SLVS-EC Sensor to PCIe Bridge with CertusPro-NX

Reference Design and Demo

FPGA-RD-02261-1.0

September 2022

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, with all faults and associated risk the responsibility entirely of the 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

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
DT	Data Type
EBR	Embedded Block RAM
IP	Intellectual Property
ISP	Image Signal Processor
LUT	Look Up Table
PLL	Phase Locked Loop
RX	Receiver
SLVS-EC	Scalable Low Voltage Signaling with Embedded Clock
TX	Transmitter

Supported Device and IP

This reference design supports the following devices with IP versions.

Device Family	Part Number	Compatible IP
CertusPro-NX	LFCPNX-100-7LFG672I	SLVS-EC Receiver IP version 1.2.0

The IPs above are supported by Lattice Radiant™ software version 3.2.1.217.3 or later.

1. Introduction

The SLVS-EC to PCIe Bridge Reference Design consists of the SLVS-EC Receiver IP to receive CMOS sensor input, a ISP, line buffers, and a DMA plus PCIe module to send the video data out to the host machine. This design is similar to automotive and machine vision designs. This document describes the architecture and function of the reference design applicable to the Lattice CertusPro-NX device.

1.1. Features List

The key features of the SLVS-EC to PCIe Bridge Reference Design are:

- Compliant with SLVS-EC Protocol specification v2.0
- Back compatibility with SLVS-EC Protocol specification v1.2
- Supports PCIe Gen1 x4 and Gen2 x4
- Supports SLVS-EC RAW8 video format

1.2. Block Diagram

Figure 1.1 shows the block diagram of the SLVS-EC to PCIe Bridge Reference Design mainly consisting of the SLVS-EC IP, ISP, Line Buffer, and a Lattice proprietary design called VDMA which consists of a high-performance video DMA and PCIe hard IP. The pixel data output from SLVS EC IP is sent to the ISP to convert the Bayer image to RGB image. The converted data goes into the Line buffer. The buffered data is fed to VDMA PCIe Subsystem IP.

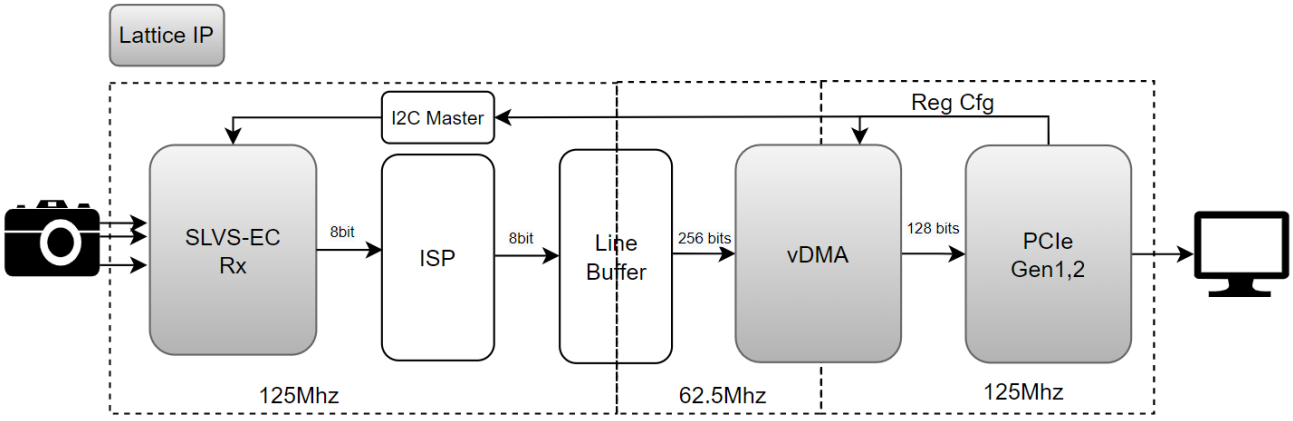

Figure 1.1. SLVS-EC to PCIe Bridge Reference Design Block Diagram

1.3. Functional Description

The SLVS-EC to PCIe Bridge Reference Design converts incoming serial video data from the CMOS image sensor into PCIe output. The SLVS-EC receiver receives serial input from the sensor including the embedded clock. It supports a configurable number of data lanes, and provides pixel data in terms of the signals frame valid, pixel_data_en_o, pixel_valid_o, and parallel pixel_data_o.

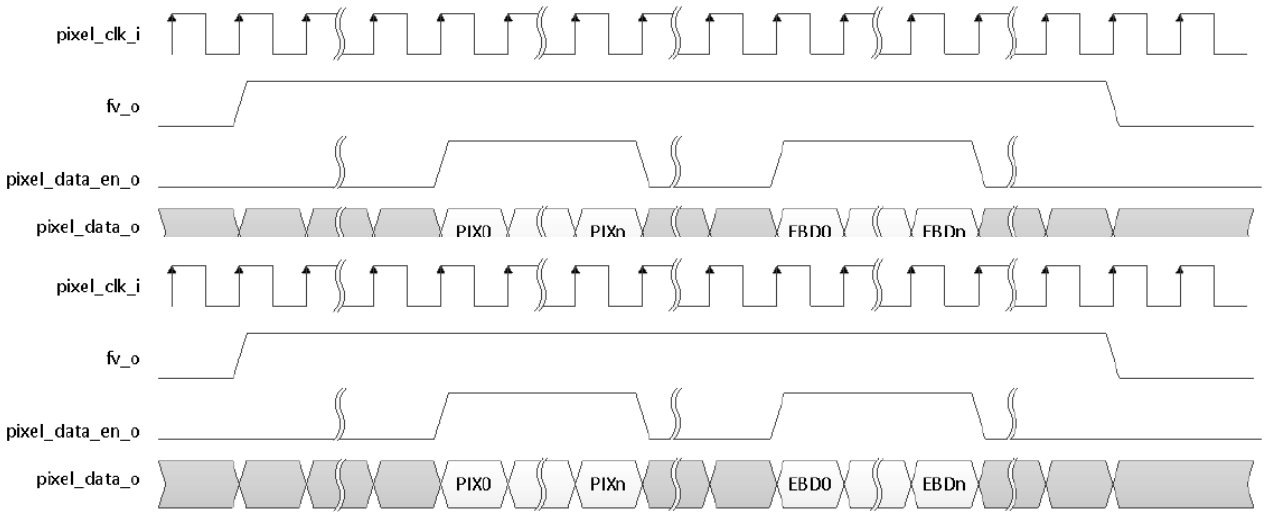

Figure 1.2. SLVS-EC IP Output Waveform

In addition to these signals, it provides other detailed header information as specified in the SLVS-EC Receiver IP Core - Lattice Radiant Software (FPGA-IPUG-02125). The pixel data output is encapsulated into PCIe TLPs (transaction layer packets) and sent through the PCIe interface by the VDMA subsystem.

1.4. Conventions

1.4.1. Nomenclature

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

1.4.2. Data Ordering and Data Types

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

Table 1.1 lists pixel data order from the core module.

Table 1.1. Pixel Data Order

Data Type	Format
RAW	RAW [MSB: 0]

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

2. Parameters and Port List

2.1. Parameters

There is a directive file for this reference design:

- defines.v – used for design synthesis by Lattice Radiant software

These directives can be modified according to the required configuration. The settings in these files must match SLVS-EC IP settings created by Lattice Radiant.

Table 2.1 shows the synthesis directives that affect this reference design. They are used for both synthesis and simulation.

Table 2.1. Synthesis Directives

Category	Directive	Remarks
Number of RX Lanes	NUM_RX_LANE_1	Only one of these directives must be selected as per SLVS-EC lanes configuration
	NUM_RX_LANE_2	
	NUM_RX_LANE_4	
Number of Bits per Pixel lane	BIT_PER_PIXEL {value}	Number of Bits per Pixel lane for SLVS EC, Set value {n} for Raw{n} data type.
Pixel count per line	LINE_LENGTH {value}	The number of pixel counts in SLVS-EC, Set to 1920.
Baud Grade	BAUD_GRADE_1	Only one of these directives must be selected as per SLVS-EC baud grade configuration
	BAUD_GRADE_3	
	BAUD_GRADE_3	
Video Data Type	RAW8	Type of video data to convert from Serial to Pixel for SLVS –EC and from pixel format to byte format for Pixel to Byte converter, always RAW8 used.
Number of Pixels Per Pixel Clock	NUM_PIX_LANE_1	Only one of these four directives must be defined, the number of pixels per pixel clock used for the input to the Pixel to Byte converter
	NUM_PIX_LANE_2	
	NUM_PIX_LANE_4	
	NUM_PIX_LANE_6	
Number of Pixels	NUM_PIXELS {value}	Number of active Pixels per Line, set to 1920

2.2. Top-Level I/O

Table 2.2 shows the top-level I/O of this reference design. This may vary depending upon user's design configurations and parameters.

Table 2.2. SLVS-EC to PCIe Bridge Top Level I/O

Port Name	Direction	Description
PCIe IOs		
perst_n_i	I	PCIe fundamental reset.
sd3_rxdp_i	I	SerDes Rx differential input
sd2_rxdp_i	I	SerDes Rx differential input
sd1_rxdp_i	I	SerDes Rx differential input
sd0_rxdp_i	I	SerDes Rx differential input
sd3_rxdn_i	I	SerDes Rx differential input
sd2_rxdn_i	I	SerDes Rx differential input
sd1_rxdn_i	I	SerDes Rx differential input
sd0_rxdn_i	I	SerDes Rx differential input

Port Name	Direction	Description
sd3_txdp_o	O	SerDes Tx differential output
sd2_txdp_o	O	SerDes Tx differential output
sd1_txdp_o	O	SerDes Tx differential output
sd0_txdp_o	O	SerDes Tx differential output
sd3_txdn_o	O	SerDes Tx differential output
sd2_txdn_o	O	SerDes Tx differential output
sd1_txdn_o	O	SerDes Tx differential output
sd0_txdn_o	O	SerDes Tx differential output
refclkp_i	I	Reference Clock of SerDes PLL in one Quad.
refclkn_i	I	Reference Clock of SerDes PLL in one Quad.
sd3_rext_i	I	External Resistance.
sd2_rext_i	I	External Resistance.
sd1_rext_i	I	External Resistance.
sd0_rext_i	I	External Resistance.
sd3_refret_i	I	Reference return for PMA PLL.
sd2_refret_i	I	Reference return for PMA PLL.
sd1_refret_i	I	Reference return for PMA PLL.
sd0_refret_i	I	Reference return for PMA PLL.
status_led_o	O	Debug LEDs
SLVS-EC IOs		
slvsec_ref_clk_n	I	Differential Reference Clock, CLK-
slvsec_ref_clk_p	I	Differential Reference Clock, CLK+
slvsec_mpcs_clk	I	Single-ended MPCS clock. The clock source for PMA. Use a 125 MHz clock
slvsec_rx_d_n[N]	I	Differential Receive Serial Interface, Rx-, N: number of lanes
slvsec_rx_d_p[N]	I	Differential Receive Serial Interface, Rx+, N: number of lanes
SLVS-EC Sensor IOs		
xclr_o	O	System clear (Normal: High, Clear: Low)
omode_o	O	SLVS mode select (SLVS – EC: High, SLVS: Low)
xce_o	O	Fixed to High
x_master_o	O	Controller / Target select (Controller Mode: High, Target Mode: Low)
slave_mode_o	O	3bit- Slave address select
slvsec_i2cm_scl_io	I/O	I^2C: Serial clock line
slvsec_i2cm_sda_io	I/O	I^2C: Serial data line

3. Design and Module Description

The top-level design (slvsec2pcie_LFCPNX.v) consists of the following modules:

- slvs_ec
- ISP
- line_fwft_fifo
- pcie_vdma_top
- i2c_master_wb_top

3.1. slvs_ec

Create this module to receive serial data from the CMOS image sensor and convert it into parallel pixel data output according to configurations, such as RX lanes, Data Type, Baud grade, and others. Figure 3.1 shows an example of IP GUI interface settings in Lattice Radiant for the RX SLVS-EC IP. Refer to SLVS-EC Receiver IP Core User Guide (FPGA-IPUG-02125) for details.

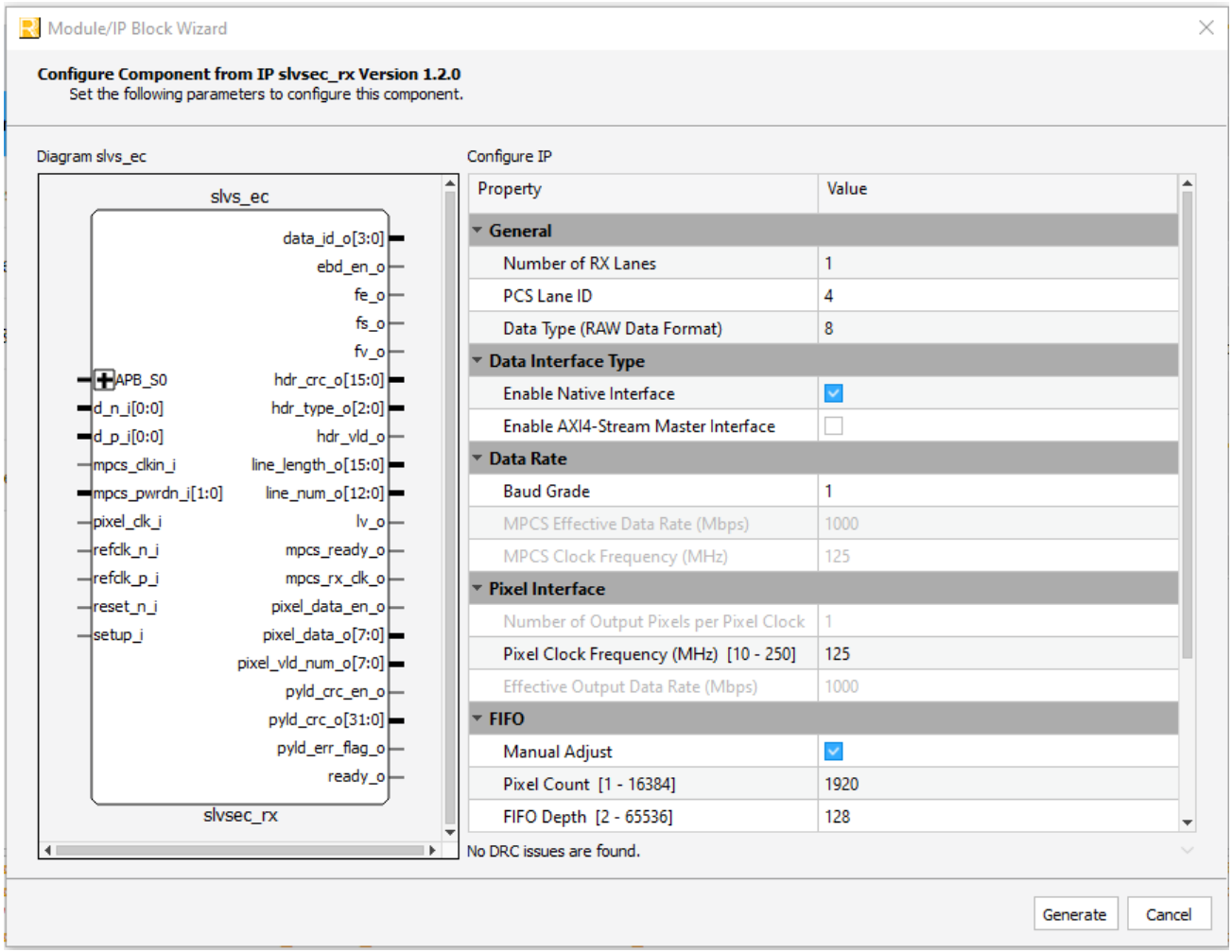

Figure 3.1. slvs_ec IP Creation in Lattice Radiant

The following are the guidelines and the required parameter settings for this reference design:

- Number of RX Lanes – Select 1,2,4,6 or 8 as per configuration
- PCS Lane ID – Select 4

- Data Type – Select as 8. These indicate RAW data types.
- Enable Native Interface – Always enabled (checked)
- Enable AXI4-Stream Master Interface – Always disabled (unchecked)
- Baud grade – Select as 1
- Pixel Clock Frequency – Recommended by IP is always 125 MHz. Value changes as per Baud grade selection.
- Pixel count – Set as 1920
- Enable Header Analysis – Always enabled (checked)
- Enable Miscellaneous signals – Always disabled (unchecked)

The .ipx file is included in the projects (slvs_ec/slvs_ec.ipx) to reconfigure the IP as per the user configuration requirements. Set the design name of this IP to slvs_ec if you need to create it from scratch so that the IP's instance name does not need to be modified.

3.2. line_fwft_fifo

Create this module to convert the data width and change the clock domain to VDMA pixel clock. Figure 3.2 shows an example of IP GUI interface settings in Lattice Radiant for the Line FIFO IP. Refer to FIFO user guide Memory Module User Guide (FPGA-IPUG-02033) for details.

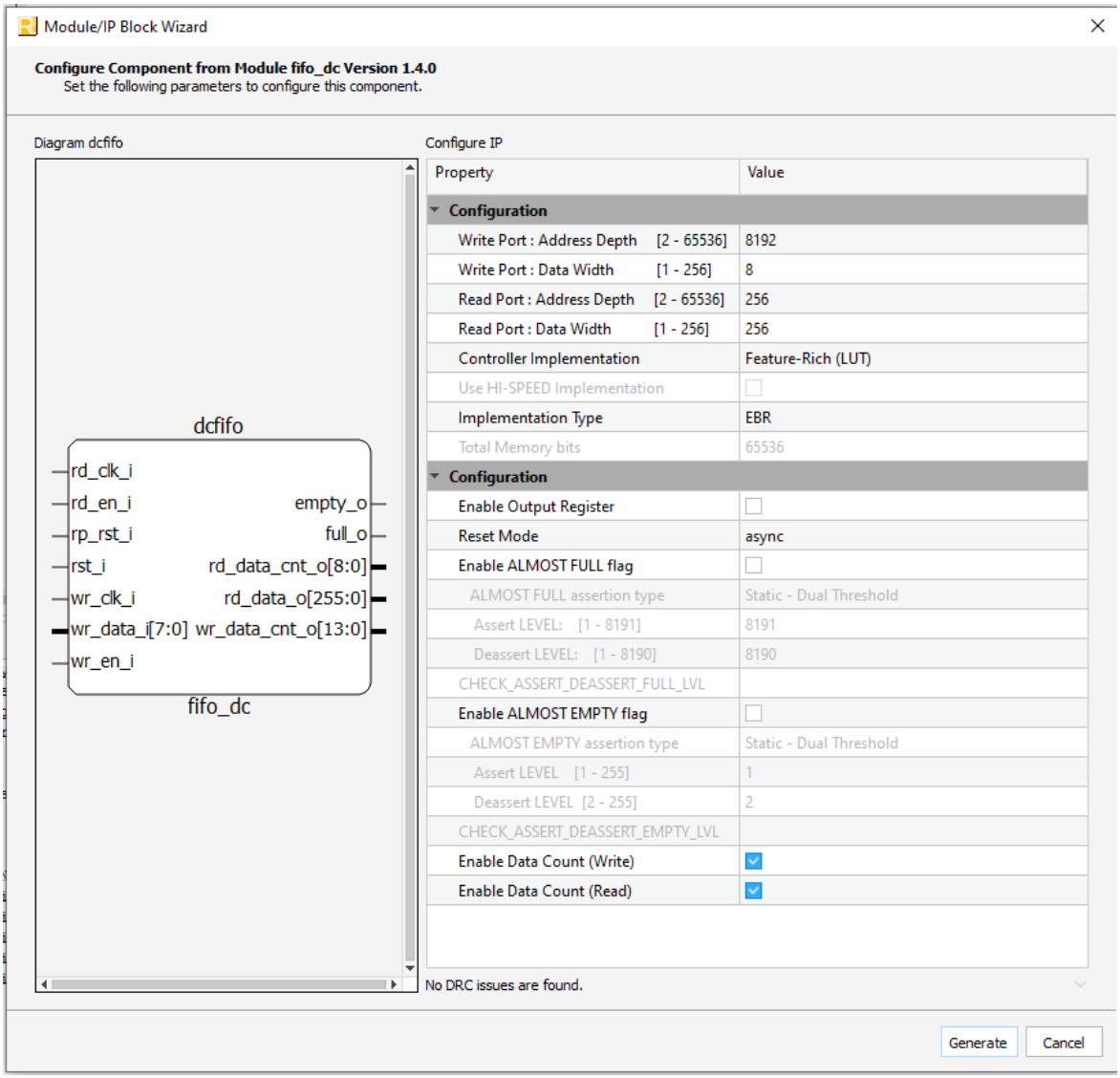

Figure 3.2. Line FIFO IP Creation in Lattice Radiant

The following are the guidelines and the required parameter settings for this module:

- Write Port: Address Depth – FIFO Total size is 8kB, it should be selected depending on Write Data Width. The total size should not exceed than 8kBytes.
- Write Port: Data Width – Select based on the SLVS-EC IP output data width. For this RD, it is 8bit.
- Read Port: Address Depth – FIFO Total size is 8kB, it should be selected depending on Read Data Width. The total size should not exceed than 8kBytes. It should be always 256 deep.
- Read Port: Data Width – 256 bit. It should be always 256bit wide.
- Controller Implementation- Feature-Rich (LUT)
- Implementation Type – EBR
- Enable Output Register – Disable (Uncheck)
- Enable Data Count (Write) – Enable(check)
- Enable Data Count (Read) – Enable(check)
- Enable ALMOST FULL flag – Disable(Uncheck)
- Enable ALMOST EMPTY flag – Disable(Uncheck)

3.3. pcie_vdma_top

The Lattice Video DMA (VDMA) is a PCIe sub-system, designed for video to PCIe or PCIe to video-related applications. VDMA provides high-bandwidth direct memory access between targets CPU memory and video type interfaces. The VDMA pixel interface is 256bit wide. It provides 4 wishbone target interfaces out of which one is used by VDMA internally, one is used by Wishbone to the I^2C bridge. The other 2 are unused.

3.4. i2c_master_wb_top

The I^2C controller core supports the critical features described in the I^2C specification and is suitable for most applications involving I^2C target control. The design responds to the read/write cycles initiated by the microcontroller through the WISHBONE interface. It provides the correct sequences of commands and data to the I^2C target device and then transfers the required data from the I^2C target device through the two open-drain wires. The I^2C controller with WISHBONE interface offloads the microcontroller from needing to administrate many details of the I^2C commands and operation sequences. It is used to configure the SLVS-EC Sensor module. Refer to the IP Core User Guide (RD1046) for details.

4. Design and File Modifications

This reference design uses version 1.2.0 of the SLVS EC IP. Update the parameters file (defines.v) based on SLVS-EC IP configurations.

4.1. Top-level RTL

The current top-level file (slvsec2pcie_LFCPNX.v) integrates all the above-mentioned design IPs required for this reference design. Modifying any IP configuration may require corresponding changes in other modules.

5. Design Package and Project Setup

The SLVS-EC to PCIe Bridge with CertusPro-NX Reference Design is available on www.latticesemi.com. Figure 5.1 shows the directory structure. The targeted design is for LFCPNX-100-7LFG672I. The provided defines.v file configures the design with the following configuration:

- SLVS-EC: 1-lane, RAW8, Baud Grade 1

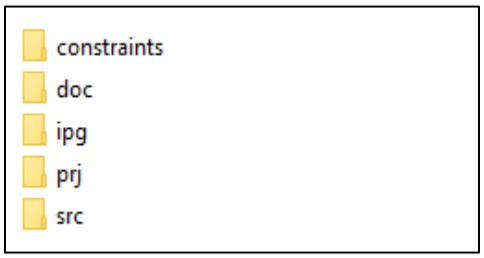

Figure 5.1. Directory Structure

Figure 5.2 shows the design files used in the Lattice Radiant project. Including PLL, Lattice Radiant creates four .ipx files. By specifying slvsec2PCIe_LFCPNX as a top-level design, the tool ignores all unnecessary files. The constraint file (slvsec2pcie_LFCPNX.pdc) is also included in the project for reference.

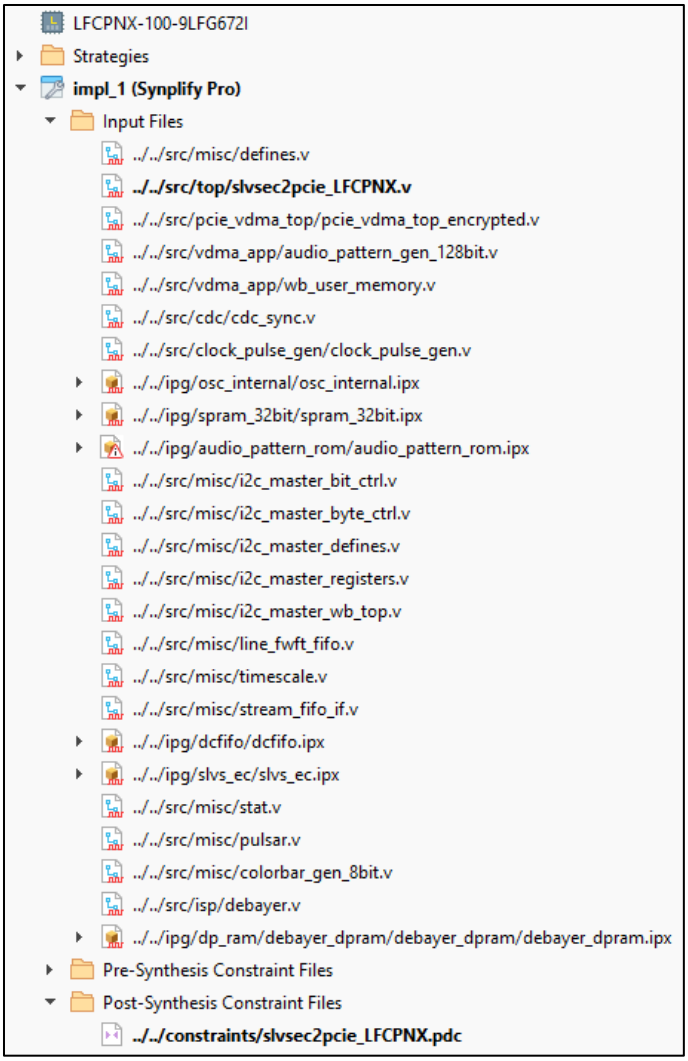

Figure 5.2. Project Files

6. Resource Utilization

Resource utilization depends on the configuration used. Figure 6.1 shows resource utilization examples under certain configurations targeting LFCPNX-100. This is just a reference and actual usage varies.

Device utilization summary:

VHI	1/1	100% used
PCLKDIV	1/2	50% used
DCS	1/2	50% used
EBR	35/208	16% used
MULT9	16/312	5% used
MULT18	8/156	5% used
REG18	16/312	5% used
ACC54	4/78	5% used
PREADD9	16/312	5% used
SEIO18A	10/132	7% used
	10/132	7% bonded
SEIO33	10/299	3% used
	10/167	5% bonded
OSC	1/1	100% used
APIO	30/60	50% used
PCS	2/2	100% used
PCIELL	1/1	100% used
PCSREFMUX8	1/1	100% used
SLICE	8198/39936	20% used
LUT	6166/79872	7% used
REG	6246/79872	7% used

Figure 6.1. Device Resource Utilization

7. Demo

7.1. Introduction

This section describes the design and setup procedure for the Lattice CertusPro-NX Video Bridge Board to demonstrate the SLVS-EC image sensor to PCIe bridging function.

The demonstration hardware consists of three boards:

- IMX535 Sensor Module
- SLVS-EC FMC Card
- CertusPro-NX Video Bridge Board.

Figure 7.1 shows the stacked arrangement of the hardware: IMX535 sensor module with lens is attached to the SLVS-EC FMC card which in turn is attached to the Video bridge board.

Figure 7.1. Stacked Hardware Arrangement

Refer to the following documents for detailed information on related boards:

- CertusPro-NX Video Bridge Board User Guide
- SLVS-EC FMC Card User Guide
- Butterfly1 Series Sensor Module Specifications

7.2. Demo Requirements

The following components are required for the demo:

- Lattice CertusPro-NX Video Bridge Board
  - Mini-USB Cable*
  - 12 V Power Supply*
- IMX535 Sensor module
- Matching lens

- Matching lens holder
- Tripod
- PCIe Extender
- SLVS-EC FMC Card
- Host system with PCIe Gen2 Support, x4 or x16 running Ubuntu operating system
- Display Monitor (supports 1080p or higher resolution)
- Lattice Radiant® Programmer version 3.x or higher*

***Note**: Required only in re-programming.

Figure 7.2. Hardware Test Setup

7.3. IMX535 Sensor Module Board

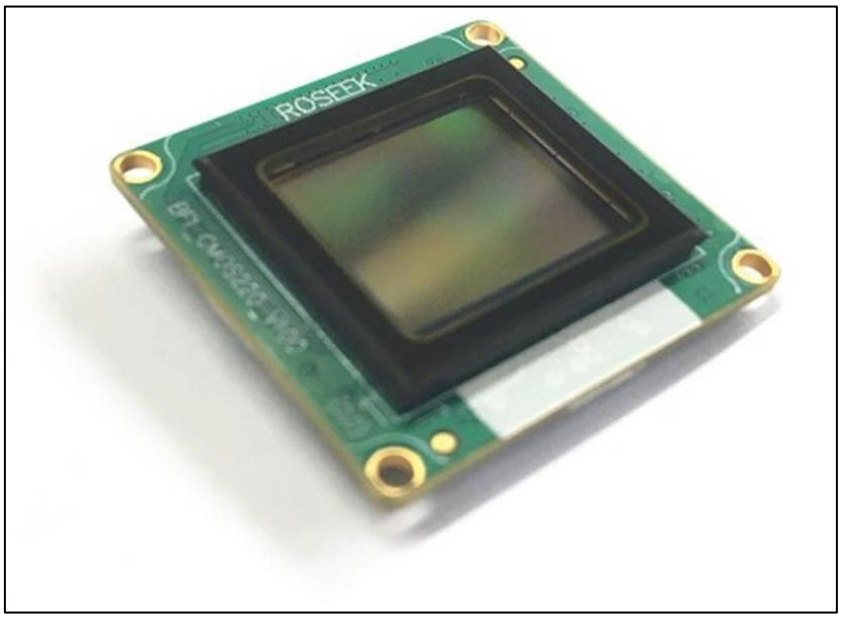

Figure 7.3. Top View of IMX535 Sensor Module Board

7.4. CertusPro-NX Video Bridge Board

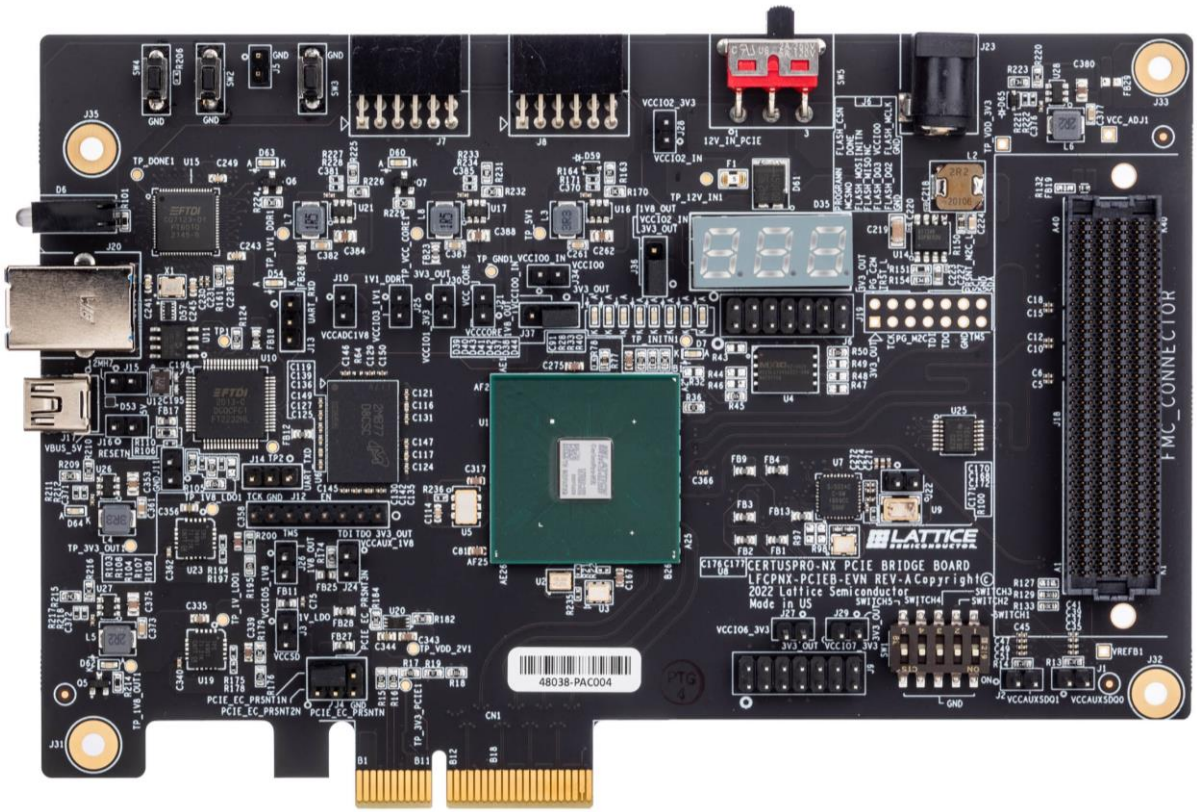

Figure 7.4. Top View of CertusPro-NX Video Bridge Board

7.5. SLVS-EC FMC Card

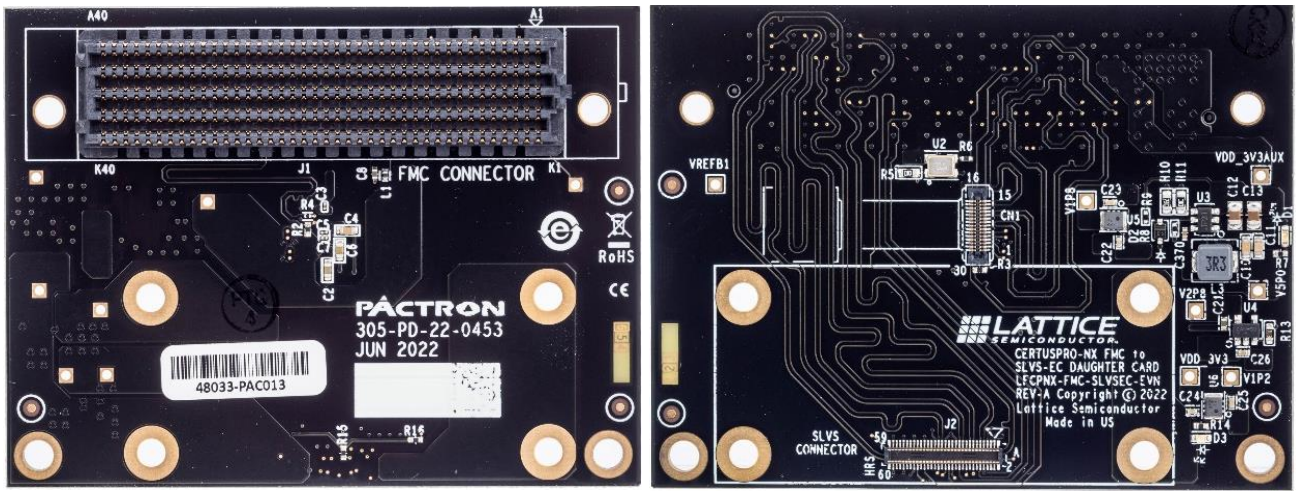

Figure 7.5. Top (Left) and Bottom (Right) View of Video Bridge FMC Card

8. Programming the Demo Design

The CertusPro-NX Video Bridge Board must be configured and programmed if there is no demo bitstream stored in external SPI Flash.

8.1. Programming the CertusPro-NX SPI Flash

8.1.1. Erasing the CertusPro-NX SRAM Prior to Reprogramming

If the CertusPro-NX device is programmed in advance, either programmed directly or loaded from SPI Flash, follow the procedure below to erase the CertusPro-NX SRAM memory before re-programming the CertusPro-NX SPI Flash. Be sure to keep the board powered before re-programming the SPI Flash so that CertusPro-NX FPGA does not reload the legacy SPI Flash pattern when rebooting.

To erase CertusPro-NX SRAM memory:

1. Launch Lattice Radiant Programmer with **Create a new blank project**.
2. Select **LFCPNX** from the **Device Family and LFCPNX-100 from the Device** (Figure 8.1), skip this step if the part is scanned.

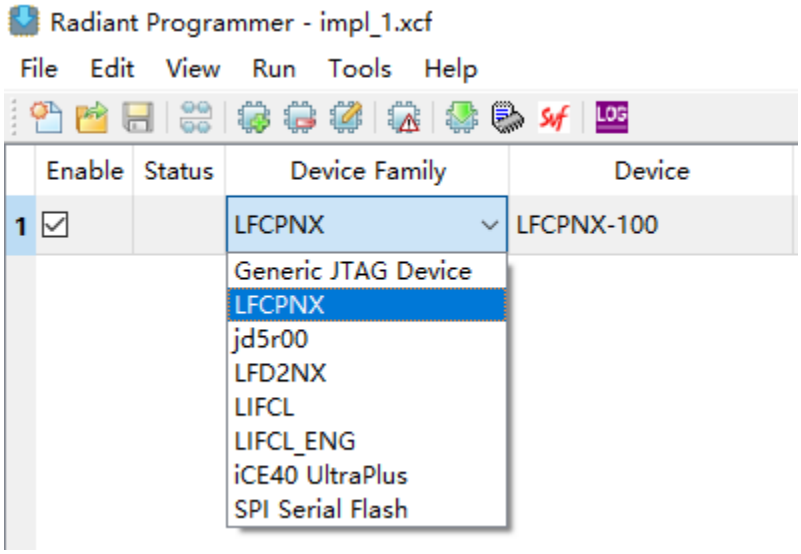

Figure 8.1. Select Device

3. Right-click the desired device family or the device, and select **Device Properties.**
4. In the pop-up Device Properties dialog (Figure 8.2), select **JTAG** for **Port Interface**, **Direct Programming** for **Access Mode,** and **Erase Only** for **Operation**.

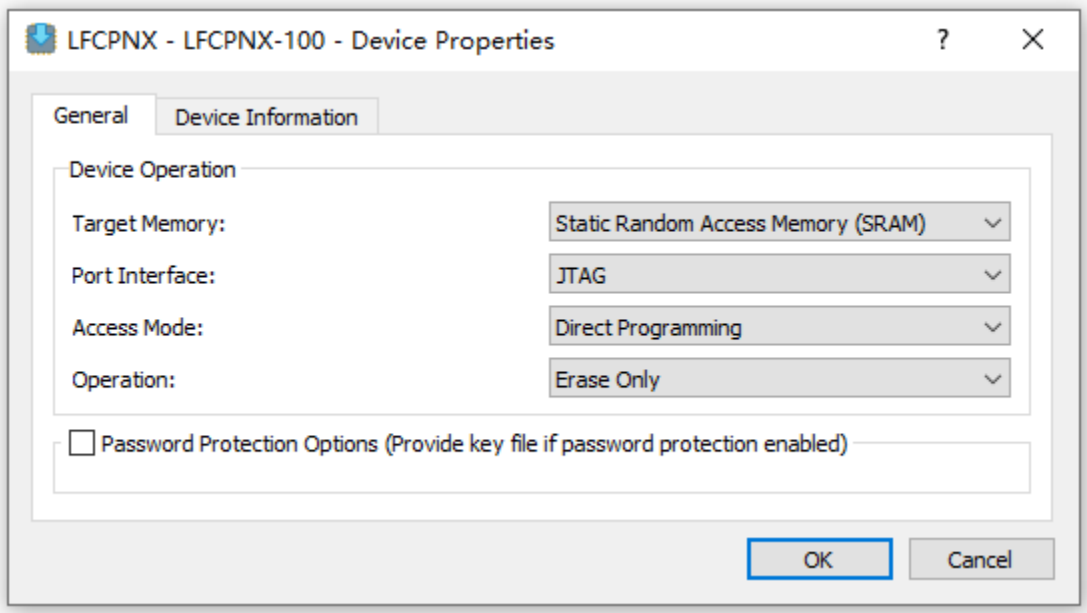

Figure 8.2. Device Operation

5. Click **OK** to close the Device Properties dialog.
6. Click the **Program** icon from the Lattice Radiant Programmer toolbar to start the Erase sequence.

8.1.2. Programming the SPI on the CertusPro-NX Sensor Bridge Board

To program the SPI:

1. Ensure the CertusPro-NX device is erased by performing Steps 1 to 6 in the previous section.
2. Right-click the desired Device Family or the device, and select **Device Properties**.
3. In the pop-up Device Properties dialog (Figure 8.3), select **External SPI Flash Memory (SPI FLASH)** for **Target Memory, JTAG2SPI** for **Port Interface**, and **Direct Programming** for **Access Mode.**
4. For **Programming File**, browse and select the CertusPro-NX bitstream file.
5. For **SPI Flash Options**, select **WinBond W25Q512JV** for **Device**, **16-pin SOIC** for **Package,** and make other selections as shown in Figure 8.3 below.

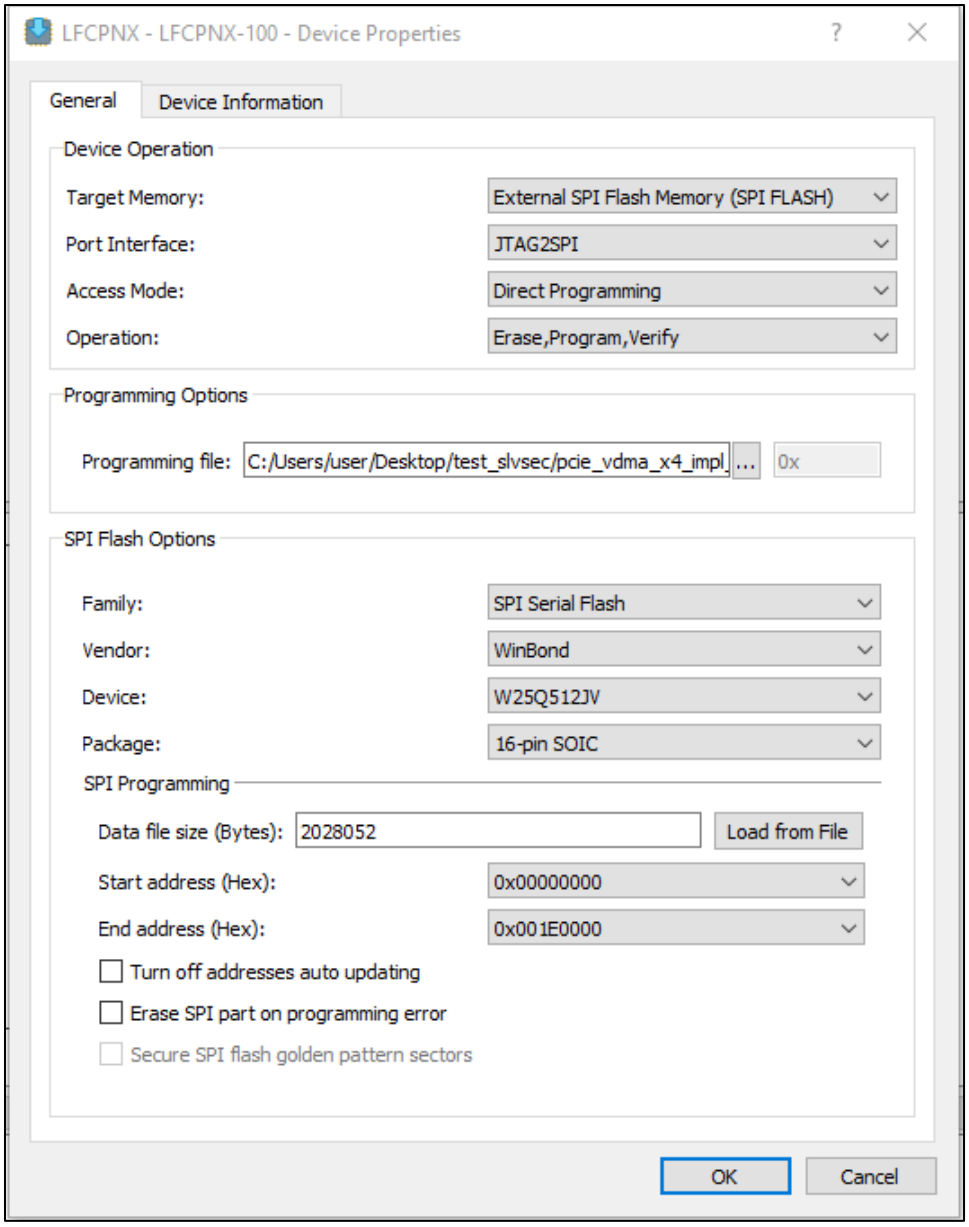

Figure 8.3. Device Properties

6. Click **OK** to close the **Device Properties** window.
7. Click the **Program** icon from the Lattice Radiant Programmer toolbar to start the programming sequence.
8. After successful programming, the **Output** console displays the results as shown in Figure 8.4.

Figure 8.4. Output Console

9. Running the Demo

To set up the demonstration design:

1. Connect IMX535 Image Sensor, SLVS-EC FMC Card, and the CertusPro-NX Video Bridge Board
2. Connect the CertusPro-NX Video Bridge Board to the wall socket using a 12 V DC power adapter.
3. Connect Video Bridge Board PCIe to host system x4 Or x16 PCIe slot using PCIe Extender Cable.
4. Power up the demo kit by turning on the host system.
5. When setting up the system, do the required installations using the following commands:

   $ sudo apt update

   $ sudo apt-get install build-essential

   $ sudo apt-get install pkg-config

   $ sudo apt-get install libglfw3-dev

   $ sudo apt-get install mesa-utils

   $ sudo apt-get install libglew-dev

   $ sudo dpkg --configure -a

   $ sudo apt-get install libasound2-dev

6. When setting up the system, decompress the provided compressed software release file – E.g. slvsy22.tar file, build the driver and application software as illustrated below:

   decompressing the compressed software files,

```
~/slvsrel$ tar xvf ./slvsy22.tar
./slvsy22/
./slvsy22/Makefile
./slvsy22/app/
./slvsy22/app/3rdpart/
./slvsy22/app/3rdpart/imgui/
./slvsy22/app/3rdpart/imgui/imgui.h
./slvsy22/app/3rdpart/imgui/LICENSE.txt
./slvsy22/app/3rdpart/imgui/stb_truetype.h
./slvsy22/app/3rdpart/imgui/imgui_impl_glfw_gl3.h
./slvsy22/app/3rdpart/imgui/imgui_draw.cpp
./slvsy22/app/3rdpart/imgui/imgui_impl_glfw_gl3.cpp
./slvsy22/app/3rdpart/imgui/imconfig.h
./slvsy22/app/3rdpart/imgui/imgui_demo.cpp
./slvsy22/app/3rdpart/imgui/gl3w.h
./slvsy22/app/3rdpart/imgui/stb_rect_pack.h
./slvsy22/app/3rdpart/imgui/imgui.cpp
./slvsy22/app/3rdpart/imgui/glcorearb.h
./slvsy22/app/3rdpart/imgui/imgui_internal.h
./slvsy22/app/3rdpart/imgui/stb_textedit.h
```

Build the driver and application software by executing the build.sh script,

```
~/slvsrel$ ./build.sh
make clean -C lib/
make[1]: Entering directory '/home/bitstar/slvsrel/slvsy22/lib'
rm -f liblscvdma.so *.o
make[1]: Leaving directory '/home/bitstar/slvsrel/slvsy22/lib'
make clean -C drvr/
make[1]: Entering directory '/home/bitstar/slvsrel/slvsy22/drvr'
make -C /lib/modules/5.15.0-46-generic/build M=/home/bitstar/slvsrel/slvsy22/drvr clean
make[2]: Entering directory '/usr/src/linux-headers-5.15.0-46-generic'
make[2]: Leaving directory '/usr/src/linux-headers-5.15.0-46-generic'
rm -rf *.o *.ko *.mod.c
make[1]: Leaving directory '/home/bitstar/slvsrel/slvsy22/drvr'
make clean -C app/GUI/AVCapture/
make[1]: Entering directory '/home/bitstar/slvsrel/slvsy22/app/GUI/AVCapture'
rm -rf  AVCapture ./AVCapture.o ./Console.o ./MiscUtils.o ./OpenGLRender/OpenGLRender.o ./AlsaPlayer/Al
saPlayerDevice.o ./Sensor/Sensor.o ../../3rdpart/imgui/imgui.o ../../3rdpart/imgui/imgui_impl_glfw_gl3.
o ../../3rdpart/imgui/imgui_draw.o
make[1]: Leaving directory '/home/bitstar/slvsrel/slvsy22/app/GUI/AVCapture'
make -C lib/
make[1]: Entering directory '/home/bitstar/slvsrel/slvsy22/lib'
g++ -std=c++11 -fPIC -Wall -Wextra -o2 -g -Wno-write-strings -shared -o liblscvdma.so LSCVDMA_IF.cpp
LSCVDMA_IF.cpp: In member function ‘int LSCVDMA_IF::OpenDriver(HANDLE&, const char*, const char*, uint3
2_t, const char*, const char*)’:
LSCVDMA_IF.cpp:611:16: warning: unused parameter ‘pFriendlyName’ [-Wunused-parameter]
```

7. Whether the card is properly mounted and detected by the system may be checked by executing det.sh script,

- If no card is detected, there is no output.
- If the card is detected a line similar to the one below is displayed,
  01:00.0 Signal processing controller: Lattice Semiconductor Corporation Device 0001 (rev 01)

```
~/slvsrel$ ls
build.sh  det.sh  slvsy22  slvsy22.tar  ydsp.sh
~/slvsrel$ ./det.sh
01:00.0 Signal processing controller: Lattice Semiconductor Corporation Device 0001 (rev 01)
~/slvsrel$
```

8. Run the demo by executing the ydsp.sh script,

```
~/slvsrel$ ./ydsp.sh
[sudo] password for :
Installing driver module lscvdma into kernel.
Done.
BoardID: LSC DemoID: VDMA devNum: 1
Opening /dev/LSC_VDMA_1
LSCVDMA_IF created.
hasInterrupt: 1  Vect: 255  Num BARs: 1
BAR0  Addr: 8a400000  Size: 16384  Mapped: 1

Expected Frame Rate: 22

Max Lines in Frame: 1188
Done.

About VDMA info:
PCI Driver Version: Lattice PCIe VDMA Device Driver lscvdma v1.02.0.0001

PCI IP Version: Lattice PCIe VDMA IP v1.02.0.0001

Video resolution: 1920 x 1072
Support max buffer number: 16
```

When the demo is running, the screen is populated with the video from the Sensor.The software prints any information related to the running conditions to the console on the terminal that started the demo. Users may toggle between the Video screen and the terminal by using [Alt]+[tab] key presses.

Appendix A. Interoperability Testing

Table A. 1. Interoperability Testing Results

CPU	Discrete Graphics	Motherboard	Host Memory	Operating System	PCIe Mode	Results
Intel Core i7-3770	NVIDIA GT210	hp Phoenix h9-1186in	8GB (4GBx2) DDR3	Ubuntu 20.04.5 LTS	Gen3	Pass
Intel Core i5-11400	Mesa Intel Graphics (RKL GT1)	Acer Veritron M200-H510	8GB (8GBx1) DDR4	Ubuntu 20.04.4 LTS	Gen3	Pass

References

- SLVS-EC Receiver IP Core User Guide (FPGA-IPUG-02125)
- PLL Module User Guide (FPGA-IPUG-02063)

For more information on the CertusPro-NX FPGA device, visit
https://www.latticesemi.com/Products/FPGAandCPLD/CertusPro-NX

For complete information on Lattice Radiant Project-Based Environment, Design Flow, Implementation Flow, Tasks, and Simulation Flow, see the Lattice Radiant Software User Guide.

Revision History

Revision 1.0, September 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com