Reliable Functionality Verification System for MIPI CSI-2 (Camera Serial Interface)

M.Tech. Thesis

By

RUCHIKA GAUTAM

DISCIPLINE OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2020

Reliable Functionality Verification System for MIPI CSI-2 (Camera Serial Interface)

A THESIS

Submitted in partial fulfillment of the
requirements for the award of the degree
of
Master of Technology
in
Electrical Engineering
with specialization in
VLSI Design and Nanoelectronics

by

RUCHIKA GAUTAM

DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
MAY 2020

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Reliable Functionality Verification System for MIPI CSI-2** in the partial fulfillment of the requirements for the award of the degree of **MASTER OF TECHNOLOGY** and submitted in the **DISCIPLINE OF ELECTRICAL ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from July 2018 to May 2020 under the supervision of Dr. Abhinoy Kumar Singh, Inspire Faculty, Discipline of Electrical Engineering, IIT Indore and Dr. Saptarshi Ghosh, Assistant Professor, Discipline of Electrical Engineering, IIT Indore and under guidance of Mr. Manoj Manu, Senior Manager, Mentor Graphics, Noida.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Ruchika

19/06/2020

Signature of the student with date
RUCHIKA GAUTAM

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

22/06/2020

Dr. Abhinoy Kumar Singh

Saptarshi Ghosh 22/06/2020

Dr. Saptarshi Ghosh

RUCHIKA GAUTAM has successfully given his/her M.Tech. Oral Examination held on **June 2020**.

(Abhinoy kr. Singh)

Saptarshi Ghosh

Signature(s) of Supervisor(s) of M.Tech. thesis
Date: 22/06/2020 22/06/2020

Convener, DPGC
Date: 22-06-2020
S. Mukherjee 22/6/2020

Somnath Dey

Signature of PSPC Member #1 22/06/2020
Date:

Signature of PSPC Member #2
Date:

ACKNOWLEDGEMENT

I want to take this opportunity to express my deepest gratitude to my research supervisor **Dr. Abhinoy Kumar Singh**, Inspire Faculty at IIT Indore and **Dr. Saptarshi Ghosh**, Assistant Professor at IIT Indore. I am indebted to them for giving their valuable time to guide me in all the way technically as well as non-technically. I am very much thankful to him for motivating me in my distressed conditions. Finally, encouragement and guidance helped me to shape my future and the completion of my M. Tech.

I would like to thank Mr. **Manoj Manu** for providing this opportunity to work in Mentor Graphics which helped me to work on real life problems and enhance my learning. He convincingly guided and encouraged me in the project as well as helped me to gain a big leap from campus to corporate. His pieces of advice kept me going on in this work, and this would not have been possible without his inputs and counseling. I would also like to thank my friends and batchmates **Abhishek**, **Sagar,** and **Sheeba** for the idea's discussion, support, and needful comments on research work.

I sincerely acknowledge the support of the **IIT Indore community** to provide resources, lab equipment, and facilities for my research work. Also, I would like to thank **Ministry of Human Resource and Development** (MHRD) for providing TA Scholarships.

Last but not the least, my work would not have been possible without the encouragement of my **parents** and my brother, **Bhuvesh**, whose tremendous support helped me stay positive and overcome the worst of hurdles.

DEDICATION

Dedicated to my parents

ABSTRACT

MIPI CSI-2 became popular in the mobile industry for interfacing camera and other host devices, e.g. mobile processor because of its ease of use, high performance, and its ability to support high-resolution photographic images and videos. However, with the increasing requirement of the amount of data stored in one image and thus resulting in high power and the need of transmitting a huge amount of data, it became important to have a fresh perspective. So, we implemented some new techniques, i.e. introducing C-PHY as an alternative PHY layer, using LRTE EPD for packet spacing, and introducing SROI to improve data rate, power, and reducing the amount of data. As a result, replacement of D-PHY with C-PHY layer improves data rate and power, using LRTE EPD for packet spacing reduces latency and improves transmission efficiency and SROI provides a method to transmit only regions of interest thus reducing the amount of data transferred, which results in high frame rate and low power dissipation.

However, implementing new features in a protocol requires tracking of human errors and bugs introduced in the system, which may result in unpredictable behavior of design which is done through implementing tests in a verification system. Manual tracking of failure or success of tests is difficult and is not very accurate, and hence it became imperative to design an automatic system for keeping track of bugs and total features tested. So, a new verification architecture is proposed, which contains a scoreboard and coverage class, improving the accuracy and efficiency of design.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

MIPI: Mobile Industry Processor Interface

CSI: Camera Serial Interface

CCI: Camera Control Interface

SOT: Start Of Transmission

EOT: End Of Transmission

LLP: Low level Protocol

HS: High Speed

LP: Low Power

TX: Transmitter

RX: Receiver

CIL: Control and Interface Logic

ECC: Error Correction Code

CRC: Checksum

VC: Virtual Channel

VCX: Virtual Channel Extension

PH: Packet Header

PF: Packet Footer

WC: Word Count

DI: Data Identifier

LPS: Low Power State

FS: Frame Start

FE: Frame End

LRTE: Latency Reduction and Transport Efficiency

SROI: Smart Region Of Interest

SEDP: SROI Embedded Data Packet

DUT: Design Under Test

TLM: Transaction Level Modeling

UVM: Universal Verification Methodology

Chapter 1

1. Introduction

MIPI CSI-2 is an interface used for transmission of video or images between the camera and host devices. Thus, CSI-2 specifies a standard for control interfaces and data transmission between camera/sensor acting as transmitter and processor as the receiver. Two interface options are defined for high-speed serial data transmission. “D PHY physical layer”, the first option, is a differential and unidirectional interface with one or more 2-wire data lanes and with one 2-wire clock lane. Figure 1.1 illustrates the connections for this option between a camera module as CSI-2 transmitter and the part of mobile phone engine as CSI-2 receiver. “C-PHY physical layer”, the second option, a differential and unidirectional interface, has one or more unidirectional 3-wire serial data Lanes, each lane having its own embedded clock. Figure 1.2 illustrates CSI 2 receiver and transmitter connections for this option. A bidirectional control interface called as CCI is used for both C-PHY and D-PHY physical layer options which are used for controlling the transmitter[1].

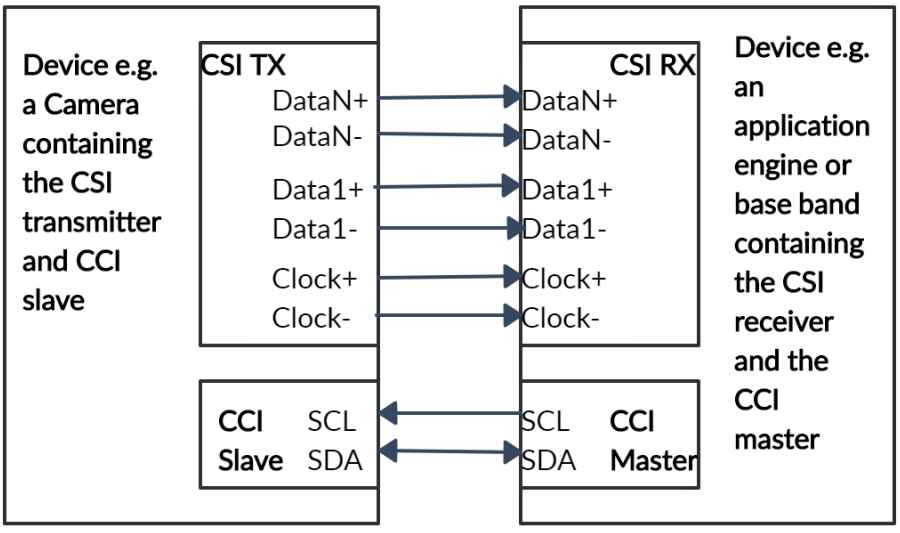


Figure 1.1 CSI 2 and CCI Transmitter and Receiver Interface for D-PHY

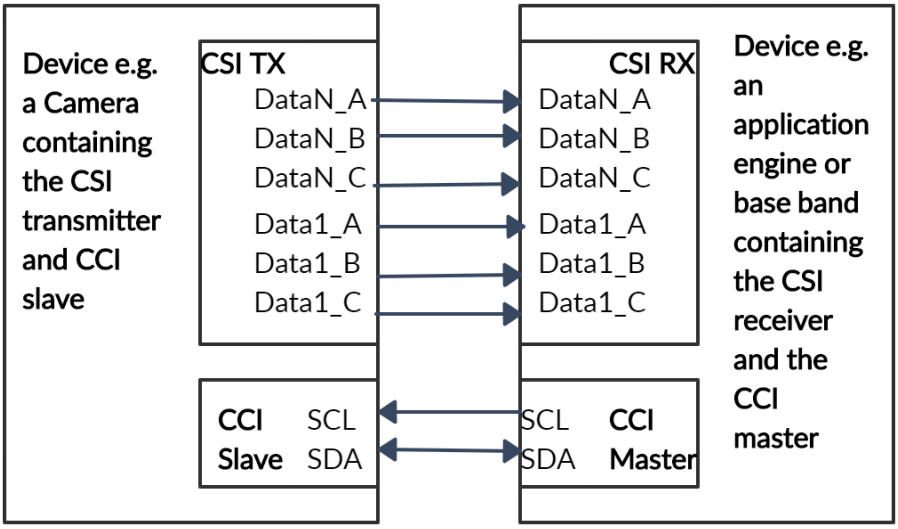


Figure 1.2 Typical CSI 2 and CCI Transmitter and Receiver Interface for C-PHY

1.1. Applications of MIPI CSI-2

CSI-2 is mainly used in interfacing of the camera and other host devices like the processor or mobile devices which can be used for the following applications:

- It can be used for implementing cameras (single or multi) in mobile devices.
- This interface can be used in VR (Virtual Reality) devices for connecting the camera in it.
- Smart cars can use this interface for various applications related to entertainment and collecting information or for safety control applications.
- It can also be used for camera in drones.
- It can also be used in several IOT applications and security applications using human detection and face recognition.

1.2. CSI-2 Layer Definitions

- PHY Layer: It specifies the output/input circuitry, transmission medium e.g. electrical conductors and clocking mechanism. It specifies the mechanism for EOT and SOT. Byte level and bit-level synchronization mechanisms are also included in the PHY layer.
- Protocol layer: Various layers having several and different responsibilities compose this layer. Multiple data streams can be transmitted on the host processor using a single interface. This layer defines the proper reconstruction of several streams on the receiver side when these streams are tagged and interleaved at the transmitter side.
    - Byte/Pixel Unpacking/ Packing Layer: CSI 2 supports varying image formats for different image applications. Pixels are packed in the transmitter by protocol layer from the application layer into bytes so that it can be sent to the Low-Level Protocol Layer in byte format. This layer unpacks bytes at the receiver end from the Low-Level protocol Layer into pixels so that it can be sent to the application layer in pixel format.
    - Low-Level Protocol: This layer specifies the method of establishing byte-level synchronization and bit-level synchronization between EOT and SOT for serial data. This layer also includes the assignment of bit values interpretation to byte.
    - Lane management: For increasing performance, CSI 2 is a lane scalable protocol. The number of data Lanes may vary and can be chosen depending on the requirements of application e.g. bandwidth requirement. The interface on the transmitter side distributes bytes to one or more lanes

from the data stream. The interface on the receiver side collects data from all Lanes and merges them together as a recombined data stream so that the original data stream sequence can be restored. For the C-PHY layer option, this layer collects or distributes byte pairs i.e. 16 bits from or to the data lanes.

- Application Layer: It defines the encoding and analysis of high-level data stored in the data stream [1].

1.3. Verification Architecture

Before any design or chip undergoes fabrication, verification of correctness and accurate functionality of the system is mandatory for its proper functionality. Since verification takes around 60-70% of the total time required for completion of a design, a fast and reliable verification system plays a very important role.[2] Thus a UVM based test bench is used for verification since it has the following advantages:

- Reusability and modularity: UVM is based on modular classes (environment, agent, driver, etc.) which can be reused at different levels.
- Separating tests: Tests comprise of stimulus generation which can be separated from the whole test bench, thus a single test bench can be re-used for multiple test cases, where only stimulus generation and test class can be changed.
- Stimulation generation using sequences: Several techniques like randomization, virtual sequences, and layered sequences can be used for generating rich stimulus.

1.4. Background

Over the past few years, the increasing demands for data density, high quality, and high-resolution images have raised the bandwidth capacity of

mobile and image sensor applications. It was difficult to maintain the bandwidth requirements because of parallel interfaces present in such applications which consumes more space and requires more power due to more number of interconnects. So, to solve this problem, a high-speed differential interface, subLVDS was introduced in 2006 to replace the parallel bus [3]. Later multi-lane structures for the MIPI interface started gaining popularity than single-lane structures due to improved data rate [4][5]. Currently, MIPI D-PHY is widely used for interfacing between camera and host processor devices [6].

For monitoring areas of interest, multiple cameras have been used. For transmitting multiple data streams from different cameras or image sensors, the concept of virtual channel has been developed in MIPI CSI-2 interface [7].

Since surveillance has become an integral part for security concerns, research has been going on improving the smart cameras. Since transmission of any raw image having high resolution requires more data and more bandwidth, so methods have been introduced to transfer only regions of utmost importance. Therefore, several methods have been also introduced to detect the regions of interest and transmitting only those regions using smart cameras [8][9].

Since verification plays a major role in designing of Verification IPs or any digital or analog ICs. Since digital designs have been becoming more complex over time, so the need of reducing the verification time and increasing the efficiency and accuracy of verification techniques has become important. Research has been conducted over the years on automated verification plans to control the verification process directly which is reusable [10][11]. So coverage models have been introduced over the years for verification of designs and to check the completeness of tests [12]. Two methods for coverage have been developed i.e. code coverage and functional coverage [13]. In semiconductor industry assertion-based

verification technique has become very important because of its ability to detect errors for efficient and advanced debugging [14][15]. Assertions are HDL based representation of design specifications for functional verification which are fired during a violation of design specification, thus improving debugging [16][17]. The traditional approach to achieve coverage is not efficient as it may lack covering hard corner case scenarios, So, for functional verification randomization technique has started gaining popularity as it guarantees the completeness of coverage [18][19]. Random stimuli or sequences are generated using System Verilog techniques which also reduce the effort for generating test cases [20]. So assertion and code coverage techniques have been combined for increasing the efficiency of verification methodology [21][22].

1.5. Motivation

Since improved data rate and low power have been a major concern over the past years in mobile camera industry, this thesis introduces C-PHY physical layer interface for improving data rate and reducing power consumption. In this thesis, a method has been developed for extracting and transmitting only regions of interest through CSI-2 interface due to the increased requirement of transmitting only regions of interest. For improving the efficiency in verification of these features, a verification architecture has been proposed in this thesis which uses a combination of functional and coverage based verification with randomization technique and uses a scoreboard for equivalence checking of design, thus improving the robustness of the design.

1.6. Objective

The goal of this thesis is to improve existing verification architecture for more efficient bug tracking and introducing features in CSI-2:

- For improving the data rate and power for physical layer interfacing.
- For improving the transport efficiency and packet delimiter overhead in packet spacing.
- For improving the adaptive transfer of regions of interest
- The existing Verification architecture has been improved for increasing efficiency and accuracy for bug tracking of design.

1.7. Organization Of thesis

As we have already discussed the motivation and objective of this thesis in chapter 1. Chapter 2 focuses on the improvement in PHY layer of CSI-2, it compares the existing D-PHY layer with the newly introduced C-PHY layer. Chapter 3 describes the protocol layer of CSI-2 and introduces EPD as a replacement of LPS packet spacing for latency reduction and transport efficiency. It also introduces the method used for transmitting SROI. Chapter 4 focuses on improvement done in existing verification architecture. Chapter 5 presents results, discussion and comparison with existing techniques. Chapter 6 presents the conclusion of this thesis work and the direction of future work.

Chapter 2

2. Physical Layer

2.1. D-PHY Physical Layer

D-PHY physical layer option used for interfacing CSI-2 is composed of 1 clock lane and several data lanes which are unidirectional in nature. D-PHY supports continuous clock lane behavior. Clock lane stays in high-speed for clock having continuous behavior and active clock signal is generated between the transmission of data packets. Clock lane enters LP 11 state between data packets transmission for clock having non-continuous behavior [23].

Prefix	**Lane Interconnect Side**	**High Speed Capabilities**	**Forward Direction Escape Mode Features Supported**	**Reverse Direction Escape Mode Features Supported**
CIL	M – Master S – Slave X	F – Forward Only R X	A – All E X	A – All E N – None X

Note: R- Reverse and Forward, E- Events, Triggers and ULPS Only, Y- Any(A,E) X-Don’t Care

Table 2.1 Lane Type descriptors

The minimum requirement for a CSI-2 transmitter in D-PHY physical layer is:

• Data lane module: LP-TX, HS-TX with unidirectional master and CIL-MFEN function
• Clock lane module: LP-TX, HS-TX with unidirectional master and CIL-MCNN function

The minimum requirement for a CSI-2 transmitter in D-PHY
physical layer is:

• Data lane module: LP-RX, HS-RX with unidirectional slave and CIL-SFEN function
• Clock lane module: LP-RX, HS-RX with unidirectional slave and CIL-SCNN function

2.1.1. Lane States and Line Levels

Lane states are determined by transmitter functions driving certain line levels. The Line Levels are used to determine the two possible transmission schemes. The primary transmission scheme uses combination of LP mode single-ended signaling and HS differential signaling. An optional secondary transmission scheme uses a combination of Alternate Low power mode and HS differential signaling [24][25].

2.1.1.1. HS and LP Mode Lane States and Line Levels

Either LP-TX or an HS-TX drives the lane during normal operation. An HS transmitter controls the lane differentially. The two LP transmitter's controls the lines of a lane single-ended and independently. This results in four possible low power lane states and two possible high-speed lane states. Differential-1 and differential-0 are the two high-speed data lanes. The low power receiver views both the differential high-speed state as LP 00.

The stop state has a central function. If the line levels display a stop state for a minimum time state, the PHY state returns to the stop state,

regardless of the previous. This can be TX or RX mode which depends on the most recent direction of operation. Table 2.2 lists all states appearing on a lane.

The duration of all low power state periods should be at least for time TLPX. State transitions must exclude glitch effects and shall be smooth. Exclusive OR of Dn and Dp lines are used for reconstructing clock signal. Ideally, the duration of reconstructed clock is at least 2*TLPX, but due to trip levels and signal slope effects, it may have a duty cycle other than 50% [26].

State Code	Line voltage level Dp line	Line voltage level Dn line	High Speed Burst Mode	Low Power Control Mode	Low Power Escape Mode
HS 0	HS Low	HS High	Differential 0	N/A, Note 1	N/A, Note 1
HS 1	HS High	HS Low	Differential 1	N/A, Note 1	N/A, Note 1
LP 00	LP Low	LP Low	N/A	Bridge	Space
LP 01	LP Low	LP High	N/A	HS Request	Mark 0
LP 10	LP High	LP Low	N/A	LP Request	Mark 1
LP 11	LP High	LP High	N/A	Stop	N/A, Note 1

Note:

1. LP 00 state is observed during HS transmission by LP receivers.
2. When data lane encounters LP 11, it returns to the stop state during escape mode.

Table 2.2 Lane State Descriptions

2.1.2. Operating Modes: Control, High-Speed, and Escape

2.1.2.1. LP and HS Operating Modes

During normal operation in LP and HS modes, a data lane is always either in high-speed or control mode. Transmission of high-speed data in busts begins from stop state and finishes at stop state, of control mode. During Data bursts, the lane is only in high-speed mode. LP 11, LP 01, LP 00 is the sequence to enter high-speed mode. The escape mode can only be entered within control mode via a request. After detection of a stop state, the data lane should always exit escape mode before returning to the control mode. The data lane should remain in control mode, if not in escape mode or high-speed. For clock lanes and data lanes, the stop state may last for any period of time > TLP-EXIT or > TLPXTHS EXIT and serves as a general standby state. Escape mode request (LP 11, 10, 00, 01, 00) High-Speed Data Transmission request (LP 11, 01, 00), or Turnaround request (LP 11, 10, 00, 10, 00) are some of the possible events which start from stop state.

2.2. C-PHY: Improved Data Rate

When there is a limitation of channel rate, a rate efficient and high-speed PHY called as C-PHY is preferred. For a group of 3 wire conductors, C-PHY uses the technology of three-phase encoding which delivers 2.28 bits/symbol, thus improving the channel rate.

The minimum requirement for a CSI-2 transmitter in C-
PHY physical layer is:

• Data lane module: LP-TX, HS-TX with a unidirectional master and CIL-MFEN function
• While transmitting data payload, it supports the insertion of a sync word.

The minimum requirement for a CSI-2 transmitter in C-
PHY physical layer is:

• Data lane module: LP-RX, HS-RX with a unidirectional slave, and CIL-SFEN function

• While transmitting data payload, it supports the insertion of a sync word.

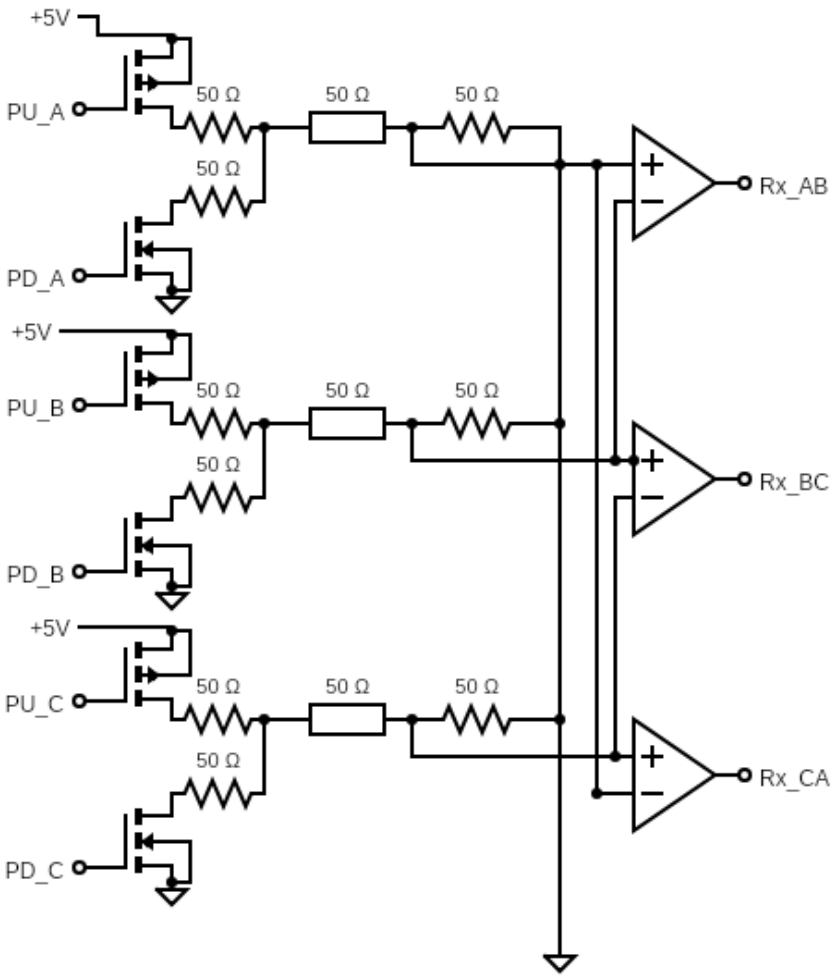


Figure 2.1 C-PHY wire states

C-PHY consists of three wires, as shown in Figure 2.1. The difference between these two wires out of the three gives one state as the output at the receiver. The six-wire states of C-PHY are shown in Table 2.3.

Wire State	V_A	V_B	V_C	PU_A	PD_A	PU_B	PD_B	PU_C	PD_C
+x	high	Low	medium	ON	OFF	OFF	ON	OFF	OFF
-x	Low	High	medium	OFF	ON	ON	OFF	OFF	OFF
+y	Medium	High	Low	OFF	OFF	ON	OFF	OFF	ON
-y	Medium	Low	High	OFF	OFF	OFF	ON	ON	OFF
+z	Low	Medium	High	OFF	ON	OFF	OFF	ON	OFF
-z	High	Medium	Low	ON	OFF	OFF	OFF	OFF	ON

Table 2.3 Six wire states of C-PHY

Since there are six different wire states and for the transition from one signal to another, each state has five possible states to move. Since these five-wire states are all different and unique, C-PHY is a system of base five. So for 5 base system the symbol rate i.e.

bit/symbol = log5/log2 = 2.3219.

But the C-PHY must map to a multiple of bytes i.e. 2 bytes (16 bits). So for calculating how many bytes or bits compose a symbol we can use the following equation

2^16 <= 5^S, S is number of symbols per 16 bits

S = 7.

So, to accomplish this, it requires that a mapper must consist of 2 bytes, the bits/symbol ratio is chosen as 16/7 = 2.2857. Thus for transmitting 16 bits, seven symbols are used in C-PHY.

For transmission and reception of 2-byte words (16 bits), a mapper is used for converting these 2 bytes into a symbol. At the transmitter side, after serializing the 7 symbols, each symbol is sent one by one serially to the receiver and thus each symbol drives the three wires of a lane i.e. A, B and C. Three differential outputs are received at the receiver side i.e. A-B, B-C, and C-A, which is in turn sent to the symbol decoder which decodes the 7 symbols. The output of the symbol decoder is sent to a De-mapper through serial to parallel converter, thus converting the 7 symbols again to 16 bits.

2.3. Comparison of D-PHY and C-PHY

Six-lane D-PHY using 14 wires with each lane having a speed of 1 Gbps is required for D-PHY to obtain a data rate of 6 Gbps. Similarly, three lanes are required for C-PHY using 9 wires with each lane having a speed of 0.875 Gbps to obtain the same data rate of 6 Gbps. In this case, the bandwidth for C-PHY is computed as 3*0.875*2.28 = 6 Gbps.

So, the following conclusions are deduced from the above calculation:

- For a bandwidth of 6 Gbps, C-PHY requires 3 lanes as compared to 9 lanes required in D-PHY.
- For a bandwidth of 6 Gbps, 9 wires are required for C-HY as compared to the requirement of 14 wires in D-PHY.
- C-PHY has a data rate of 0.875 Gbps as compared to 1 Gbps data rate of D-PHY.
- So, for same bandwidth, the number of lanes is reduced by 50% in C-PHY as compared to D-PHY.

- The number of wires is reduced by 57% in C-PHY as compared to D-PHY.
- The data rate is reduced by 12.5% in C-PHY as compared to D-PHY.

Thus for the same bandwidth, C-PHY has better performance than D-PHY because of reduced data rate, reduced power, and less number of wires used, thus reducing the area requirement.

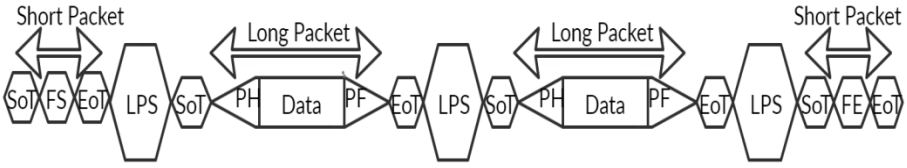
Chapter 3

3. Low-Level Protocol

Data is transmitted through thus layer using short or long packet formats, where data is byte-oriented.

Basic features of the Low-Level protocol are:

- Data is transmitted independently of the payload values i.e. arbitrary data.
- The word size is 8-bit.
- Sixteen virtual channels are supported for D-PHY and up to thirty-two virtual channels are supported by C-PHY.
- For frame end, frame start, line end, and line start, special packets are used.
- For detecting an error, 16-bit checksum code is used.
- For detection and correction of an error, 6-bit Error correction code (ECC) is used [27].



Key:

LPS: Low Power State

ST: Start Of Transmission

ET: End Of Transmission

PH: Packet Header

PF: Packet Footer

SP: Short Packet

Figure 3.1 Low-Level Protocol Packet

3.1. Low-Level Packet Format

In Low-Level protocol format as shown in Figure 3.1, long packet and short packet formats are defined. Depending on the physical later, length and format of packets may vary. The start of each packet is indicated by SOT sequence, which is followed by LPS. The packet end is identified by EOT sequence.

3.1.1. Low-Level Protocol Long Packet Format

Data types 0x10 to 0x38 are identified as long packet. The description of data types for the long data type is given in Table 3.1. As shown in Figure 3.2, for D-PHY layer option, the long packet comprises of the following three elements:

- Thirty two-bit packet header
- Eight-bit variable data which contains an application-specific payload data
- Sixteen-bit packet footer

32-bit packet header consists of for elements:

- Eight-bit data identifier
- Sixteen-bit word count
- Two-bit virtual channel extension field
- Six-bit Error Correction Code

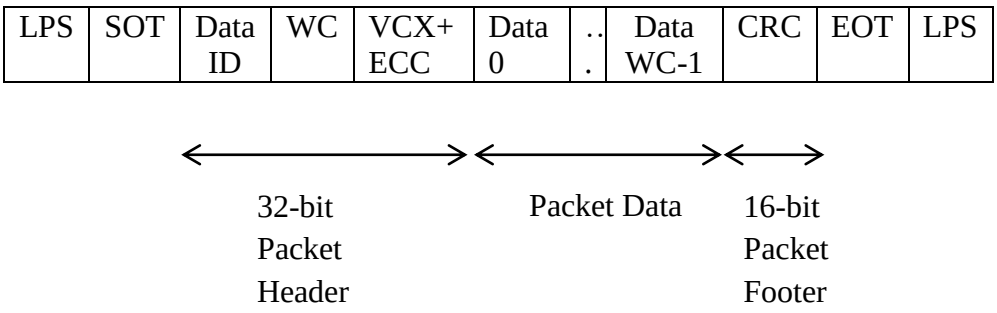


Figure 3.2 Long Packet Structure for D-PHY

For C-PHY, long packet structure format is composed of the following four elements:

- Packet Header
- Eight-bit variable data which contains an application-specific payload data
- Sixteen-bit packet footer
- Filler bytes (can be zero or more)

Two similar 6N byte halves compose packet header and each half compromises of N copies of following fields: a reserved field composed of 16-bit, Virtual Channel extension composed of 3-bits Payload data word count composed of 16 bits, a Data Identifier composed of 8 bits and checksum of packet header composed of 16 bits. Checksum (CRC) is calculated over whole packet payload data which and composes a 16-bit packet footer. Insertion of filler byte is also done when needed.

Two-bit virtual channel and data type of payload data compose the 8-bit data identifier for both physical layer options. The virtual channel is composed of 2 bits for D-PHY, and 3 bits for C-PHY and hence is common in both options. Virtual channel number for a particular packet is defined by combining virtual channel field and virtual channel extension fields composed of 4 or 5 bits. The number of data words (each of 8-bit) in data packet payloads defines the word count for both physical layer options. Packet filler, packet footer and packet header are not included in the word count. Correction of single-bit error and detection of two-bit errors is done by ECC in packet header for the D-PHY option.

A single error in a symbol of C-PHY layer can cause multiple errors in the packet header, so the C-PHY layer does not use the ECC field. So to calculate the error, C-PHY calculates CRC (Checksum Correction) over data identifier, VCX, Reserved and word count field which comprises the four bytes of the packet header.

Filler bytes are added only in the C-PHY layer option. Value of all filler bytes is always zero. If the total number of words in a packet data payload is odd i.e. LSB of word count is 1, then one packet filler is inserted after packet footer for ensuring that packet footer finishes on the boundary of 16 bit word. For ensuring that the C-PHY lane transmits the same number of 16-bit words on each lane, additional filler bytes are also inserted by the CSI-2 transmitter.

3.1.2. Low-Level Protocol Short Packet Format

The structure for the short packet of D-PHY is shown in Figure 3.3. The structure for the short packet in both the options is similar to packet header of the long packet except that the short packet data field replaces the word count field. Data types 0x00 to 0xFF are identified as short packets. A short packet only consists of a packet header. It does not contain packet footer and filler bytes.

Data field of the short packet is frame number for frame synchronization and its value is line number for line synchronization.

The data field of the short packet is user-defined for generic short packet data types. Correction of single-bit error and detection of two-bit error is accomplished by the presence of Error Correction Code field for D-PHY physical layer option. Detection of one or more bit errors can be achieved by the presence of 16-bit Checksum in the short packet of C-PHY layer, but correction of error is not supported by CRC.

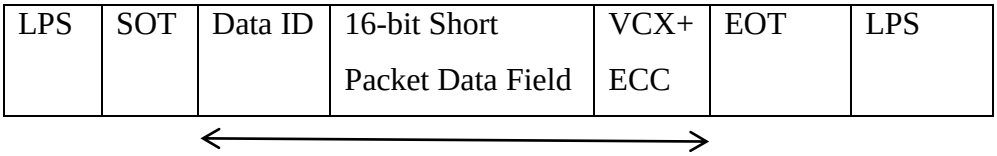


Figure 3.3 Short Packet Structure for D-PHY

3.2. Data Identifier (DI)

Data type and Virtual channel fields compose the byte of Data identifier, which is shown in Figure 3.8. The two most significant bits of Data identifier contains the Virtual Channel field. Six lower significant bytes of Data Identifier contains the virtual channel field.

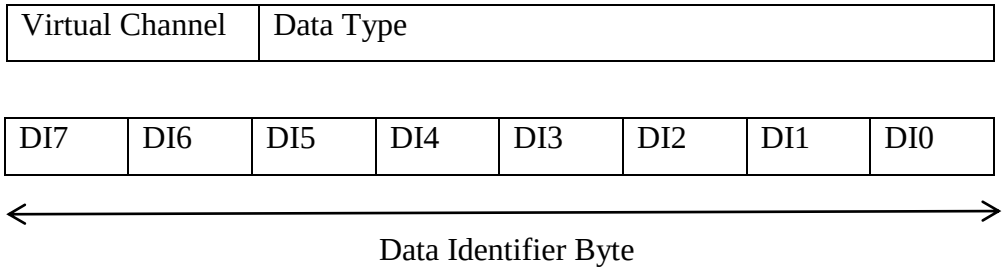


Figure 3.4 Data Identifier Byte

3.3. Virtual Channel Identifier

Different streams of data interleaved in a single stream of data can be logically separated by designating a separate identifier for each data flow using virtual channel identifier, which is composed of 4 or 5 bits. The most significant three bits of virtual channel number are taken from 3 bits VCX field, and the least significant two bits are the 2 bits of VC field. For both physical layers, VCX is situated in packet header. The virtual channel identifiers are extracted by the receiver from the packet headers, and thus the interleaved data stream is de-multiplexed to their corresponding channels. For D-PHY sixteen virtual channels are supported, and for C-PHY thirty virtual channels are supported. At the receiver side, peripherals are programmed so that it allows the host processor so that data streams of different virtual channels can be de-multiplexed. The block diagram of logical channel for selecting virtual channel is shown in Figure 3.5.

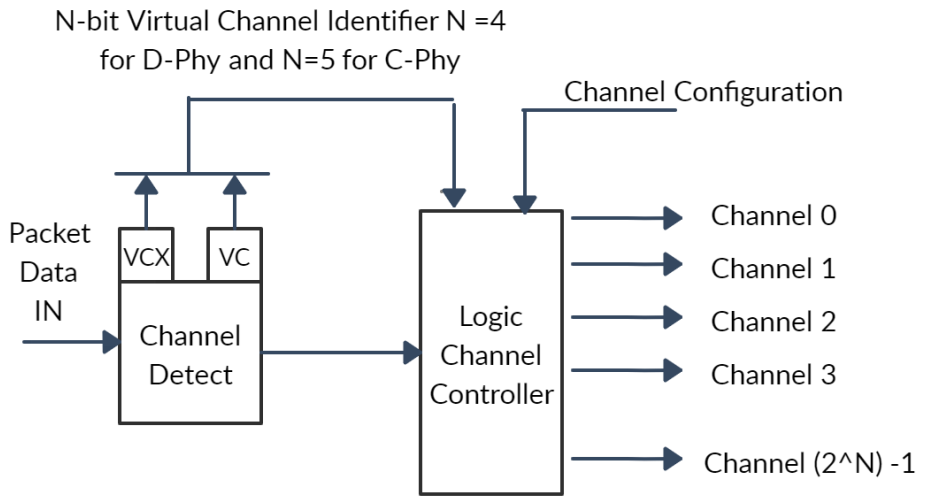


Figure 3.5 Block Diagram of Logical Channel

3.4. Data Type

The content and format of data payload are specified by data type value. The maximum number of data types supported by this protocol is sixty-four. Eight different types of data class are supported, as shown in Figure 3.11. Each data class can be further classified up to 9 different types of data definitions. There are two classes which denote data types of short packets and six classes which denote data types of long classes.

Data Type	**Description**
0x00 to 0x07	Synchronization Short Packet Data Types
0x08 to 0x0F	Generic Short Packet Data Types
0x10 to 0x17	Generic Long Packet Data Types
0x18 to 0x1F	YUV Data
0x20 to 0x26	RGB Data
0x27 to 0x2F	RAW Data
0x30 to 0x37	User Defined Byte-based Data
0x38	USL Commands

0x39 to 0x3E	Reserved for future use
0x3F	For CSI-2 over C-PHY: Reserved for future use For CSI-2 over D-PHY: Unavailable (0x3F is used for LRTE EPD Spacer)

Table 3.1 Data Type Classes

3.5. Packet Header Error Correction Code for D-PHY

It is very important that only correct values of word count, virtual channel, and data identifier are interpreted by the packet. Correction of single-bit error and detection of 2-bit errors in WC, VC, and DI fields is done by 6-bit ECC present in the packet header and a 26-bit hamming code is used. The Data Identifier field DI[7:0] maps the ECC input D[7:0], the Word Co unt LS Byte (WC[7:0]) to D[15:8], the Word Count MS Byte (WC[15:8]) to D[23:16] and the Virtual Channel Extension (VCX) field to D[25:24].

3.6. Checksum Generation

For detecting possible errors in a stream of transmission, a checksum is computed over all the bytes of packet data composing each long packet. Similarly, for C-PHY a checksum is computed over WC, VC, DI, and reserved fields. Generator polynomial x16+x12+x5+x0 is used for computing 16-bit checksum. Byte order transmission of checksum is shown in Figure 3.6.

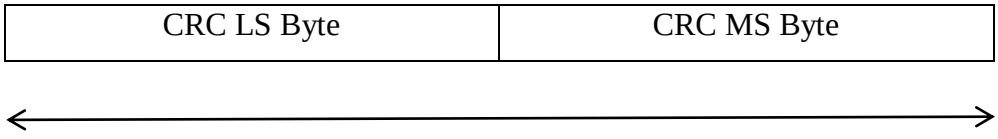


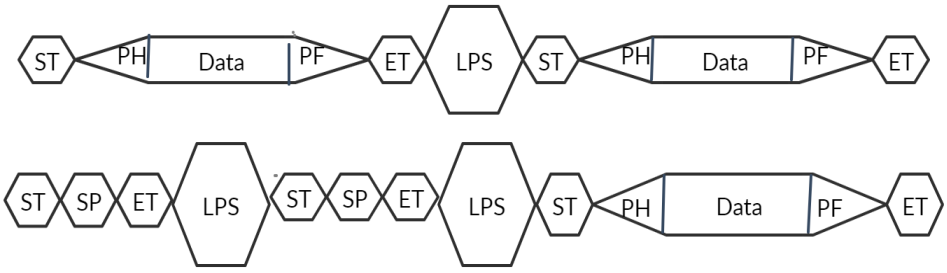
Figure 3.6 Checksum Transmission Byte Order

The checksum is transmitted in the packet footer after being calculated over the data payload packets. CRC is always 0xFFFF when the total count of words is zero. For C-PHY option checksum is transmitted in the packet header when calculated over DI, VCX, WC, and reserved fields.

Serial CRC is implemented by using the shift register. At the starting of each packet, the initial value of the checksum (CRC) shift register is assigned to 0xFFF. The initial value of CRC is assigned twice (each for CRC in the packet header and packet footer) for each packet if the same circuitry is utilized for calculating CRC in packet footer and packet header for C-PHY option. The checksum is finally computed and stored in CRC circuitry when all data payload has been passed through the circuitry of CRC. To verify that no error has been transmitted, the CSI-2 physical layer transmits the checksum to the receiver of CSI-2.

3.7. Packet Spacing

Between several protocol packets, CSI-2 makes transitions from Low Power State (LPS) to the packet and then packet to again LPS. Packet spacing with the inclusion of LPS is shown in Figure 3.7.



Key:

LPS: Low Power State

ST: Start Of Transmission

ET: End Of Transmission

PH: Packet Header

PF: Packet Footer

SP: Short Packet

Figure 3.7 Short/Long Packet Spacing

3.8. Synchronization Short Packet Data Type Codes

Table 3.2 illustrates the description of data types for synchronization short packet. Only short packet formats are used to transmit short data types.

Data Type	Description
0x00	Frame Start Code
0x01	Frame End Code
0x02	Line Start Code (Optional)
0x03	Line End Code (Optional)
0x04	End of Transmission Code (Optional)
0x05 to 0x07	Reserved

Table 3.2 Synchronization Short Packet Data Codes

3.8.1. Frame Synchronization Packets

Each frame is transmitted by beginning with the packet of frame start (FS) which contains the frame start code. Long packets contain the data of image or for transmitting synchronization codes which is followed by the frame start packet. Every image frame finishes with a frame end (FE) code. The data field of the short packet contains a frame number of 16-bit for synchronizing frame start and frame end packets. For a given frame, the frame number is always the same for synchronization packets of frame end and frame start. Frame number should be always non-zero when the use case is operative frame number and for the use case where frame no. is inoperative, it should be set to zero.

The behavior of the frame number consisting of 16-bit is given below:

- For inoperative frame number, frame no. is zero.
- Within the same virtual channel, for each frame start packet frame no increments by 1 or 2 and is rest to 1 periodically e.g. 1,2,3,4 or 1,2,1,2 or 1,3,5,1,3,5 or 1,2,4. When an image frame is not

transmitted due to corruption, frame no. can be incremented by 2. So increment of frame number by 1 or 2 can be intermixed within a frame.

3.8.2. Line Synchronization packets

Inclusion of line synchronization packets is optional. Data field of the short packet contains a line number of 16-bit for synchronizing line start and line end packets. For a given line the line number is always the same for synchronization packets of line end and line start.

The behavior of the line number consisting of 16-bit is given below:

- For inoperative frame number, line no. is zero.
- Within the same virtual channel, for each frame start packet frame no. increments by 1 or by some arbitrary step greater than 1 which is constant and is rest to 1 periodically.

3.9. Generic Short Packet Data Type Codes

Data types for generic Short Packet are listed in Table 3.3.

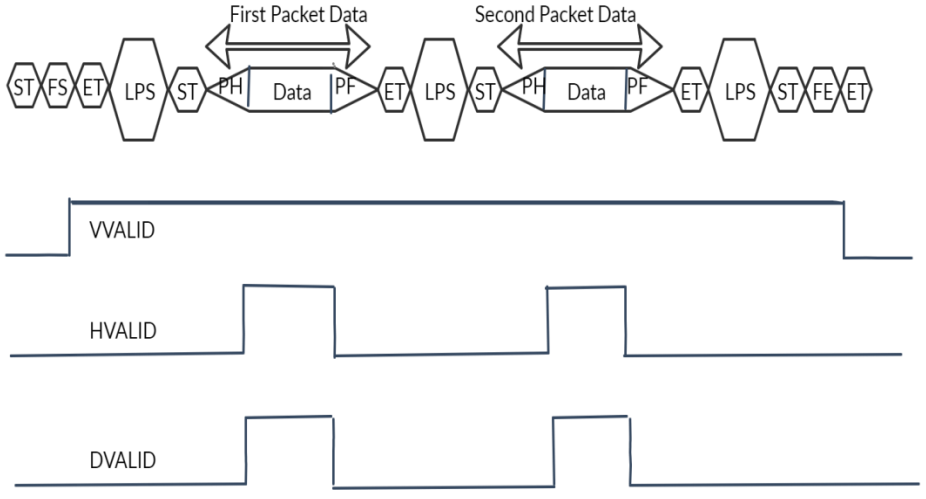
Data Type	Description
0x08	Generic Short Packet Code 1
0x09	Generic Short Packet Code 2
0x0A	Generic Short Packet Code 3
0x0B	Generic Short Packet Code 4
0x0C	Generic Short Packet Code 5
0x0D	Generic Short Packet Code 6
0x0E	Generic Short Packet Code 7
0x0F	Generic Short Packet Code 8

Table 3.3 Generic Short Packet Data Type Codes

Information for flash triggering, closing and opening of shutters can be included in the data stream by the mechanism provided by generic short packet data types. A user-defined data type and 16-bit data can be transmitted to the application layer by inclusion of user-defined data field in generic short packets.

3.10. Examples of Packet Spacing Using LPS

For both the physical layer option, separation of the packet is accomplished by inserting End of Transmission (EOT), Low Power State (LPS), Start of Transmission (SOT) sequence. Examples for data frames consisting of single and multiple packets are illustrated in Figure 3.17 and Figure 3.18. VVALID, DVALID and HVALID show the behavior of line start, line end, frame start and frame end packets. Line blanking period is defined as the time period between the packet footer of N packet and packet header of N+1 packet. Frame Blanking Period is defined as the time period between Frame End in one packet and the Frame Start of the next packet. For the frame blanking period, the minimum time is defined for the transmitter. An example of multi packet spacing is shown in Figure 3.8.



Key:

LPS: Low Power State	PH: Packet Header
ST: Start Of Transmission	PF: Packet Footer
ET: End Of Transmission	SP: Short Packet
FS: Frame Start	FE: Frame End

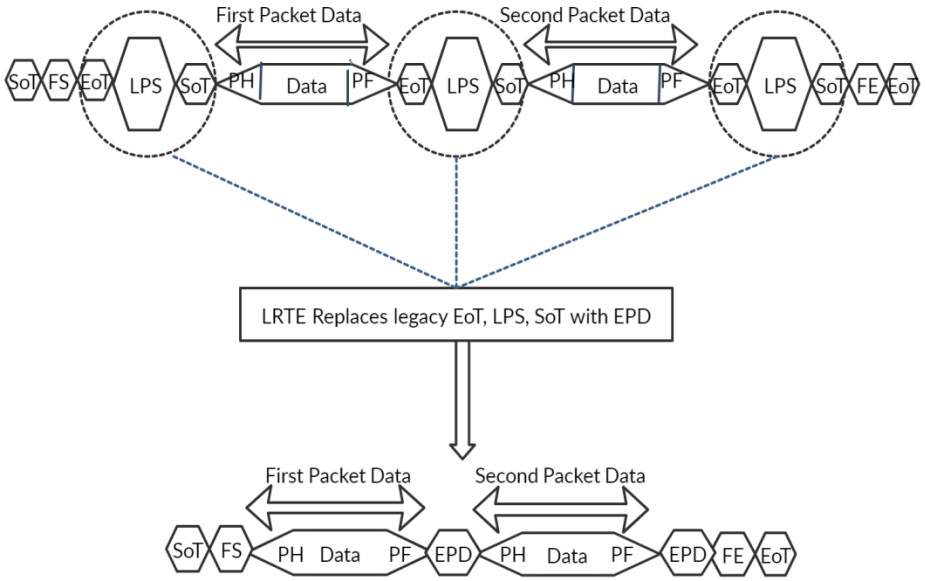
Figure 3.8 Example of Multiple Packets

3.11. Improved method for packet Spacing Latency Reduction and Transport Efficiency

An improved method called Latency Reduction and Transportation (LRTE) is introduced for the separation of packets for optimal transport of packets, which has two important features: Latency Reduction and Transport Efficiency.

3.11.1. Interpacket Latency Reduction (ILR)

In both physical layers, long packets and short packets of CSI-2 are separated by End of Transmission, Low Power State and Start of Transmission. The overhead of these packet delimiters is reduced by the mechanism of LRTE PDQ. The need for transitions from High State to low power and then to High State (HS-LPS-HS) is avoided by replacing EOT, LPS and SOT by LRTE PDQ which is more efficient. An EPD is composed of protocol generated and/or PHY generated elements. Packet Delimiter Quick (PDQ) is PHY generated and spacers are protocol generated. The transmission of EPD for separating packets is shown in Figure 3.9.



Key:

LPS: Low Power State

SoT: Start Of Transmission

EoT: End Of Transmission

FS: Frame Start

EPD: Efficient Packet Delimiter

PH: Packet Header

PF: Packet Footer

SP: Short Packet

FE: Frame End

Figure 3.9 LRTE EPD

EPD is composed of a PHY generated PDQ (Packet Delimiter Quick) which can be optionally preceded by one or more spacers. When EPD is enabled, filler bytes are inserted to align all the lanes accordingly. There are two options for EPD in D-PHY:

- EPD option 1
- EPD option 2

Two registers are used for both options of EPD

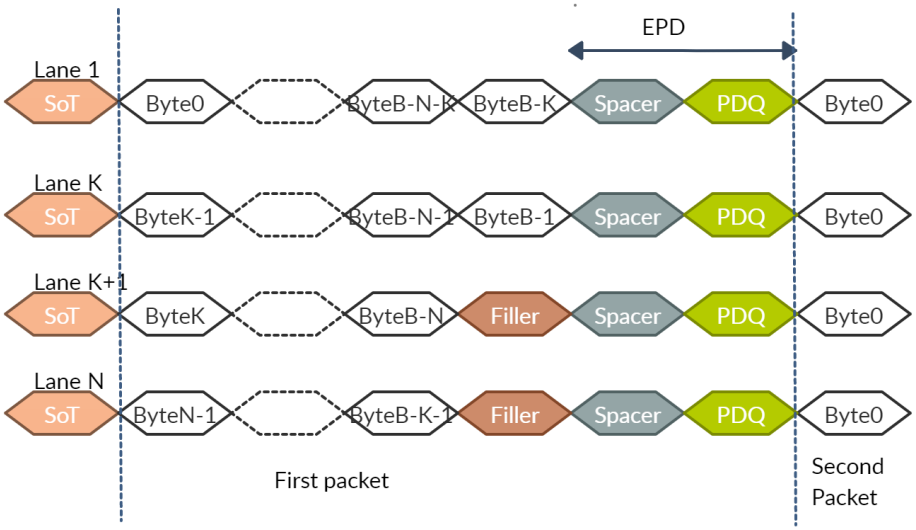
- One register is for enabling EPD and transmitting spacers in the short packet. The MS bit of this register is used for enabling

transmission of EPD consisting of Sync word and spacers. If MS bit is 1, EPD is enabled and if MS bit is 0, legacy EOT, LPS and SOT is transmitted as packet delimiter. The remaining 15 bits of the register is used for specifying the minimum number of spacers that can be inserted per lane.

- The second register is used for selecting EPD options for D-PHY and transmitting spacers in a long packet. The MS bit of this register is used for selecting the option used for EPD. If MS bit is 0, EPD option 1 is used, or if MS bit is 1, EPD option 2 is used. The remaining 15 bits of the register is used for specifying the minimum number of spacers that can be inserted per lane in the long packet.

3.11.1.1. EPD option 1

EPD for D-PHY option 1 is composed of a PHY generated HS-idle PDQ (Packet Delimiter Quick) which can be optionally preceded by fillers and one or more spacers. PDQ is a robust packet delimiter since it is PHY generated and PHY consumed. For PDQ signaling clock is used, so using spacers can help in using additional clock cycles which can be used for flushing payload data before PDQ is disabled.



Key:

LPS- Low Power State SoT- Start of Transmission

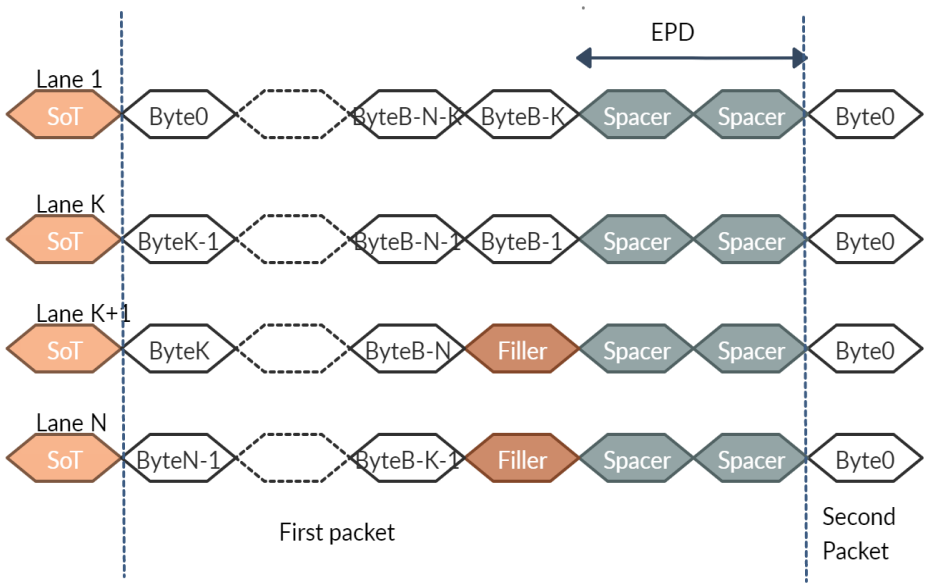
EoT- End Of Transmission EPD- Efficient Packet Delimiter

PDQ- PHY Generated and Consumed Packet Delimiter Quick

Figure 3.10 EPD option 1

3.11.1.2. EPD option 2

In option 2 only spacer bytes which are protocol generated and protocol consumed are inserted when PDQ cannot be generated.



Key:

LPS- Low Power State SoT- Start of Transmission

EoT- End Of Transmission EPD- Efficient Packet Delimiter

Figure 3.11 EPD Option 2 for D-PHY

3.12. Comparison of LPS spacing and LRTE EPD

Consider an image sensor supporting 5.3 Megapixel (2592 x 2056 pixels), with 60 frames per second and 12 bits transmitted per pixel. For CSI-2 over C-PHY having a speed of 3.5 Gbps per lane. Thus the total bandwidth of 3 lane C-PHY is 3*3.5*2.28 = 24 Gbps.

Packet Transport of Image Sensor:

- 2592 pixels x 12 bits per pixel = 31,104 bits per row i.e. long packet
- Number of bits in packet header = 6*3*16 = 288 and number of bits in packet footer = 16 bits
- Time per packet including packet footer and header = 31,408 bits/24 Gbps = 1.3 us.

CSI-2 with C-PHY legacy packet delimiters:

- Start of Transmission and End of Transmission with LP delimiter = ~0.5us.
- Total time per packet including LP delimiter = 1.8us
- Overhead of packet delimiter = 27.7 %
- Times per image frame = 1.8us * 2056 = 3.7ms
- Total time required for streaming 60 frames from a single image sensors = 60 * 3.7ms = 222ms
- Total number of image sensors per aggregator that can be supported = FLOOR[1/0.222] = 4

CSI-2 with LRTE PDQ:

- Time per image frame = 1.3 us * 2056 = 2.6ms
- Total time required for streaming 60 frames from a single image sensors = 60 * 2.6ms = 156ms
- Total number of image sensors per aggregator that can be supported = FLOOR[1/0.156] = 6

Benefits of CSI-2 with LRTE:

- Improvement in transport efficiency of frame = 27.7 %
- Number of image sensors additionally supported per aggregator = 2

3.13. Smart Region of Interest (SROI)

Regions of interest/importance, extracted out in rectangular portions can be transmitted selectively out of the complete image instead of sending the whole image frame. Data bandwidth of the frame can be reduced by selectively transmitting only regions of interest after carving it out from an image frame, such as the number plate of a vehicle or human faces.

Advantages of SROI are:

- High Frame Rate: By reducing the amount of data, the frame rate is also increased and the workload on processors is also reduced
- High Resolution: Since the amount of data is reduced, high resolution can be transmitted without increasing much data
- Low Power dissipation: By reduction of workload on processors, system power is also saved.

Applications of SROI are:

- Better deductions can be made by analyzing regions of interest using algorithms. By extracting and carving only important information from images it can help in
    - Defects on a conveyer belt can be identified more easily and quickly by machines.
    - Medical anomalies like tumour can be recognized more efficiently.

3.13.1. SROI Frame Format

Each region of interest is defined by its size and position, i.e. height, width, X, and Y coordinates of ROI rectangle. Regions of interest are

either identified or defined by image processor which are set in advance or application processor by a detection function integrated into the processor.

Three data packets are used by SROI:

- SROI Short Packet: In an image frame, the SROI Long packet is transmitted with frame start and frame end. Line end and line start are also transmitted as SROI short packets when line synchronization is required.
- SROI Embedded Data Packet: The information of ROI i.e. position and size is transmitted in SROI Embedded Data Packet. It is always located at the beginning of the image frame.
- SROI Long Packet: The image data of ROI is transmitted using SROI long packet in an image frame.

The same image data type is used for transmitting SROI long packet as used to transmit long packets of normal image. In a normal image, the length of long packets is always the same within a frame which is not required in long packets of SROI.

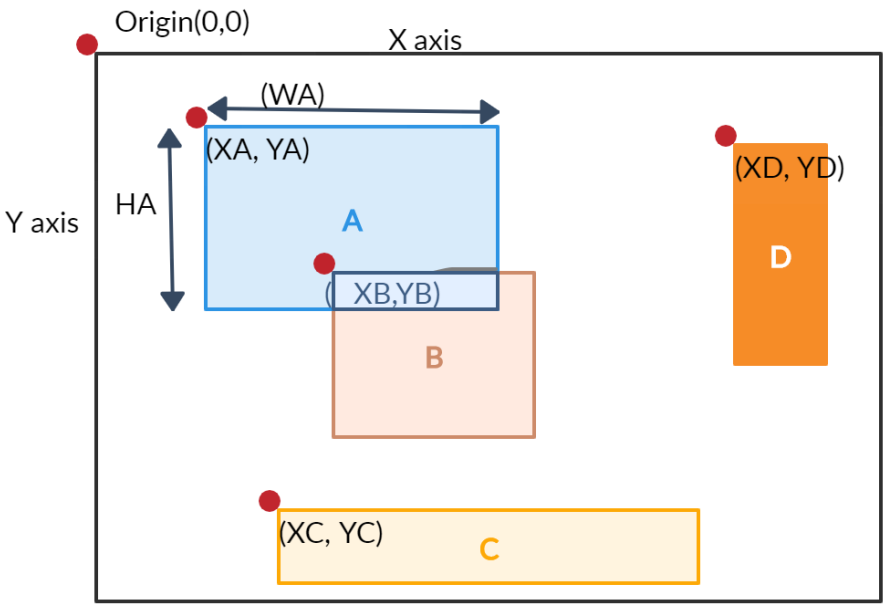


Figure 3.12 ROIs in an image frame

When multiple regions of interest are encountered within one line, overlapped data of all ROIs are merged into a single packet of SROI, and blanking region between them is removed.

As shown in Figure 3.12 and 3.13, the blanking region between A(0) and D(0) is not transmitted and the overlapped regions of multiple ROIs, for example B(0) and A(n-2) is transmitted only once. So while counting the word count of the long packet, the overlapped regions of all ROIs are counted only once.

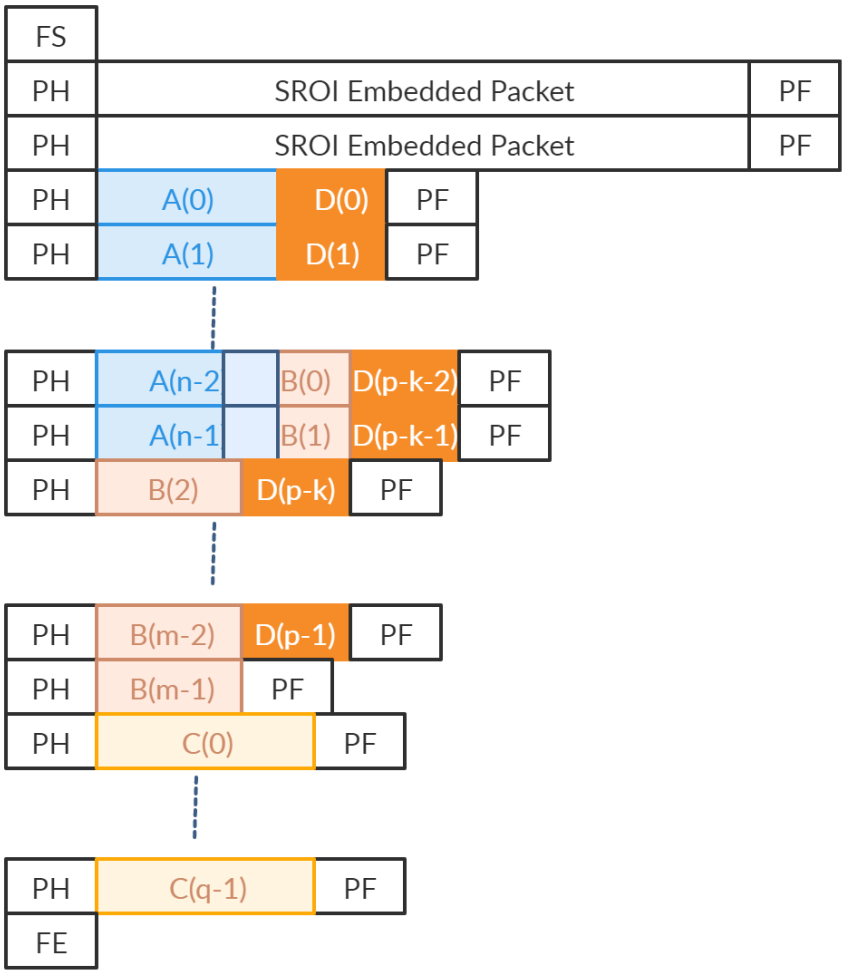


Figure 3.13 Transmission of SROI Frame and its format

3.13.2. Algorithm used for extracting and transmitting SROI

Test Class: As shown in Figure 3.14, an image and the information of ROIs i.e. X, Y, height and width coordinates in that image are given as input by the user in Test class. SROI class is called with initializing its data members X, Y, height, width and image according to the inputs given by the user.

SROI Class: Pixels information of the image is extracted from the BMP class. This pixel information including height and width of the original image, a two-dimensional array storing the pixel values of the image. Two algorithms (horizontal overlapping and vertical overlapping described in Section 3.13.2.1 and 3.13.2.2) are combined for extracting and transmitting only regions of interest with overlapped portion transmitted only once and removing any blanking portions.

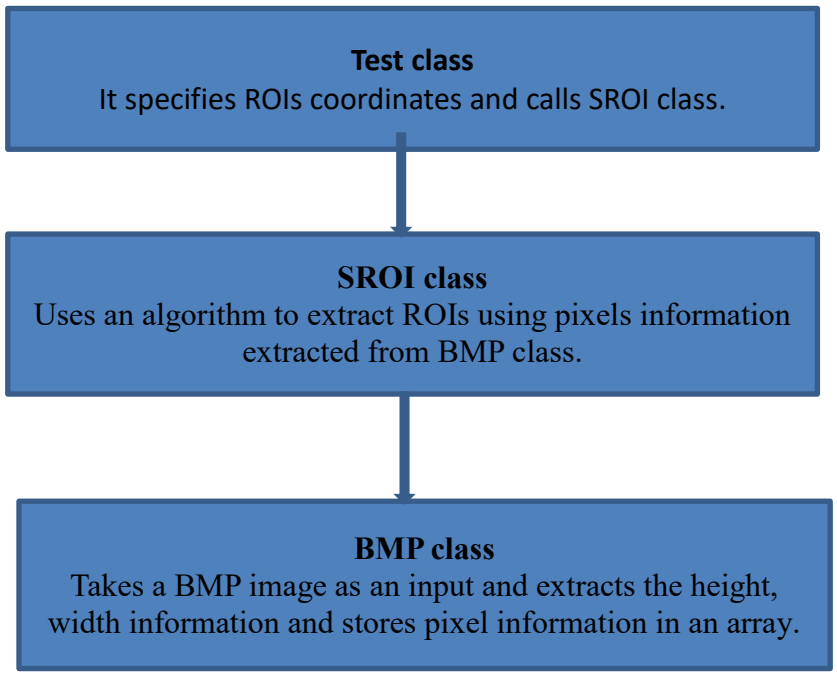


Figure 3.14 Algorithm for extracting ROIs

BMP Class: This class takes the original BMP image as input and calculates the height and width of the original image. It extracts the pixel values of the image from the DIB header of the bitmap file. These pixel arrays are converted to the different data formats like RGB888, RGB565, RGB666, RGB444 etc. depending on the data format used. Thus this pixel array can be further used to form a new pixel array only consisting of ROIs information.

3.13.2.1. Horizontal Overlapping

As shown in Figure 3.27, the first step to identify the overlapped region between two ROIs in a horizontal line is to start with a loop for each line and identify which ROIs lie in that line.

It then sorts out the ROIs lying in a region based on the increasing order of their x coordinate thus arranging them in the order in which they lie in the line. Then the pixels of ROIs will be stored in sroi_pixels array. Then starting for a particular line, the ROI having the smallest x coordinate will be filled first in the sroi_pixels array. Then it is checked whether the next ROI overlaps the previous ROI. If the next ROI overlaps the previous ROI, then sroi_pixels array is filled only with the non-overlapped region. If more ROIs lie in that line, the same thing is checked for the next ROI. It also checks for the blanking region between two ROI whether there is a blanking region (a region which is not considered in any ROI) or not between any two ROI, then this region is removed. For the last ROI it checks whether there is any region after it or not, it removes that region and then moves to the next line.

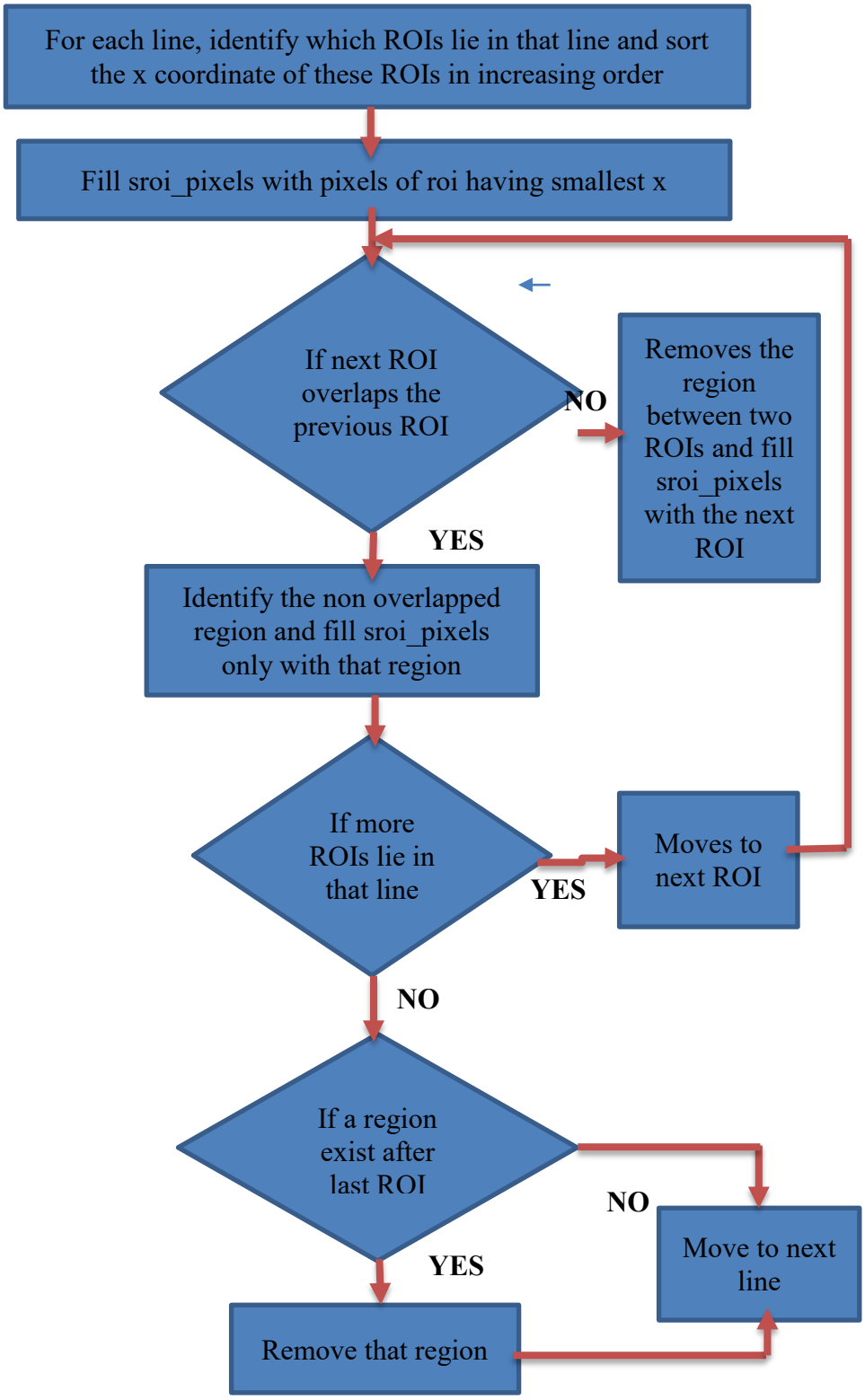


Figure 3.15 Algorithm for Horizontal Overlapping

3.13.2.2. Vertical Overlapping

For vertical overlapping, it checks each line, whether any ROI lies in that line or not. If there isn't any ROI in that line, then that line is removed, or if there is any ROI then it applies the horizontal overlapping algorithm for that line and hence moves on to next line and repeats this process for each line until it reaches the last line in that frame.

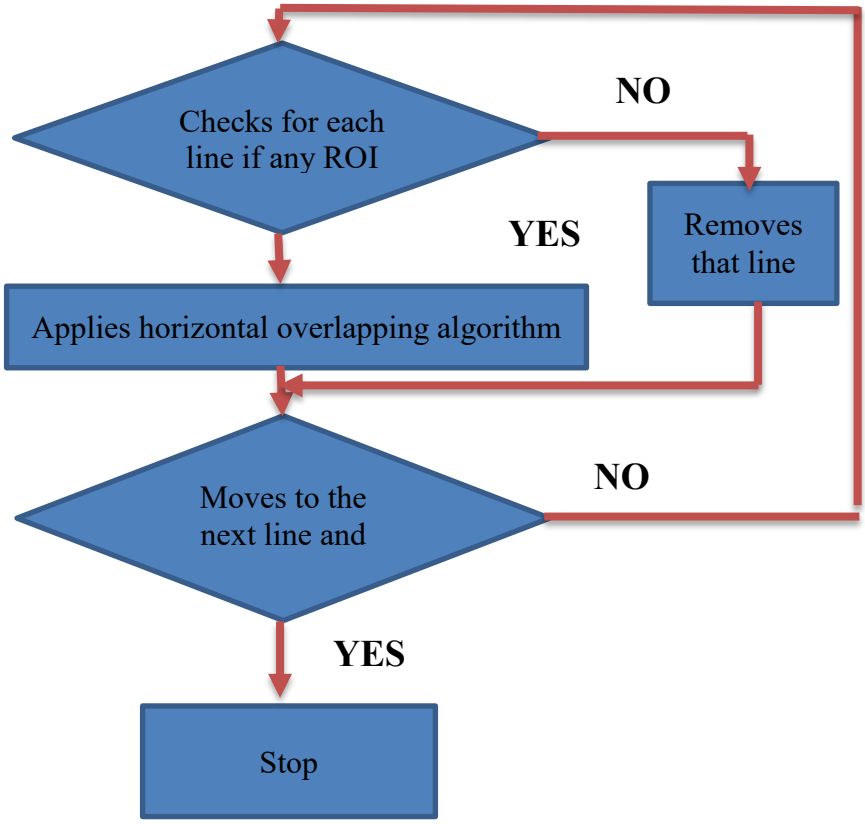


Figure 3.16 Algorithm for Vertical Overlapping

3.13.3. Transmission of SROI Embedded Data Packet

The condition or requirement for sending SROI Embedded Data Packet (SEDP) by an image sensor is described in this section which depends on the condition where the Application processor knows ROI or not.

There is no compulsion to send the SEDP when the application processor knows the information of all ROIs to be received. However, SEDP can be optionally transmitted, and if SEDP is transmitted and SROI packet option 1 is used, then the application processor can detect SROI by inspecting the values of the virtual channel.

It is required to transmit SEDP when the application processor does not know the information of all ROIs. Application processor can identify SROI packets either by virtual channel for option 1 r by inspecting the data type for option 2.

		AP Knowledge of All Required ROI Information	
		AP Knows	AP Does Not Know
SEDP Required		No	Yes
SROI Packet Option	Option 1	AP either knows SROI Packet, or can detect SROI Packet by Virtual Channel	AP can detect SROI Packet by Virtual Channel
	Option 2	AP knows SROI Packet	AP can detect SROI Packet by Data Type

Table 3.4 Transmission of SROI Embedded Data Packet

3.13.4. SROI Packet Detection Option

There are two options for detection of SROI by application processor:

- SROI Packet Option 1
- SROI Packet Option 2

3.13.4.1. SROI Packet Option 1

In option 1, SROI packets are distinguished from the non-SROI packets by using a different virtual channel value.

The same virtual channel is used for all SROI packets in a SROI frame and this virtual channel is different from other non-SROI packets. So, the application processor can identify the virtual channel of SEDP and all the packets having that same virtual channels are identified as SROI packets. So, virtual channel interleaving can also be used in such frames.

3.13.4.2. SROI Packet Option 2

In option 2, SROI packets are distinguished from non-SROI packets using the data type. It identifies the frame containing SEDP, and all the packets in that frame are detected as SROI.

So, for option 2 virtual channel interleaving cannot be used hence all the packets whether it is SROI or non-SROI should have the same value of the virtual channel.

3.13.5. Format of SROI Embedded Data packet (SEDP)

SEDP is of embedded data type and has data type code 0x12. Every embedded data type is followed by its data format code used for specifying its application. For SROI Embedded Data Packet, the data format code is 0x0D

Multiple ROI fields that contain information about the extracted region constitute the SROI Embedded Data Packet. Every ROI field starts with ROI ID of it, and then other information like height, width, X and Y coordinates consisting of type ID of the field and its data stored in the specific payload are followed by ROI ID. The format of SEDP is shown in Figure 3.17.

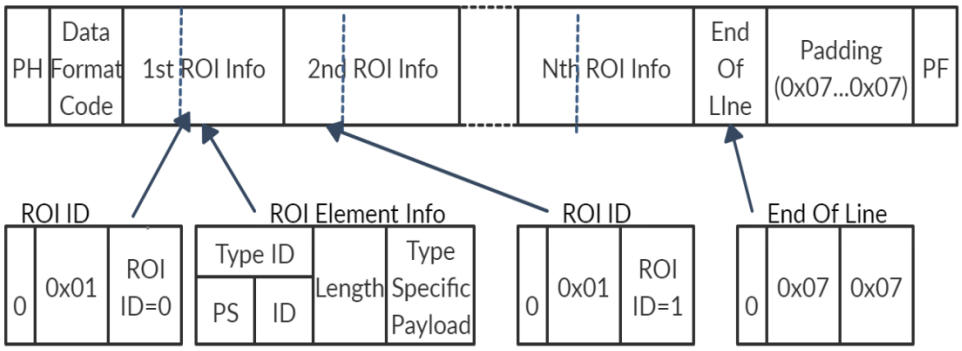


Figure 3.17 SROI Embedded Data Packet Format

Field Name	Size (bits)	Description
Type ID (Payload Size)	2	Size of the Type Specific Payload 0: 1 byte 1: 2 bytes 2: 4 bytes 3: Size is given by the Length field(below)
Type ID(ID)	6	Type Identifier (bottom bits of first byte): 0x00 – 0x1F MIPI Defined ID 0x20 – 0x3E User Defined ID 0x3F MIPI Defined ID
Length	8	**If Payload Size is 3 (2'b11):** Number of 8 bit data bytes in the type specific payload field. Example: A value of 0x20 in the Length field indicates that the type specific payload contains 32 bytes. **If Payload size is 0,1 or 2:**

		Length field is not included in the ROI Element Information Field.
Type Specific payload	Variable	The payload data is byte based i.e. data size shall be divisible by 8 bits.

Table 3.5 ROI Element Information Field

At the end of the embedded line, End Of Line is inserted after the last ROI field. The length of the embedded line does not exceed the length of the full resolution image. After last ROI field, the remaining line is padded with End Of Line with data 0x07 is the length of SROI embedded data packet line is less than the full length of resolution image.

Type ID	**Name**	**Description**
8'b00_0_00000	**Reserved**	Reserved for future use
8'b00_0_00001	**ROI ID**	Identification number (1 Byte) of each ROI. (Mandatory)
8'b00_0_00010	**ADC Bit Depth**	Bit depth of Analog Digital Converter of each ROI
8'b00_0_00011 through 8'b00_0_00110	**Reserved**	Reserved for future use
8'b00_0_00111	**End of Line**	The end of SROI Embedded Data line. The value of Type Specific Payload is 0x07.
8'b00_0_01000 through 8'b00_0_11111	**Reserved**	Reserved for future use
8'b01_0_00000	**Reserved**	Reserved for future use
8'b01_0_00001	**ROI ID (2 Bytes)**	Identification number (2 Bytes) of each ROI. (Optional)

8’b01_0_00010 through 8’b01_0_00111	**Reserved**	Reserved for future use
8’b01_0_01000	**X**	X coordinate of upper left region of image [pixel]
8’b01_0_01001	**Y**	Y coordinate of upper left region of image [pixel]
8’b01_0_01010	**Height**	Height of region [pixels]
8’b01_0_01011	**Width**	Width of region [pixels]
8’b01_0_01100 through 8’b01_0_11111	**Reserved**	Reserved for future use

Table 3.6 ROI Element Type ID Definitions

Chapter 4

4. Verification Architecture for CSI-2

4.1. Existing verification architecture

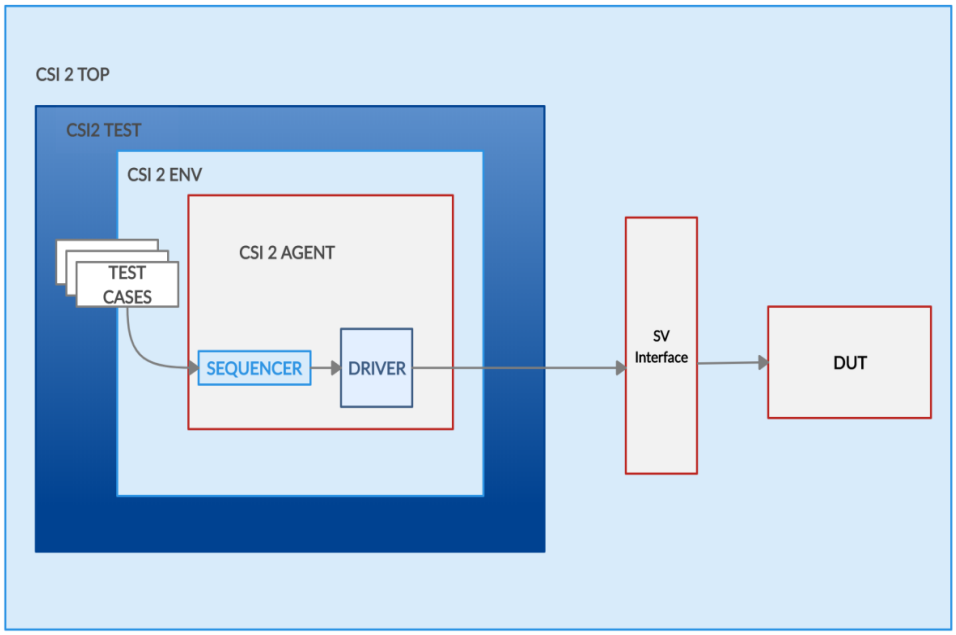


Figure 4.1 Existing Verification Architecture

DUT: Device Under Test is the main design that needs to be tested for improving its functionality. The Hardware IP consists of a set of modules and classes designed in System Verilog and System C languages. All the design features of CSI -2 like SOT, EOT mechanism, low power state transmission, EPD transmission and SROI etc. are contained in the DUT.

Several test cases are required to check all the functionalities of design in each and every corner case, thus improving its quality.

The existing verification architecture shown in Figure 4.1 consists of the following classes [28][29]:

- Top Class: This class act as the container class, which contains other classes like test class, environment, agent, driver and monitor. This class is majorly responsible for the connection between test class and DUT through virtual interfaces. Clock and reset signals are also generated in this class.

- Test Class: Test Class contains the environment class. Functionalities of test class are:
    - All the configurations like number of lanes used in CSI-2 are set in this class and all these configurations are passed onto the agent class.
    - This class is used for creating a particular test case scenario which is achieved by using user-defined sequences to drive specific stimulus.
    - Thus all the test cases are written, initiated and generated in this class.
    - In this class, virtual interfaces are retrieved and configuration objects are set into the database using set config_db.

- Environment Class: This class can contain multiple agents which are reusable components for verification wherein each agent has its own driver, sequencer and monitor. All the connections between other classes like connection of scoreboard and configuration classes with agent class. In this class, the configuration is also passed to the agent class from the test class.

- Configuration Class: This class will contain features or configurations required for a design or chip. In CSI-2 following configurations are used.
    - To set the number of lanes used.

- To set the number of virtual channels used.
- To set the options used in EPD
- To set the number of spacers used.
- To set the data type
- To set the resolution of image frame
- To set the line synchronization feature.
- To set the width of PHY like 16 or 32 bus width.

- Agent Class: Sequencer, driver and monitor are encapsulated and connected in this class. Analysis components include coverage collector and scoreboard. The agent has some configurability features like operating the agent in active mode or passive mode. It has configurations to turn on the coverage collector and scoreboard checker. If agent is set to work in active mode:
  - Driver and Sequencer are created
  - Driver is connected to Interface
  - Driver and sequencer are connected in connect phase
  - Collector is created in both active and passive mode

- Driver Class: Driver describes how signals need to be driven to the DUT interface. Thus this class defines how it processes the input coming through the sequencer and converts them to the required output which is driven to the interface. For communication between sequencer and driver, TLM ports are defined Following methods are used by driver to interact with the sequencer:
  - Get_next_item: This function waits and hence blocks until it gets a sequence item available through the sequencer.
  - Item_done: This is a non-blocking method, which indicates that the item has been successfully received by the driver through sequencer and hence is called after get_next_item.

- Sequence Items: The sequence items contain data items required for generating stimulus. Like in this protocol, sequence items for camera packet, SOT and EOT phase, Low power phase has been used. Camera packet has item fields like EPD is enabled or not, data type, and other packet information like frame number for Frame Start, etc.

- Sequences: For generating a specific stimulus, a series of sequence items are used in sequences. These series of sequence items are sent to the driver through sequencer. For writing a particular test case, its stimulus is generated in sequence class which is then called in test class.

A System Verilog Interface is used for connecting the Design Under Test (DUT) and the top class. In a traditional test bench all the components are static in nature including the DUT and test bench components where all the data (input/output) are in the form of either wire, net or signals[30].

But in the UVM test bench, DUT is static in nature but the components of test bench are in the form of classes and inputs are applied through transactions which all are dynamic in nature. So, a virtual interface is required for all the connections.

- Virtual interface is set in the top class using set config_db
- Using get config_db, the same virtual interface is retrieved (get) from the database in test class and passed on the configuration class.
- In agent, virtual interface is passed onto the driver, sequencer and monitor using configuration class.

4.2. Improved Verification Architecture

Following Classes are added in existing Verification Architecture for its improvement:

- Monitor Class: This class acts like a passive agent from which output from DUT interface can be extracted. So, this class captures output from the DUT interface and reference model interface.

- Scoreboard Class: This class tracks the record of failed and passed tests by checking whether the expected output matches the actual output or not. The expected output is taken from a reference model through monitor and since the DUT gives the actual output. The output data is taken from the DUT to DUT interface and then is fed to the scoreboard for comparison with the expected output through the monitor as shown in Figure 4.2.

- Assertions: Assertions in test bench are used to verify the correct functionality of the design. It checks whether a particular sequence of required events is followed or not or it checks whether a specific condition is satisfied or not. So, a flag like warning or error is raised whenever a specific condition or sequence of events is followed. So, a list of assertions is made to list all the rules of the CSI-2 protocol and then these assertions are implemented and verified through negative testing. Incorrect scenarios are created for negative testing. Thus assertions play an important role in identifying bugs in a design.

- Coverage Class: Coverage class checks how much functionality of design has been covered. For example, if there are 20 features in a protocol and if all the 20 features are mentioned and written in coverage code. If only 17 features are covered through all the tests

that mean 85% coverage has been achieved i.e. more test cases are required to cover all the important features.

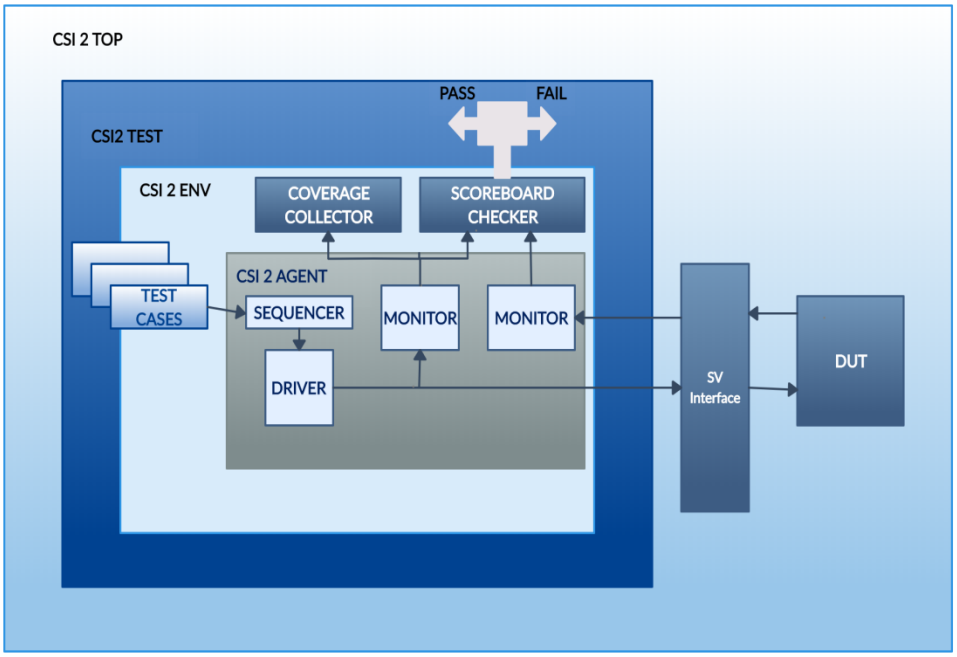


Figure 4.2 Improved Verification Architecture of CSI-2

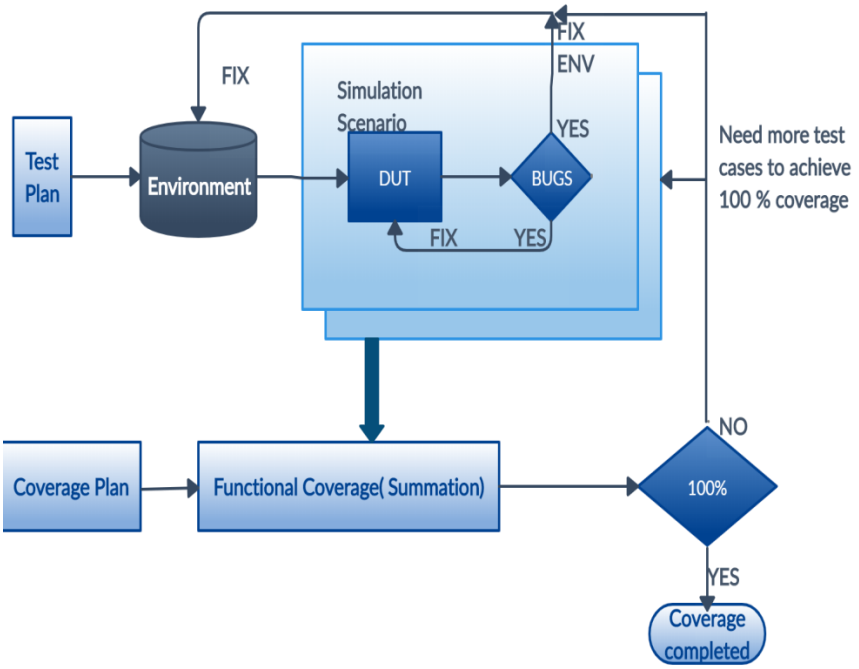


Figure 4.3 Flow diagram of coverage and bug fixing

So, the first step for verifying all the functionality is to make the test plan which consists of all test scenarios to be covered, assertion plan which consists of all the conditions of protocol to be checked and coverage plan which consists of all the features to be covered in testing. The process of achieving 100% coverage and bug fixing is illustrated in Figure 4.3.

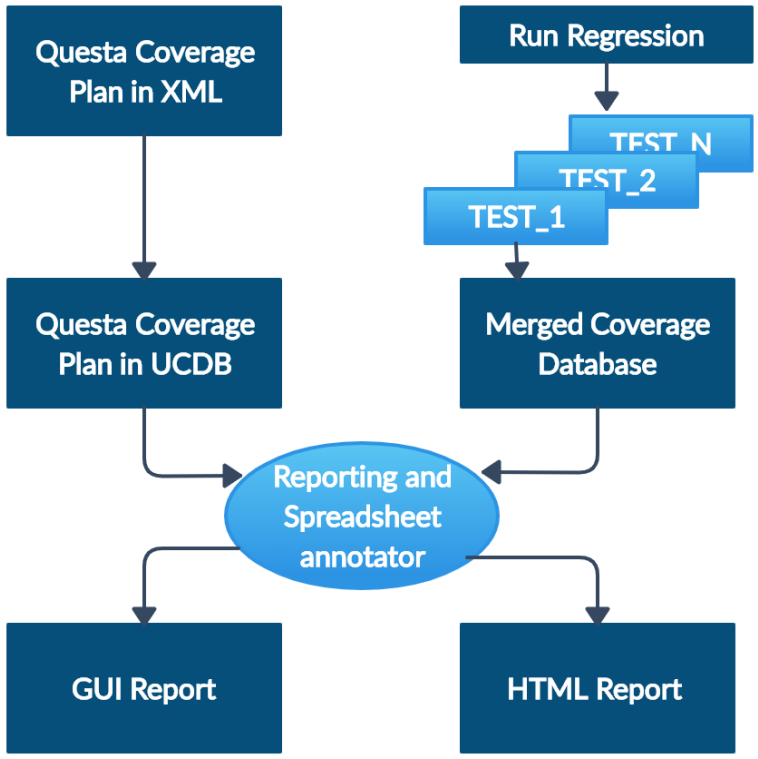


Figure 4.4 Flow for coverage collection using Questa

First, all the test scenarios are created according to the test plan and their functionality is checked through scoreboard and for checking that the design works only under some specific condition or specific sequence of events, negative tests are made using the assertion plan. When all the tests are executed, bugs are identified and fixed. After bug fixing all the tests are run in a regression (in one go) through which coverage result is calculated according to the coverage plan and coverage class defined. If

the total coverage does not come out to be 100 percent, then more scenarios and tests are generated for generating more stimulus which can cover the remaining features to achieve 100 percent coverage. The process/flow for collecting coverage through Questa sim is shown in Figure 4.4.

4.3. Comparison of existing and improved Architecture

In existing architecture, there was no automatic system to detect failure and success of a test, it had to be decided by the verification engineer by looking at GUI report and comparing the actual output with expected output manually.

In improved architecture, the scoreboard is added in combination with coverage and assertion-based verification model for improving the efficiency, easy debugging and clean implementation of the design.

By including scoreboard class, a reference model calculates the expected output using input transactions and scoreboard compares it with the actual output from DUT thus giving the result as success or failure of the test. Thus scoreboard provides the following advantages:

- Improved Efficiency
- Improved Accuracy
- Reduced Time
- Easy debugging
- Clean Implementation

Limitations of using scoreboard:

- Time consumed in implementing a reference model.

In existing architecture, there was no coverage class to check the progress of the functionalities executed. So the progress tracking was not very accurate and identification of missing functionalities or missing corner case testing was difficult.

In improved architecture, coverage class has been included to check the progress of functionalities executed in a more efficient way. Coverage plan is made, and coverpoints are written according to by defining how many scenarios need to be covered for a particular point. So, while executing test cases, the tool automatically checks how many functionalities have been covered by using coverage plan and comparing it with how many bins and coverpoints have been hit by all the tests. Thus any missing corner case or missing scenario or functionality in the design can be easily identified. Thus adding coverage class provides the following advantages:

- Accurate progress tracking of functionalities executed.
- Easy identification of any missing functionality.
- Easy debugging by identification of any missing scenario or corner case testing.

Limitations of coverage are:

- If the user does not make an exhaustive coverage plan, important features and test cases can be missed.

Chapter 5

5. Results

5.1. C-PHY over D-PHY

Since C-PHY uses the mechanism of differential inputs with clock embedded in each 3 wire lane and a symbol encoding mechanism for converting 16 bits to 7 symbols, it transmits 2.28 bits/symbol thus improving:

- Transmission Efficiency
- Data Rate
- Requires less area
- Reducing Power

A comparison of D-PHY and C-PHY for a bandwidth of 6 Gbps is shown in Table 5.1 and Table 5.2.

Physical Layer	Bandwidth	Number of Lanes	Number of Wires	Data Rate
D-PHY	6 Gbps	6	14	1 Gbps
C-PHY	6 Gbps	3	9	0.875 Gbps

Table 5.1 Comparison of C-PHY and D-PHY

Percentage reduction in the number of lanes	Percentage reduction in the number of wires	Percentage reduction in data rate
50%	57%	12.5 %

Table 5.2 Improvement by using C-PHY

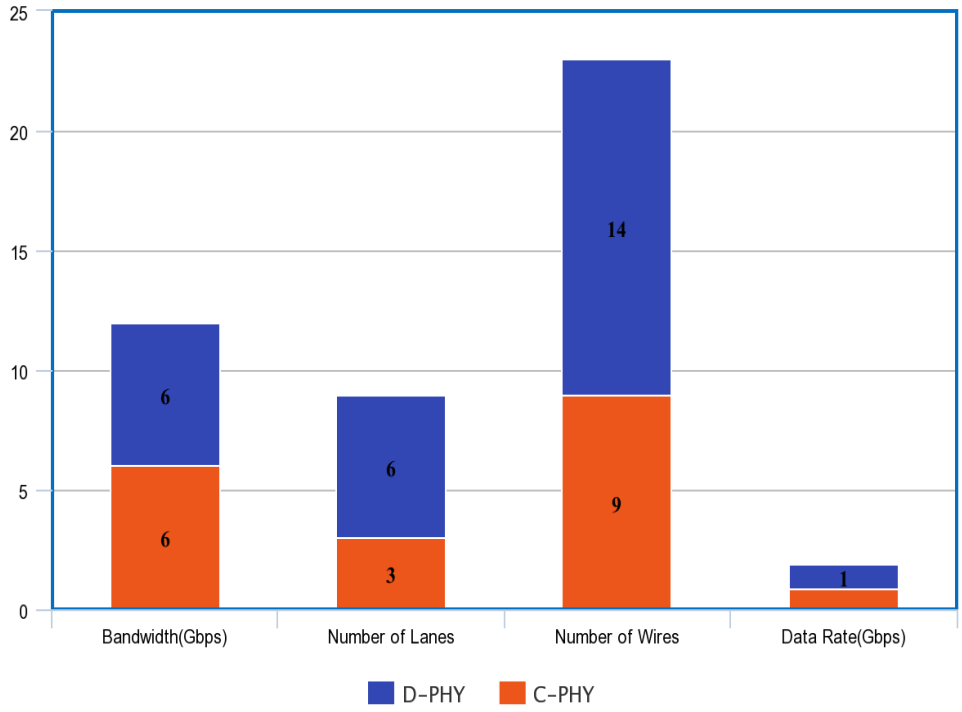


Figure 5.1 Comparison Graph of C-PHY and D-PHY

Thus for the same bandwidth, the number of data lanes is reduced by a percentage of 50%, the number of wires are reduced by 57 % and data rate is reduced by 12.5 % in C-PHY. The comparison graph of C-PHY and D-PHY is shown in Figure 5.1.

5.2. LRTE-EPD over HS-LPS-HS packet spacing

The comparison of LRTE-EPD with LPS spacing for an image sensor supporting 5.3 Megapixel (2592 x 2056 pixels), with 60 frames per second and 12 bits transmitted per pixel and C-PHY having a speed of 3.5 Gbps per lane is shown in Table 5.3 and 5.4.

As, it can be seen from the comparison graph illustrated in Figure 5.2 that while using LRTE EPD, more number of sensors can be supported and it reduces the packet delimiter overhead also.

Considerations	**CSI-2 with LPS**	**CSI-2 with LRTE**

	spacing	
Legacy Packet Delimiter Horizontal Row Overhead	27.7 %	0.20 %
Maximum Sensors Supported	4	6

Table 5.3 Comparison of CSI-2 with LPS spacing and CSI-2 with LRTE

Considerations	**CSI-2 with LRTE over C-PHY**
LRTE PDQ Frame Transport Efficiency Impact	27.5 %
LRTE PDQ Additional Supported Sensors	33.3 %

Table 5.4 Improvement due to LRTE EPD

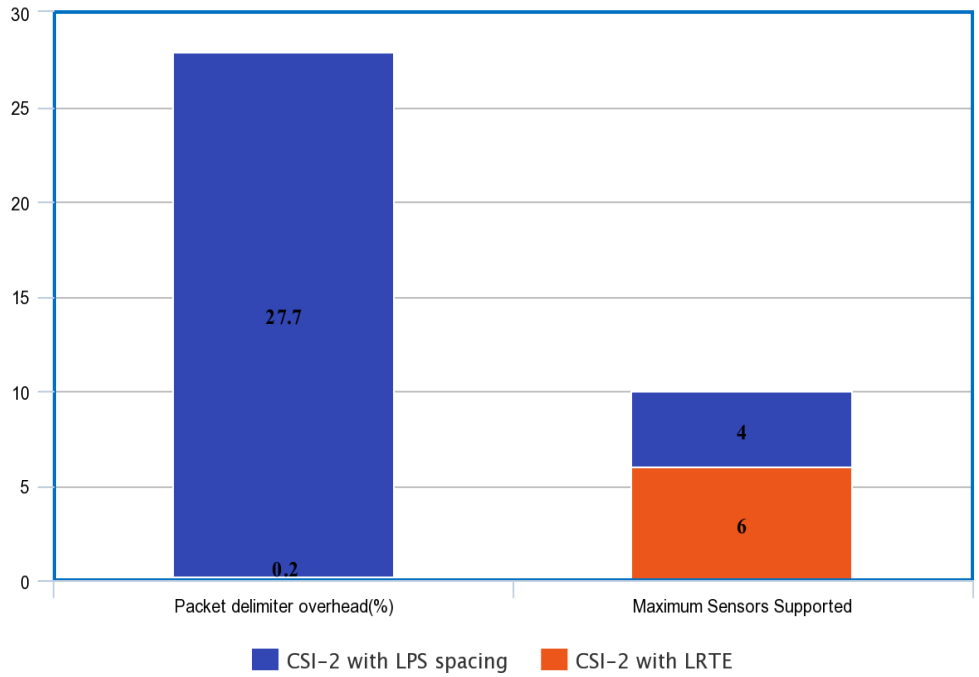


Figure 5.2 Comparison Graph of CSI-2 with LPS and LRTE

So, by using CSI-2 with LRTE EPD can improve frame transport efficiency by a percentage of 27.5 % and 33.3 % of additional sensors can be supported.

Thus by using LRTE EPD instead of LPS packet spacing has the following advantages

- Improved Transport Efficiency
- Additional sensors supported
- Latency Reduction

5.3. SROI (Smart Region of Interest)

5.3.1. Horizontal Overlapping and Blanking

Figure 5.3 Input image for horizontal overlapping and blanking

For verifying horizontal overlapping and blanking algorithm, input image as shown in Figure 5.3 is used where some region in ROI 1 and 2 depicts

horizontal overlapping and the overlapped region must be transmitted only once and ROI 2 and 3 depicts horizontal blanking and the unnecessary region between 2 and 3 should not be transmitted and the same can be shown in output image illustrated in Figure 5.4.

Figure 5.4 Output image for horizontal overlapping and blanking

5.3.2. Vertical Overlapping and blanking

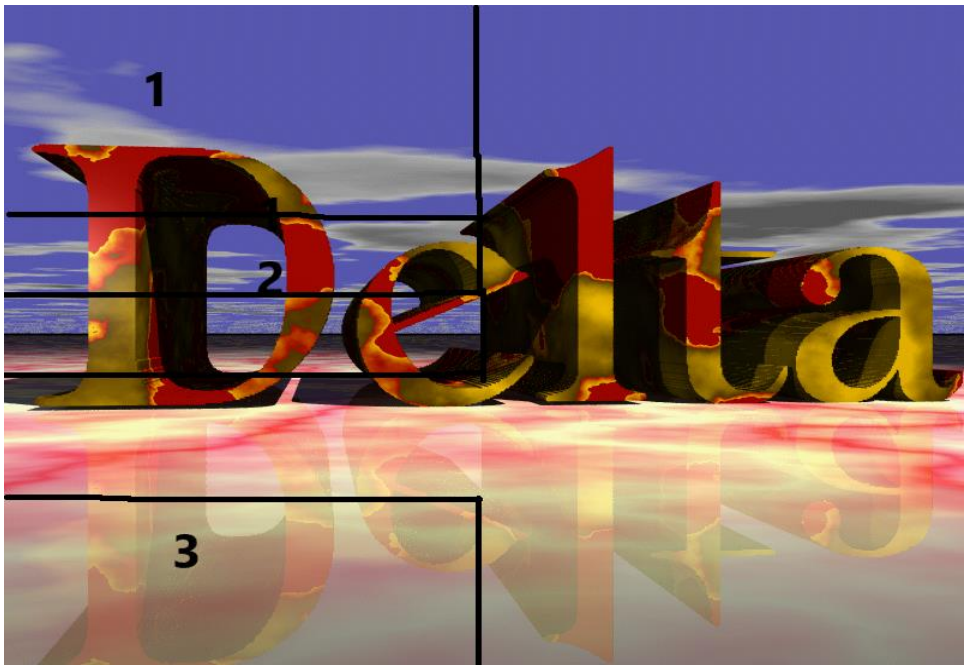


Figure 5.5 Input image for vertical overlapping and blanking

For verifying vertical overlapping and blanking algorithm, input image as shown in Figure 5.5 is used where ROI 1 and 2 depicts vertical overlapping and the overlapped region must be transmitted only once and the ROI 2 and 3 depicts vertical blanking and the unnecessary region between them must be removed and the same can be seen from the output image as shown in Figure 5.6.

Figure 5.6 Output image for vertical overlapping and blanking

5.3.3. Horizontal and vertical overlapping

For verifying the combined algorithm of horizontal and vertical overlapping and blanking, input image as shown in Figure 5.7 is used where ROI 1 and 2 depict both horizontal and vertical overlapping where the overlapped region between them must be transmitted only once. ROI 3 and 4 depict both horizontal and vertical overlapping and the same can be shown from the output image as shown in Figure 5.8.

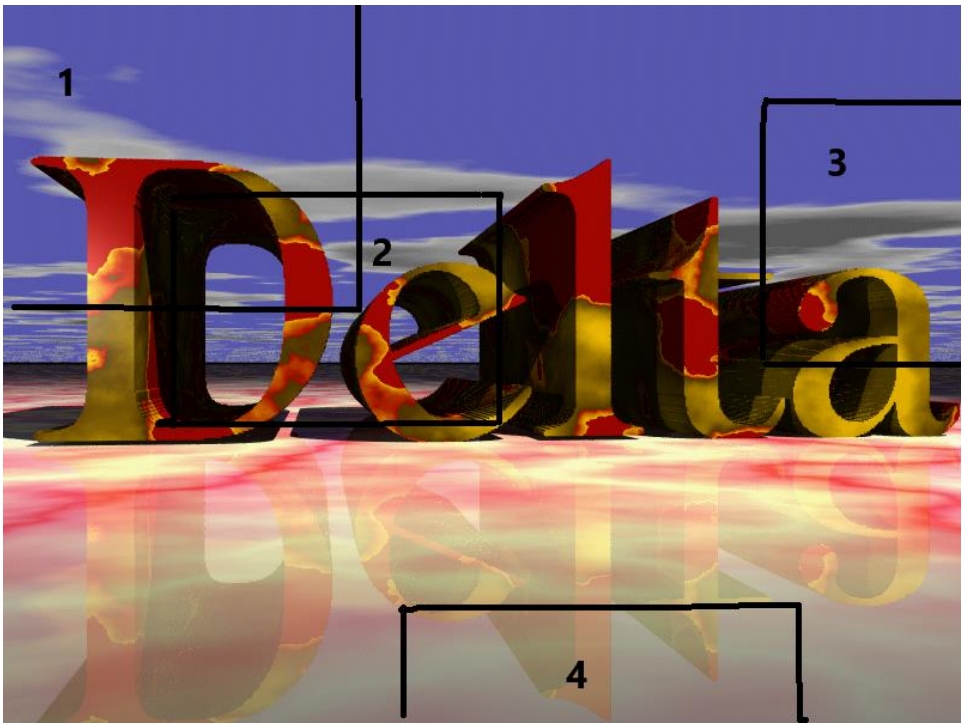


Figure 5.7 Input image for horizontal overlapping

Figure 5.8 SROI Image transmitted

By using the existing system, instead of sending only the regions of interest, the complete image was transmitted.

Thus, by using SROI feature, the regions of interest are extracted using their X, Y, height and width coordinates of the ROIs. After identifying these regions, only ROIs are transmitted with the overlapped region transmitted only once.

Thus using SROI reduces the amount of data, thus providing following advantages:

- High Frame Rate
- High Resolution
- Low Power Dissipation

5.4. Improved Verification Architecture over Existing System

The improved verification architecture uses scoreboard in addition combined with coverage and assertion model to provide following advantages:

- Improved Efficiency
- Improved Accuracy
- Easy Debugging
- Accurate progress tracking of functionalities executed

Randomization technique is used to create more diverse test cases and creating corner case scenarios for more efficient bug tracking. Following bugs have been tracked by this system:

- Incorrect design behavior at low frequencies of skew calibration.
- Incorrect Pixel to byte mapping for RAW 24 data type.
- Invalid assertion firing for RGB 666 and RGB 444 data type.
- Invalid Line synchronization.
- Incorrect design behavior at PPI in alternate calibration mode.
- Maximum Range of skew calibration was not supported.

- Incorrect assertion was fired for using 16 virtual channels
- Incorrect recognition of data at the receiver side while using EPD at 16-bit bus width
- Invalid Frame Start recognition at the receiver side for EPD at 32-bit bus width
- Incorrect packing of bytes in 16 and 32-bit bus width D-PHY
- Invalid assertion firing for SROI for using different length of lines in a frame

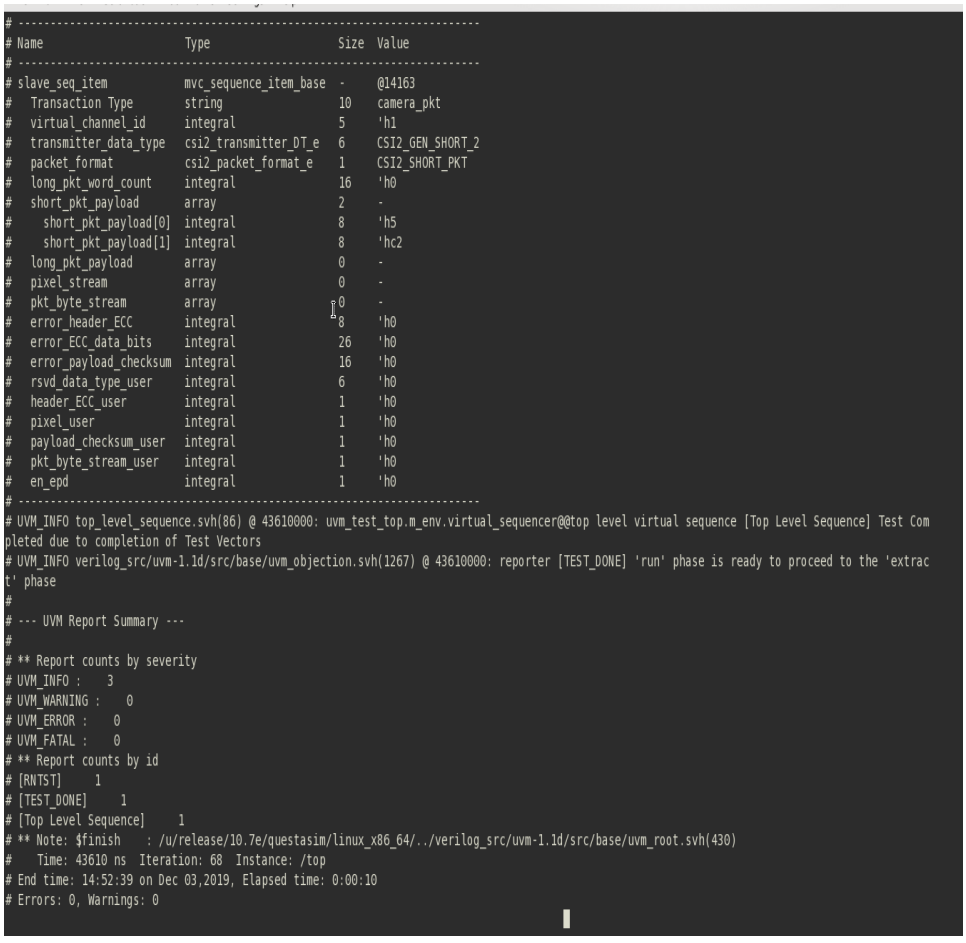


Figure 5.9 Scoreboard output

The scoreboard output which prints the compared input and output and thus giving the output as failed or passed test is shown in Figure 5.9. An

illustration for firing of assertion when an invalid line number is set during transmission is shown in Figure 5.10.

```
# ** Error: (vsim-60041) MVC @ 13316000 ps: /top/csi2_if_peripheral_device ca
mera_pkt: MGC_CSI2: CSI2_TX_SHALL_SEND_INCR_LINE_NUMBER_FOR_SAME_VC_LS - Line
 number increments by one for every LS packet within the same Virtual Channel
 and the same Data Type. The line number is periodically reset to one for the
 first LS packet after a FS packet. # see CSI2 Protocol Specification 2.0 Sec
tion : # 9.8.v2.
#  Line virtual channel id = 0b00000
#  Last Line number        = 0b0000000100000000
#  Incremental value       = 0
#  Observed current Line number = 0b0000000000000011
#  Expected current Line number = 0b00000000000000000000000100000000
```

Figure 5.10 Assertion for invalid line number

Total 41 new assertions have been listed out and 52 coverpoints have been added in coverage class, thus achieving 100 % coverage using 34 test cases as shown in Figure 5.11.

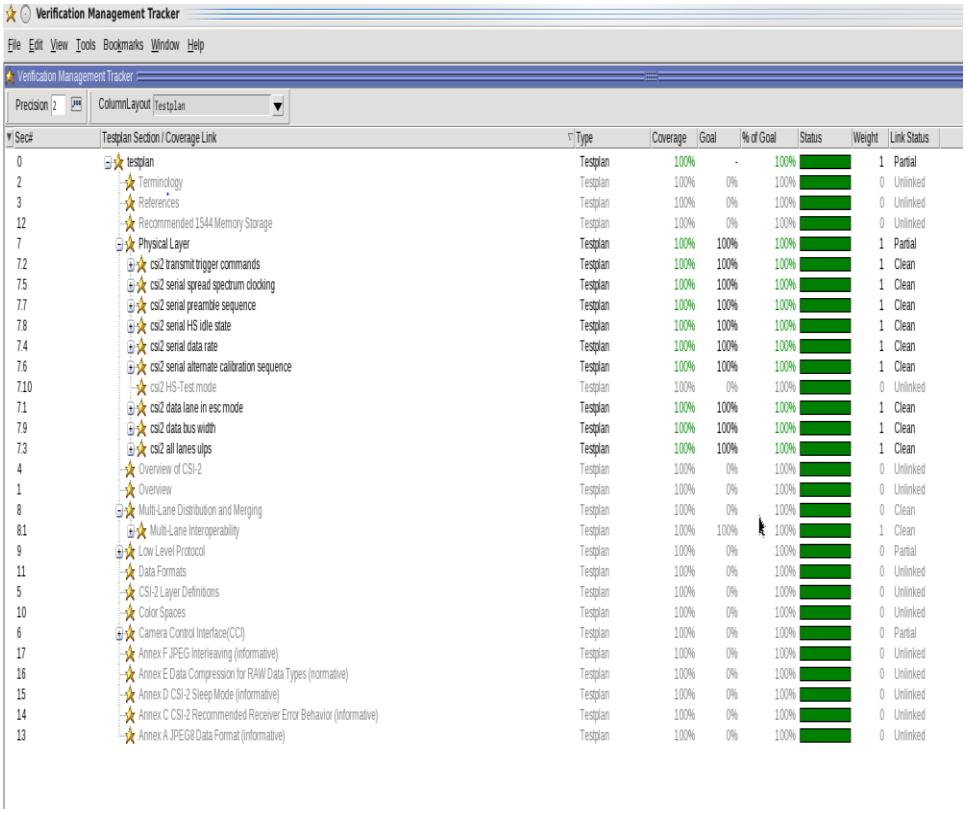


Figure 5.11 Coverage Result

Chapter 6

6. Conclusion and Future Work

6.1. Conclusion

This thesis focuses on improving the existing functionalities of CSI-2 and developing an efficient verification and bug tracking system using System Verilog and System C. The following enhancements have been introduced in the design.

- D-PHY was used in the existing design which reduces data rate and consumes more power. So, C-PHY has been introduced as an alternative PHY layer which improves the data rate, transport efficiency and power of the design.
- The LPS packet spacing was used in the existing design which requires Low Power – High-speed – Low power transition thus increasing the packet delimiter overhead and reducing transport efficiency. So a new method is developed for packet spacing which uses EPD spacers instead of HS-LPS-HS packet delimiter for packet spacing which improves transport efficiency and reduces latency.
- SROI feature has been introduced for the adaptive transfer of rectangular regions of interest instead of transferring the complete image which results in data reduction, high frame rate, reduced power and high resolution of the image.
- The existing Verification architecture has been improved for increasing efficiency and accuracy for bug tracking of design.

6.2. Future Work

- An additional feature can be introduced which can replace CCI as a controlling interface that can reduce the number of interface wires.
- The algorithm for calculating SROI pixels can be improved by reducing time complexity.
- An algorithm can be introduced for classifying the type of bugs and fixing them automatically.

REFERENCES

[1] Specification for Camera Serial Interface 2 (CSI-2SM) Version 1.1 09 November 2010.

[2] S. Fine and A. Ziv, "Coverage directed test generation for functional verification using Bayesian networks," Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451), Anaheim, CA, 2003, pp. 286-291, doi: 10.1145/775832.775907.

[3] C. L. Lee, K. Hsiao and M. Chou, "A Low Power Mobile Camera Processor Design with SubLVDS Interface," 2006 International Symposium on VLSI Design, Automation and Test, Hsinchu, 2006, pp. 1-4, doi: 10.1109/VDAT.2006.258111.

[4] K. Lim, G. S. Kim, S. Kim and K. Baek, "A multi-lane MIPI CSI receiver for mobile camera applications," in IEEE Transactions on Consumer Electronics, vol. 56, no. 3, pp. 1185-1190, Aug. 2010, doi: 10.1109/TCE.2010.5606244.

[5] Y. Lu, Z. Chen and P. Chang, "Low power multi-lane MIPI CSI-2 receiver design and hardware implementations," 2013 IEEE International Symposium on Consumer Electronics (ISCE), Hsinchu, 2013, pp. 199-200, doi: 10.1109/ISCE.2013.6570183.

[6] Beom-Dae Kim, Sang-Jin Lee, Doo-Hwan Kim and Kyoungrok Cho, "Design of a D-PHY chip for mobile display interface supporting MIPI standard," 2012 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2012, pp. 660-661, doi: 10.1109/ICCE.2012.6162016.

[7] H. Sabri, S. Khattak, B. Kapralos, K. El-Khatib and M. Guennoun, "Virtual Reality-Based Interface for the Control of Multiple Surveillance Cameras," 2007 IEEE International Workshop on Haptic, Audio and

Visual Environments and Games, Ottawa, Ont., 2007, pp. 76-79, doi: 10.1109/HAVE.2007.4371591.

[8] F. Yonga, C. Bobda and A. Zarazadeh, "Improving video communication in distributed smart camera systems through ROI-based video analysis and compression," 2012 Sixth International Conference on Distributed Smart Cameras (ICDSC), Hong Kong, 2012, pp. 1-6.

[9] B. Rinner and W. Wolf, "A Bright Future for Distributed Smart Cameras," in Proceedings of the IEEE, vol. 96, no. 10, pp. 1562-1564, Oct. 2008, doi: 10.1109/JPROC.2008.928741.

[10] J. V. E.V., P. V. and K. Balamurugan, "Design of Generic Verification Procedure for IIC Protocol in UVM," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 1146-1150, doi: 10.1109/ICECA.2019.8821815.

[11] A. Hunter, A. Piziali, A. Ziv, K. Larson and S. Hemmady, "Ensuring Functional Closure of a Multi-core SoC through Verification Planning, Implementation and Execution," 2008 Ninth International Workshop on Microprocessor Test and Verification, Austin, TX, 2008, pp. 7-13, doi: 10.1109/MTV.2008.20.

[12] A. M. Cruz, R. B. Fernández and H. M. Lozano, "Automated functional coverage directed for complex digital systems," 2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC), Playa del Carmen, 2014, pp. 155-156, doi: 10.1109/VLSI-SoC.2014.7004172.

[13] YANG Yawen , ZHOU Shan and KONG Lu, “Coverage Test Technology Based on ONESPIN Verification Platform” 2004 IOP Conf. Series: Journal of Physics: Conf. Series 1026 (2018) 012004, doi :10.1088/1742-6596/1026/1/012004

[14] A. T. Sonny and S. Lakshmiprabha, "OVL, PSL, SVA: Assertion based verification using checkers and standard assertion languages," 2013 International Conference on Advanced Computing and Communication Systems, Coimbatore, 2013, pp. 1-4, doi: 10.1109/ICACCS.2013.6938754.

[15] M. Siegel, A. Maggiore and C. Pichler, "Untwist your brain - Efficient debugging and diagnosis of complex assertions," 2009 46th ACM/IEEE Design Automation Conference, San Francisco, CA, 2009, pp. 644-647.

[16] M. Riazati, S. Mohammadi, A. Afzali-Kusha and Z. Navabi, "Improved Assertion Lifetime via Assertion-Based Testing Methodology," 2006 International Conference on Microelectronics, Dhahran, 2006, pp. 48-51, doi: 10.1109/ICM.2006.373264.

[17] H. El-Kharashy, M. Khami, A. Sala and M. Korany, "A novel assertions-based code coverage automatic CAD tool," IEEE EUROCON 2017 -17th International Conference on Smart Technologies, Ohrid, 2017, pp. 277-281, doi: 10.1109/EUROCON.2017.8011119.

[18] V. S. Rashmi, G. Somayaji and S. Bhamidipathi, "A methodology to reuse random IP stimuli in an SoC functional verification environment," 2015 19th International Symposium on VLSI Design and Test, Ahmedabad, 2015, pp. 1-5, doi: 10.1109/ISVDAT.2015.7208152.

[19] M. Ashwathanarayan and G. Jayakrishna, "A method/approach leading to controlled randomization in validation of an IP," 2017 12th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS), Palma de Mallorca, 2017, pp. 1-2, doi: 10.1109/DTIS.2017.7930165.

[20] S. Karlapalem and S. Venugopal, "Scalable, Constrained Random Software Driven Verification," 2016 17th International Workshop on

Microprocessor and SOC Test and Verification (MTV), Austin, TX, 2016, pp. 71-76, doi: 10.1109/MTV.2016.19.

[21] A. Hany, A. Ismail, A. Kamal and M. Badran, "Approach for a unified functional verification flow," 2013 Saudi International Electronics, Communications and Photonics Conference, Fira, 2013, pp. 1-6, doi: 10.1109/SIECPC.2013.6550753.

[22] N. Sheeley, N. Pena, I. Waheed and M. Nodine, "Enhancing Sequential LEC Using a Cumulative Verification Methodology," 2008 Ninth International Workshop on Microprocessor Test and Verification, Austin, TX, 2008, pp. 39-42, doi: 10.1109/MTV.2008.10.

[23] Specification for D-PHYSM Version 1.2 14 April 2014 Release 05.

[24] Yoo-Jin Kim, Doo-Hwan Kim, Seok-Man Kim, Kyoung-Rok Cho "Low Power Design of a MIPI Digital D-PHY for the Mobile Signal Interface" December 2010, DOI: 10.5392/JKCA.2010.10.12.010

[25] Yong-Hwan Lee, Hyun-Woong Ju "PHY Adapter Layer Design for Low-power fast serial bus protocol" January 2008

[26] Specification for D-PHYSM Version 2.1 15 December 2016.

[27] Specification for Camera Serial Interface 2 (CSI-2SM) Version 2.0 28 March 2017.

[28] Universal Verification Methodology (UVM) 1.1 Class Reference, Accellera Organization, June 2011

[29] Universal Verification Methodology (UVM) 1.1 User's Guide, Accellera Organization, May 2011

[30] System Verilog 3.1a Language Reference Manual, Accellera Organization, 2004.