Designing LVDS transmitter in FPGA to interface with EZ-USB™ FX20/FX10/FX5/FX5N controller

About this document

Scope and purpose

This application note explains the LVDS interface of EZ-USB™ FX20/FX10/FX5/FX5N (USB 3.2 peripheral controller family) controller and how the LVDS transmitter (LVDS TX) can be implemented in FPGA to transmit the data to EZ-USB™ FX20/FX10/FX5/FX5N controller over LVDS interface up to 20 Gbps. The reference FPGA project provided along with this application note is based on the Efinix Ti180 FPGA.

Intended audience

This document is primarily intended for customers who wish to design and implement an LVDS interface on an FPGA to transmit data to the EZ-USB™ FX20/FX10/FX5/FX5N controller.

Table of contents

Table of contents

About this document **1**
Table of contents **2**
1 Introduction **4**
2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller **5**
2.1 Key features of LVDS RX interface 5
2.2 Link initialization and training 6
2.2.1 PHY training 6
2.2.2 LINK training 6
2.3 Flow control 6
2.4 LVDS RX power modes 7
2.4.1 Active mode – L0 7
2.4.2 Low-power mode – L1 7
2.4.2.1 Entry to L1 mode 7
2.4.2.2 Exit from L1 mode 8
2.4.3 Low-power mode – L3 8
2.4.3.1 Entry to L3 mode 8
2.4.3.2 Exit from L3 mode 8
2.5 Thread and socket configuration 9
2.6 Data transmission flow 10
2.6.1 Basic LVDS transaction flow 10
2.6.2 Detailed LVDS transaction flow 11
2.7 Supported gear ratios 13
2.8 Data integrity 17
2.8.1 Data CRC 17
2.8.2 Metadata (header) CRC 17
3 Implementation of LVDS transmitter in FPGA **18**
3.1 System level block diagram 18
3.2 I2C Slave Register interface 19
3.2.1 FPGA's I2C Registers 20
3.2.1.1 Base offset for device-specific registers 20
3.2.1.2 Common control and status registers 20
3.2.1.3 Device-specific registers 23
3.3 Functional description of LVDS transmitter (LVDS TX) 27
3.3.1 LVDS transmitter block diagram and features 27
3.3.1.1 Features of LVDS transmitter 28
3.3.2 Device selection FSM 28
3.3.3 PHY and LINK training FSM 30
3.3.4 Frame data transmitter FSM 31
3.3.4.1 Initial socket sequence FSM 32
3.3.4.2 Frame header information transmitter FSM 33
3.3.4.3 LVDS data packet transmitter FSM 34
3.3.4.4 Frame end transmitter FSM 35
4 Reference FPGA project **36**
5 Troubleshooting LVDS interface **37**
5.1 PHY training failure 37
5.2 LINK training failure 38

Table of contents

5.3 Incorrect sampling of buffer ready (BUFFER_RDY) 40
5.4 Slow or sluggish video data streaming on USB Host 40
5.5 Choppy video data streaming on USB Host 41
References 42
Revision history 43
Disclaimer 44

1 Introduction

EZ-USB™ FX20/FX10/FX5N/FX5 is a family of USB peripheral controllers compliant with USB 3.2 Gen 2x2 specification and backward compatible with the USB 3.1 and USB 2.0 specifications. EZ-USB™ FX20[1] consists of dual Arm® Cortex®-M4 and M0+ core CPUs, a 512 KB flash, 128 KB SRAM, 128 KB ROM, seven serial communication blocks (SCBs), a cryptography accelerator, and a high-bandwidth data subsystem providing DMA data transfers between LVDS/LVCMOS and USB ports at speeds up to 20 Gbps. An additional 1 MB SRAM is included in the high-bandwidth data subsystem to provide buffering for USB data. EZ-USB™ FX20 also supports USB Type-C plug orientation detection and flip-mux function without the need for external logic.

The EZ-USB™ FX20 controller is typically used in cameras, video, imaging, and data acquisition type of applications. The EZ-USB™ FX20 controller supports LVDS receiver (LVDS RX) and LVCMOS transmitter and receiver (TX/RX) interface to connect with an external data source such as FPGA.

This application note describes the LVDS RX interface in detail along with the various configurations, and its features. This document also provides detailed state diagrams to explain how data is being transmitted by the FPGA over the LVDS interface of the EZ-USB™ FX20 controller.

Additionally, the FPGA system-level block diagram describes how data is captured, processed, buffered, and sent by the FPGA to the EZ-USB™ FX20 controller. For better understanding, the reference FPGA project is provided to explain the complete architecture of the design.

This document also lists a few troubleshooting techniques to overcome common mistakes while using the LVDS interface.

This application note does not include the information on LVDS control byte encoding. For more details, refer to the “LVDS control byte encoding” section from the architecture reference manual [1].

[1] In the rest of the document, EZ-USB™ FX20 is used as a short form to refer EZ-USB™ FX20/FX10/FX5N/FX5.

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

2.1 Key features of LVDS RX interface

1. The EZ-USB™ FX20 controller supports only RX mode and FPGA transmits data using LVDS TX interface.
2. The EZ-USB™ FX20 controller has two LVDS RX ports, named Port 0 and Port 1. Each port supports a maximum of up to eight data lanes, one clock, and one control lane.
3. Each LVDS port has seven dedicated single-ended LVCMOS control signals. Out of seven, three can be configured only as an input and the rest as input or output. Two output control signals are used as a trigger (LINK_RDY) to start training from FPGA and as a buffer ready (BUFFER_RDY) to indicate whether the EZ-USB™ FX20 controller is ready to receive data from FPGA or not.
4. Each lane supports a 1:1, 2:1, 4:1 or 8:1 gear ratio and supports clock frequency ranging from 74.25 MHz to 625 MHz.
5. Each lane supports maximum throughput (that is, clock frequency x gear ratio) up to 1250 Mbps (156.25 MHz x 8 bits) per data lane. In simple words,
   - If the clock frequency is 156.25 MHz or less, the maximum supported gear ratio is 8:1.
   - If the clock frequency ranges from 156.25 MHz to 312.5 MHz, the maximum supported gear ratio is 4:1.
   - If the clock frequency is greater than 312.5 MHz, the maximum supported gear ratio is 2:1.
6. Each port can be configured individually (narrow link) with one control lane, one clock lane, and one or more data lane(s) or both ports can be combined and be treated as a single wide link port supporting 16 data lanes, two clock lanes, two control lanes and 14 dedicated LVCMOS control signals.
7. Data lanes carry data bytes and a full byte in the control lane identifies whether data lanes carry the data or control information.
8. Each lane (control or data) should carry information in multiple bytes (8 bits) only and the most significant bit must be sent first.
9. The control byte indicates if the data transfers are full-width-transfers or partial transfers. A full-width transfer carries data bytes in all data lanes. A partial transfer must be the last transfer to a buffer before End of Packet (EOP) is transmitted on the control byte to indicate the end of data transfer to the current buffer. When partial data[2] is sent, it must be present on the least significant lanes and must be less than the number of active data lanes.
10. The LVDS interface on EZ-USB™ FX20 controller requires PHY and LINK training before starting receiving data and control bytes.
11. The LVDS interface on the EZ-USB™ FX20 controller supports Active mode (L0) and two low-power modes (L1 and L3).

[2] Partial data refers to data bytes less than the width of the active data lanes.

2.2 Link initialization and training

LVDS training comprises two parts, that is, PHY training and LINK training. LVDS training is mandatory on power-up or after any low-power mode (L1 or L3) exit.

2.2.1 PHY training

PHY training is required in the LVDS mode to ensure that the LVDS RX properly samples the data sent by the FPGA. The FPGA is expected to continuously send a known data pattern (matching the LVDS PHY Training Mode Config register) on the control and all data lanes for a minimum 50 µs. As soon as FPGA is configured, it may start sending PHY Training Sequence byte (PHY_TS) on the control and all the data lanes or when the EZ-USB™ FX20 controller is initialized and ready for the PHY training, it may inform FPGA to start sending PHY Training Sequence byte using LVCMOS control signal or an I2C/UART. Once FPGA is informed, FPGA shall start transmitting PHY Training Sequence byte (PHY_TS) for minimum TS_Time (50 us) on all the data lane(s) as well as the control lane(s). It helps the EZ-USB™ FX20 controller to correct its byte boundary on each lane.

Note: *In the reference project, to inform FPGA to start PHY training, the LVCMOS control signal (LINK_RDY) is used.*

2.2.2 LINK training

LINK training is used to ensure that data received by the EZ-USB™ FX20 controller from each of the lanes is properly aligned. Immediately after PHY training, FPGA must start transmitting LINK Training Block sequence (P0, P1, P2, and P3) to compensate skew and align data bytes across all data and control lanes.

Note that in wide link mode, the PHY Training Sequence byte and LINK Training Block sequence need to be sent on both control lanes, even though the Port1 control lane will not be used for data transfers.

Note: *Upon successful completion of training, the EZ-USB™ FX20 controller informs the FPGA by writing status to PHY/Link Training Control and Status register through I2C. If the training completion acknowledgement mechanism is not defined, the FPGA must wait for 3µs before moving to data transmission.*

2.3 Flow control

A flow control mechanism is available for each LVDS port using a buffer-ready signal. Any of the LVCMOS control pins associated with the Port can be used with appropriate firmware programming.

Buffer ready (BUFFER_RDY) signal can be used to verify that the EZ-USB™ FX20 controller is ready to receive data before starting the data transfer. Due to latencies between the LVDS transfers and the LVCMOS control signal updates, this cannot be used as an indication of when the transfer should be stopped. Instead, the packet size (a.k.a. device buffer size, that is, the number of bytes that FPGA must transmit in a single transfer) must be informed by the EZ-USB™ FX20 controller via I2C/UART or SPI. FPGA shall check for buffer ready (BUFFER_RDY) to be asserted before initiating any packet transfer. Once a transfer is initiated, FPGA shall not look at buffer ready (BUFFER_RDY) status until the next packet transfer.

FPGA shall always check buffer ready (BUFFER_RDY) signal before transferring any new packet. This ensures that there is no data drop because of the unavailability of space in the buffer.

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

2.4 LVDS RX power modes

As mentioned in the introduction, the LVDS interface supports different power modes, L0, L1, and L3. L2 mode is not defined. L0 is the active (Normal) mode whereas L1 and L3 are the low power modes. These modes are applicable to port 0 or port 1 independently and wide link mode also. Low power mode entry or exit is controlled by the FPGA. FPGA can enter low power mode to save power when there is no data to transmit.

Figure 1 describes the different power modes. Each port can be in a different power mode independent of the other port except in a wide link configuration.

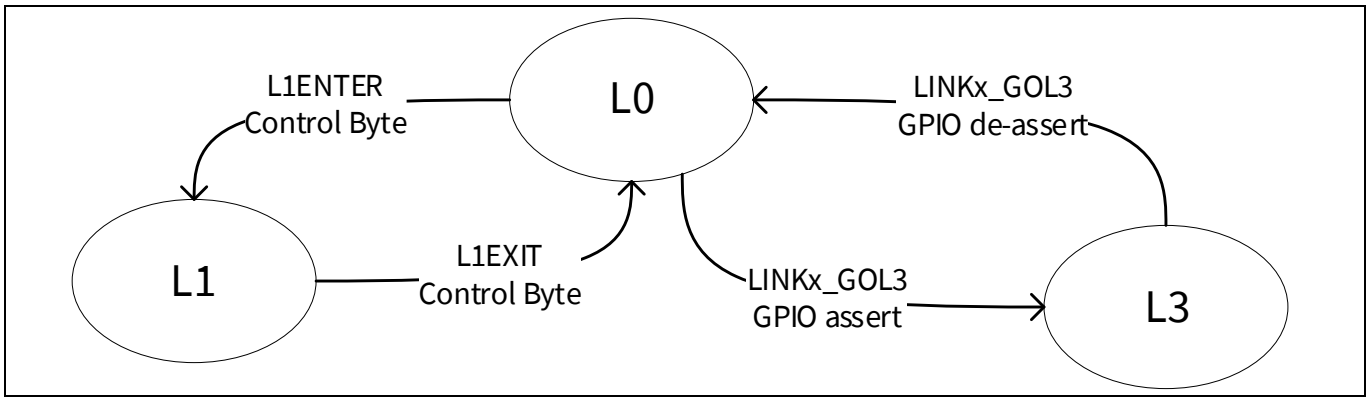


Figure 1 LVDS RX power modes

2.4.1 Active mode – L0

In this mode, all data lanes, one control lane, and clock lane(s) are active. It should be noted that one clock lane shall be used for each port with up to eight data lanes and both the clock lanes shall be used for the wide link (with 16 data lanes) mode. In case of a wide link, both clock lanes shall be driven by the same FPGA clock source.

In this mode, FPGA shall actively drive the clock, control, and data lanes. If FPGA does not have valid data to transmit, it shall transmit an IDLE control byte on the control lane whereas data lanes can carry any value. The data present on the data lanes will be discarded by the EZ-USB™ FX20 controller because the IDLE control byte is present on the control lane.

2.4.2 Low-power mode – L1

In this mode, all the active data lanes are powered down whereas the clock and control lanes continue to be active.

The time to enter L1 mode (L1ENTER_TIME) would be around 20x interface cycles. The time to exit L1 mode (L1EXIT_Time) and get back to L0 mode would be around 50 µs.

2.4.2.1 Entry to L1 mode

To enter L1 mode, FPGA shall transmit the L1ENTER control byte. It will cause the EZ-USB™ FX20 controller to initiate entry to L1 mode and eventually, data lanes will be turned off.

During the L1 entry period (L1ENTER_TIME), which can take up to 300 ns, the FPGA must continue to transmit L1ENTER control bytes on the control lane while data and clock lane(s) continue to be active. The data received with L1ENTER control bytes will be discarded.

After L1ENTER_TIME, FPGA may put the data lanes in a low-power state while continuing to transmit L1ENTER control bytes on the control lane with an active clock lane.

There will be no acknowledgement from the EZ-USB™ FX20 controller to FPGA which indicates that data lanes are turned off. It is up to FPGA to decide how much time to be in this mode.

2.4.2.2 Exit from L1 mode

To exit L1 mode, the FPGA must transmit L1EXIT control byte on the control lane and start transmitting PHY Training Sequence byte (PHY_TS) for TS_Time (50 us) on all the data lane(s) as well as the control lane.

Upon completion of PHY training, FPGA shall immediately start LINK training by transmitting LINK Training Block sequence (P0, P1, P2, and P3) on all the data lane(s) and control lane.

Once LINK training is completed, FPGA shall start transmitting valid data with valid control bytes.

2.4.3 Low-power mode – L3

In this mode, all the data, control, and clock lanes are powered down. The time to enter L3 mode (L3ENTER_TIME) would be around 20x interface cycles and the time to exit L3 mode (L3EXIT_Time) and get back to L0 would be around 5 milliseconds.

FPGA shall use a LINKx_GOL3 control to signal for the entry and exit to and from L3 mode. LINKx_GOL3 will be one of the LVCMOS control signals associated with each port. This GPIO will be capable of waking the EZ-USB™ FX20 controller from deep sleep, and it can also be any of the generic GPIOs.

2.4.3.1 Entry to L3 mode

FPGA may choose to go to low power L3 mode by asserting LINKx_GOL3 control signal. For wide link configuration, LINK0_GOL3 shall be used. During the L3 entry time (L3ENTER_TIME), FPGA shall continue to transmit IDLE control bytes on the control lane while data and clock lanes continue to be active.

After L3ENTER_TIME, FPGA shall put the data, control, and clock lanes in a low power state or power them down. There will be no acknowledgement from the EZ-USB™ FX20 controller to FPGA to indicate that the EZ-USB™ FX20 controller has been in L3 mode.

It is up to FPGA to decide how much time to be in this mode.

2.4.3.2 Exit from L3 mode

To exit L3 mode, FPGA shall de-assert the LINKx_GOL3 control signal which was used to enter the same mode. Before de-asserting, FPGA activates the clock lane(s) and transmits PHY Training Sequence byte (PHY_TS) for L3EXIT_Time (5 milliseconds) on data and control lanes. De-asserting LINKx_GOL3 causes the EZ-USB™ FX20 controller to get out of deep sleep.

Upon completion of PHY training, FPGA shall immediately start LINK training by transmitting LINK Training Block sequence (P0, P1, P2, and P3) on all the data lane(s) and control lane.

Once LINK training is completed, FPGA shall start transmitting valid data with valid control bytes.

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

2.5 Thread and socket configuration

Each LVDS RX port of the EZ-USB™ FX20 controller supports two threads. Port 0 supports thread 0 and 1, Port 1 supports thread 2 and 3, however, wide link mode supports all four threads (thread 0 to 3). Each thread supports a maximum of up to eight sockets (socket 0 to 7). Therefore, there are a total of 32 sockets for four threads. Each socket has been allocated with multiple buffers to store the data.

Based on design requirements, the designer(s) can decide how many thread(s) or socket(s) should be used for the transfer. FPGA must send the same thread (STAD[tt]) and socket number (SSAD[kkk]) through the control lane before each data transfer.

For example, if thread 0 and socket 0 are decided to be used for the transfer, the FPGA must send STAD[00] and SSAD[000] control bytes to select the thread and socket of the EZ-USB™ FX20 controller. Section 2.6.2 provides more information on the LVDS transaction flow between the FPGA and the EZ-USB™ FX20 controller.

Note: *For information on thread, socket, and control byte, refer to the "Sockets, descriptors, buffers and packets" section and "Socket thread mapping in FX20" table in the architecture reference manual [1].*

The thread and socket configuration are portrayed in Figure 2.

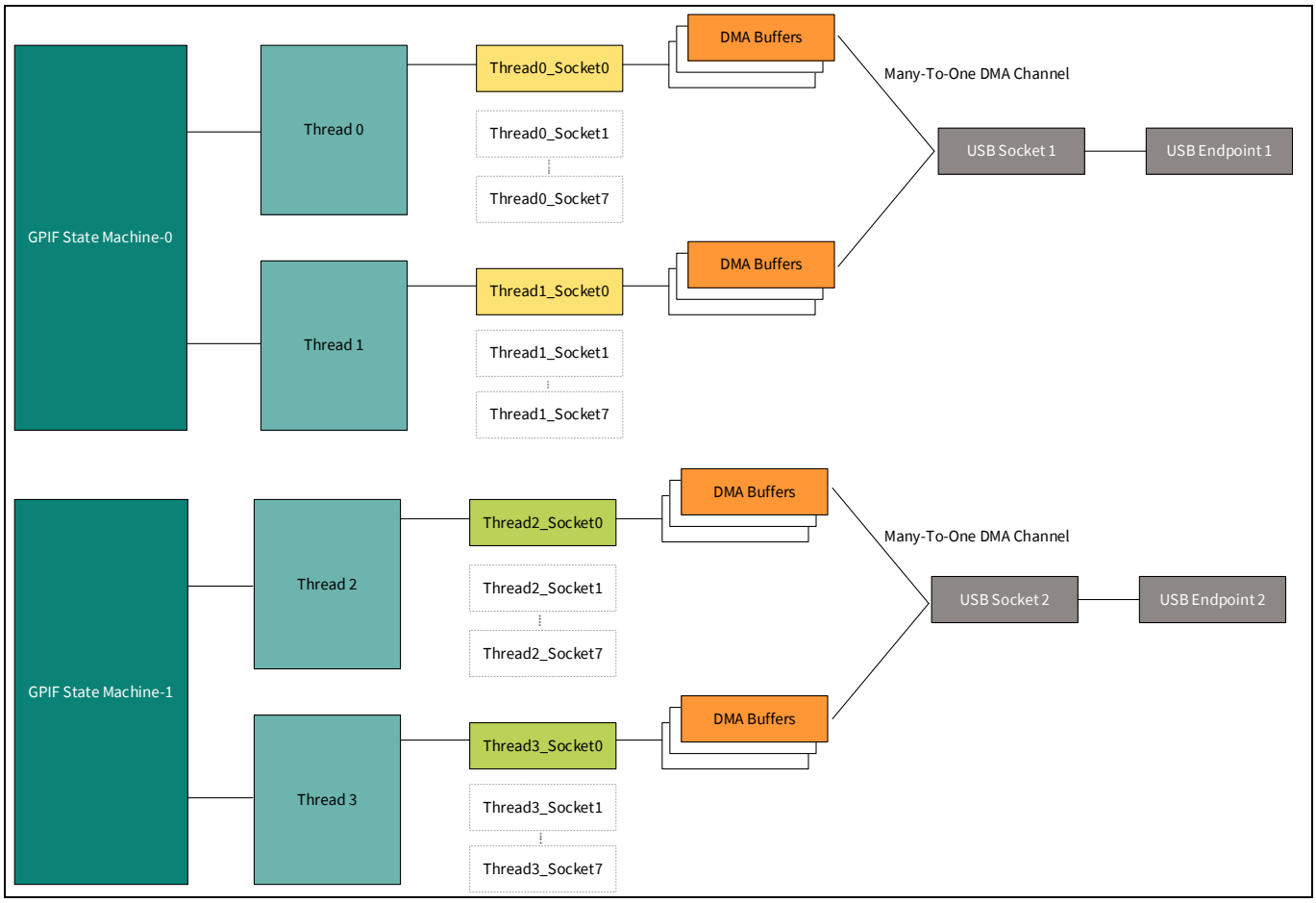


Figure 2 Thread and socket configuration

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

2.6 Data transmission flow

After the completion of link training, FPGA can start transmitting data and control information to the EZ-USB™ FX20 controller on data and control lanes, respectively. Control byte received on the control lane will indicate whether the data lanes carry data or control-related information. The following sections explain the basic as well as detailed transaction flow between FPGA and EZ-USB™ FX20 controller.

2.6.1 Basic LVDS transaction flow

Figure 3 describes the basic LVDS transaction flow between the EZ-USB™ FX20 controller and FPGA along with the Low-power modes. On power-up, FPGA performs the training before starting transmitting data. This mode is known as Active (L0) mode. If the FPGA does not have data or if the EZ-USB™ FX20 controller informs via I2C/UART or SPI, the FPGA can enter any Low-power modes to save power. Upon any Low-power mode exit, training is required.

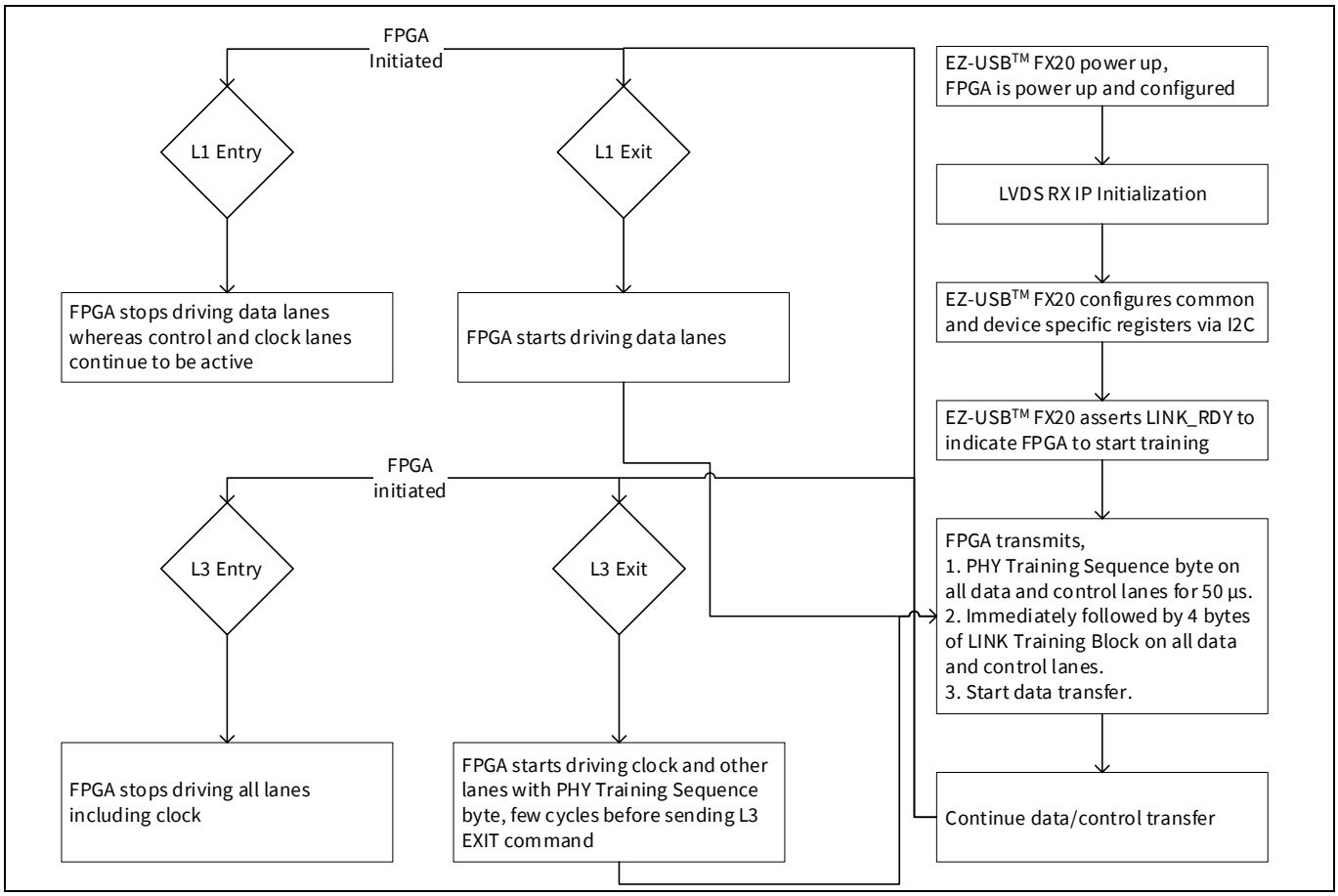


Figure 3 Basic LVDS flow diagram with power modes

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

2.6.2 Detailed LVDS transaction flow

Figure 4 shows the detailed LVDS write transaction flow between the FPGA and the EZ-USB™ FX20 controller. It describes the order in which the FPGA (LVDS TX) sends the control and data information to LVDS RX.

Note that the BUFFER_RDY, LVCMOS control signal is used for the flow control. It indicates to FPGA whether a buffer within the EZ-USB™ FX20 controller is ready to receive data or not.

1. After the EZ-USB™ FX20 controller is up and FPGA is configured, the EZ-USB™ FX20 controller should configure the FPGA registers such as PHY Training Mode Config and LINK Training Block sequence (P0 to P3), thread and socket information, buffer size over I2C and assert the LINK_RDY signal to start the training. Once the LINK_RDY signal is asserted (HIGH), the FPGA must send the PHY Training Sequence byte (PHY_TS) for a minimum 50 µs, followed by the LINK Training Block sequence (P0, P1, P2, and P3) on all data lanes and control lanes. Refer to section 3.2.1 for more information on the FPGA's I2C registers.
2. The FPGA waits for the acknowledgement from the EZ-USB™ FX20 controller. Once PHY and LINK training is successfully completed, the EZ-USB™ FX20 controller acknowledges the FPGA by updating the PHY/LINK Training Control and Status register. Along with this, the EZ-USB™ FX20 controller configures FPGA's Active Device Mask (list of connected source devices), thread/socket information and buffer size registers. Active Device Mask register defines how many source devices are connected to the FPGA. The Thread and Socket Info register is unique for each device, and it defines on which socket and thread address, source device's frame data needs to be transferred by the FPGA.

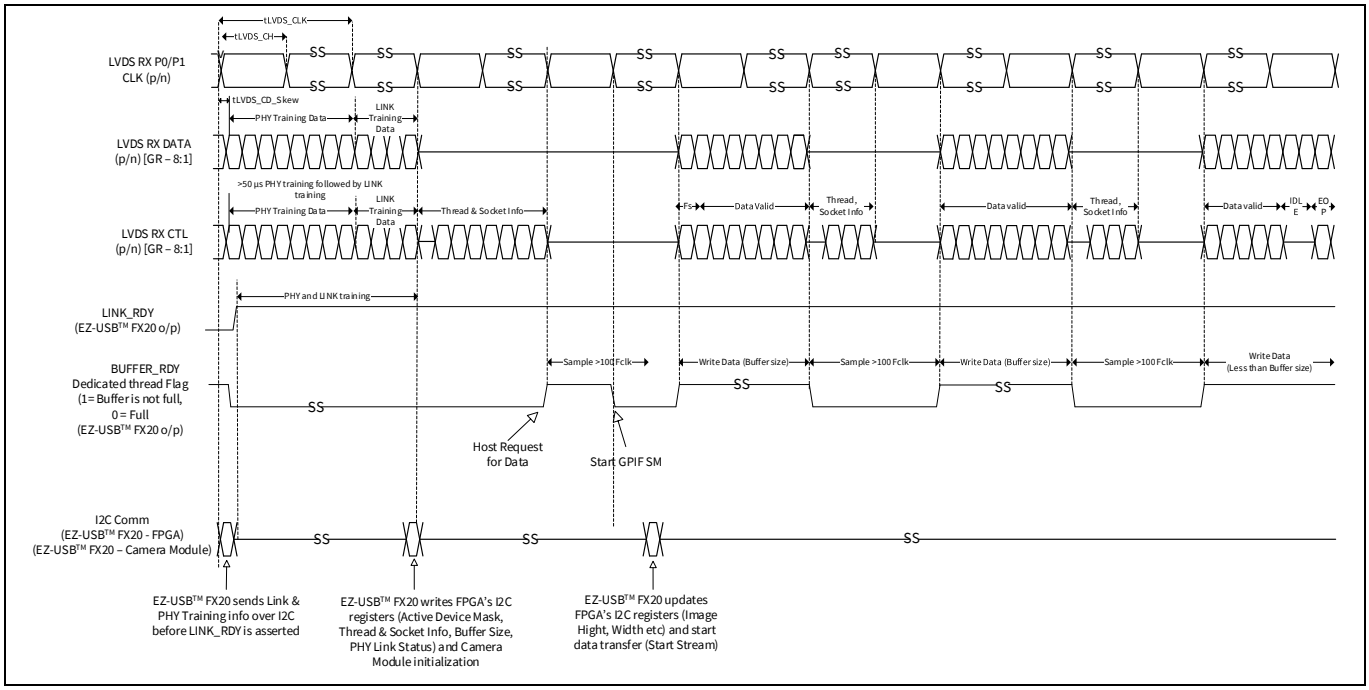


Figure 4 Detailed LVDS transaction diagram

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

3. The FPGA uses the received information to configure the EZ-USB™ FX20 thread and socket of the selected device by sending commands over the LVDS control lane and waits for minimum 100 LVDS clock cycles before sampling the buffer ready (BUFFER_RDY) control signal.
4. When the host application requests data, the EZ-USB™ FX20 starts the GPIF state machine and enables the DMA channel. This step asserts the buffer ready (BUFFER_RDY) control signal, indicating that the DMA buffer is available to the FPGA to write the frame data of the selected device.
5. The EZ-USB™ FX20 controller configures device-specific registers like image height, width, and frame rate over I2C. Along with it, it also instructs the FPGA to start data transfer of enabled source devices by updating the Start Stream register.
6. After receiving the 'Start/Stop stream' signal (HIGH) and the buffer ready (BUFFER_RDY) signal is asserted by the EZ-USB™ FX20 controller, the FPGA starts transmitting the DATA_CMD control byte on the control lane and data bytes on the data lane equals to the buffer size register. The DATA_CMD control byte on the control lane indicates to the EZ-USB™ FX20 controller that the data present on the data lanes is valid. Once the FPGA sends the full packet (equal to the buffer size) is sent, it starts sending the IDLE_CMD control bytes until the buffer ready (BUFFER_RDY) is de-asserted by the EZ-USB™ FX20 controller. When the IDLE_CMD control byte is received by the EZ-USB™ FX20 controller, it just discards the data present on the data lanes.
7. Once buffer ready (BUFFER_RDY) is de-asserted, and two threads are assigned to the selected source device, the FPGA sends the new thread and socket number over the control lane and waits for 100 LVDS clock cycles to sample buffer ready (BUFFER_RDY). When the buffer for the sent thread and socket address is allocated by the EZ-USB™ FX20 controller, the buffer ready (BUFFER_RDY) will be asserted and the FPGA starts writing data by sending DATA_CMD on the control lane and data on the data lanes. After sending the exact buffer size of data, the FPGA starts sending IDLE_CMD control bytes. Repeat these steps until they reach the end of the frame.
8. The FPGA keeps track of bytes being written to the EZ-USB™ FX20 controller for the selected device. When it completes writing the full frame data, in the end, it sends the EOP_CMD control byte on the control lane to inform the EZ-USB™ FX20 controller that it is the end of the packet and waits for the buffer ready (BUFFER_RDY) to be de-asserted. When the EZ-USB™ FX20 controller receives the EOP_CMD control byte, it de-asserts the buffer ready (BUFFER_RDY) control signal and commits the buffer to the USB Host even if it is not full (partially full). If the last buffer is the full buffer, no need to send the EOP_CMD as the EZ-USB™ FX20 controller will commit that buffer to the USB Host and de-asserts the buffer ready (BUFFER_RDY) control signal.
9. After sending a full frame of the selected device, the FPGA moves to the next enabled device. This information is already available on the Active Device Mask register. If only a single device is enabled (connected with the FPGA), it starts with the same device to send the next frame data. Thus, the FPGA serves a maximum of eight devices in a round-robin manner.

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

2.7 Supported gear ratios

The Gear ratio indicates the number of bits that can be transferred per data lane per clock on the interface. 1:1, 2:1, 4:1 and 8:1 are supported by gear ratio values for all the port configurations, individual Port 0, Port 1, or wide link. FPGA shall transmit a minimum of 1 byte on each data and control lane and partial byte transfer is not allowed.

LVDS RX supports a maximum 1.25 Gbps data rate per lane. If the clock frequency is 156.25 MHz or less, the maximum supported gear ratio is 8:1. If the clock frequency ranges from 156.25 MHz to 312.5 MHz, the maximum supported gear ratio is 4:1 and if the clock frequency is greater than 312.5 MHz, the maximum supported gear ratio is 2:1.

Figure 5 illustrates how data or control information is sent on each data or control lane per clock cycle for each supported gearing ratios. It must be noted that the most significant bit of the byte is sent first.

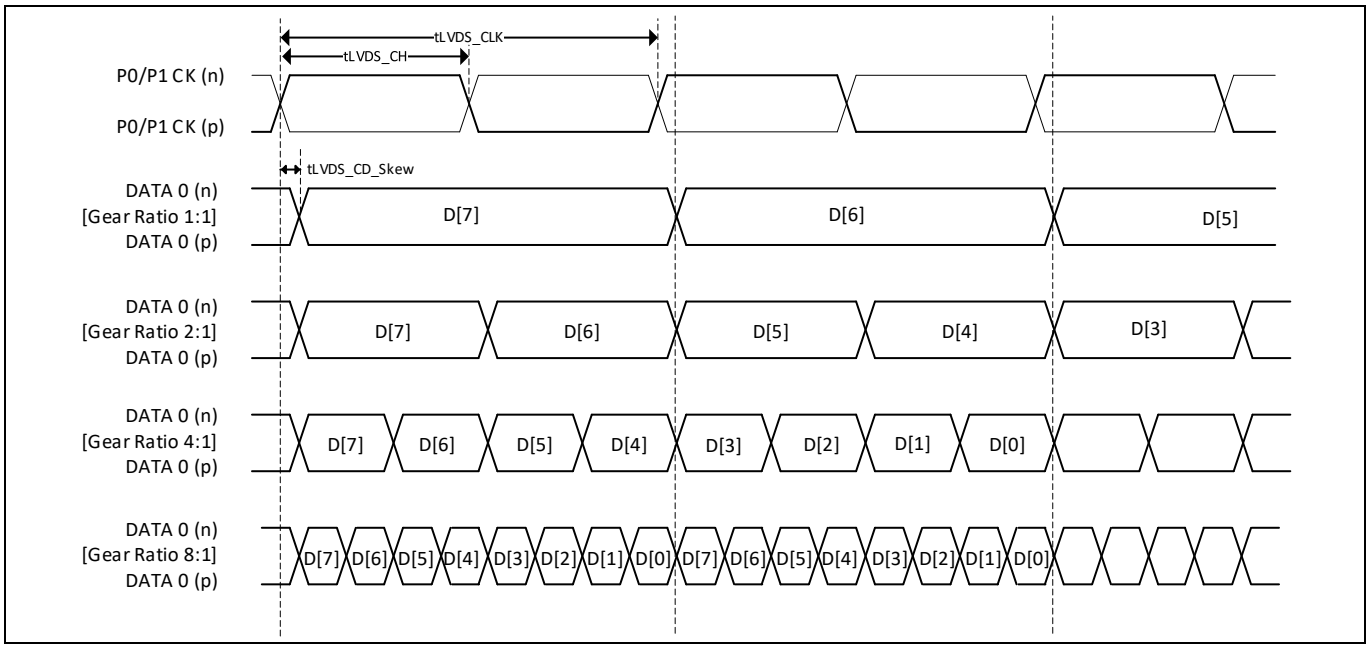


Figure 5 Gear Ratio 1:1, 2:1, 4:1 and 8:1 illustrated on LVDS Control and Data lane 0

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

Figure 6 shows how data and control information is written by the FPGA when the gear ratio is selected as 8:1 and Port 0 is used. As mentioned in Figure 5, in LVDS, data will be sent serially only and for better understanding, in each clock, the bit mask is written on each data and control lane. It explains how data byte is written on each data lane.

In this scenario, only a single LVDS clock cycle is required by the FPGA to send a full byte on each data and control lane. Therefore, a total of 8 bytes will be sent in each LVDS clock cycle.

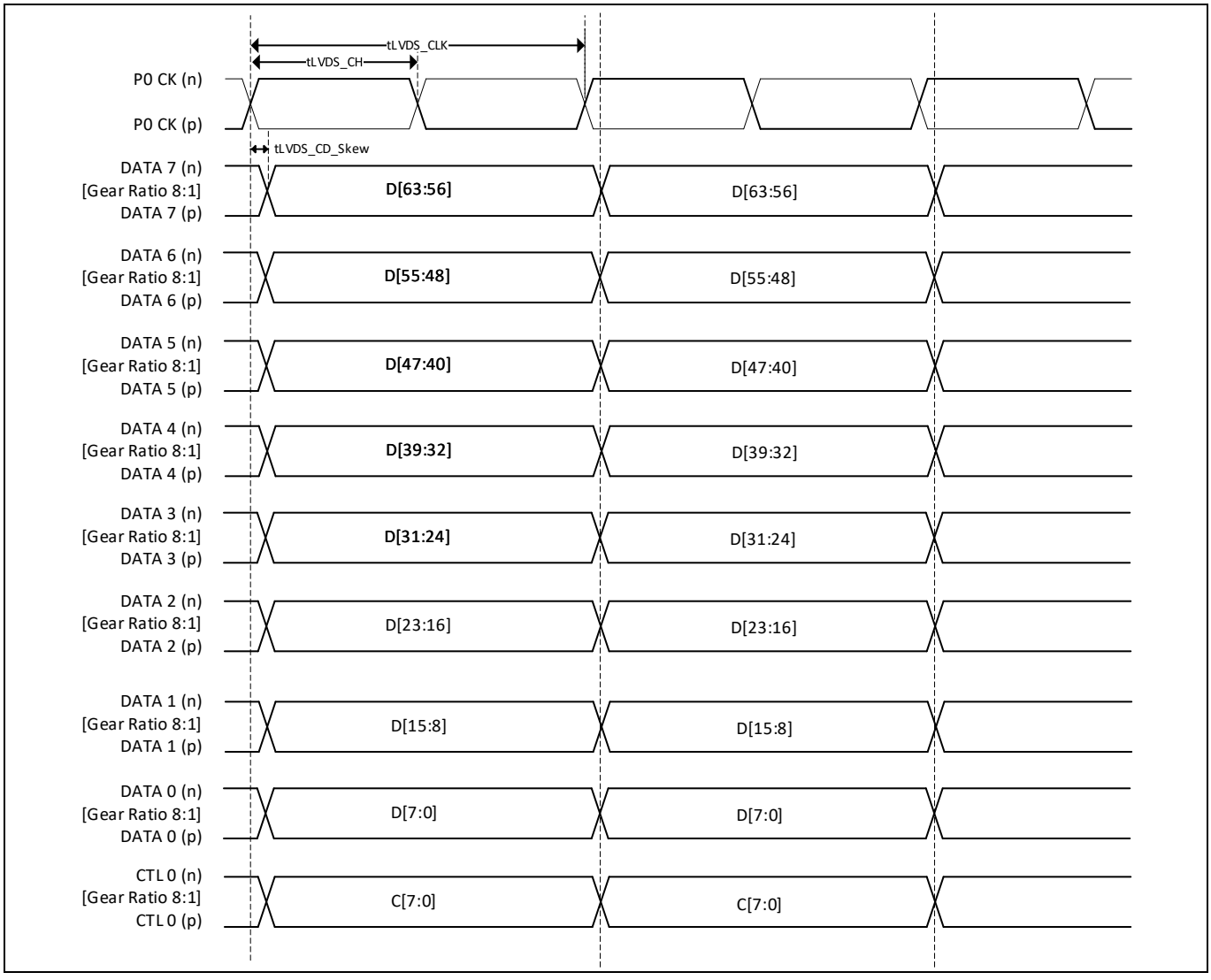


Figure 6 Port 0 data transfer with 8:1 gear ratio

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

Figure 7 shows the data transfer scenario on Port 1 for the gear ratio 4:1. The FPGA requires two LVDS clock cycles to send a full byte on each data and control lane. Therefore, two LVDS clock cycles are required to send 8 bytes of data.

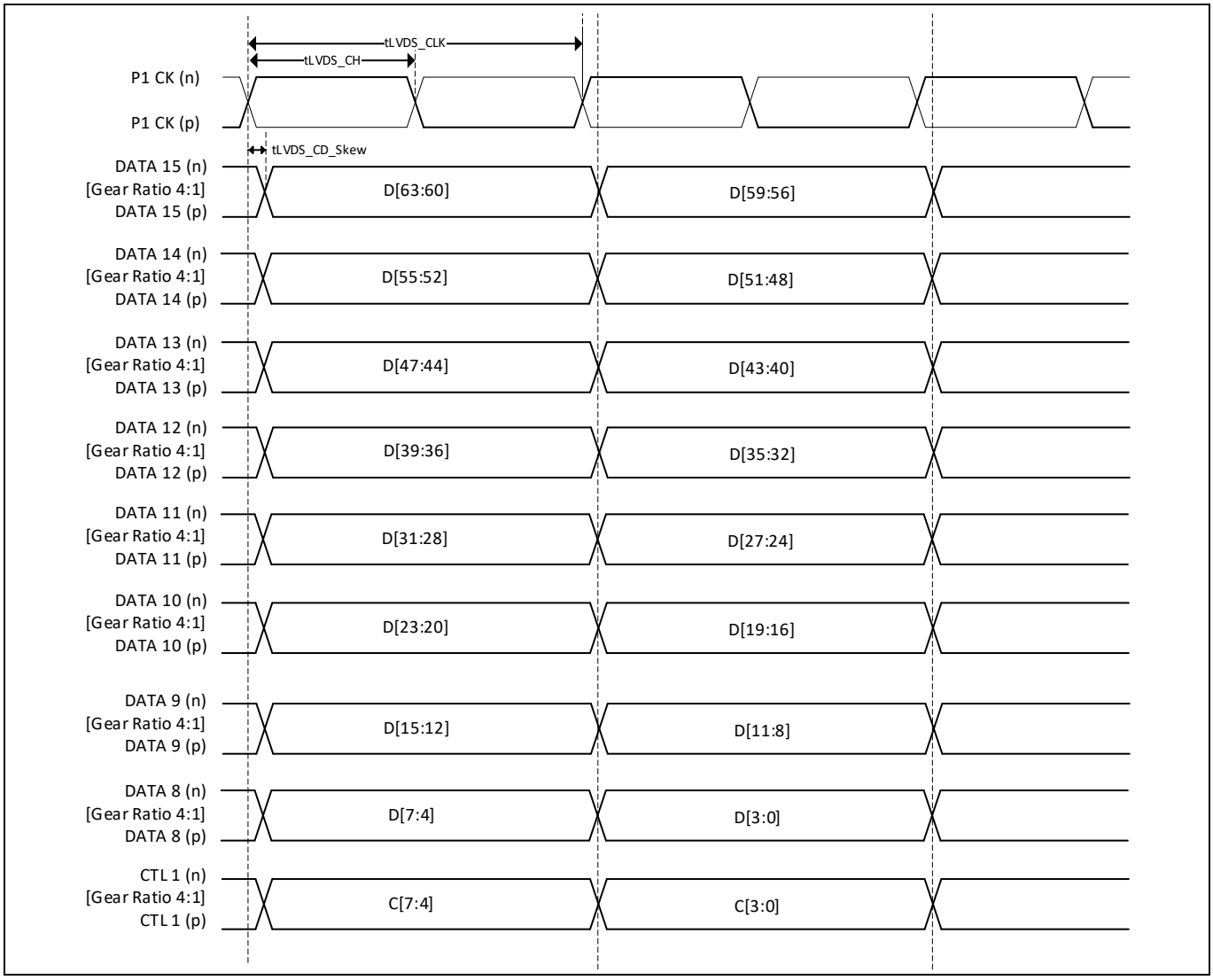


Figure 7 Port 1 data transfer with 4:1 gear ratio

2 LVDS RX interface of EZ-USB™ FX20/FX10/FX5/FX5N controller

Figure 8 shows the wide link data transfer with 4:1 gear ratio. In wide link mode, the same LVDS clock must be present on both clock lanes whereas Port 1 control lane is also shown as it is required during PHY and LINK training.

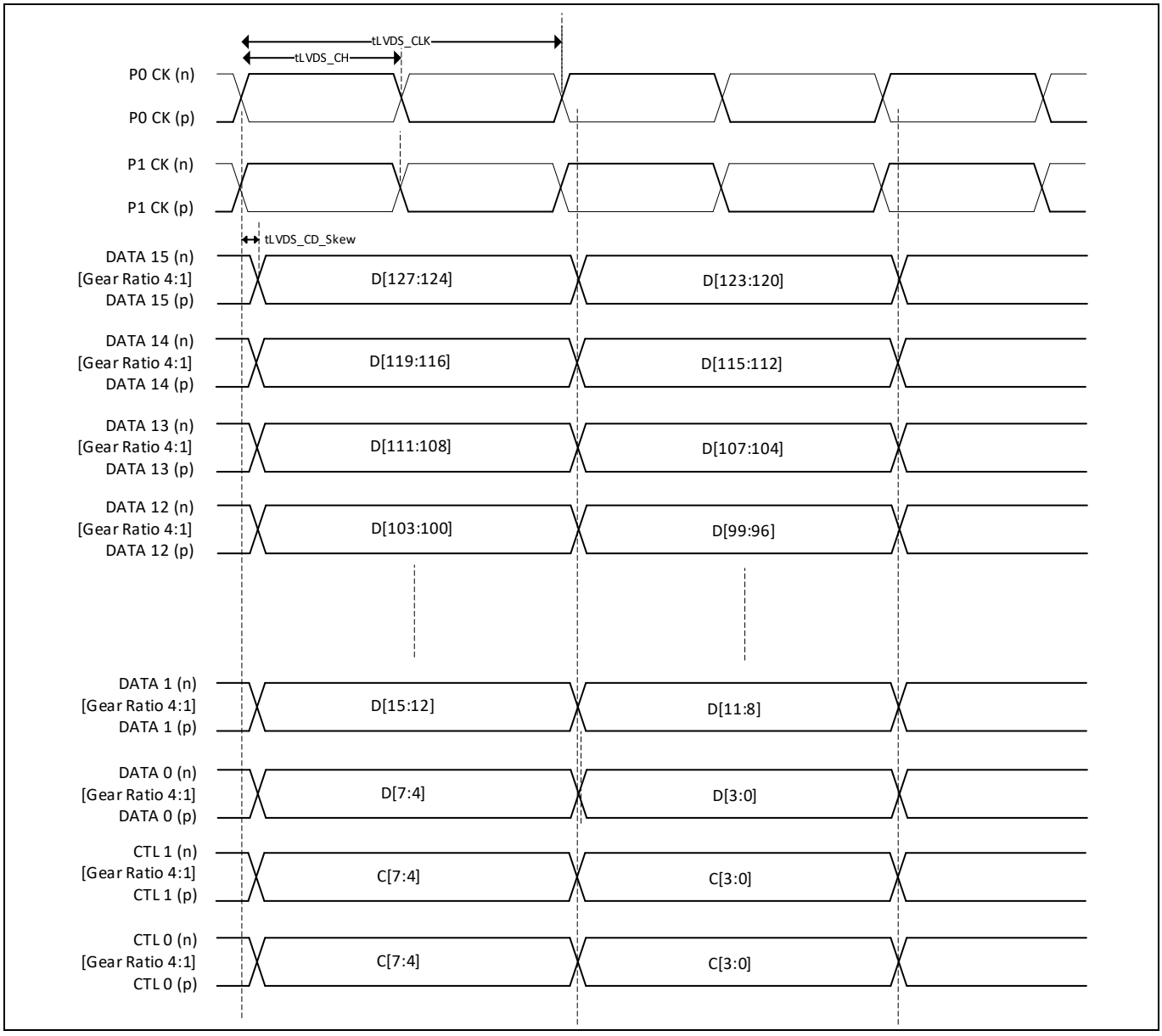


Figure 8 Wide Link data transfer with 4:1 gear ratio

2.8 Data integrity

The control information on the control lane between the FPGA and the EZ-USB™ FX20 controller is always protected using an odd parity bit. The most significant bit of control byte is used as the parity bit. Whereas the data transmission is protected using a 16-bit cyclic redundancy check (CRC) and it is an optional feature which can be used to detect data integrity errors during the hardware debug.

Note: The reference design provided with this application note does not support data and header CRC.

2.8.1 Data CRC

To utilize this feature, in the beginning of the transfer, the FPGA shall send the INITDCRC control byte on the control lane to start the CRC calculation on the EZ-USB™ FX20 controller side. The EZ-USB™ FX20 controller starts updating the CRC calculation on receiving each new byte. FPGA shall also calculate CRC on valid data bytes being transmitted to the EZ-USB™ FX20 controller. At the end of data transmission, FPGA shall transfer calculated 16-bit CRC on the least significant data lanes (data lanes 0 and 1) and the control lane shall carry CHKDCRC control byte to indicate that data being transferred on data lanes is CRC value. As CRC is 16-bit and each data lane can carry only 1 byte worth of data, in scenarios where only a single data lane is activated, CRC shall be transferred in consecutive cycles, using 2xCHKDCRC control bytes.

The EZ-USB™ FX20 controller calculates CRC on all valid data bytes received and the calculated CRC is compared against CRC transmitted by FPGA. If they do not match, the data sent by FPGA got corrupted in transmission and the THREADx_DATA_CRC_ERR interrupt will be raised.

The data CRC value computed by the EZ-USB™ FX20 controller can be added into the data buffer as part of metadata using the INMD control byte.

The polynomial used for the data CRC is $X^{16}+ X^{12} + X^5 + 1$.

2.8.2 Metadata (header) CRC

To protect the integrity of the header, including the leader and trailer, the EZ-USB™ FX20 controller explicitly supports CRC for header verification. For header CRC as well, the same polynomial $X^{16}+ X^{12} + X^5 + 1$ as data CRC is used.

Note: For more information on how data and header CRC can be used, refer to the "LVDS control byte encoding" section from the architecture reference manual [1].

3 Implementation of LVDS transmitter in FPGA

3.1 System level block diagram

This section provides an overview of the system to capture data from the video source using the EZ-USB™ FX20 controller. It uses FPGA as a mediator, which captures data from multiple video sources, such as MIPI or HDMI, processes, and transmits to the EZ-USB™ FX20 controller over the LVDS interface at the maximum speed of 20 Gbps.

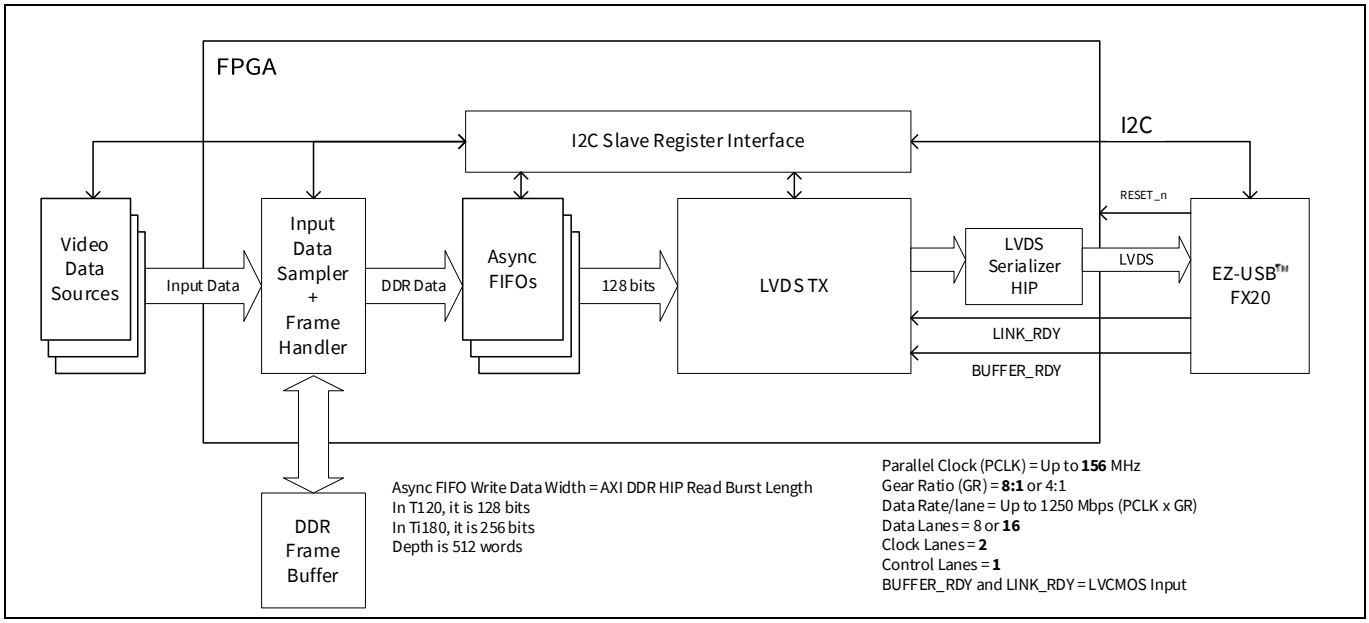


Figure 9 High-level block diagram of video data capture using EZ-USB™ FX20 controller

As illustrated in Figure 9, the multiple video (maximum 8) sources are connected to the FPGA. It can be either MIPI or HDMI-based camera sensor. The key role of the FPGA is to capture or sample data from each video source, process, store, and transmit to the EZ-USB™ FX20 controller over the LVDS interface in a round-robin manner.

As soon as the EZ-USB™ FX20 controller is initialized and the FPGA is configured, the EZ-USB™ FX20 controller initiates I2C writes to configure the LVDS PHY Training Mode Config and LVDS LINK Training Block Bytes P0 to P3 register defined in section 3.2.1 and asserts the LINK_RDY control signal to inform the FPGA to start the PHY and LINK training.

Based on the LVDS PHY Training Mode Config and LVDS LINK Training Block Bytes P0 to P3 register data, the LVDS TX block trains the LVDS RX and waits until the EZ-USB™ FX20 controller informs the FPGA that training is successful through PHY/Link Training Control and Status register. Along with it, the Active Device Mask and Thread/Socket Info registers are also written. The Active Device Mask register defines how many source devices are connected with the FPGA and the Thread/Socket Info register defines the thread and socket address for each device source.

Once training is successful, the FPGA sends thread/socket address control bytes and waits for a minimum of 100 LVDS clock cycles before the sampling buffer-ready (BUFFER_RDY) control signal. If it is still not asserted by the EZ-USB™ FX20 controller, it waits until it gets asserted.

3 Implementation of LVDS transmitter in FPGA

If the EZ-USB™ FX20 controller is ready to receive data, that is, buffer allocated to thread and socket is empty, it informs the FPGA by asserting buffer ready (BUFFER_RDY) control signal and writing Stream Enable register. It must be noted that the Stream Enable register informs the FPGA whether to start or stop data transfer whereas buffer ready (BUFFER_RDY) indicates that the EZ-USB™ FX20 controller has a buffer available for the selected thread. Therefore, it will be de-asserted after every packet (buffer) transfer. Figure 4 explains the detailed behavior of the buffer ready (BUFFER_RDY) control signal.

Upon a data transfer request from the EZ-USB™ FX20 controller, the Input Data Sampler samples the data from each video source and stores it in an intermediate buffer. The sampled data is transferred to DDR memory by the Frame Handler in a round-robin manner.

The main intention of using DDR memory as a frame buffer is to store multiple full frames of each video source. If the USB Host is slower or occupied with other tasks or the EZ-USB™ FX20 controller is enumerated in either USB3.2 Gen 2 or lower than the input data speed (USB3.2 Gen2x2), the USB interface between the USB Host and the EZ-USB™ FX20 controller creates the bottleneck. In this scenario, the entire frame within the frame buffer will be overwritten by the Frame Handler and as a result, the output frame rate on the USB Host side is reduced. However, the data within the frame will not be dropped.

The buffered data of each video source is read and stored into an asynchronous FIFO, which basically manages the data rate between the Input Data Sampler block and LVDS transmitter (LVDS TX). Apart from training, one of the main tasks of the LVDS TX block is to read the frame data from asynchronous FIFO and send it to the EZ-USB™ FX20 controller in a round-robin manner. It can also add the header, leader, and footer bytes if required according to the configured USB class (Streaming Mode register).

In the end, data received by the EZ-USB™ FX20 controller is sent to the USB Host. If the Host wants to stop the transfer, it informs the EZ-USB™ FX20 controller through USB command and eventually, the EZ-USB™ FX20 controller informs the FPGA by writing Stream Enable register through I2C.

In this example, LVDS TX is configured in wide link mode with 8:1 gear ratio and 156.25 MHz clock. The data rate would be 1.25 Gbps per lane (156.25M x 8) and in a wide link mode, a total of 16 data lanes are used, therefore, the total data rate is 20 Gbps between the FPGA and the EZ-USB™ FX20 controller.

3.2 I2C Slave Register interface

I2C (Inter-Integrated Circuit) is a widely used communication protocol that enables data exchange between master and slave devices. Initially, the FPGA requires a range of configuration data, including PHY Training Sequence byte and LINK Training Block sequence (P0, P1, P2, and P3), image height and width, frames per second (FPS), bits per pixel, thread/socket information, when to start or stop sending data and other relevant parameters. To store these types of configuration parameters in FPGA, registers are implemented which can be accessed by the EZ-USB™ FX20 controller using I2C.

In this specific application, the EZ-USB™ FX20 controller acts as a master device, while the FPGA acts as the slave. After configuring or resetting the FPGA, the EZ-USB™ FX20 controller transmits all necessary information to the FPGA before asserting the LINK_RDY control signal.

The example code provided with this application note uses the following pin assignments for SCL and SDA signals.

- SCL: P10_0 (SCB0)
- SDA: P10_1 (SCB0)

3 Implementation of LVDS transmitter in FPGA

3.2.1 FPGA's I2C registers

FPGA's I2C registers are divided mainly into two categories:

- Common control and status registers. These include LVDS TX's configuration, control, and status registers.
- Device-specific registers. These are a set of registers specifically for each device (audio/video source) starting from the device offset. Because LVDS TX supports a total of eight devices, there are a total of eight sets of these registers.

The following sections will describe both types of registers in detail.

3.2.1.1 Base offset for device-specific registers

The LVDS transmitter is designed to manage up to eight devices. Table 1 shows the base offset for each device's device-specific registers.

Table 1 Base offset for device-specific registers details

Sl.#	Register	Register offset address	Description
1	Common Register Base Address	0x00	Offset of Common Control and Status Registers
2	Device 0 Base Address	0x20	Device 0 Buffer Size Info address is 0x36 and 0x37.
3	Device 1 Base Address	0x3C	Device 1 Buffer Size Info address is 0x52 and 0x53.
4	Device 2 Base Address	0x58	Device 2 Buffer Size Info address is 0x6E and 0x6F.
5	Device 3 Base Address	0x74	Device 3 Buffer Size Info address is 0x8A and 0x8B.
6	Device 4 Base Address	0x90	Device 4 Buffer Size Info address is 0xA6 and 0xA7.
7	Device 5 Base Address	0xAC	Device 5 Buffer Size Info address is 0xC2 and 0xC3.
8	Device 6 Base Address	0xC8	Device 6 Buffer Size Info address is 0xDE and 0xDF.
9	Device 7 Base Address	0xE4	Device 7 Buffer Size Info address is 0xFA and 0xFB.

3.2.1.2 Common control and status registers

Table 2 Common control and status register details

Sl.#	Register name	Register offset address	Type	Description
1	RTL Version	0x00	RO	FPGA RTL Version. It needs to be updated after every change in RTL. Bit 7-4: Major version. Bit 3-0: Minor version.
2	Streaming Mode	0x01	RW	Mode of USB Class. It can be either UVC, U3V or Vendor. Bit 7-1: Not used. Bit 0: '1' for UVC or Vendor, '0' for U3V.

3 Implementation of LVDS transmitter in FPGA

Sl.#	Register name	Register offset address	Type	Description
3	Frame Header Info	0x02	RW	Whether frame header should be sent by FPGA or not. If '0', FPGA should not send frame header, it will be taken care by the EZ-USB™ FX20 controller. If '1', FPGA should send header data. Bit 0: FPGA should not add header data. Bit 1: FPGA should add header data along with payload. Bit 2: FPGA should add header data as metadata.
4	LVDS PHY Training Mode Config (PHY_TS)	0x03	RW	LVDS PHY Training Sequence (PHY_TS) byte. It is sent during PHY training on all data and control lanes. It is same as EZ-USB™ FX20 controller Register PHY Training Mode Config Register. Recommended values are 0x9C, 0xB3, 0x4C, 0xCA, 0x35, 0xBA, 0x45, 0x63, 0x39, and 0xC6.
5	LVDS LINK Training Block Byte P0	0x04	RW	LINK Training Block (P0 to P3) needs to be sent through all the active data and control lanes. It is same as EZ-USB™ FX20 controller Register LVDS LINK Training Block Sequence Byte. Recommended value for P0 is 0x78, P1 is 0x4B, P2 is 0x5A, and P3 is 0x12.
6	LVDS LINK Training Block Byte P1	0x05	RW	
7	LVDS LINK Training Block Byte P2	0x06	RW	
8	LVDS LINK Training Block Byte P3	0x07	RW	
9	Active Device Mask	0x08	RW	Each bit corresponds to the connected source device. Here, device is referred to a peripheral from which FPGA receives a data and transmits to EZ-USB™ FX20 controller. It could be either various types of cameras or internal color bar generator which generates data. A maximum of eight source devices can be connected. Bit 0 is for device 1 and bit 7 is for device 8. If the corresponding bit is high, it means, the device is active and mapped to threads. Therefore, LVDS TX FSM will serve this device. If the corresponding bit is 0, it will be ignored by FSM. FSM serves devices in round-robin manner.
10	Low Power Mode Support	0x09	RW	Low power mode support. Bit 7-2: Not used. Bit 1: L3 Power Mode Support. Set means enter L3 mode and reset means exit L3 mode. Bit 0: L1 Power Mode Support. Set means enter L1 mode and reset means exit L1 mode.

3 Implementation of LVDS transmitter in FPGA

Sl.#	Register name	Register offset address	Type	Description
11	PHY/Link Training Control and Status	0x0A	RW	PHY/Link training control and status register. If the corresponding bit is high, only training will be carried out by FPGA. Upon training completion, status/acknowledgement will be given by EZ-USB™ FX20 controller. Bit 7: EZ-USB™ FX20 controller Port 1 training is completed successfully. If it is low after reset, FPGA starts training. Bit 6: EZ-USB™ FX10/20 Port 0 training is completed successfully. If it is low after reset, FPGA starts training. Bit 5-2: Not used. Bit 1: Link Training is required or not. Set means, while training, FPGA transmits the LINK Training Block sequence. Reset means, FPGA excludes the LINK training part. Bit 0: PHY Training is required. Set means, FPGA transmits PHY Training byte and if reset, it excludes PHY training and starts with LINK Training. This is just for testing purpose. The EZ-USB™ FX20 controller LVDS interface requires the training.
12	External Controller Status Info	0x0B	RO	It provides the status of DDR HIP Controller. Bit 7-5: Not used Bit 4: Datapath is idle or not Bit 3: Command queue full status Bit 2: DDR Controller busy status Bit 1: DDR configuration status Bit 0: DMA Ready flag status

3.2.1.3 Device-specific registers

Table 3 Device-specific register details

Sl.#	Register	Register offset address	Type	Description
1	Stream Enable	0x00	RW	This register is for controlling streaming. The EZ-USB™ FX20 controller FW must set start/stop stream bit once all relevant registers are written. Bit 7-3: Not used Bit 2: Stop Stream Immediately (App stop). It is useful for U3V frame only. In U3V, it is mandatory to send full frame and then stop. When EZ-USB™ FX20 controller receives the information from Host (Set/Clear Feature command), to stop the streaming during ongoing frame, it asserts this bit. Based on this, FPGA understands that this is the last frame, and next leader should not be sent. EZ-USB™ FX20 controller can de-assert this bit when it receives trailer for the ongoing U3V frame. For UVC, no need to use this bit. Bit 1: Start/Stop stream. 1 = Start and 0 = Stop. When this bit is set, FPGA starts sending frame data to EZ-USB™ FX20 controller. If it is reset, FPGA stops immediately. Bit 0: Soft reset. It is DMA Reset to restart streaming. It is not being used as of now.
2	Mode Enable	0x01	RW	This register is for enabling/disabling different image formats. Bit 7-4: Not used Bit 3: Enable YUV422-420 Conversion. Bit 2: Enable Mono8 Conversion. Bit 1: Enable Still Capture Bit 0: Enable Interlaced streaming
3	Image Height Info LB	0x02	RW	Register for image height. Bit 7-0: Height lower byte
4	Image Height Info UB	0x03	RW	Register for image height. Bit 7-0: Height upper byte
5	Image Width Info LB	0x04	RW	Register for image width. Bit 7-0: Width lower byte
6	Image Width Info UB	0x05	RW	Register for image width. Bit 7-0: Width upper byte
7	Image Frame Rate Info	0x06	RW	Register for image frame rate. Bit 7-0: frame rate
8	Image Pixel Width Info	0x07	RW	Register for image pixel width in bits. Bit 7-0: Pixel width in bits. For example, 16, 24, 32.

3 Implementation of LVDS transmitter in FPGA

Sl.#	Register	Register offset address	Type	Description
9	Source Type Info	0x08	RW	This register defines which type of device (image data source) is connected with FPGA. It can be MIPI, HDMI, or test pattern generator within FPGA only. Bit 7-2: Not used. Bit 1:0: • 0 - Test Pattern Generated from RTL, that is, internal test pattern or color bar. • 1 - Over HDMI • 2 - Over MIPI • 3 - Other
10	Status Info	0x09	RO	This register shows the status of different FIFOs used. Intermediate data FIFO is the part of Input Data Sampler and Frame Handler. Bit 7-6: Not used. Bit 5: DDR read status (Full frame read complete). Bit 4: DDR write status (Full frame write complete). Bit 3: Not used. Bit 2: Intermediate data FIFO full status. Bit 1: Intermediate data FIFO empty status. Bit 0: Async FIFO (Slave FIFO) almost empty status.
11	MIPI Status	0x0A	RO	It gives the status of MIPI error. It is valid only if source type is MIPI.
12	HDMI/MIPI Source Info	0x0B	RW	This register provides information regarding source device. If source type is HDMI, it defines how many channels have been used and source connection status. If source type is MIPI, it gives information regarding ISP and image cropping algorithm should be enabled or not within FPGA. Bit 7-6: Not used. Bit 5: Enable Crop Algorithm (Valid for only MIPI source). Bit 4: Enable CIS ISP IP (Valid for only MIPI source). Bit 3: Not used. Bit 2: 0 for YUV and 1 for RGB. Bit 1: 0 for single channel and 1 for dual channel. Bit 0: Source connection status.

3 Implementation of LVDS transmitter in FPGA

Sl.#	Register	Register offset address	Type	Description
13	U3V Stream Mode	0x0C	RW	This register defines the U3V streaming mode. Bit 7-3: Number of frames for “multi frame” mode. Bit 2: Multi frame mode. Bit 1: Single frame mode. Bit 0: Continuous mode.
14	U3V Chunk Mode	0x0D	RW	This register provides information regarding U3V chunk mode. Bit 7-3: Not used. Bit 2: Chunk type is time stamp. Bit 1: Chunk type is Frame ID. Bit 0: Chunk mode enable.
15	U3V Trigger Info	0x0E	RW	It defines the U3V trigger-related information. Bit 7: Software trigger. Bit 6: Hardware trigger 3. Bit 5: Hardware trigger 2. Bit 4: Hardware trigger 1. Bit 3: Hardware trigger 0. Bit 2: Not used. Bit 1: Trigger Type: 0- Frame type, 1- Acquisition type. Bit 0: Enable trigger mode.
16	Device X Active Thread Info	0x0F	RW	This register defines number of active threads for a device. If its value is, 1 = Only single thread is assigned to device. 2 = Two threads are assigned to device. Maximum two threads can be mapped to device. Device X Thread 1 Info and Thread 2 Info registers inform that which thread is mapped. If two threads needed to be used, first thread value shall be defined in Device X Thread 1 Info register and second thread value shall be defined in Device X Thread 2 Info register. In case, only single thread needed to be used, value shall be defined in Device X Thread 1 Info register only.
17	Device X Thread 1 Info	0x10	RW	It defines the first thread value which is mapped to a device. Its value can be from ‘0’ to ‘3’ as EZ-USB™ FX20 controller supports maximum four threads. Thread 0 and 1 shall be used for independent Port 0 and thread 2 and 3 shall be used for independent Port 1. In wide link, all four can be used.

3 Implementation of LVDS transmitter in FPGA

Sl.#	Register	Register offset address	Type	Description
18	Device X Thread 2 Info	0x11	RW	It defines the second thread value which is mapped to a device. Its value is valid only if Device X Active Thread Info register has value '2'. Its value can be from '0' to '3' as EZ-USB™ FX20 controller supports maximum four threads. Thread 0 and 1 shall be used for independent Port 0 and thread 2 and 3 shall be used for independent Port 1. In wide link, all four can be used.
19	Device X Thread 1 Socket Info	0x12	RW	It defines the socket number assigned to thread defined in Device X Thread 1 Info register. Each thread supports a maximum of eight sockets. Therefore, its value can be from '0' to '7'.
20	Device X Thread 2 Socket Info	0x13	RW	It defines the socket number assigned to thread defined in Device X Thread 2 Info register only if Device X Active Thread Info register has value '2'. Each thread supports a maximum of eight sockets. Therefore, its value can be from '0' to '7'.
21	Device X Flag Info	0x14	RW	This is a control register to enable the flags. The EZ-USB™ FX20 controller supports maximum four flags per thread but here, three are used and it is common for both the threads. Whichever bit is high, that flag is active for device. Bit 7-4: Not used. Bit 3: First frame start frame. Bit 2: Not used. Bit 1: New packet start flag. Bit 0: New frame start flag.
22	Device X Counter and CRC Info	0x15	RW	Defines the event counters, payload counters, Data CRC and Metadata CRC for device. There are 2 event counters, 1 payload counter, Data CRC and Metadata CRC available for each thread but here, it is assumed it is same for all threads. Bit 7-5: Not used Bit 4: Metadata CRC. Not supported. Bit 3: Data CRC. Not supported. Bit 2: Enable Payload Counter Bit 1: Not used Bit 0: Enable Frame Count
23	Device X Buffer Size Info LB	0x16	RW	Device X buffer (packet) size. It is same for all threads mapped to this Device. Its maximum value can be up to 65535 bytes.
24	Device X Buffer Size Info UB	0x17	RW	

3 Implementation of LVDS transmitter in FPGA

3.3 Functional description of LVDS transmitter (LVDS TX)

This section describes how the LVDS transmitter block works in detail. It incorporates multiple Finite State Machines (FSM) to facilitate the training and the transmission of frame data to the EZ-USB™ FX20 controller. These are described in more detail in subsequent sections. The LVDS TX is designed to handle up to eight devices.

3.3.1 LVDS transmitter block diagram and features

Figure 10 shows the high-level block diagram of the LVDS transmitter. The main role of LVDS TX is to train the LVDS RX upon power-up or any low power mode exit and transmit the frame data to the EZ-USB™ FX20 controller.

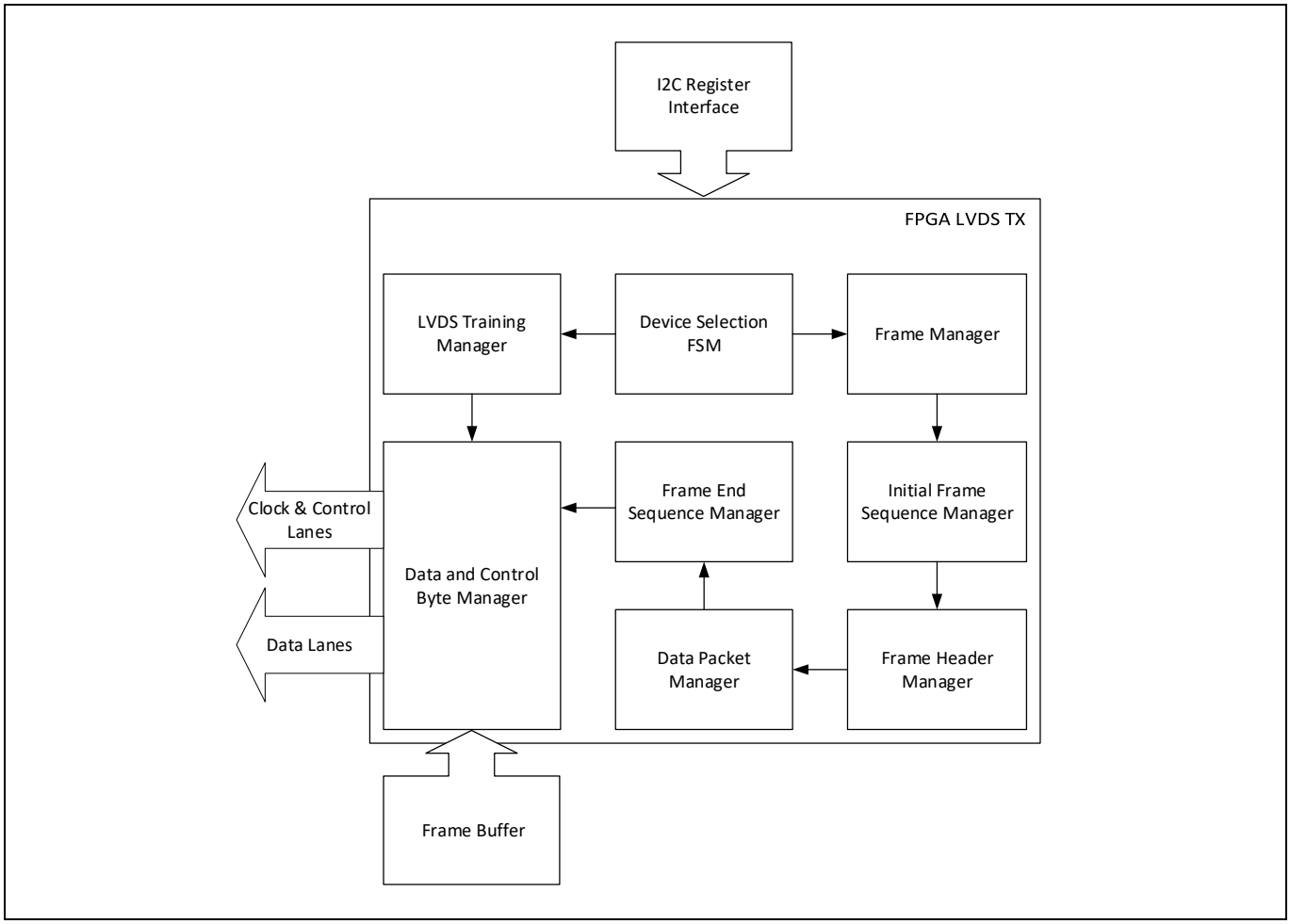


Figure 10 LVDS transmitter

It uses the configuration data stored in the I2C registers. It also accesses the frame buffer where frame data is stored for each device source connected with the FPGA. It serves each device in a round-robin manner. In the end, it provides the data on the data lane and control information on the control lane according to the selected gear ratio and mode.

3 Implementation of LVDS transmitter in FPGA

3.3.1.1 Features of LVDS transmitter

- Supports a maximum of eight devices, which is the video source connected with the FPGA or internal test pattern generator.
- Supports LVDS clock range from 74.25 MHz to 156.25 MHz.
- Supports independent narrow link (Port 0 or Port 1) and wide link modes.
- Supports 4:1 and 8:1 gear ratio.
- Supports header data insertion along with payload and as metadata for UVC class.
- Supports leader and trailer data insertion for USB3Vision (U3V) class.
- Supports thread interleaving.
- Supports Precision Time Management (PTM).

3.3.2 Device selection FSM

Figure 12 shows the device selection FSM, and it is the main FSM. Upon power-up or logical reset, it will be in the IDLE state. Once the EZ-USB™ FX20 controller is up and FPGA is configured, the EZ-USB™ FX20 controller (I2C master) configures the registers via I2C and asserts LINK_RDY control signal to inform FPGA to start training.

The Device selection FSM checks the PHY/Link Training Control and Status register to determine the type of training required for the EZ-USB™ FX20 controller and instructs the PHY/Link Training FSM to start the training and waits until its successful completion.

After training, the device selection FSM starts serving the devices in a round-robin manner as shown in Figure 11. The LVDS TX supports a maximum of eight devices.

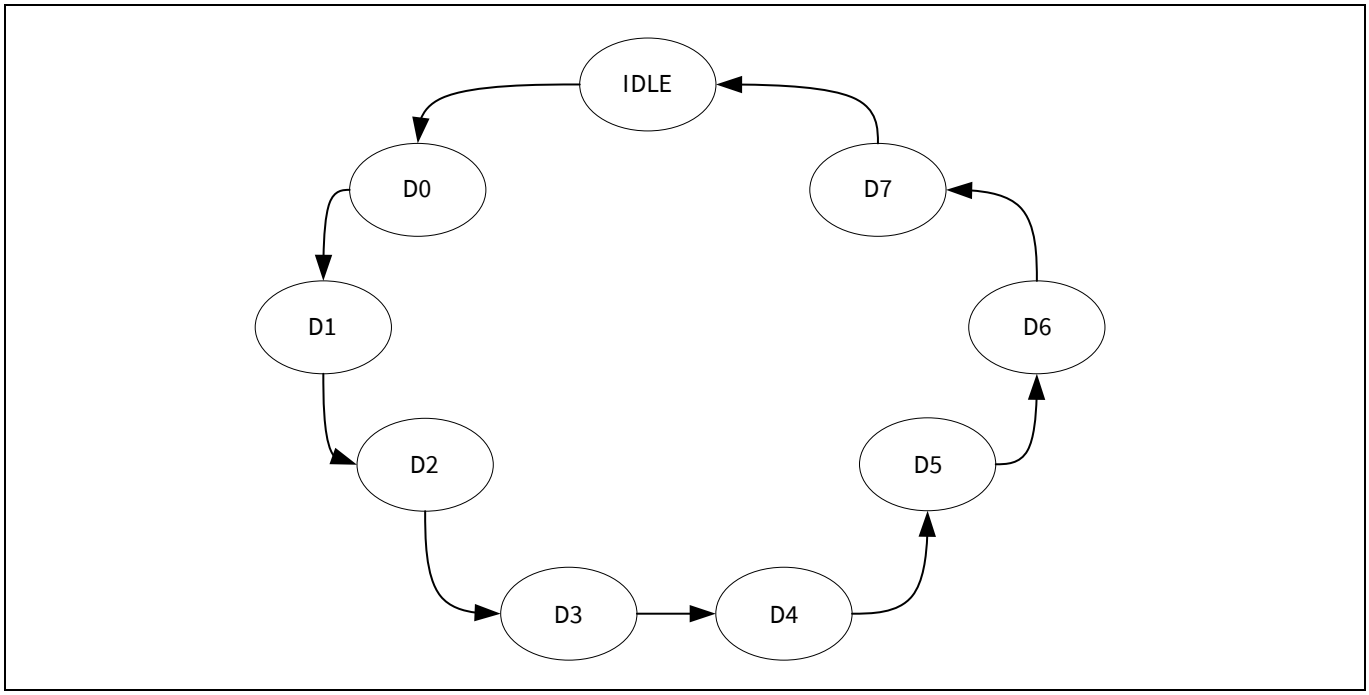


Figure 11 Device selection FSM serving multiple devices

3 Implementation of LVDS transmitter in FPGA

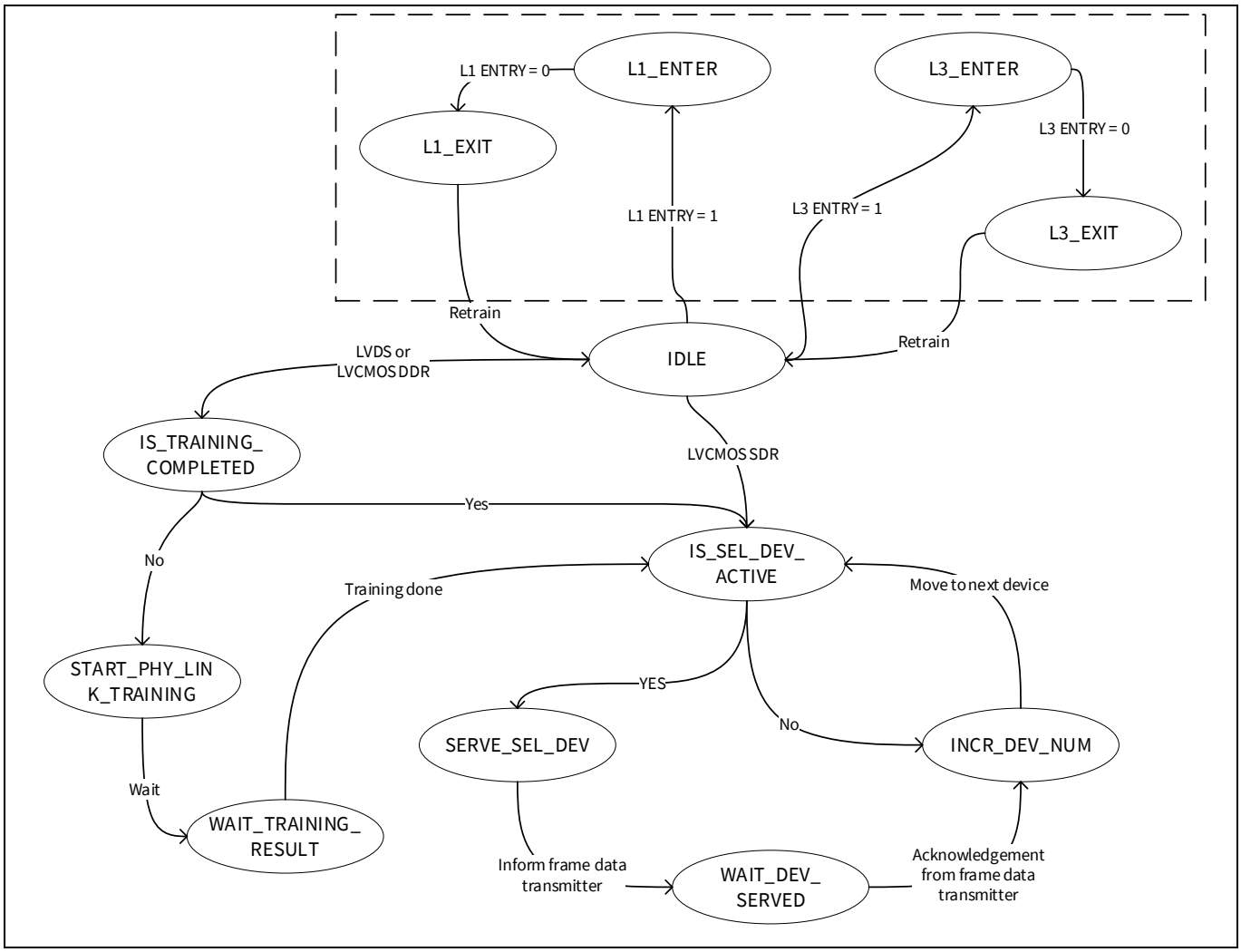


Figure 12 Device selection FSM

If the device is a video source, it can be either an image sensor (such as the THine camera module with built-in ISP or the Sony IMX715 image sensor board) connected to the FPGA, or an internal test pattern (color bar) generated within the FPGA.

The Device Selection FSM starts with the first device. If the device has data to send, it instructs the Frame Data Transmitter FSM to transmit a frame and waits for an acknowledgement. Once acknowledgement is received, the Device Selection FSM moves on to the next device and follows the same steps until all active devices (up to a maximum of 8) have been served. If any device does not have data to send, it simply moves on to the next device.

The Device Selection FSM moves to low power mode upon receiving an L1 Entry or L3 Entry signal from the EZ-USB™ FX20 controller. In response, it moves to the L1_ENTER or L3_ENTER state and instructs the LVDS IP to power down the LVDS lanes according to the mode. The Device Selection FSM moves to the L1_EXIT or L3_EXIT state when the L1 or L3 Entry signal is de-asserted by the EZ-USB™ FX20 controller respectively via I2C. Upon exit from L1 or L3 mode, PHY and Link training is mandatory.

3 Implementation of LVDS transmitter in FPGA

3.3.3 PHY and LINK training FSM

This FSM sends the training pattern required to train the LVDS RX PHY and LINK of the EZ-USB™ FX20 controller. The PHY and LINK training is required for the LVDS receiver (RX) as soon as the EZ-USB™ FX20 controller is powered up or LVDS low power modes (L1 or L3) exit. Upon power-up, the EZ-USB™ FX20 controller initializes its PHY and asserts LINK_RDY. Once LINK_RDY is asserted, the PHY and LINK training FSM transmits PHY Training Sequence (PHY_TS) byte for minimum TS_Time (50 µs) followed by LINK Training Block consisting of a 4-byte sequence (P0, P1, P2 and P3) on all the data and control lanes. The LINK training is used to compensate skew on all lanes and align data bytes of all lanes.

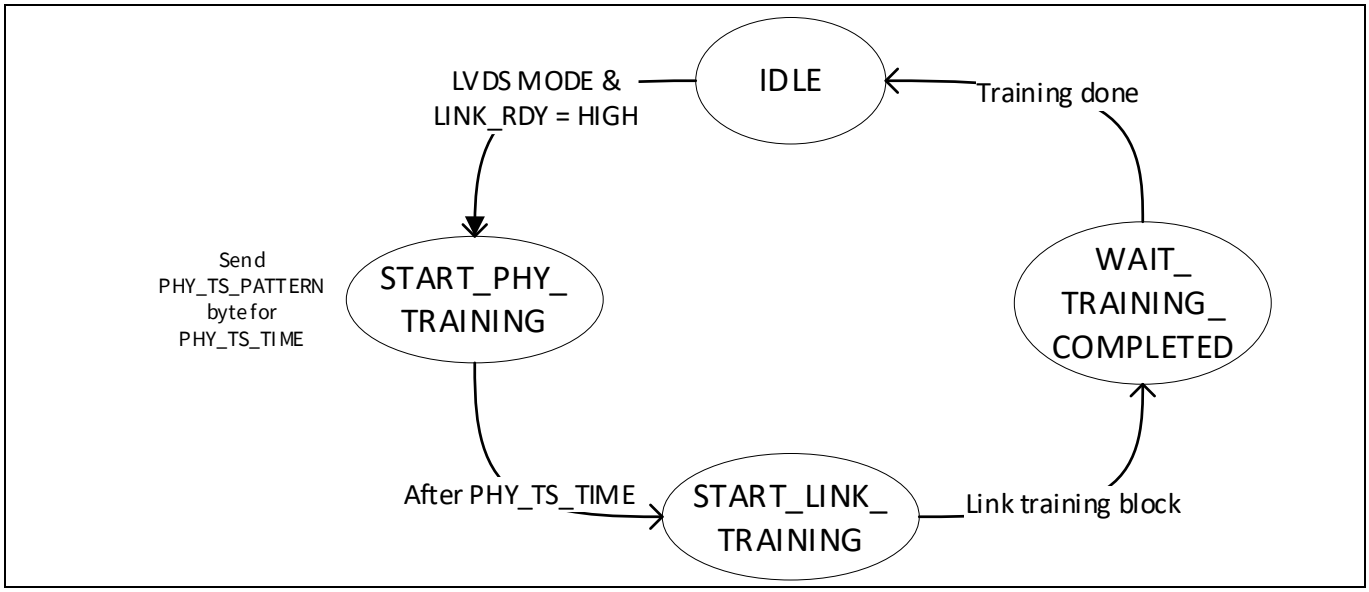


Figure 1 LVDS PHY and LINK training FSM

The FPGA receives training requirement information, including the PHY Training Sequence byte (1 byte) and LINK Training Block sequence (P0 to P3), via I2C. For more details on the FPGA I2C slave register, refer to the Section 3.2.1.

Upon receiving the acknowledgement from the EZ-USB™ FX20 controller, it acknowledges to the main FSM that training is completed.

3 Implementation of LVDS transmitter in FPGA

3.3.4 Frame data transmitter FSM

Frame data transmitter FSM is responsible for frame data transmission to the EZ-USB™ FX20 controller. Once PHY and LINK training is completed, and the selected device has data ready to transmit, the Device Selection FSM instructs the frame data transmitter FSM to transmit a frame.

The frame data transmitter FSM utilizes sub-FSMs to manage specific tasks, including the transmission of initial socket/thread information, frame header information, data, and frame end information. Figure 13 shows all the states.

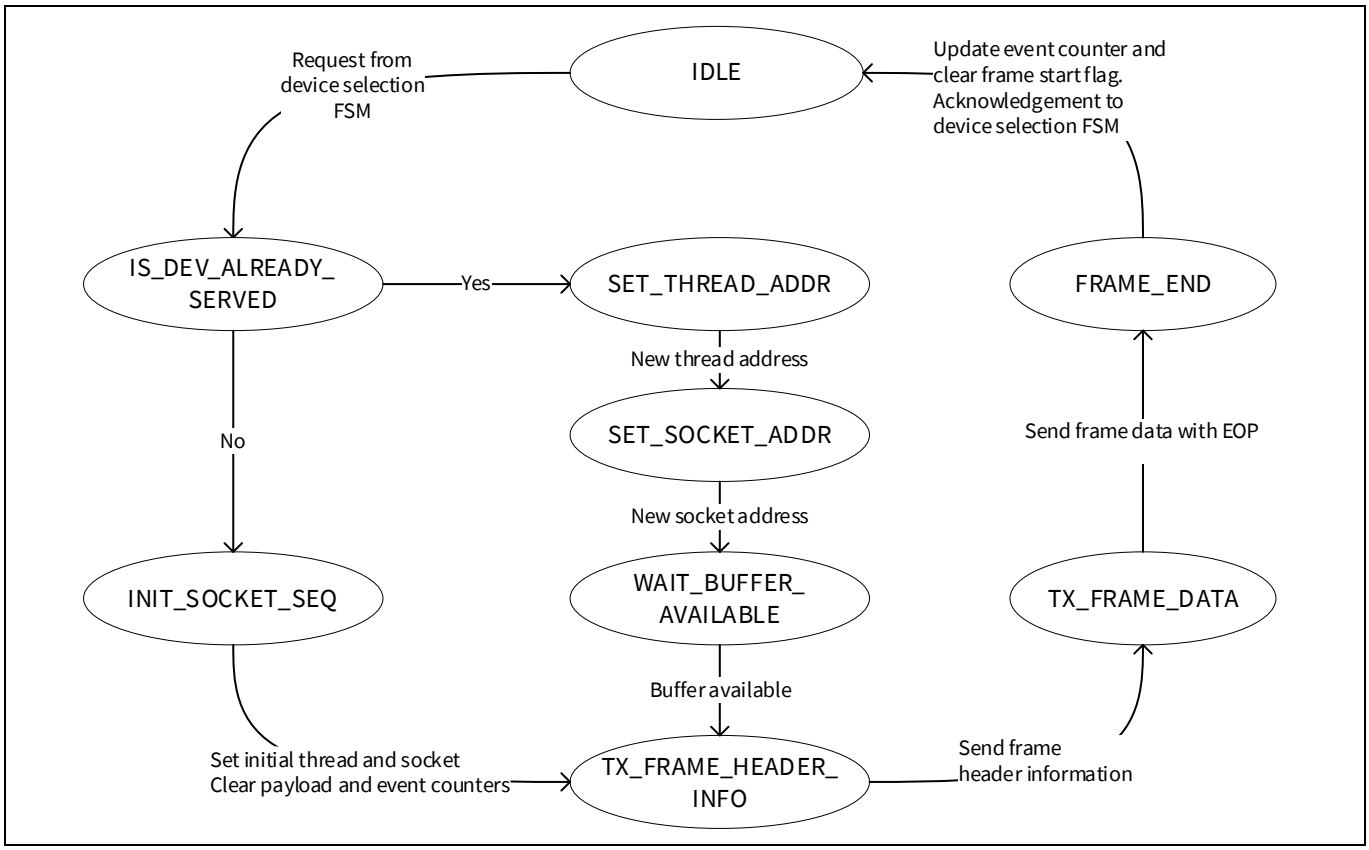


Figure 13 Frame data transmitter FSM

3 Implementation of LVDS transmitter in FPGA

3.3.4.1 Initial socket sequence FSM

The initial socket sequence FSM handles the initialization sequence. It assigns the initial thread and socket address. Additionally, it initializes the payload and event counters if they are activated through FPGA I2C slave registers. Besides these tasks, it also sets the lower byte of the metadata variable word with the length of the frame header.

At the end, it acknowledges the frame data transmitter FSM indicating that the initial thread and socket information has been sent. Figure 14 shows all the states of the initial socket sequence FSM.

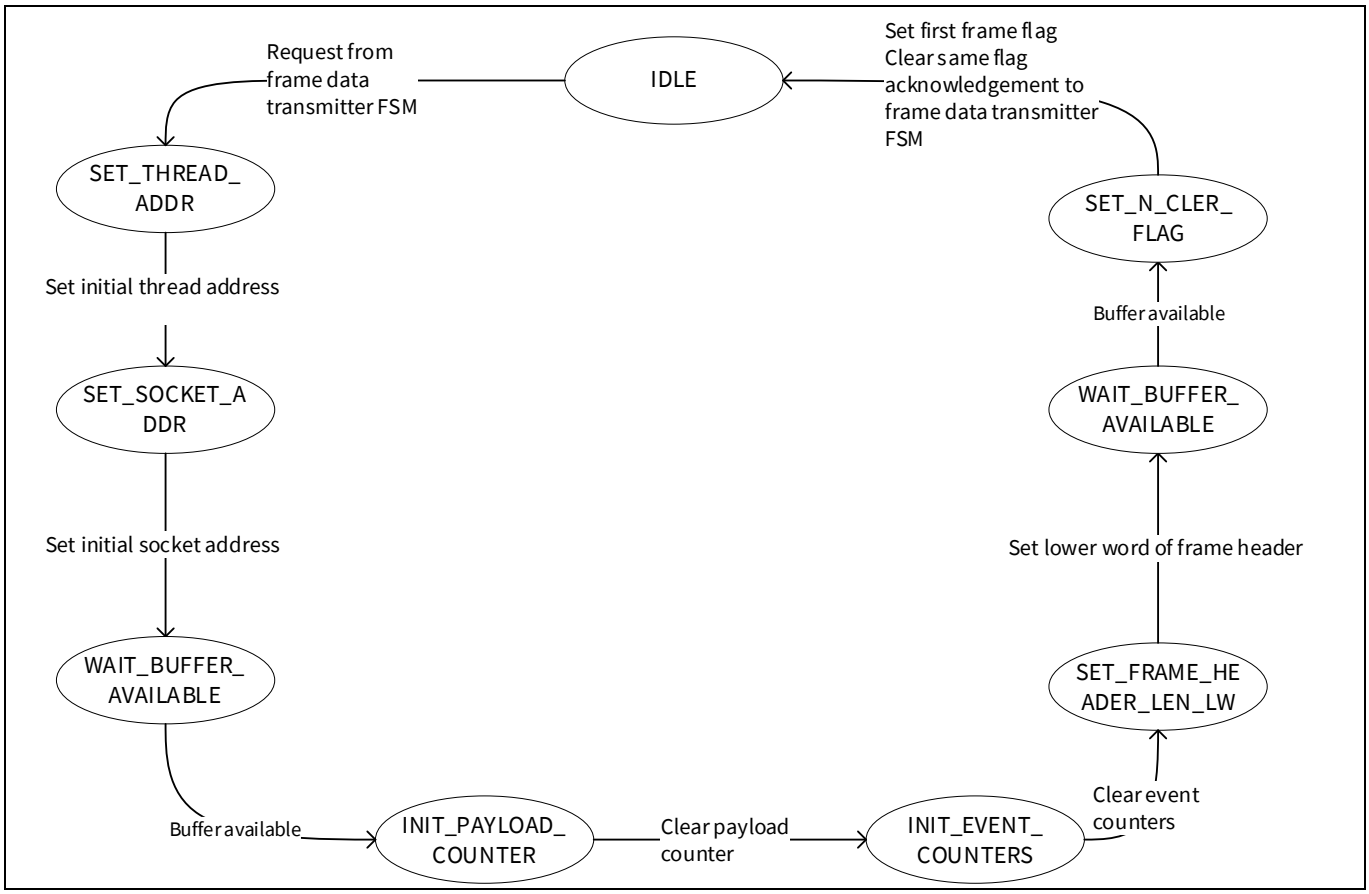


Figure 14 Initial socket sequence FSM

3.3.4.2 Frame header information transmitter FSM

The frame header information transmitter FSM is responsible for transmitting information related to the UVC frame header. It sends an IDLE control byte until the source device (THine camera module with built-in ISP or the Sony IMX715 image sensor board or internal test/color bar pattern generator) begins capturing data. As soon as the selected device begins capturing data, the PTS control byte is sent, followed by IDLE control bytes until the camera device starts sending data to the FPGA.

Once the FPGA receives data, the Frame Header Information Transmitter FSM transmits the SCR control byte. These control bytes are used to manage Precision Time Stamp (PTS) values when multiple devices are transmitting data.

Once a buffer is available on the EZ-USB™ FX20 controller side, it transmits the SET FLAG control byte to indicate that the first frame data will be transmitted soon if FLAGs are enabled through the FPGA I2C slave register.

At the end, it acknowledges the Frame Data Transmitter FSM, confirming that the header information has been sent.

Each GPIF thread has four flags that are visible to GPIF and CPU. A maskable interrupt will be raised to CPU, whenever these flags are set or cleared by FPGA. No counters are associated with flags, for detailed information on the control byte, refer to the “LVDS control byte encoding” section from the architecture reference manual [1].

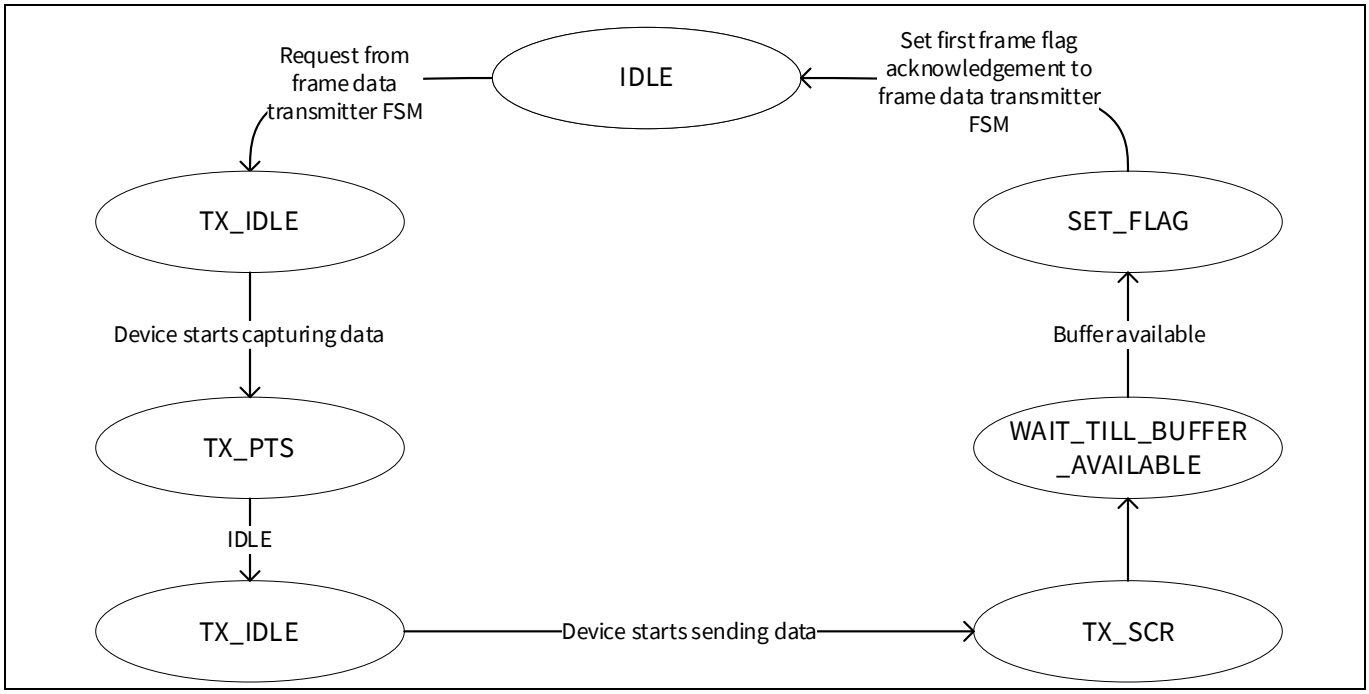


Figure 15 Frame header transmitter FSM

3 Implementation of LVDS transmitter in FPGA

3.3.4.3 LVDS data packet transmitter FSM

The data packet transmitter FSM manages the transmission of data packets of any class like UVC, U3V, or vendor-specific along with the header insertion.

It sends the packet information before each data packet if the FPGA is required to transmit packet details. Additionally, it transmits a control byte to enable the payload counter and flags to inform the EZ-USB™ FX20 controller that a new packet is coming, provided the corresponding bits are set in the FPGA I2C register. It also manages to send data packets and control bytes, such as clearing flags to indicate the completion of the current data packet and hold payload count if respective bits are configured in the FPGA I2C register.

After the last data packet, the data packet transmitter FSM transmits an EOP (End of Packet) control byte to commit the buffer to the EZ-USB™ FX20 controller. Subsequently, it also switches to a new thread and socket address if it is configured with a multithread.

Moreover, In U3V, first, it sends the leader and trailer packets as well at the start and end of the U3V frame. Leader and trailer packet is nothing but the ZLP (Zero Length Packet) to indicate the EZ-USB™ FX20 controller to add header and footer data in a frame.

At the end, it informs the frame data transmitter FSM that full frame data has been sent for the selected device.

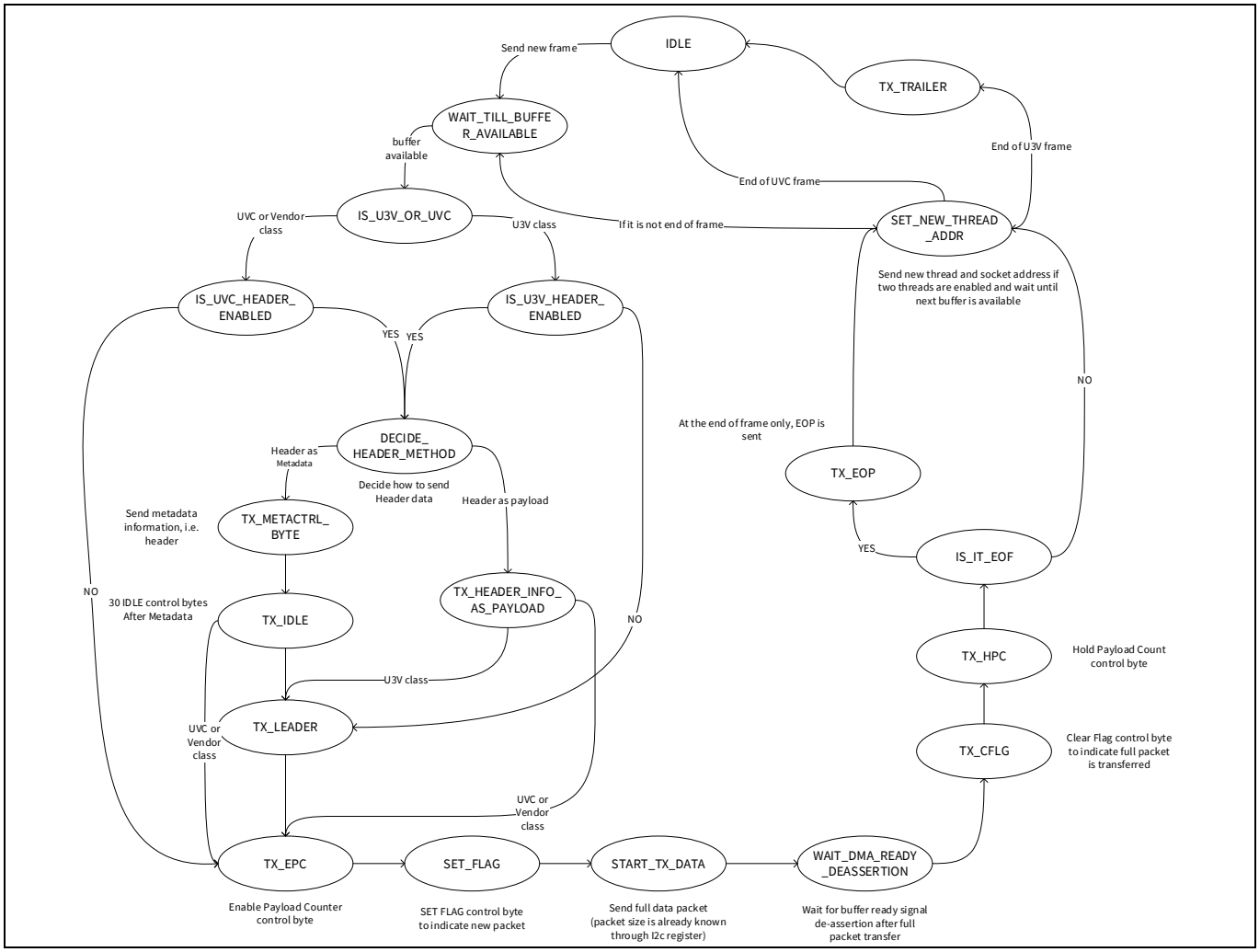


Figure 16 Data packet transmitter FSM

3 Implementation of LVDS transmitter in FPGA

3.3.4.4 Frame end transmitter finite state machine (FSM)

The frame end transmitter FSM is responsible for sending a sequence of control bytes to indicate the end of a frame (EOP). It first transmits a control byte to clear a flag (CFLAG), which indicated that all bytes in a frame have been transferred. Subsequently, it updates the event counter to count the current frame. This update occurs if the respective bits are set in the FPGA I2C slave register by the EZ-USB™ FX20 controller.

At the end, it informs the frame data transmitter that the current frame is completed. Upon receiving acknowledgment, the frame data transmitter informs the device selection FSM that the selected device is served. Therefore, it moves to the next activated device. As a result, it serves all the activated devices one by one in a round-robin manner.

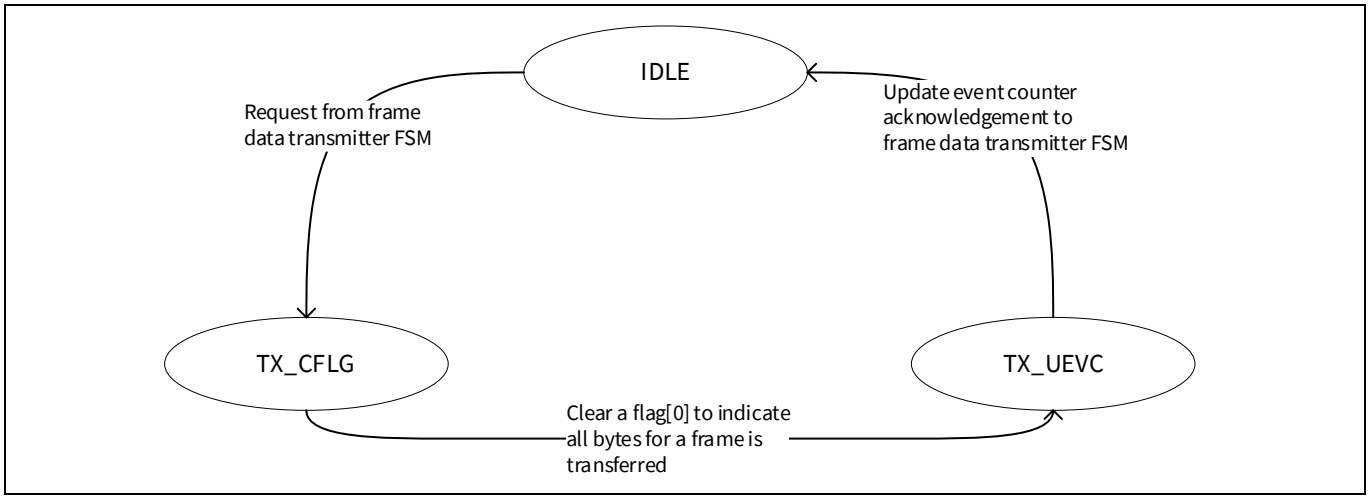


Figure 17 Frame end transmitter FSM

4 Reference FPGA project

Contact Infineon's technical support team to receive the reference design/project to stream video from four different MIPI-based camera sensors. The reference design supports the wide link (16 data lanes) and 8:1 gear ratio at 148.5 MHz.

The reference design package includes the Efinix Ti180 FPGA project including RTL files, ready to evaluate demonstration, FW HEX and document to understand the procedure to run the demonstration on the hardware kit.

Note: *The provided reference FPGA project is validated for a maximum of four video devices. It does not support data and header CRC insertion along with the low power mode as of now.*

5 Troubleshooting LVDS interface

This section briefs a few checkpoints about the LVDS interface between the FPGA and the EZ-USB™ FX20 controller. It must be noted that the following configuration parameters must be pre-defined between the FPGA and the EZ-USB™ FX20 controller.

- LVDS interface clock frequency
- Port configuration, like narrow link (Port 0 or 1) or wide link
- Gear ratio
- I2C slave address

Moreover, the following registers must be configured by the EZ-USB™ FX20 controller before LINK_RDY assertion.

- PHY Training Sequence (PHY_TS) byte
- LINK Training Block sequence (P0 to P3)

Please make sure that only the recommended values are used for PHY Training Sequence byte (LVDS PHY Training Mode Config Register) and LVDS LINK Training Block Sequence (P0 to P3).

5.1 PHY training failure

As mentioned in section 2.2, the FPGA starts with training and followed by data transfer. Therefore, for error free data transfer between the FPGA and the EZ-USB™ FX20 controller, PHY and LINK training must be completed successfully. After sending PHY training bytes and LINK training pattern, the FPGA waits for the acknowledgement from the EZ-USB™ FX20 controller.

For the EZ-USB™ FX20 controller, training is successful only if data bytes received during PHY and LINK training must be same as it has written to LVDS PHY Training Mode Config (PHY_TS) and LVDS LINK Training Block Sequence (P0 to P3) respectively. Only when data is matched, it informs the FPGA by updating its PHY/Link Training Control and Status register. The FPGA moves forward for data transfer only if it receives the acknowledgement from the EZ-USB™ FX20 controller else it waits.

In Figure 18, FW debug log shows the status of the PHY training along with the byte used for PHY training (PHY_TS). The reference design provided with this application note includes the document explaining how to take the debug log.

The possible check points for the unsuccessful PHY training:

- The FPGA and the EZ-USB™ FX20 controller must be using the same value of the pre-defined parameters like LVDS clock frequency, gear ratio, Port configuration and I2C slave address.
- The EZ-USB™ FX20 controller must use one of the recommended values for the LVDS PHY Training Mode Config register. Check the description of LVDS PHY Training Mode Config register to find the different recommended values.
- If wide link mode is used, the FPGA must send the PHY training sequence byte on both the control lanes during PHY training.

5 Troubleshooting LVDS interface

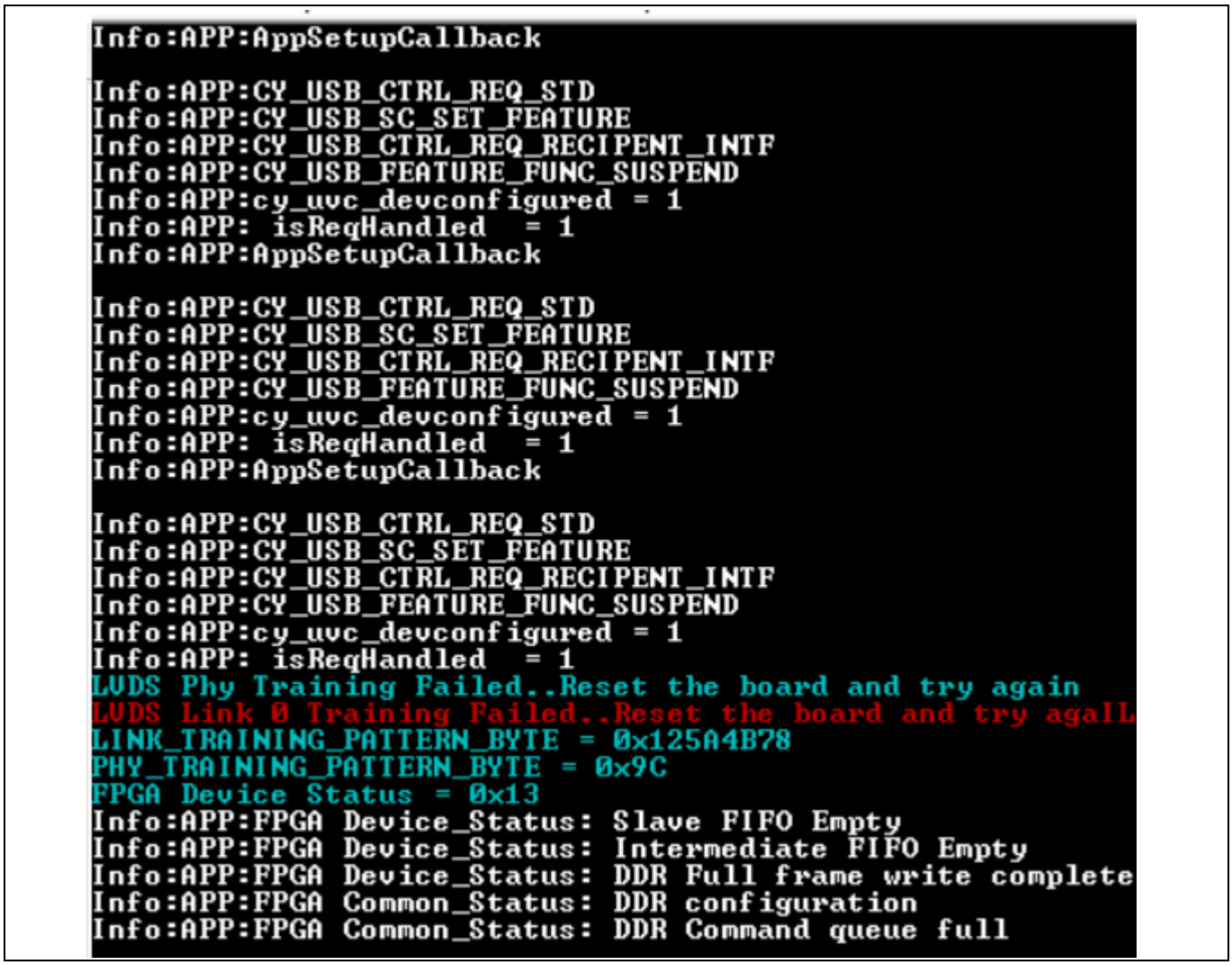


Figure 18 PHY training log

5.2 LINK training failure

FW debug log in Figure 19 shows the LINK training status. Along with the overall status, it can also inform the individual lane status for both the ports. As mentioned in the debug log, the lower 8 bits represent the status for data lanes (bit 0 for data lane 0 and bit 7 for data lane 7) and bit 8 represents the status of control lane.

The value 0x1F0 represents that data lanes 0 to 3 are failed on both the ports.

5 Troubleshooting LVDS interface

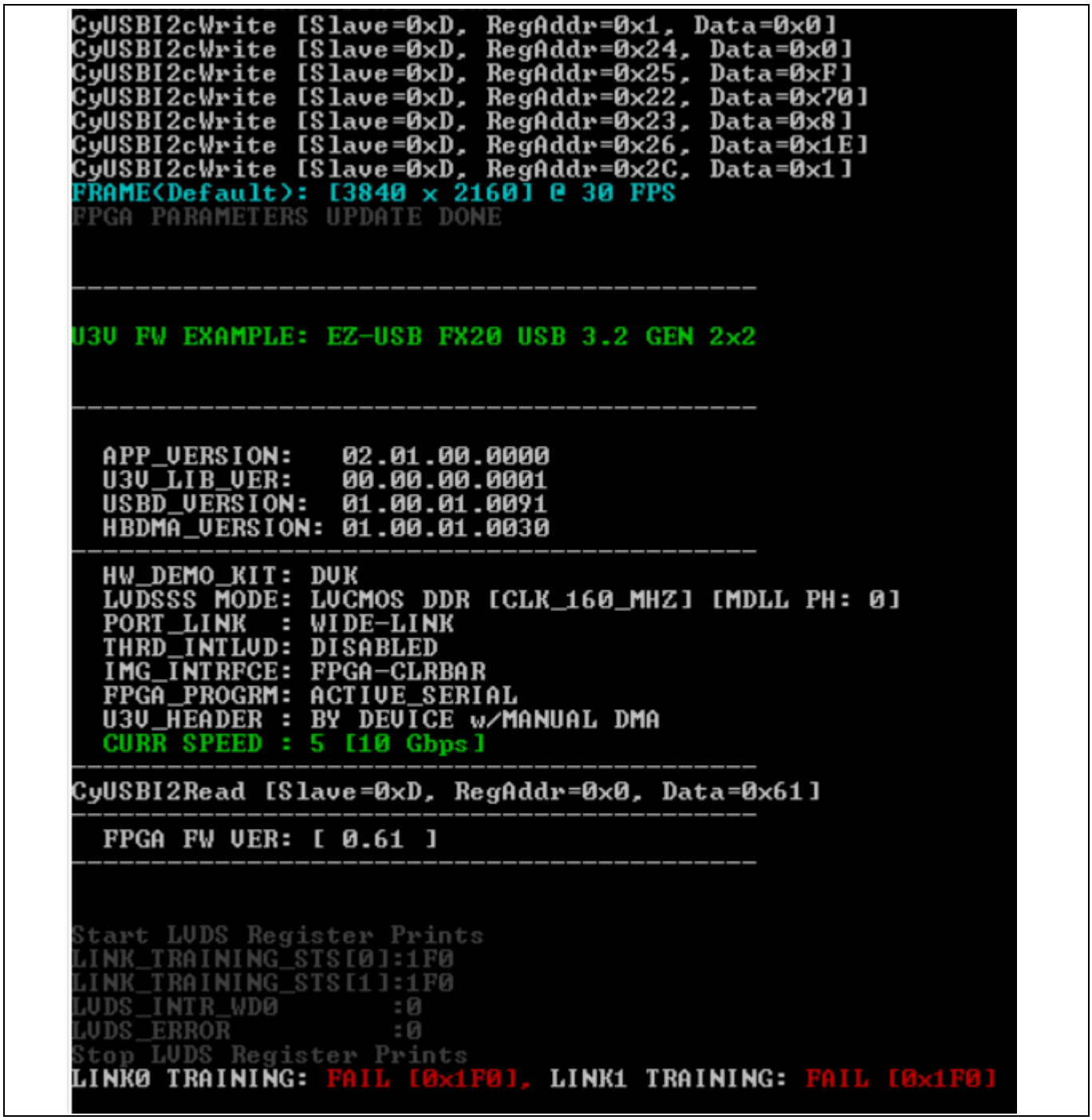


Figure 19 LINK training log

The possible check points for the unsuccessful LINK training:

- The FPGA and the EZ-USB™ FX20 controller must be using the same value of the pre-defined parameters like LVDS clock frequency, gear ratio, Port configuration and I2C slave address.
- The EZ-USB™ FX20 controller must use one of the recommended values for the LVDS Training Block Sequence (P0 to P3) registers. Check the description of LVDS Training Block Sequence (P0 to P3) registers to find the recommended values.

5 Troubleshooting LVDS interface

- If wide link mode is used, the FPGA must send the PHY training sequence byte on both the control lanes during PHY training.
- Increasing skew between the clock and data lane in FPGA may help the EZ-USB™ FX20 controller to sample data correctly.

Note: *In the reference project provided with this application note, a delay of 500 ps is added between the clock lane and data/control lanes, allowing the data and control lanes to lead the clock lane.*

5.3 Incorrect sampling of buffer ready (BUFFER_RDY)

Once the FPGA sends the thread and socket address information through the control lane, it must wait for a minimum 100 LVDS clock cycles before sampling the buffer ready (BUFFER_RDY) control signal from the EZ-USB™ FX20 controller. After 100 LVDS clock cycles, if the buffer ready (BUFFER_RDY) control signal is asserted, the FPGA can move ahead with the data transfer else wait till it gets asserted. Figure 4 explains it with detailed LVDS transaction flow.

5.4 Slow or sluggish video data streaming on USB host

The video data received by the EZ-USB™ FX20 controller from the FPGA is sent to USB Host. If the USB Host is busy or occupied with other tasks, it affects the throughput. As a result, the output video frame rate on a USB Host drops and video streaming becomes slower than the original input.

The possible check points for the slow video data streaming:

- The USB Host must not be occupied with too many tasks which can affect the throughput between the USB Host and the EZ-USB™ FX20 controller.
- The EZ-USB™ FX20 controller must be enumerated at the correct speed. FW debug log shows that speed at which the EZ-USB™ FX20 controller is enumerated. In Figure 19, the EZ-USB™ FX20 controller is enumerated in USB3.2 Gen2 (10 Gbps) mode as speed is mentioned as 5. The Table 4 shows the speed macro value used in FW to define the enumeration speed of the EZ-USB™ FX20 controller.

  For example, the LVDS interface between the FPGA and the EZ-USB™ FX20 controller runs at 156.25 MHz and 8:1 gear ratio in a wide link mode, it means, the data rate the FPGA and the EZ-USB™ FX20 controller is 20 Gbps. On the other hand, if the EZ-USB™ FX20 controller is connected to USB3.2 Gen2 Host, the maximum data rate is 10 Gbps. In this scenario, the video data rate at the USB Host side may drop.

Table 4 EZ-USB™ FX20 controller enumeration speed information

Sl.#	FW speed #	EZ-USB™ FX20 controller enumeration speed
1	0	Device is not connected
2	1	Full-Speed
3	2	High-Speed
4	3	USB3.2 Gen1 (5 Gbps)
5	4	USB3.2 Gen1x2 (10 Gbps)
6	5	USB3.2 Gen2 (10 Gbps)
7	6	USB3.2 Gen2x2 (20 Gbps)

5 Troubleshooting LVDS interface

5.5 Choppy video data streaming on USB host

As mentioned in section 3.1, the Input Data Sampler samples the data and stores it into an intermediate buffer or FIFO before it is transferred to DDR frame buffer by the Frame Handler block. When the EZ-USB™ FX20 controller is ready to receive the data, it asserts the buffer ready (BUFFER_RDY) and LVDS TX block starts transmitting data from the DDR frame buffer to the EZ-USB™ FX20 controller.

In the FPGA, if the frame buffer is not used to store the entire frame data, there is a possibility that when LVDS TX block is waiting for the buffer ready (BUFFER_RDY) control signal to get asserted and intermediate FIFO gets overflow because of continuous incoming data. As a result, intermediate frame data is lost before it is sent to the EZ-USB™ FX20 controller and eventually to the USB Host. It results in choppy video streaming.

The frame buffer is not required in the design if the throughput requirement is relatively lower than what the EZ-USB™ FX20 controller offers. Otherwise, it is highly recommended to use it in a design.

The Status Info register shows the status of the Intermediate as well as the Asynchronous (Asyc) FIFOs. The EZ-USB™ FX20 controller can read this register to get the information.

References

[1] Infineon Technologies AG: *EZ-USB™ FX20 (USB 20 Gbps) device controller architecture reference manual;* Available online

Revision history

Document revision	Date	Description of changes
**	2025-05-09	Initial release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2025-05-09

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2025 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-41535 Rev. **

Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.