_p_i	1	I	Data +	Positive subLVDS Input data lane 0 to subLVDS Rx Wrapper
d0_n_i	1	I	Data -	Negative subLVDS Input data lane 0 to subLVDS Rx Wrapper, complement of d0_p_i
d1_p_i	1	I	Data +	Positive subLVDS Input data lane 1 to subLVDS Rx Wrapper
d1_n_i	1	I	Data -	Negative subLVDS Input data lane 1 to subLVDS Rx Wrapper, complement of d1_p_i
d2_p_i	1	I	Data +	Positive subLVDS Input data lane 2 to subLVDS Rx Wrapper
d2_n_i	1	I	Data -	Negative subLVDS Input data lane 2 to subLVDS Rx Wrapper, complement of d2_p_i
d3_p_i	1	I	Data +	Positive subLVDS Input data lane 3 to subLVDS Rx Wrapper
d3_n_i	1	I	Data -	Negative subLVDS Input data lane 3 to subLVDS Rx Wrapper, complement of d3_p_i
d4_p_i	1	I	Data +	Positive subLVDS Input data lane 4 to subLVDS Rx Wrapper
d4_n_i	1	I	Data -	Negative subLVDS Input data lane 4 to subLVDS Rx Wrapper, complement of d4_p_i
d5_p_i	1	I	Data +	Positive subLVDS Input data lane 5 to subLVDS Rx Wrapper
d5_n_i	1	I	Data -	Negative subLVDS Input data lane 5 to subLVDS Rx Wrapper, complement of d5_p_i
d6_p_i	1	I	Data +	Positive subLVDS Input data lane 6 to subLVDS Rx Wrapper
d6_n_i	1	I	Data -	Negative subLVDS Input data lane 6 to subLVDS Rx Wrapper, complement of d6_p_i
d7_p_i	1	I	Data +	Positive subLVDS Input data lane 7 to subLVDS Rx Wrapper
d7_n_i	1	I	Data -	Negative subLVDS Input data lane 7 to subLVDS Rx Wrapper, complement of d7_p_i
d8_p_i	1	I	Data +	Positive subLVDS Input data lane 8 to subLVDS Rx Wrapper
d8_n_i	1	I	Data -	Negative subLVDS Input data lane 8 to subLVDS Rx Wrapper, complement of d8_p_i
d9_p_i	1	I	Data +	Positive subLVDS Input data lane 9 to subLVDS Rx Wrapper
d9_n_i	1	I	Data -	Negative subLVDS Input data lane 9 to subLVDS Rx Wrapper, complement of d9_p_i
pixclk_o	1	O	Clock	Pixel clock
SubLVDS Word Alignment				
dvalid_o	1	O	Flag	Indicates valid pixel data
fv_o	1	O	Flag	Indicates valid frame
lv_o	1	O	Flag	Indicates a valid line
pixdata_o	BUS_WIDTH*LANE_WIDTH	O	Data	Multi-pixel data bus
IMX Framer				
inck_i	1	I	Clock	IMX Framer input clock. This clock is shared with the Sony Image Sensor.
xvs_o	1	O	Control	Sony slave readout vertical control signal
xhs_o	1	O	Control	Sony slave readout horizontal control signal

2.3.1. Detailed Description

Figure 2.5 shows the detailed block diagram of Deserializer block when LANE_WIDTH = 4. It is composed of GDDRx4 gearbox for 1:8 gearing and GDDRx16 gearbox for 1:16 gearing. The number of DDR components is determined by the parameter LANE_WIDTH. There is only one instance of CLKDIV and ECLKSYNC in any number of Rx lanes, which means there is only one source of ECLK (input serial clock) and SCLK (pixel clock).

To avoid additional clock resource, GDDR_SYNC is not used in this module. Alignment of clock and data is ensured in the subLVDS word alignment module.

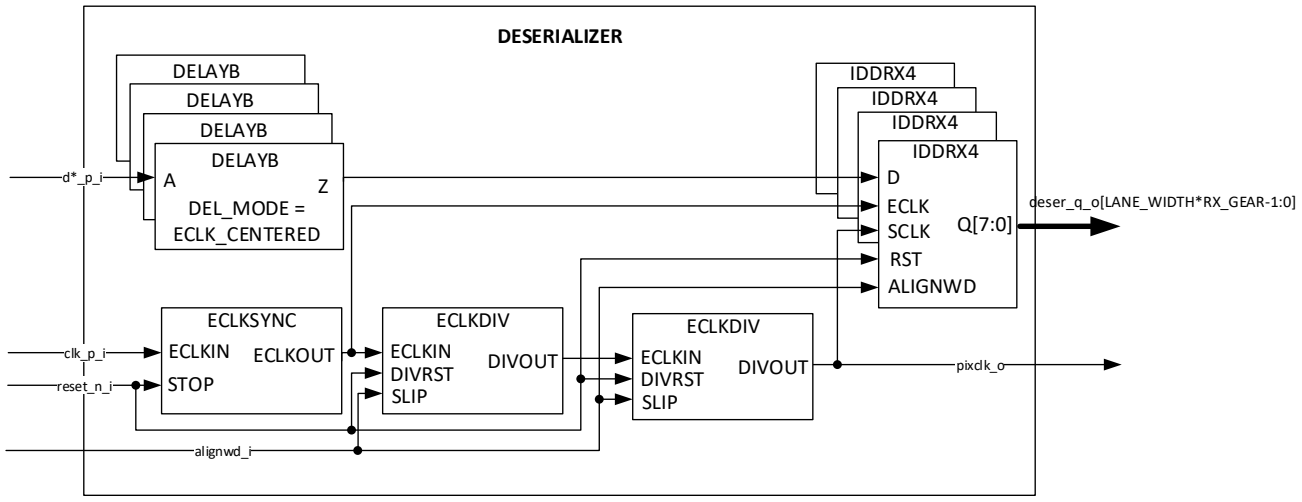


Figure 2.5. Deserializer of 1:8 Gearing Block Diagram

Figure 2.6 shows a detailed block diagram of Deserializer block when LANE_WIDTH = 4 for the 1:16 gearing case. FIFO module is used to get to 1:16 gearing.

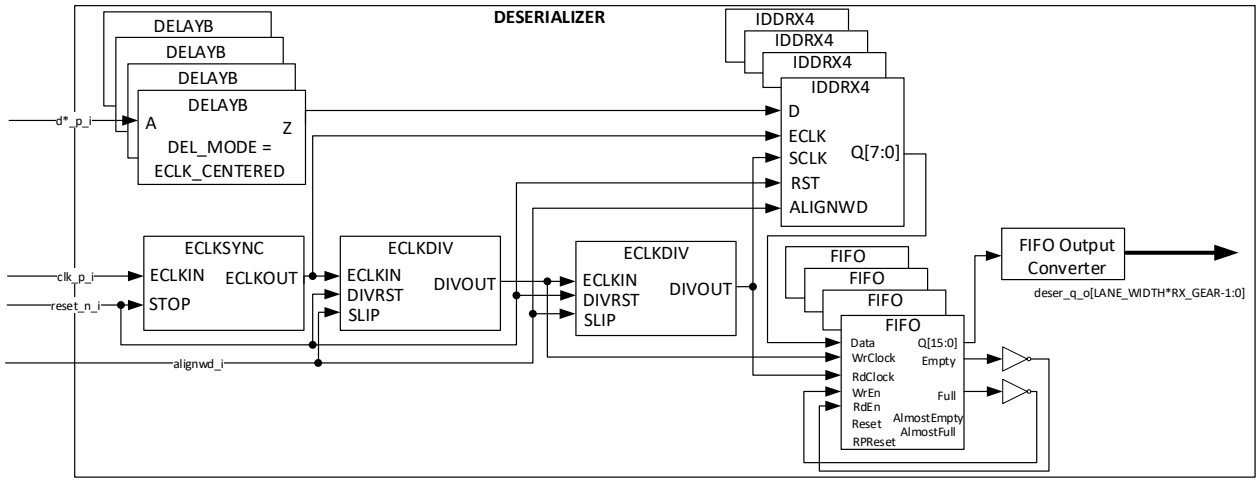


Figure 2.6. Deserializer of 1:16 Gearing Block Diagram

Figure 2.7 shows the detailed block diagram of SubLVDS Word Alignment block when LANE_WIDTH = 4. The number of word_aligner instances is determined by the parameter LANE_WIDTH. Sync codes are embedded in each serial data lane by the Sony Image Sensor. The word aligner block detects these sync codes and aligns the deserialized data to a 10-bit or 12-bit pixel data.

The 10-bit or 12-bit data are fed to the parser block. It checks the recognition (sync) codes from the beginning (SAV) and the end (EAV) of each packet, if they are part of an active video line or not. The fv_o goes high at the beginning of an active video frame, and low at the end of the frame. Similarly, the lv_o goes active high or low at the beginning or end of an active video line, respectively. The dvalid_o control signal goes active high on clock cycles that have valid pixel data.

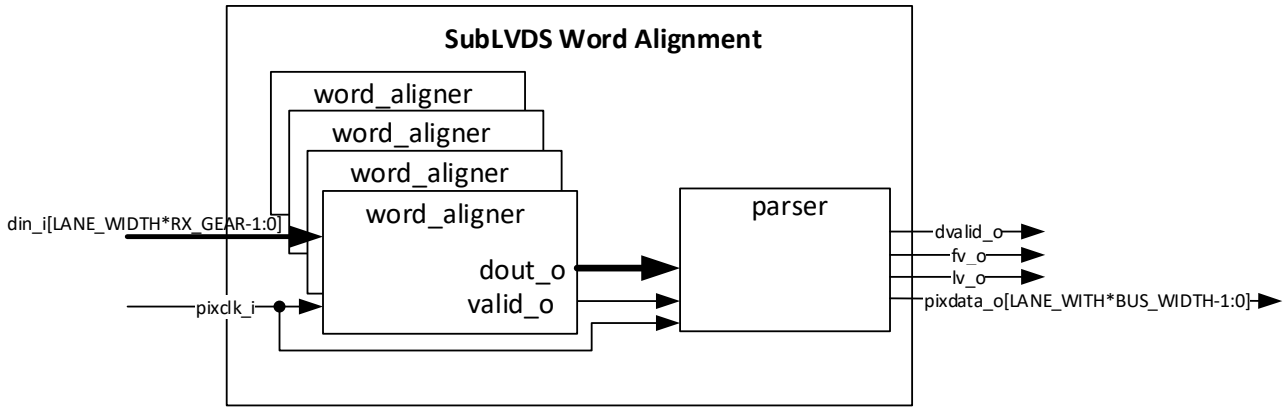


Figure 2.7. SubLVDS Word Alignment Block Diagram

Table 2.5. Indicator States

Sync Code	fv State	lv State
SAV (valid line)	1	1
EAV (valid line)	1	0
SAV (invalid line	0	0
EAV (invalid line)	0	0

The IMX Framer module is for Sony Image Sensors that operate in Slave mode. It provides a control mechanism for the rate at which each line and frame is read out. Timing of these two signals is defined in the Sony Image Sensor datasheet.

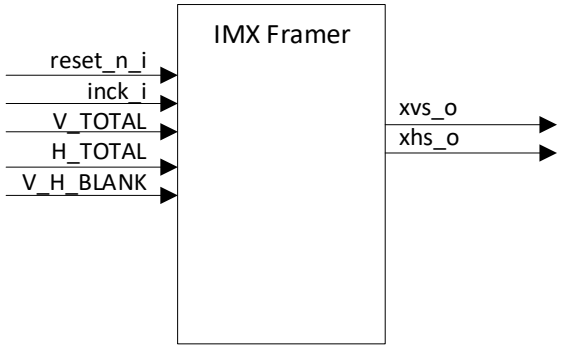


Figure 2.8. IMX Frame Block Diagram

2.4. Sample Configurations

Waveforms below show the SubLVDS Image Sensor Receiver output behavior with different Word Count and Dropped Pixel Count but having same Number of Rx Lanes == 4, Data Type == RAW10 and number of pixels sent by sensor is 40 pixels.

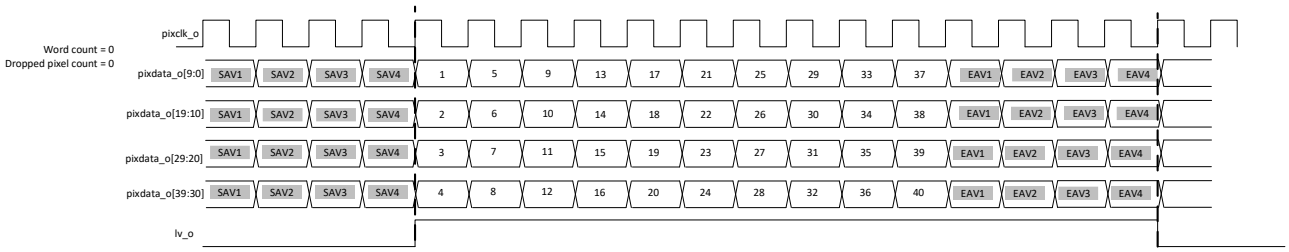


Figure 2.9. Output Waveform when Word Count == 0 and Dropped Pixel Count ==0

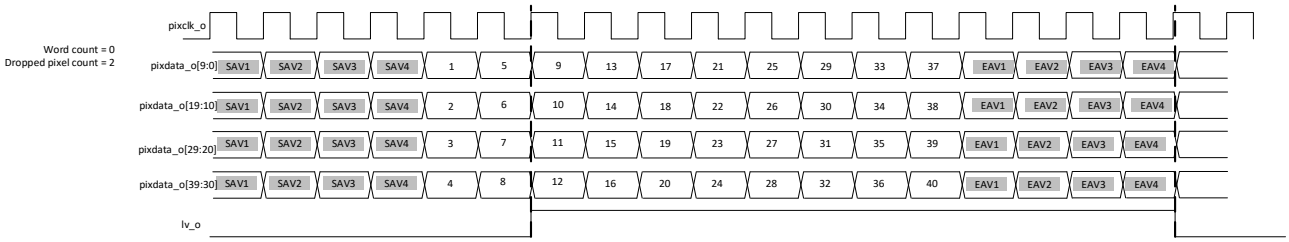


Figure 2.10. Output Waveform when Word Count == 0 and Dropped Pixel Count ==2

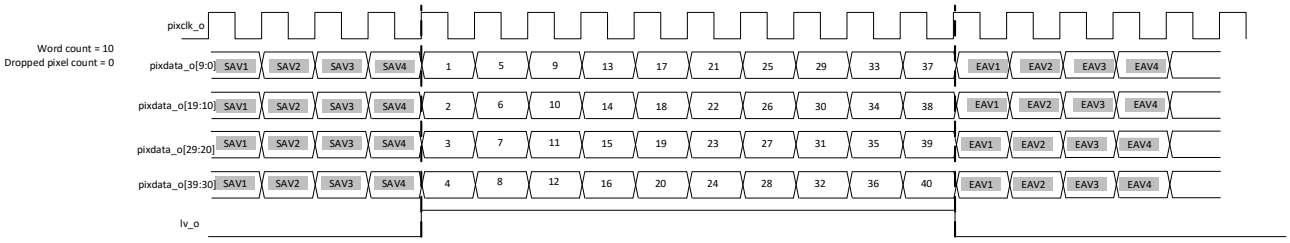


Figure 2.11. Output Waveform when Word Count == 10 and Dropped Pixel Count ==0

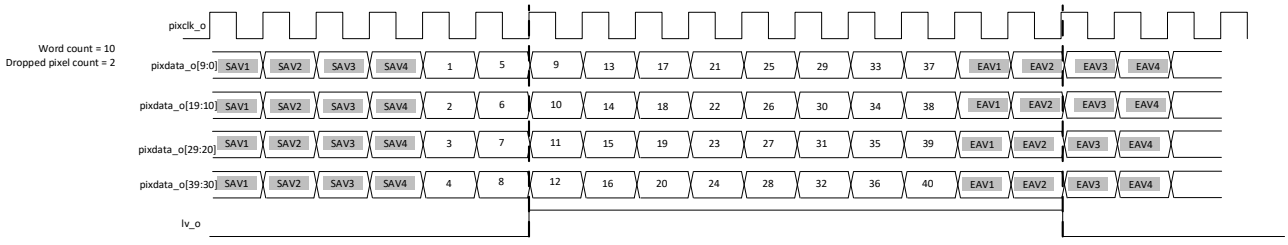


Figure 2.12. Output Waveform when Word Count == 10 and Dropped Pixel Count ==2

3. Parameter Settings

This section lists the compiler directives and parameters used to configure the SubLVDS Image Sensor Receiver Submodule IP.

Table 3.1 lists the parameters used to generate the SubLVDS Receiver Submodule IP. All parameters are either set automatically or input in the user interface during the SubLVDS Receiver Submodule IP generation.

Table 3.1. SubLVDS Image Sensor Receiver Submodule IP Parameter Settings in User Interface

User Interface Config. Tab	Tab Sub-sections	Parameter	Attribute	Options	Description
Configuration	Receiver	Number of Rx Lanes	User-Input	4, 6, 8, 10	Generates subLVDS I/O.
		Rx Gear	Read-Only	8, 16	Specifies the Rx gearing. Only the 4-lane configuration has the option to choose between 8 or 16 gearing.
	Clock	Rx Line Rate	User-Input	<Value>	Target Rx line rate per lane.
		SubLVDS Input Clock Frequency	Read-Only	<Value>	SubLVDS clock. Automatically computed based on target Rx line rate.
		Pixel Clock Frequency	Read-Only	<Value>	Pixel clock. Automatically computed based on target Rx line rate.
	Data	Dropped Line Mode	User-Input	Static, Dynamic	Allows user to choose between Static (predetermined values for the number of dropped lines) and Dynamic (user determines values via an added port) Modes
		Dropped Line Count	Read-Only if Dropped Line Mode is Dynamic and User-Input if Dropped Line Mode is Static	0 – 7 (x Line Count)	Determines the number of lines to be dropped at the start of the frame
		Dropped Pixel Mode	User-Input	Static, Dynamic	Allows user to choose between Static (predetermined values for the number of dropped pixels) and Dynamic (user determines values via an additional port) Modes
		Dropped Pixel Count	Read-Only if Dropped Pixel Mode is Dynamic and User-Input if Dropped Pixel Mode is Static	0 – 65535 (x Number of Lanes)	Crops the number of pixels after SAV (the OPB and OPB ignore pixels). Kindly refer to the Sony IMX sensor specification for information on these pixels. The input value should be equal to desired number of pixels to drop / *Number of RX Lanes*. For example, to drop 8 pixels when *Number of RX Lanes* == 4, the input to *Dropped Pixel Count* should be 2.
	Data	Word Count Mode	User-Input	Static, Dynamic, Off	Allows user to choose between Static (predetermined values for the word count), Dynamic (user determines values via an additional port) and Off (logic not used) Modes

User Interface Config. Tab	Tab Sub-sections	Parameter	Attribute	Options	Description
		Word Count	User-Input if Word Count Mode is Static, otherwise, this is Read-Only	0 – 65535 (x Number of Lanes)	Number of active video pixels per line after the dropped pixels (when *Dropped Pixel Count* > 0). Reducing this effectively drops the OPB ignore bits right before the EAV. Kindly refer to the Sony IMX sensor specification for information on these pixels. If *Word Count* == 0 and *Dropped Pixel Count* == 0, the total number of pixels coming out of the design is the total number of active pixels sent by sensor + EAV pixels. The input value should be equal to desired total number of pixels / *Number of RX Lanes*. For example, if the desired total number of pixels is 40 and Number of RX Lane == 4, *Word Count* should be 10.
Video	Video Packet	Data type	User-Input	RAW10 RAW12	Selects desired data type.
	IMX Framer Settings	Image Sensor Mode	User-input	Master Slave	Sets the mode of the image sensor. In slave mode, it enables the IMX framer.
		V_TOTAL	User-input	Decimal value	Sets the number of lines XVS is driven high. This parameter is needed in slave mode. In master mode, disregard this entry.
		H_TOTAL	User-input	Decimal value	Sets the number of INCK clocks XHS is driven high. This parameter is needed in slave mode. In master mode, disregard this entry.
		V_H_BLANK	User-input	Decimal value	Sets the number of INCK clocks XVS and XHS is driven low. This parameter is needed in slave mode. In master mode, disregard this entry.

4. IP Generation and Evaluation

This section provides information on how to generate the Lattice SubLVDS Receiver Submodule IP code using Lattice Diamond Clarity Designer and how to run simulation, synthesis, and hardware evaluation.

4.1. Licensing the IP

The SubLVDS Image Sensor Receiver IP is available free of charge , but an IP-specific license is required to enable full, unrestricted use of the SubLVDS Image Sensor Receiver IP in a complete, top level design.

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/CrossLinkPlus IP for your design.

You may download and generate the SubLVDS Receiver IP and fully evaluate the IP through functional simulation and implementation (synthesis, map, place and route) without an IP license. The SubLVDS Receiver IP also supports Lattice IP hardware evaluation capability. See the Hardware Evaluation section 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 SubLVDS Image Sensor Receiver 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.

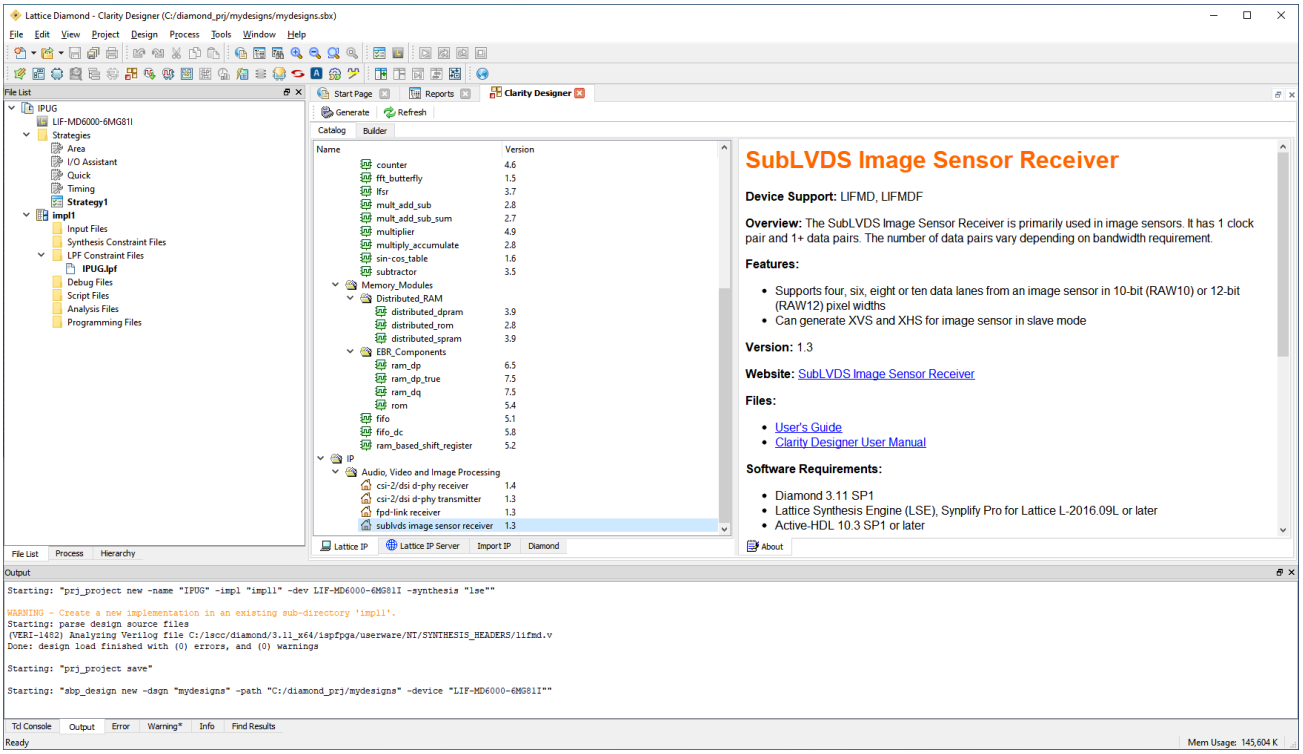


Figure 4.1. Clarity Designer Window

4.3. Generating IP in Clarity Designer

The Clarity Designer tool is used to customize modules and IPs and place them into the device 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 procedure for generating SubLVDS Receiver IP in Clarity Designer is described below. Clarity Designer can be started from Lattice Diamond design environment.

To start Clarity Designer:

1. Create a new Diamond project for CrossLink or CrossLinkPlus 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** – Select this to create a new Clarity Design project directory in which the SubLVDS Receiver IP is generated.
   - **Design Location** – Clarity Design project directory path.
   - **Design Name** – Clarity Design project name.
   - **HDL Output** – Hardware Description Language Output Format (Verilog HDL).

   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.

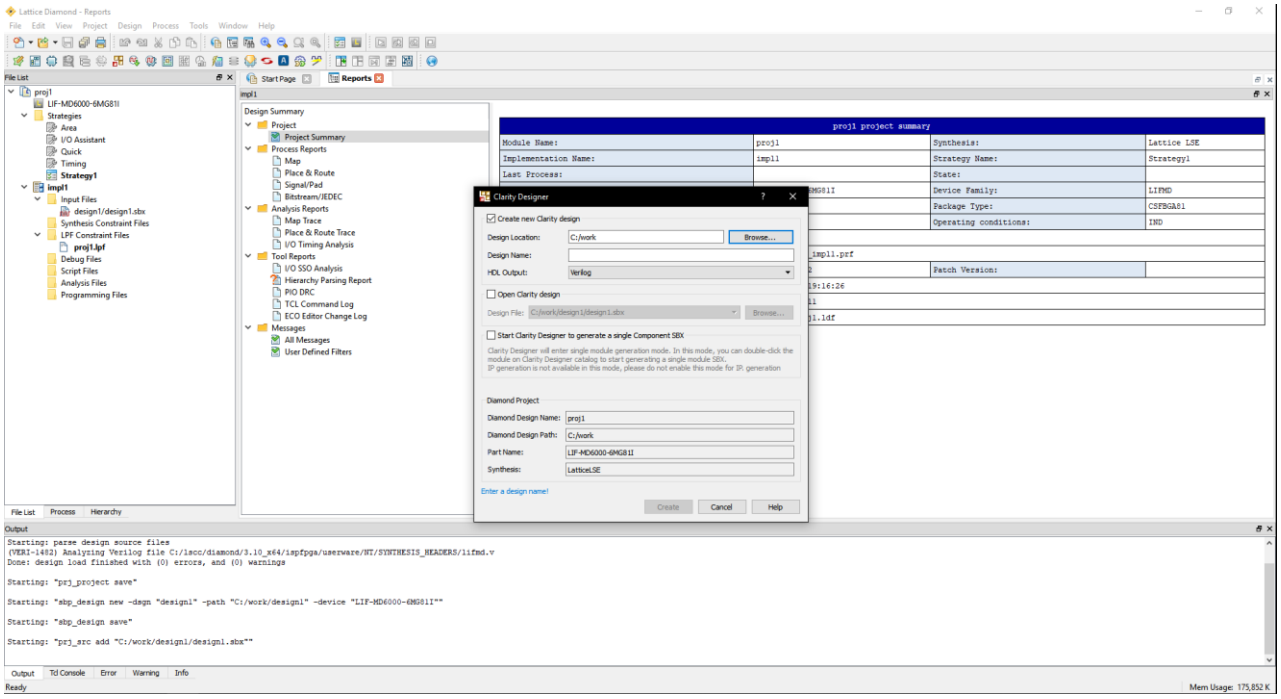


Figure 4.2. Starting Clarity Designer from Diamond Design Environment

To configure SubLVDS Receiver IP in Clarity Designer:

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

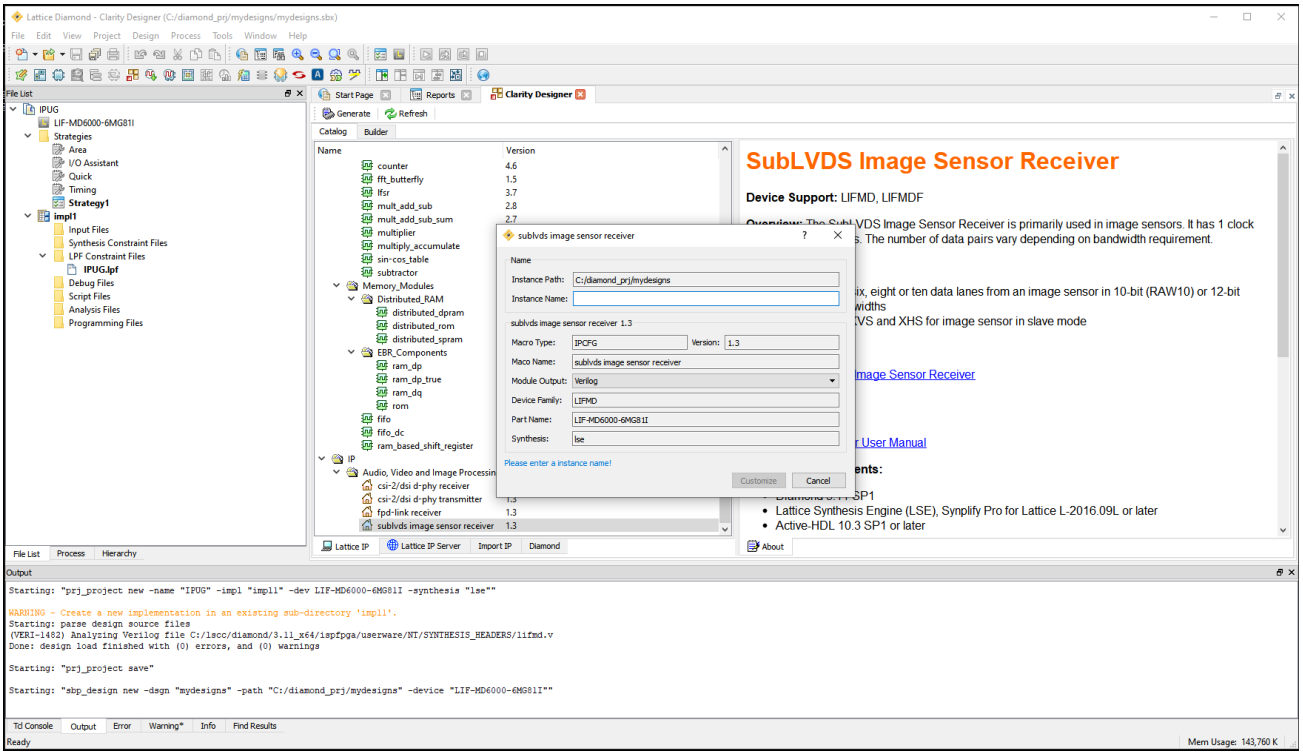
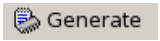


Figure 4.3. Configuring SubLVDS Receiver IP in Clarity Designer

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.
4. Input valid values in the required fields in the **Configuration** tab.
5. Go to **Video** tab and input valid values in the required fields. See Figure 4.5.
6. After selecting the required parameters, click the **Configure** button.
7. Click **Close**.
8. Click 

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.

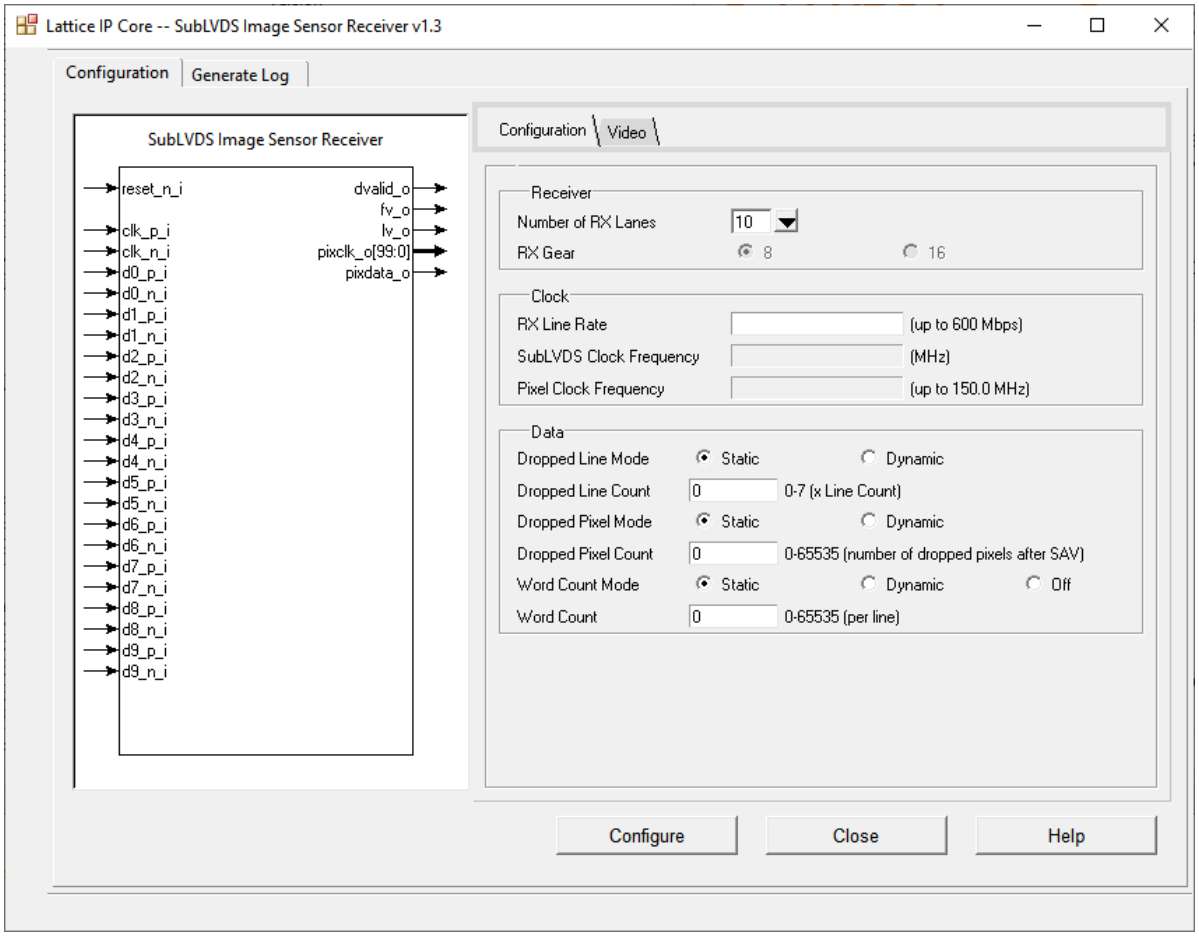


Figure 4.4. Configuration Tab in IP User Interface

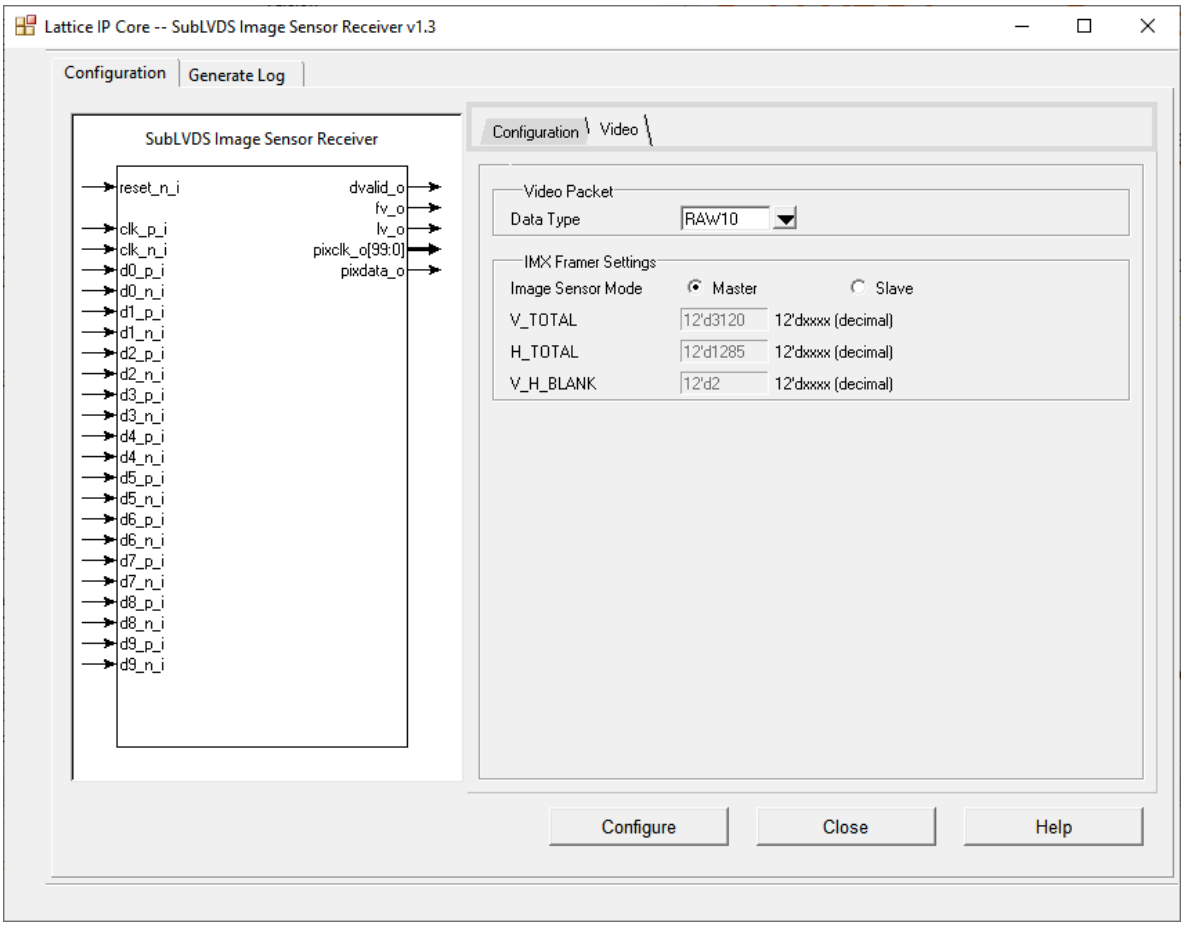


Figure 4.5. Video Tab in IP User Interface

4.4. Generated IP Directory Structure and Files

Figure 4.6 shows the directory structure of generated IP and supporting files.

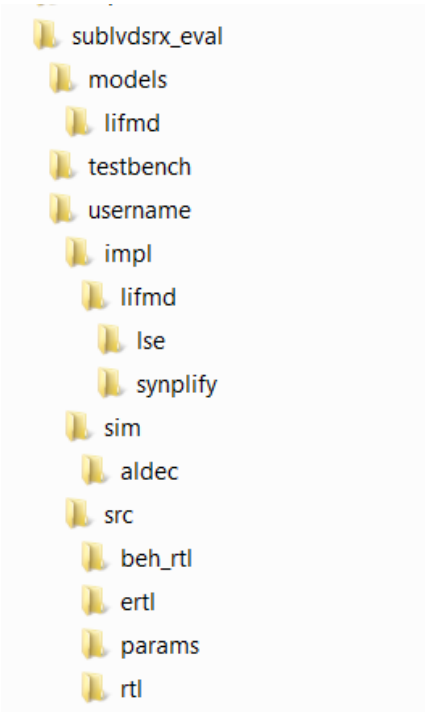


Figure 4.6. SubLVDS Receiver IP Directory Structure

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

Aside from the files listed in the tables, most of the files required to evaluate the SubLVDS Image Sensor Receiver IP are available under the directory *\<sublvdsrx_eval>*, including the simulation model. Lattice Diamond project files are also included under the folder at *\<sublvdsrx_eval>\<instance_name>\impl\<device_family>\<synthesis_tool>*, where *<device_family>* can either be **lifmd** for CrossLink or **lifmdf** for CrossLinkPlus devices.

The *\<instance_name>* folder (username folder in Figure 4.6) 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.

The folder *\<instance_name>*, the *\sublvdsrx_eval* and sub directories provide files supporting SubLVDS Receiver IP evaluation that includes files/folders with content that is constant for all configurations of the SubLVDS Receiver IP. The *\sublvdsrx_eval* directory is created by Clarity Designer the first time the IP is generated, when multiple SubLVDS Receiver IPs are generated in the same root directory and updated each time the IP is regenerated.

You can use the prebuilt Diamond projects provided at *\<project_root>\sublvdsrx_eval\<instance_name>\impl\<device_family>\<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. Running Functional Simulation

To run simulations using Active-HDL:

1. Under the **Tools** menu in Diamond, select **Active-HDL**.
2. In **Active-HDL** window, under the **Tools** tab, select **Execute Macro**.
3. Select the **.do** file **\<project_dir>\sublvdsrx_eval\<instance_nam>\sim\aldec*_run.do** file.
4. Click **OK**.
5. Wait for simulation to finish.
6. To override default TB parameters, modify the **\<project_dir>\sublvdsrx_eval\testbench\tb_setup_params.v** file.

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
VFRONT_BLNK	Vertical front blanking
VREAR_BLNK	Vertical rear blanking
XHS_ASRT	Number of clock DCK cycles the xhs signal is asserted
XHS_PERIOD	XHS period in terms of DCK cycles
INIT_DRIVE_DELAY	Delay from reset deassertion or tinit_done assertion before model starts driving data
IMXFRAMER	Used to enable sampling of xhs and xvs when design is in slave mode

4.6. Simulation Strategies

This section describes the simulation environment which demonstrates basic SubLVDS Receiver functionality. Figure 4.7 shows the block diagram of simulation environment.

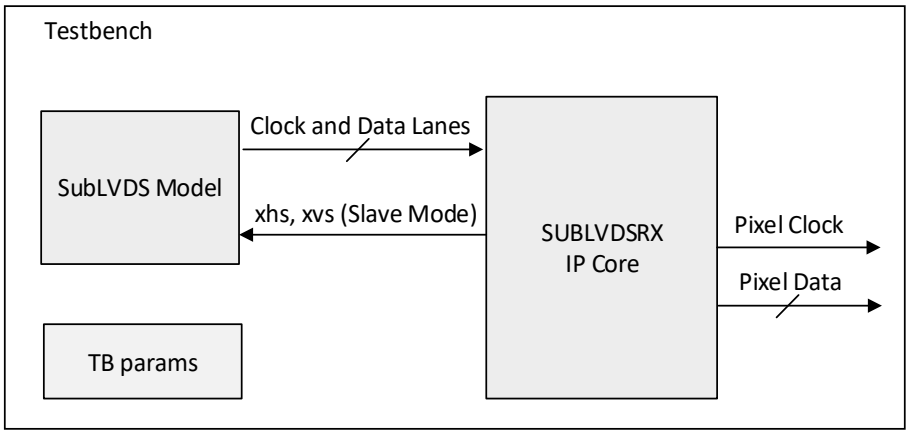


Figure 4.7. Simulation Environment Block Diagram

4.7. Simulation Environment

The simulation environment is made up of a Sublvds model instance connected to the input of SubLVDS Image Sensor receiver IP core instance in the testbench. The SubLVDS model is configured based on the SubLVDS Image Sensor Receiver IP configurations and testbench parameters. It can be configured as 4, 6, 8 or 10 Rx lanes. The testbench waits for user-programmable delay (tinit_delay) before transmitting the subLVDS data to the input of the design.

The testbench also provides xhs and xvs signals to the SubLVDS model when the design is configured as master. Otherwise, the model uses the xhs and xvs signals from the design as reference for data transmission when configured as slave. Refer to *tb_setup_params.v* for details of other TB parameters.

Figure 4.8 shows an example simulation of SubLVDS Image Sensor receiver in slave mode.

Figure 4.8. Sony Sensor in Slave Mode Configuration

You should set initial driving delay before the testbench starts sending out data to the IP core. In this example, the design is configured as slave, which means it is also transmitting the XVS and XHS signals. The SubLVDS model monitors the XVS signal assertion before starting to transmit data to the IP core.

Figure 4.9 shows an example simulation of SubLVDS Image Sensor receiver in master mode.

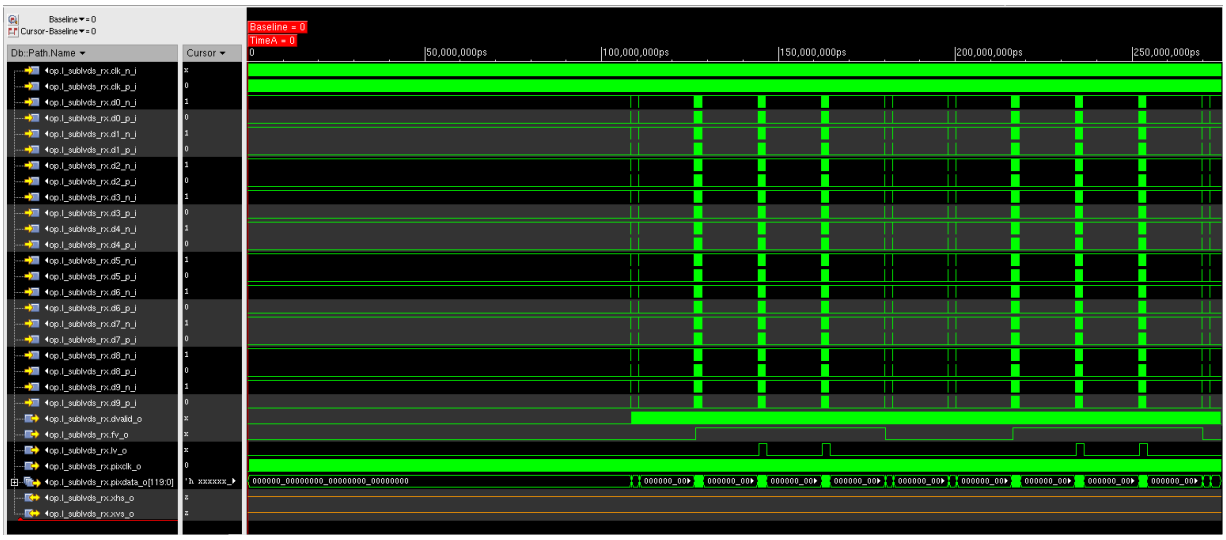


Figure 4.9. Sony Sensor in Master Mode Configuration

4.8. Instantiating the IP

The core modules of the SubLVDS Image Sensor Receiver IP are synthesized and provided in NGO format with black box Verilog HDL source files for synthesis. A Verilog HDL source file named *<instance_name>_sublvds_rx.v* instantiates the black box of core modules. The top-level file *<instance_name>.v* instantiates *<instance_name>_sublvds_rx.v*.

A Verilog HDL 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 other Verilog HDL source files are provided in *\<project_dir>*. These files are refreshed each time the IP is regenerated.

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 the *\<project_dir>\sublvdsrx_eval\<instance_name>\impl\<device_family>\lse* or *\<project_dir>\sublvdsrx_eval\<instance_name>\impl\<device_family>\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*, which is located in the *\<project_dir>\sublvdsrx_eval\<instance_name>\impl\<device_family>\<synthesis_tool>* directory.

To use the pre-built Diamond project files:

1. Choose **File > Open > Project**.
2. In the **Open Project** dialog box browse to **\<project_dir>\sublvdsrx_eval\<instance_name>\impl\<device_family>\<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 SubLVDS Image Sensor Receiver IP supports Lattice IP hardware evaluation capability. You can create versions of the IP that operate in hardware for a limited period of time without 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. The updated IP can be used 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** from the menu as shown in Figure 4.10.

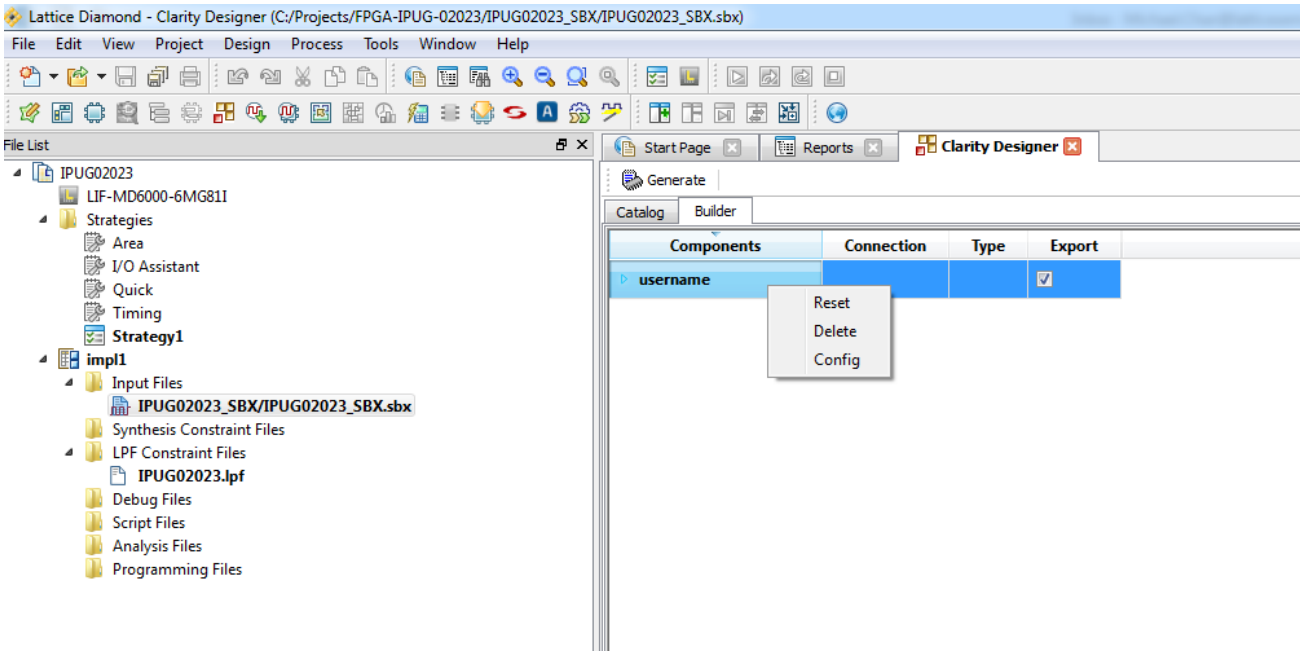


Figure 4.10. IP Regeneration in Clarity Designer

2. The IP Configuration user interface is displayed. Change the parameters as required and click the **Configure** button.
3. Click Generate in the toolbox. Clarity Designer regenerates all the instances which are reconfigured.

References

For more information about CrossLink and CrossLinkPlus devices, refer to CrossLink Family Data Sheet (FPGA-DS-02007) and CrossLinkPlus Family Data Sheet (FPGA-DS-02054).

Software documentation:

- Clarity Designer User Manual
- Lattice Diamond User Guide

Technical Support Assistance

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

Appendix A. Resource Utilization

Table A.1 lists resource utilization information for Lattice CrossLink FPGA using the SubLVDS Receiver IP.

Clarity Designer is the Lattice IP configuration utility, and is included as a standard feature of Lattice Diamond tool. For details about the usage of Clarity Designer, refer to 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]
10-lane configuration	994	1362	1037	0	142.450	75
8-lane configuration	844	1163	894	0	142	90
6-lane configuration	684	964	661	0	142.005	75
4-lane configuration	528	745	473	0	149.499	112

Notes:

1. Performance and utilization data target an LIF-MD6000-6MG81I device using Lattice Diamond 3.9 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 CrossLink family. This does not show all possible configurations of the SubLVDS Receiver IP.
2. The f_{MAX} values are based on pixel clock. Actual fmax are attained from MAP trace report.

Appendix B. What is Not Supported

The IP does not support configuration through registers.

Revision History

Revision 1.3, IP Version 1.3, March 2020

Section	Change Summary
Introduction	Changed IP design version from 1.x to 1.3.

Functional Descriptions	• Updated the following figures:
	  • Figure 2.1. SubLVDS Receiver Submodule Top Level Block Diagram
	  • Figure 2.4. SubLVDS Rx Wrapper detailed Block Diagram
	  • Figure 2.5. Deserializer of 1:8 Gearing Block Diagram
	  • Figure 2.7. SubLVDS Word Alignment Block Diagram
	• Added Figure 2.6. Deserializer of 1:16 Gearing Block Diagram.
	• Changed Table 2.2 header to SubLVDS Output Bit No.
	• Updated Table 2.4. Deserializer and Word Alignment Module Pin List.
	• Removed Table 2.5. SubLVDS Rx Top Level Parameter List.

Parameter Settings	• Updated section heading. • Updated Table 3.1. SubLVDS Image Sensor Receiver Submodule IP Parameter Settings in User Interface • Removed Compiler Directives section.
IP Generation and Evaluation	• Updated user interface screenshots. • Removed paragraph below Figure 4.9.

Revision 1.2, IP Version 1.3, October 2019

Section	Change Summary
Disclaimer	Newly added section.
All	• Added CrossLinkPlus device support. • Minor adjustments in style and formatting.
References	Updated.

Revision 1.1, IP Version 1.0, April 2019

Section	Change Summary
Introduction	Specified that this user guide can be used for IP design versions 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 1.0, IP Version 1.0, July 2017

Section	Change Summary
All	Initial release.

www.latticesemi.com