Antmicro

SDI to MIPI CSI-2 Video Converter

2023-10-13

CONTENTS

CHAPTER

ONE

INTRODUCTION

This document describes Antmicro's SDI to MIPI CSI-2 Video Converter, which uses a Lattice Crosslink-NX FPGA and an SDI deserializer to perform the conversion. SDI is a popular standard used in video broadcasting, while MIPI CSI-2 is a mobile/embedded camera standard that is directly supported by a wide range of SoCs. The SDI to MIPI CSI-2 Video Converter also provides on-board DDR3 memory for processing the captured stream.

1.1 Documentation structure

This documentation provides implementation details and usage instructions. It is divided into the following chapters:

- *Hardware setup* - description of the hardware, physical connections and configurations,
- *Building the FPGA design* - how to generate FPGA bitstreams for all available SDI signal configurations,
- *Preparing the software* - how to build and use a Linux BSP that supports the MIPI SDI Video Converter for Jetson Xavier NX,
- *CMOS to D-PHY converter* - detailed description of the CMOS to D-PHY IP Core.

CHAPTER

TWO

HARDWARE SETUP

A complete setup using the SDI-MIPI Video Converter would typically consist of at least 3 devices:

- *SDI Video Converter* - Antmicro's SDI to MIPI CSI-2 Video Converter, responsible for deserializing SDI data and packeting it with the MIPI CSI-2 protocol,
- *Transmitter* - a device transmitting SDI video data, connected to the SDI input port of the bridge,
- *Receiver* - a device receiving MIPI CSI-2 video data, connected to the MIPI CSI-2 output port of the bridge.

2.1 SDI Video Converter

The core part of this setup is the SDI to MIPI CSI-2 Video Converter.

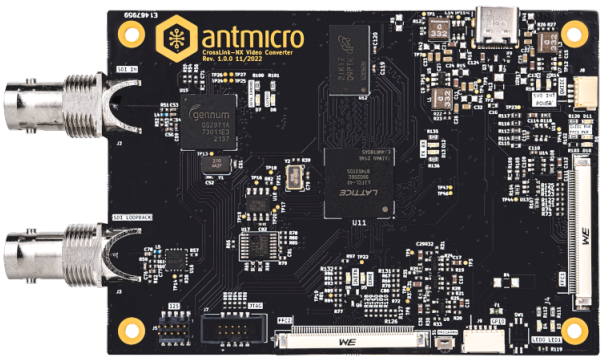

Figure 2.1: SDI to MIPI CSI-2 Video Converter board, rev. 1.0.0

2.1.1 Features

- Integrated SDI adaptive cable equalizer and output loopback connector
- Three 4-lane MIPI D-PHY transceivers at 6 Gbps per PHY exposed on two 50-pin FFC connectors
- Ability to power devices from the 5V and 3.3V rails of the side FFC connector
- Ability to power the board from either USB or Antmicro's 50-pin FFC connector
- Lattice Crosslink-NX FPGA for processing SDI and CSI streams
- 2Gbit (128Mbx16) DDR3L DRAM memory for the FPGA to support stream processing
- Ability to interface and program the FPGA via the side FFC I^2C bus

2.1.2 Architecture

The video converter board consists of the SDI deserializer, Lattice Crosslink-NX FPGA, and on-board DDR3L memory. The Figure 2.2 visualizes an architecture and data exchange between the board components.

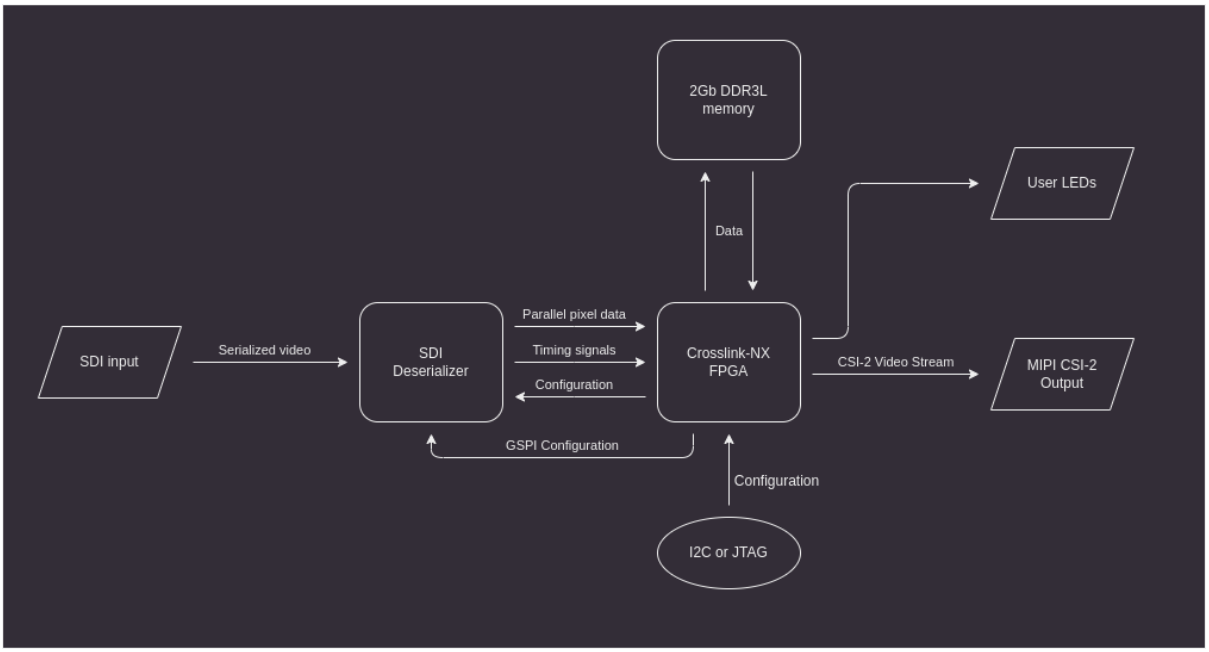

Figure 2.2: SDI to MIPI CSI-2 Video Converter block diagram

An I^2C or JTAG commands must come from outside of the board in order to program the FPGA chip.

2.2 Transmitter

The transmitter is a device (typically a camera) outputting data via the SDI interface. It must be compatible with the 1080p30Hz YUV422 video format.

Several input devices have been tested with SDI to MIPI CSI-2 Video Converter, e.g:

- Atomos Shogun Flame,
- Blackmagic HDMI to SDI Converter,
- Decimator HDMI/SDI 4K Cross Converter.

2.3 Receiver

The receiver is a device that can receive data through the MIPI CSI-2 interface. The *Software section* of this documentation covers support for the Jetson Xavier NX board.

Antmicro provides sources for a Linux distribution configured for SDI Video Converter support for Jetson Xavier NX.

CHAPTER

THREE

BUILDING THE FPGA DESIGN

The CrossLink-NX FPGA is responsible for converting data deserialized from SDI by the Semtech chip to MIPI CSI-2. The core module of the FPGA design is a CMOS to D-PHY converter which converts standard parallel video data into CSI-2 byte packets.

3.1 CMOS to D-PHY Converter

The CMOS to D-PHY module provides a bridging solution for converting parallelized pixel data from the deserializer into a MIPI CSI-2 video stream. The module supports all of the following formats:

- 720p25Hz YUV422,
- 720p30Hz YUV422,
- 720p50Hz YUV422,
- 720p60Hz YUV422,
- 1080p25Hz YUV422,
- 1080p30Hz YUV422,
- 1080p50Hz YUV422,
- 1080p60Hz YUV422.

For more details, refer to *CMOS to D-PHY converter* chapter.

3.2 Setting up the environment

The dependencies required to build the FPGA design can be installed by running:

```
pip3 install -r requirements.txt
```

In order to generate a bitstream from the generated sources, the nextpnr toolchain is required.

3.3 Building the bitstream

Once required tools are installed, project can be built by invoking:

```
make all
```

The output files are created in the `build/<video-format>-<lanes>` directory.

The Makefile provides configuration flags that can be set as presented below:

```
FLAG=flag_value make all
```

The list of available flags can be accessed with `make help` and is described below:

Table 3.1: Available `make` configuration flags.

Flag	Flag description	Default value
YOSYS	Path to yosys.	`yosys`
NEXTPNR	Path to nextpnr.	`nextpnr-nexus`
PRJOXIDE	Path to prjoxide.	`prjoxide`
ECPPROG	Path to ecpprog.	`ecpprog`
YOSYS_ARG	Additional arguments for yosys.	`-o 1080p30-2lanes_syn.v`
SYNTH_ARG	Additional arguments for Yosys the `synth_nexus` command.	`-flatten`
NEXTPNR_A	Additional arguments for NEXTPNR.	`--placer-heap-timingweight 35`
PATTERN_GEN	Set to `1` if you want to generate design with embedded pattern generator.	None
SIM	Set to `1` if you want to generate Verilog sources ready for simulation with Modelsim Lattice FPGA Edition.	None
VIDEO_FORM	Video format, one of `720p60`, `1080p30` or `1080p60`.	`1080p30`
LANES	D-PHY lanes, must be either `2` or `4`.	`2`
TRACE	Set to `1` if you want to generate simulation waveforms.	None

3.4 Testing

If you want to test the CMOS to D-PHY Core in simulation, you can run the cocotb tests:

```
make tests
```

To run a single test:

```
cd tests
TOP=<module_name> make test
```

You can choose module names from `crc16`, `packet_formatter`, `mipi_dphy` and `cmos2dphy`.

Note: Verilator tests do not cover the D-PHY module since there is no open source simulation model available.

CHAPTER

FOUR

PREPARING THE SOFTWARE

To receive video from a SDI-MIPI Video Converter, the receiver must include drivers for the Semtech GS2971A deserializer.

4.1 Building the BSP

To enable the SDI-MIPI Video Converter support, you need to build the kernel with the required drivers and the device tree for the board. Antmicro provides a preconfigured Linux kernel for the Jetson Xavier NX.

To fetch, build, and add the modified kernel to a stock L4T 32.4.4 package, you must do the following:

1. Get and extract the cross-compilation toolchain:

```
wget http://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-
↪linux-gnu/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
tar xf gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
export PATH=$(pwd)/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin:$PATH
```

2. Get and set up the L4T-based host software:

```
wget https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-
↪GMC3/T186/Tegra186_Linux_R32.4.4_aarch64.tbz2
tar xf Tegra186_Linux_R32.4.4_aarch64.tbz2
wget https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-
↪GMC3/T186/Tegra_Linux_Sample-Root-Filesystem_R32.4.4_aarch64.tbz2
sudo tar xf Tegra_Linux_Sample-Root-Filesystem_R32.4.4_aarch64.tbz2 -C Linux_for_
↪Tegra/rootfs/
pushd Linux_for_Tegra
sudo ./apply_binaries.sh
sudo chown -R $USER rootfs/lib/modules
sudo chown -R $USER rootfs/lib/firmware
popd
```

3. Clone the kernel sources and checkout the branch that matches your hardware:

```
git clone https://github.com/antmicro/sdi-mipi-bridge-linux
pushd sdi-mipi-bridge-linux
git checkout video-converter
```

4. Build the kernel using `make`:

```
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
make tegra_defconfig
make -j$(nproc)
```

5. Install the kernel image, modules and device tree blob:

```
cp ./arch/arm64/boot/Image ../Linux_for_Tegra/kernel/
cp ./arch/arm64/boot/dts/tegra194-p3668-all-p3509-0000.dtb ../Linux_for_Tegra/
↪kernel/dtb/
INSTALL_MOD_PATH=../Linux_for_Tegra/rootfs/ make modules_install
sudo chown -R root ../Linux_for_Tegra/rootfs/lib/modules
sudo chown -R root ../Linux_for_Tegra/rootfs/lib/firmware
popd
```

4.2 Uploading the BSP

To flash the software on the device, put the device in recovery mode, connect it to the host PC with a USB cable, and use the following command:

```
pushd Linux_for_Tegra
sudo ./flash.sh jetson-xavier-nx-devkit-emmc mmcblk0p1
popd
```

CHAPTER

FIVE

CMOS TO D-PHY CONVERTER

The SDI to MIPI CSI-2 Video Converter uses the SDI deserializer chip to decode the SDI serial stream into a 20-bit parallel stream with additional vertical and horizontal synchronization signals. The 20-bit parallel data consists of two 10-bit words defining the luminance and chrominance components of the YCbCr color space. Each of these components is directly connected to the FPGA input pins. The CMOS to D-PHY converter module synchronizes to frame and line valid signals, then passes the data from the deserializer to the MIPI D-PHY, following the MIPI CSI-2 protocol.

5.1 Architecture

The CMOS to D-PHY converter is capable of generating MIPI CSI-2 packets and switching MIPI D-PHY link modes. It consists of the following modules:

1. *CSI-2 Finite State Machine* - FSM that controls MIPI CSI-2 protocol flow, assuming input signals `FVAL/LVAL` it transmits frames one by one. Underlying modules are synchronized between each other in each FSM state to strictly follow MIPI CSI-2 protocol.
2. *Packet Formatter* (Low Level Protocol) - MIPI CSI-2 packet generator, it generates SoT (Start of Transmission), EoT (End of Transmission), header and footer for each packet.
3. *Checksum generator* - combinatorial CRC16 checksum generator.
4. *TX Global Operations* - controls D-PHY interface lanes switching between Low Power (LP) and High Speed (HS) modes.
5. *Hardened TX D-PHY* - the MIPI D-PHY interface provided by the FPGA fabric, configured to operate as a transmitter, controlled by TX Global Operations.

5.2 MIPI CSI-2 Finite State Machine

The MIPI CSI-2 Finite State Machine (FSM) is designed to synchronize other modules with each other. That means, this module is not strictly a part of the MIPI CSI-2 protocol. The FSM synchronizes to the beginning of the frame and then initiates D-PHY packet transfers while controlling the Packet Formatter, the Checksum Generator and the TX Global Operations modules. The diagram below illustrates the FSM flow during device operation.

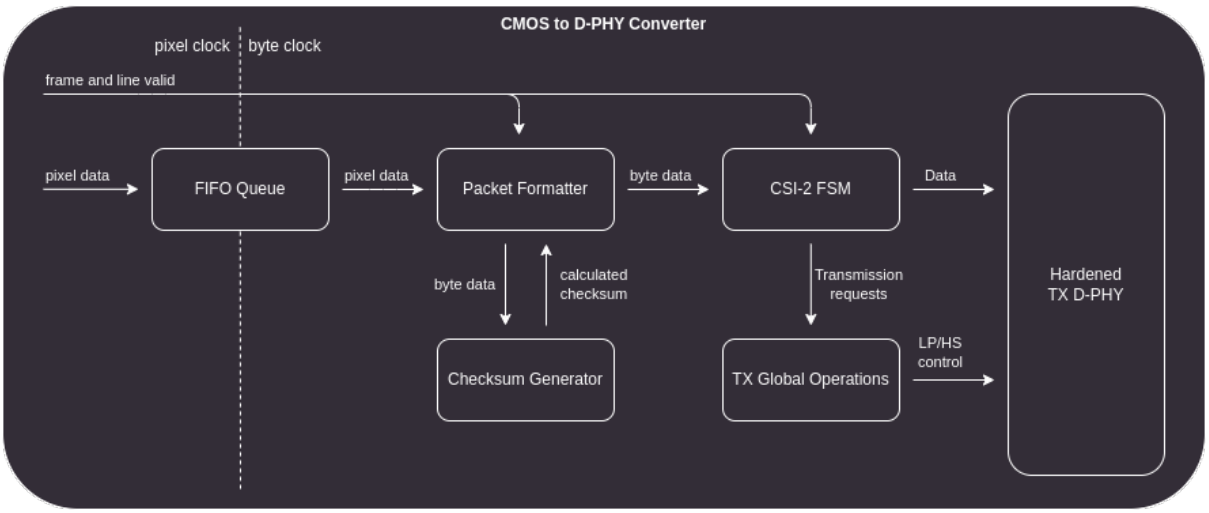

Figure 5.1: CMOS to D-PHY IP Core design

5.3 Packet Formatter

The MIPI CSI-2 protocol transmits input data in a protocol specific packets. There are two main types of packets:

1. Short packets - used for line and frame synchronization. While line start and end packets are optional, frame synchronization is mandatory. A short packet is a 32-bit word consisting of Data ID, Word Count and computed Error Correction Code (ECC).
2. Long packets - consist of packet header, payload and packet footer. The packet header is generated using exactly the same rules as the short packet. The payload is generated from pixels packed into single bytes and sent one by one between the header and the footer. In the case of the YUV422 8-bit data type, no additional pixel-to-byte conversion is required and the pixels are passed directly to the payload field. Each byte is simultaneously passed to the payload and to the CRC checksum generator. The calculated checksum is transmitted as the packet footer at the end of each long packet transmission.

Each packet starts with an initialization sequence (single byte `0xb8` on each line) and ends with the HS-Trail state (inverted MSB of the last transferred byte). The packet formatter module itself does not control the data input to the D-PHY interface. It generates the initialization sequence, header, footer, and HS-Trail state, but it is the task of a supervisor to forward these generated values. The diagram below visualizes the internal flow of the packet formatter for each requested packet.

5.3.1 Data ID

The Data Identifier is a single byte consisting of a 2-bit virtual channel identifier and a 6-bit data type. The data type is equal to `0x1e` for YUV422 8-bit format. In case of frame synchronization packets it must be set to `0x0` or `0x1` for frame start and frame end packets respectively.

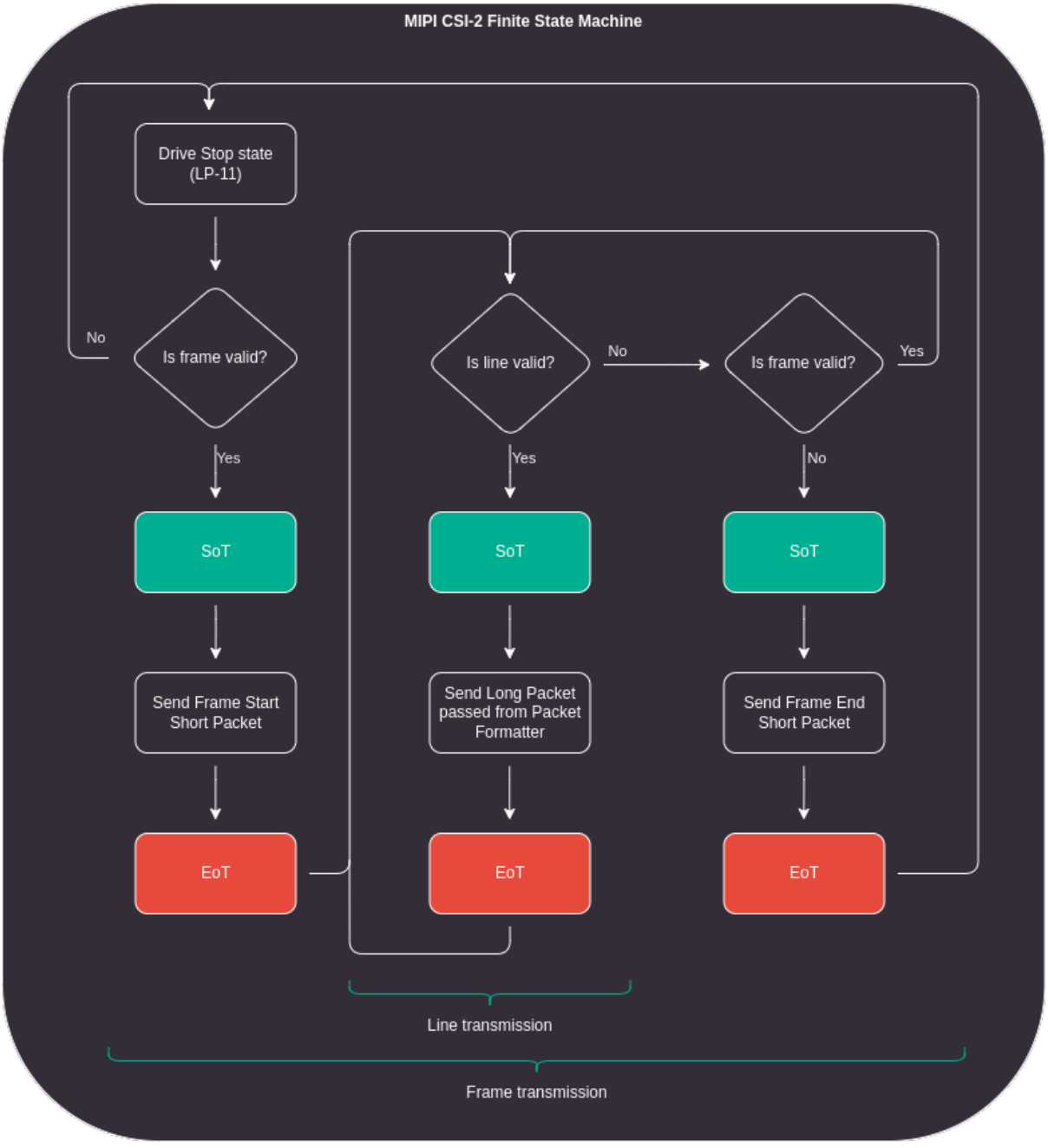

Figure 5.2: MIPI CSI-2 Finite State Machine flow diagram

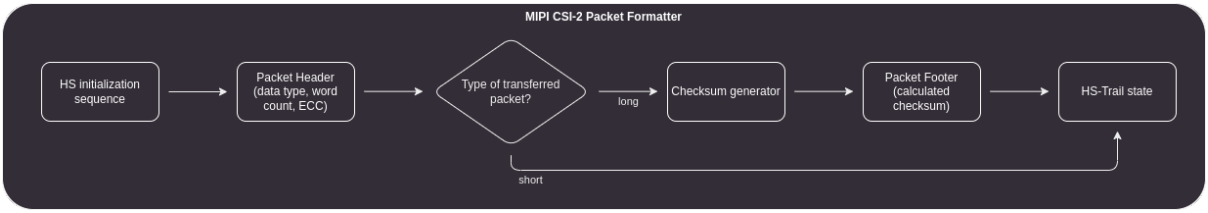

Figure 5.3: Packet Formatter IP Core flow diagram

5.3.2 Word Count

The Word Count value should be zero when transmitting short packets, unless there is a need to attach any embedded data. For long packets, this value must indicate the number of individual bytes to be transmitted.

5.3.3 ECC Generator

The Error Correction Code (ECC) is a Hamming code calculated from the Data ID and the Word Count (WC). The Hamming code is generated for 6 bits, with the two most significant bits set to 0.

5.4 Checksum Generator

In order to detect any errors during the payload transmission, a checksum is calculated over each data packet. The checksum is a 16-bit CRC defined by a polynomial $x^{16}+x^{12}+x^5+x^0$ with an initial state set to `0xffff`. The generator shift direction is assumed to be little endian.

5.5 TX Global Operations

The MIPI D-PHY interface is capable of switching between HS and LP modes. Each switch between these modes must follow a precise sequence of either Start of Transmission (SoT) to enter HS mode or End of Transmission (EoT) to enter LP mode. The High Speed mode is referred to as the transmission mode because it is used to transmit or receive either large chunks of data or synchronization packets.

The TX Global Operations can be controlled by sending transfer requests and validating byte data. It generates signals that should be connected directly to the hardened D-PHY module. The figure below visualizes an example of switching from LP to HS and back from HS to LP on the D-PHY interface.

5.6 Hardened TX D-PHY

The Lattice Crosslink-NX FPGA devices have hardened MIPI D-PHY modules that can be instantiated by the user. These interfaces support the MIPI CSI-2 and MIPI DSI protocols for Tx and Rx devices with unidirectional HS and bidirectional LP modes. The hardened D-PHY blocks deserialize and serialize data into byte data packets and distribute them between 1, 2, 3 or 4 supported lanes. The MIPI D-PHY modules available on Crosslink-NX include an internal PLL that generates the clock for both byte and D-PHY HS domains.

For CMOS to D-PHY converting purposes, hardened D-PHY is configured to work as a transmitter in MIPI CSI-2 protocol.

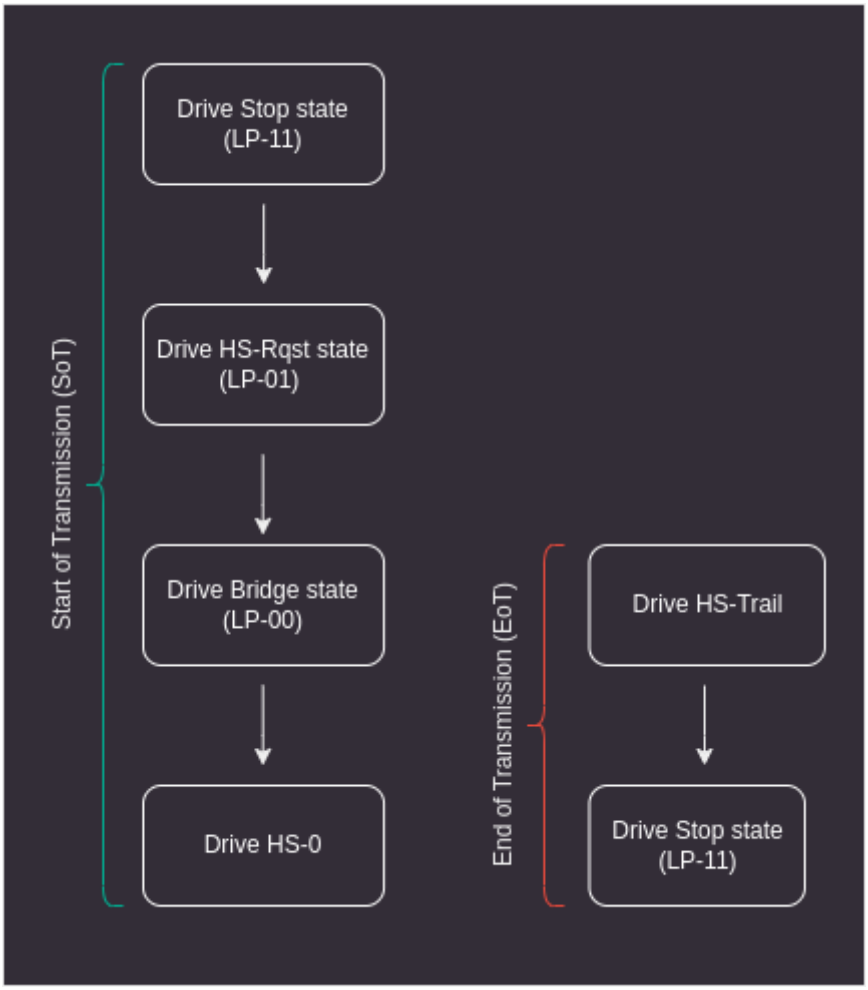

Figure 5.4: Start and End of CSI-2 transmission

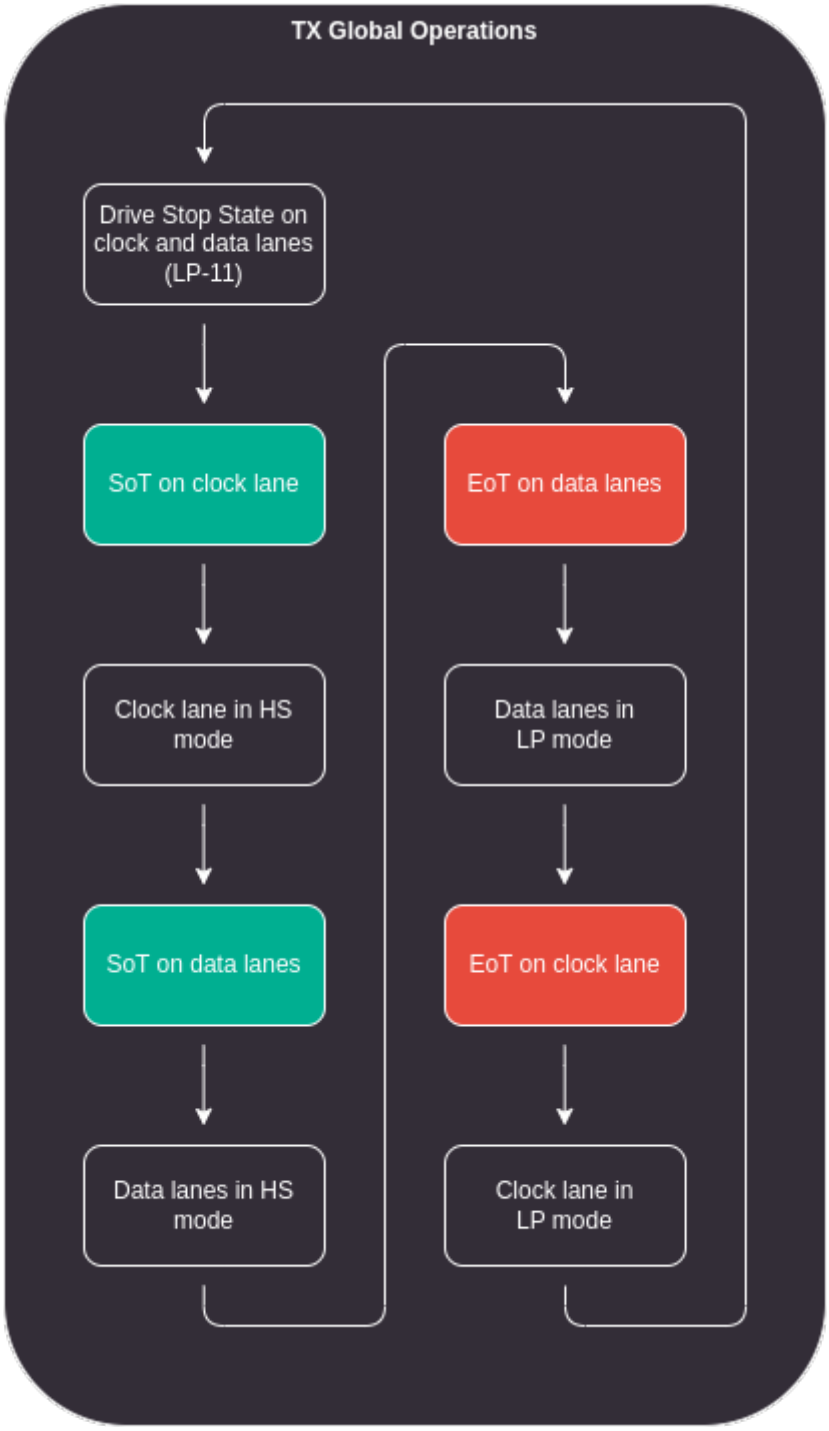

Figure 5.5: TX Global Operations module flow