DEVELOPMENT OF SWITCHING OPERATIONS BETWEEN MIPI CAMERAS USING A ZYNQ-7000 FPGA FOR STEREO VISION APPLICATIONS

By
Mowly Krishnamoorthy, BASc
Queen's University, Kingston, Canada, 2004

A Major Research Project
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Engineering
in the program of
Electrical and Computer Engineering
Toronto, Ontario, Canada, 2025

Toronto, Ontario, Canada, 2025

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF AN MRP

I hereby declare that I am the sole author of this MRP. This is a true copy of the MRP, including any required final revisions.

I authorize Toronto Metropolitan University to lend this MRP to other institutions or individuals for the purpose of scholarly research.

I further authorize Toronto Metropolitan University to reproduce this MRP by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my MRP may be made electronically available to the public.

Acknowledgements

This project would not be possible without the inspiration and vision of my supervisor Dr. Lev Kirischian to whom I would like to express my deep gratitude. Many thanks to the members of the review committee for their participation and consideration.

I owe a huge debt to my dear friend [redacted] for sitting with me in the trenches for hours on end to test, debug and discuss about the project.

A heartfelt gratitude towards [redacted] who has been unequivocally and unapologetically by my side seeing this odyssey through with me. I know you do not understand what I did, but that motivated me to make this report as comprehensive as possible so that one day you might!

Abbreviations

[]

ADAS Advanced Driver Assistance Systems. 1

ADC Analog-to-Digital Converter. 12, 16, 32, 69

ASIC Application Specific Integrated Circuit. 1

ASP Application Specific Processor. 1

AWB Auto White Balance. 31

bpp bits per pixel. 9, 14, 16, 40

BRAM Block RAM. 19

CCI Camera Control Interface. ix, 8, 9

COTS Commercial-Off-The-Shelf. 2, 19, 35

CPU Central Processing Unit. 1, 18

CSI Camera Serial Interface. vii, ix, 3, 8–11, 13–17, 20, 21, 23, 28, 31, 36, 37, 58, 66

CV Computer Vision. 1

D-PHY . viii–xii, 3, 5, 9–12, 15–17, 20, 21, 36, 37, 40, 54, 55, 59, 63–68

DDR Double Data Rate. 10, 36, 40

DMA Direct Memory Access. 3, 19

DRPL Data Rate Per Lane. 14, 15, 40

DSI Display Serial Interface. 8, 9

DSP Digital Signal Processor. 1, 19

DUT Device-Under-Test. 54

EMI Electromagnetic Interference. 8

EMIO Extended MIO. 20

EoT End of Transmission. 9, 15, 16

ERSL Embedded Reconfigurable System Laboratory. 2

FF Flip-Flops. 19

FFC Flat Flexible Cable. x, 21, 25, 27, 28, 31

FIFO Fist-In First-Out. 3

FPGA Field Programmable Gate Array. vii, ix, x, xiii, 1–5, 7–9, 11, 16, 17, 19–22, 25–27, 31, 35, 37, 42, 43, 46, 47, 52, 56, 58, 59, 63

FPS Frames per second. 11, 13, 14, 18, 31, 32, 39, 69

GPIO General Purpose IO. ix, 19–25, 27, 31, 32, 41, 42, 45, 46, 58

GPU Graphics Processing Unit. 1

HBI Horizontal Blanking Interval. 13, 14, 39

HBP Horizontal Back Porch. 13, 14

HFP Horizontal Front Porch. 13, 14, 16, 39

HFR High Frame-Rate. 1, 18

HS High-speed. viii–xi, 2, 8–10, 12, 17, 21, 22, 54, 63–68

HSYNC Horizontal Sync Pulse. 13, 14, 16, 39

IC Integrated Circuits. ix, 1–3, 13, 21, 23–25, 27, 58

ILA Integrated Logic Analyzer. x, 28, 35, 56–58

IP Intellectual Property. x, 1, 9, 17, 35–37, 41, 54

IPI IP Integrator. x, 35, 38

ISP Image Signal Processor. 5, 8, 12, 31–33, 56

LLP Low-Level Protocol. 9

LP Low-power. viii–x, 2, 9–12, 15, 16, 22, 63–68

LSB Least Significant Bit. 16, 54

LUT Look-Up Tables. 19

LVCMOS Low-Voltage CMOS. 9, 17, 19, 21

LVDS Low-Voltage Differential Signaling. 19, 21, 63, 64

MAC multiply-accumulate. 1

MCA Multi-Camera Adapter. ix, 3–5, 21, 23–25, 27, 28, 31, 33, 35, 40–42, 46, 58

MIO Multiplexed IO. ix, x, 20–22, 42, 46, 47, 59

MIPI Mobile Industry Processor Interface. vii, ix, xii, 2–5, 7–12, 14, 16, 17, 20–23, 31, 35–37, 58, 63, 65

MSB Most Significant Bit. 16, 31

OS Operating System. 3, 4, 56

PCB Printed Circuit Circuit. 2, 3, 7

PCLK Pixel Clock. 12–14, 40

PL Programmable Logic. x, 2, 5, 17, 19–21, 31, 35, 38, 40

PLL Phased Lock Loop. 13, 56

Pmod Peripheral module. ix, x, 21–23, 33, 41, 42, 46, 47, 58, 59

PPI PHY-Protocol Interface. 11, 36, 37, 54

PS Processing System. 2, 17, 19–21, 35–37, 40, 46, 56, 58, 59

RPi Raspberry Pi. ix, 21, 23–27, 29, 31

RTL Register-Transfer Level. 65, 66

SLVS Scalable Low-Voltage Signaling. 11, 17, 21

SoC System-On-Chip. vii, xii, 2, 3, 18–22, 35, 59

SoT Start of Transmission. viii, 9, 16, 67

TSP Taylor Series Polynomials. 7

VBI Vertical Blanking Interval. 14, 39

VBP Vertical Back Porch. 14, 39

VDMA Video DMA. 36

VFP Vertical Front Porch. 14, 16, 39

VHDL VHSIC Hardware Description Language. xiii, 36, 43, 54

VSYNC Vertical Sync Pulse. 14, 16, 31, 39

VTG Video Timing Generator. 36

Abstract

With the increase in demand for higher resolution mobile cameras, serial interfaces have replaced the conventional parallel interfaces in the chip-to-chip communications. MIPI is the de facto, industry-wide serial standard for mobile cameras optimized for high-bandwidth and low-power consumption. Since this standard is focused for commercial use, this project details the system integration and implementation using off-the-shelf products designed using MIPI specifications, for stereo vision applications. The scope of this project includes analysis of possible approaches and theory, selection and integration of products, firmware/software modifications of the vision processing and control components and demonstration of the experimental hardware setup. The goal is to develop a system for switching between at least two 5 MP cameras with a resolution of 1080p at 30 FPS. Typically, the switching operation for cameras is accomplished with applications running on a CPU. This project uses a hybrid architecture, that makes use of the synergy between CPU and FPGA for a SoC-based solution that realizes switching, both in hardware and software, for future stereo vision applications.

Keywords – FPGA, SoC, Stereo Vision, MIPI, D-PHY, CSI, Multi-Camera

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Computer stereo vision, one of the best researched areas in Computer Vision (CV), has been around since the 1970s and its ever evolving. Its a passive type of technology where no extra light source, apart from ambient light, is required for performing measurements. Contrary to Active Cameras, stereo vision has advantageous use cases in outdoor, long-range and depth sensing applications, not limited to Advanced Driver Assistance Systems (ADAS) and autonomous mobile robots. Despite their key advantages, high computational demand, especially for High Frame-Rate (HFR) and real-time processing requirements has prevented their widespread adoption.

Traditional CPUs struggle with vast computations required by stereo vision because of the usual fetch-decode-execute cycle architecture and can only demonstrate simpler algorithms like block matching at typical camera frame rates. Perhaps hardware that bypasses CPU architecture with parallel computations, like GPUs, can perform better and they do, but at the cost of high power consumption and portability. For comparison, the GeForce GTX TITAN X by NVIDIA requires upto 250 W akin to power used by a mini fridge, which limits the reach of any exploration regarding GPU based stereo vision.

Application Specific Integrated Circuits (ASICs) are best in terms of performance, unit cost and power consumption but they require long development cycles from initial design to final product and specialized equipment and expertise, all rather discouraging for proof-of-concept or feasibility endeavours. Many of today's applications rely on stream processing like audio, video, communication protocols, etc., and it becomes very difficult to obtain the desired performance without exploiting inherited parallelism of the algorithms in applications. Digital Signal Processors (DSPs) are efficient for functions within the application set due to their MAC style operations but cannot be practically deployed as the sole platform for stereo vision applications[1].

FPGA, generic IC with reconfigurability, are positioned well to balance performance with cost efficiency and power consumption. The top down approach is to make an application specific architecture that divides the problem into many small sub-problems solved in parallel through Application Specific Processors (ASPs) that have minimal power consumption. The main disadvantage is that the programming effort vastly exceeds that of CPUs and GPUs incentivizing many digital imaging companies to develop closed-source (IPs) specifically using technology from semiconductor manufacturers they sign an agreement with. This hardware engineering remains in the private domain restricting public support from Vendors, production-ready open source IPs and hardware documentation for FPGA implementations.

In academic research, most image transfer and configuration mechanisms between sensors and FPGA

are custom designed for the application including previous works at my lab, Embedded Reconfigurable System Laboratory (ERSL). Since they are custom built, there is limited interoperability with other FPGAs and image sensors. New driver software that communicate with the new sensor firmware should be created. Thus, software needs to be modified when hardware is upgraded. Standardized protocols and interfaces help overcome this incompatibility issue to a great degree. MIPI is an industry wide standard serial communication protocol and interface promoted by the MIPI Alliance, popular in mobile phones and Tablet computers for their fast image transfer rates within short distances between cameras and displays to the processor. The idea of interoperability using industrial proven standards underpins the exploration of possible solutions for stereo vision in this project.

1.1 Motivation

Considering the barriers and with the purpose of increasing interoperability, is it possible to create a stereo vision system using Commercial-Off-The-Shelf (COTS) devices and MIPI? This is the question this report seeks to answer.

MIPI is particularly good for HS, LP and compact data transmissions with less overhead than USB3. Unlike other HS interfaces that are hot-pluggable or plug-n-play (like HDMI, USB), MIPI interfaces are meant for fixed and permanent connection in a final design like in FPGAs. Since there is widespread adoption of MIPI in laptops, automotive systems, drones, robotics, virtual reality, IoT devices, medical imaging and predominantly in smartphones and tablets, its a natural choice to be implemented in FPGA. A distinction needs to made between FPGAs and FPGA boards: The former an IC housing the PL and possibly the Processing System (PS) while the latter is a packaged FPGA on a Printed Circuit Circuit (PCB) fitted with various I/O. Stereo vision requires two cameras and lower end FPGA boards rarely are equipped with more than a single MIPI port. Thus, a mechanism to switch the cameras is needed. Table 1.1.1 below shows the main options.

Method	Best for	Limitations
Hardware Switching	Low-resource FPGAs, simple system	Limited switching speed
Time-Driven Multiplexing	Alternating between cameras	Frames may be dropped if cameras are unsynchronized
Parallel Processing	Real-time dual camera processing	High resource usage
External Bridge	MIPI-based systems, SoC integration	Requires extra hardware
Software Switching	FPGA + CPU systems (i.e., Zynq)	Higher latency

Table 1.1.1: Comparison of Camera Switching Methods

Time-driven multiplexing approach has both cameras connected to the FPGA, but only one camera's

data is processed at a time. The FPGA uses an internal multiplexer to select between the cameras. FPGA boards do not come with two ports and therefore will not be pursued. Parallel processing is ideal but warrants a custom solution where the dual camera streams are captured simultaneously using dedicated image processing pipelines on an intermediary board and sent to FPGA where they are both processed simultaneously. Direct Memory Access (DMA) or Fist-In First-Out (FIFO) buffers needed to store the images and FPGA's internal logic will merge them or select frames based on an algorithm for processing. These operations significantly increase resource usage that many entry-level FPGAs cannot support. Therefore, this option is avoided. External bridge IC chips (i.e. MIPI aggregrators) that combine multiple streams into a single stream to the FPGA along with synchronization signals, are another option, but they need to be integrated onto a PCB at a higher system cost which is not ideal. The best candidates for a proof-of-concept design using affordable FPGA boards are Hardware and Software Switching at the expense of higher performance penalties.

1.2 Objectives

Objective of the project is to investigate efficiency of time-multiplexing (switching) of multiple video-sensors (video-cameras) equipped with MIPI interfaces using FPGA-based SoC and determine most effective Software/Hardware co-design. The constraints for the project are as follows:

- Industry-wide standard to be used for image transfer
- Use of commercially available devices to maximize interoperability
- Usage of affordable or entry-level FPGA development boards to minimize system costs
- To prioritize low-power consumption and to limit additional latency, only bare metal designs (i.e no Operating System (OS)) are considered

Given these constraints, the primary task is the successful demonstration of a multi-camera video imaging system using off-the-shelf components. This task covers a wide range of system integration challenges because these components are untested for FPGA use and have limited hardware documentation. The main tasks are:

- Detailed analysis of the MIPI Standard with a focus on D-PHY and CSI-2 protocols at the receiver side
- Selection of hardware components (cameras, FPGA board, MCAs)
- System integration, hardware implementation and test runs of proposed designs
- Preliminary simulation analysis and verification of the working design

The problem areas are identified and broken down into three broad categories that need to be solved for a successful design.

- Image Sensor
  - How to properly configure the sensor?
  - Show successfully configuration with direct connection of a single sensor to FPGA
- MCA
  - What are its sub-components and how to access them via FPGA?
  - How to locate and identify the image sensors when they are connected?
  - How to configure all of the sensors?
  - Find out the camera selection process via FPGA
- Hardware/Software Co-Design
  - Study available hardware and software designs in order to adapt them to this project's requirements
  - How to test and debug the modified designs and verify their operation?
  - Analyze resource and power usage

1.3 Original Contributions

In the original inception of this project, object tracking using a stereo vision MIPI system was the target. As far as is known, no other research, at least publicly, is undertaken with this objective. This foray into unchartered territory presented unforeseen technical setbacks in camera configuration and system integration, that led to rescaling of the original objective.

Since the commercial landscape is quite prohibitive when it comes to hardware information sharing on manufacturers' respective products as it is their 'geese that lays the golden eggs', a lot of time was spent on reverse engineering. They were happy to share the software resources but that meant integrating an OS in addition to other software tools like OpenCV [1], all on the FPGA board, making that approach in violation of the constraints. Using OS abstracts away how hardware is wired and also provides better integration and updates of evolving image processing capabilities, effectively sustaining their business models even with new hardware products. However, due to passage of time, the older versions of MIPI are released and there are some designs that implement them. However none of these

[1] https://opencv.org/

designs are ever used for a multi-camera, bare metal setup. Understanding this landscape was part of the steep learning curve that consumed ample time.

Being aware of the dynamics at play, original contributions begin with analyses of stereo vision principles, MIPI Standard, image sensor basics and D-PHY transmitter packetization details. Much of the low-level information regarding the image transfer is scattered and often glossed over. The explanation given here is pieced together over many sources.

Implementing low-level circuitry for image transport and performing basic Image Signal Processor (ISP) functions in the PL from multiple cameras requires a capable FPGA. An overview of the FPGA chip along with the development board is presented. Following that, detailed reasonings for finding the I2C configuration path and process through MCA is discussed. Next, various camera modules selected for the proposed designs are investigated and settled on a pair that works.

Once the final design is decided, detailed analysis of the implementation is presented for the surgical modifications required for the hardware and software switching to work. Snapshots of the successful demonstrations are included.

Simulation verification of the D-PHY module used, is presented along with preliminary analyses of resource usage and power consumption of the final design are discussed. The latency of that module is detailed with illustrations. Finally, a recap of the objectives achieved, challenges faced and problems overcome and future applications are discussed.

1.4 Report Organization

The remainder of this report is organized as follows. Chapter II delves into the theory and technical background required to understand the serial-based technology used in this project along with some fundamental stereo vision principles. Image sensor basics is also covered.

Chapter III explores the proposed approach for the system design and integration. The challenges include finding the means of communication to control the switching architecture laid out in designs at the end. Many diagrams and illustrations are used to convey the progression of this project.

Chapter IV delves into the finer details of both the software and hardware components used in this project. Although these systems are designed by others, integrating them into a successful design meant a thorough grasp of the underlying components for surgical changes.

Chapter V is a discussion of preliminary analysis of the working system. This project is meant to prove this design works. Optimization and streamlining are future goals.

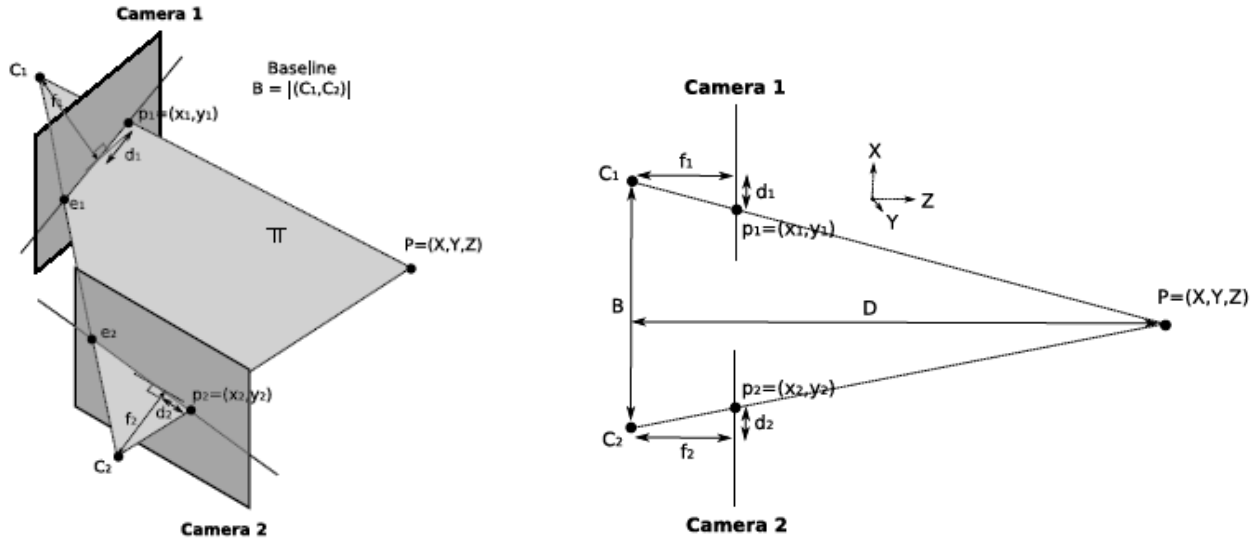
Finally, a recap of the project is presented in Chapter VI where a summary of the main results from the project is provided.

Chapter 2

Theory & Background

2.1 Principles of Stereovision

Nature uses vision to effortlessly detect, identify, and track objects in a three-dimensional (3D) world. One of the most important aspects of this vision is stereo imaging: a 3D scene projected onto two individual eyes from different viewpoints. These viewpoints are used by the brain to reconstruct the original 3D scene. But why two eyes? For depth perception. Similarly, two identical cameras displaced a distance, called Baseline B in Figure 2.1.1, are used to reconstruct 3D scenes. This setup is known as Simple Stereo or Binocular Vision.



(a) Generalized Stereo **(b)** Rectified Stereo

Figure 2.1.1: A binocular stereo camera system

Once these two cameras are calibrated, a single point of projection in the image of left camera and its *corresponding* point in the right one will produce 2 outgoing rays out of p_1 and p_2 in Figure 2.1.1a, respectively. The location where these rays intersect in a global coordinate system is known as the Scene Point, P. The plane created by the aforementioned points is known as epipolar plane, π. This plane, π contains the Scene Point and corresponding centers of the two cameras, known as Principle Points c_1 and c_2 and epipolar lines (e_1,p_1) and (e_2,p_2) in Figure 2.1.1a. This is known as Stereo Correspondence, where knowing the location of P in one image completely specifies the *matching* P and epipolar line in the other image. Now stereo correspondence is reduced to a 1-D search once the epipolar lines are known significantly speeding up calculation of depth, Z. This is the idea behind Simple Stereo which is really the idea of Triangulation. Note that π is a generalized plane and not necessarily a horizontal one.

Its crucial to minimize rotations of two cameras and ensure optical axes are parallel because this reduces the alignment to a 1st order Taylor Series Polynomials (TSP), constraining the images of the corresponding 3-D Scene Points to lie on the same scanline in both images as shown in Figure 2.1.1b. This is known as Stereo Rectification and it reduces the complexity of the stereo correspondence problem mentioned above. Un-rectified images require searching along non-horizontal epipolar lines increasing complexity as can be seen in the generalized stereo setup in Figure 2.1.1a. Binocular camera modules mounted on a planar PCB alleviate this issue considerably, hence the Waveshare Dual IMX219, 8 megapixels camera module was chosen initially[2].

This quick background on the principles underlie the basis for creating a stream processor for vision applications in an FPGA beginning with a discussion of MIPI.

2.2 MIPI System Organization

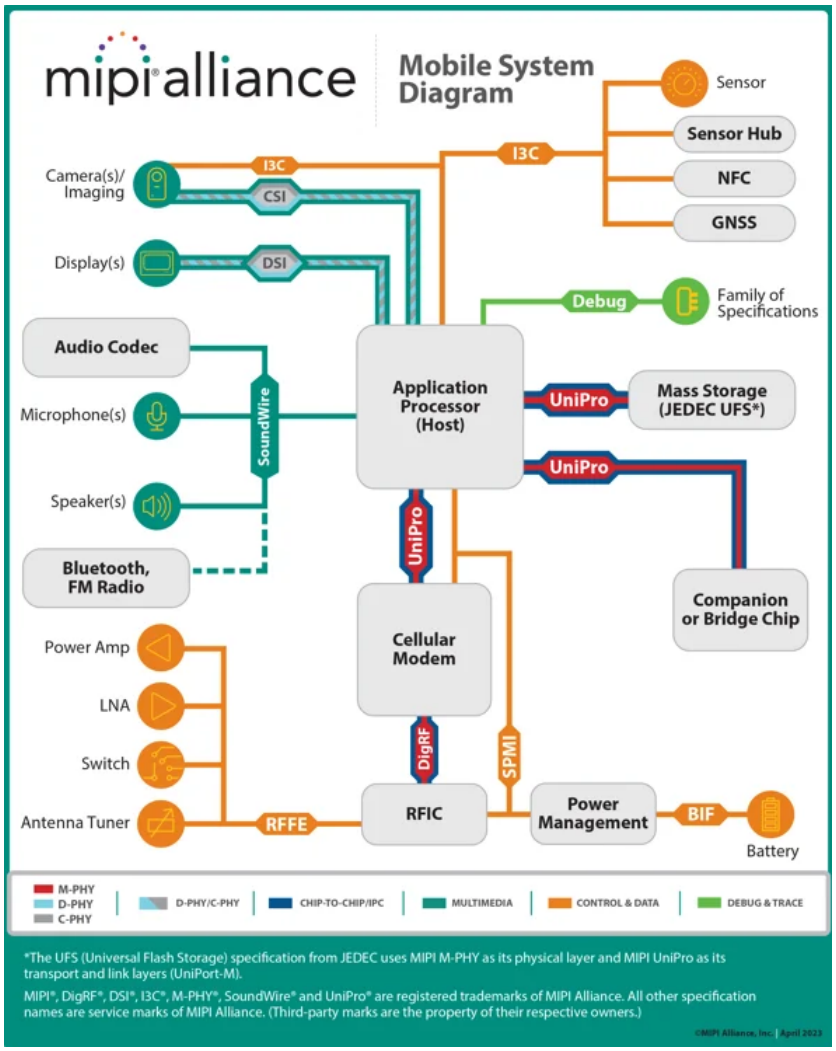


Figure 2.2.1: Ecosystem of MIPI embedded connections[2]

Through an industrial collaboration of established companies in the mobile industry, MIPI defines specifications(protocols) and interfaces for cameras and displays focusing on high-speed, low-power and reduced Electromagnetic Interference (EMI) communication. The main outcome of this alliance is to ensure interoperability between vendors, scalability based on applications, simplified integration of different components, and cost-effectiveness. A broad portfolio of interface specifications from the MIPI Alliance enables design engineers to efficiently interconnect essential components in a mobile device, from the modem and antenna to the peripherals and application processor as shown in Figure 2.2.1.

As can be seen in the top left area, Cameras are communicated and configured using a specification called CSI. Since this report focuses exclusively on cameras, Display Serial Interface (DSI) for displays will not be discussed.

2.2.1 CSI-2

This specification focuses on a data transmission layer for transferring image data (both still images and video) and another layer for basic image sensor control called CCI. The CCI is a subset of the popular I2C protocol, including the minimum combination of obligatory features for I2C slave devices specified in the I2C specification. In this way, transmitters that comply with the CCI specification can also be connected to the system I2C bus. Figure 2.2.2 illustrates the differential connections between camera which is a master and the processor, the slave. Note the I2C connection is not differential. This specification does not cover any advanced imaging system features offered with ISP units such as image exposure, focusing and processing.

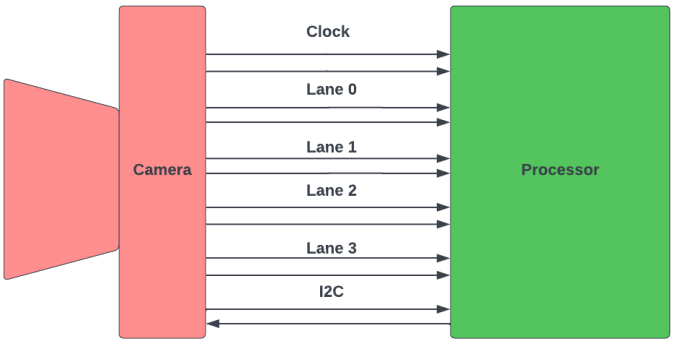


Figure 2.2.2: Typical design of a camera and a host processor of a CSI-2 interface. All four data lanes and clock lane use differential signaling and are unidirectional while CCI (I2C) lane is low-speed, bidirectional, two-wire, half-duplex connection. The processor can be an FPGA.

CSI-1 was the first standardized mobile interface but it was a parallel bus. CSI-2 introduced HS differential serial lanes with support up to 4 lanes (now 8). It takes a parallel image data bus and

[2] https://www.mipi.org/mobile

serializes it into byte size data blocks for speed transmission on a PHY, or physical transmission medium. CSI-2 follows the same general architectures as DSI specification, with the biggest differences being the addition of a separate control interface (CCI) and support for embedded data in the image frame.

The data transmission protocol layer is composed of several layers, all on top of the PHY layer. The PHY layer is discussed below in section D-PHY. This protocol layer specifies how multiple data streams may be tagged and interleaved so each data stream can be properly reconstructed at the receiver side, shown in Figure 2.2.3. In this project, the receiver portion is implemented on a FPGA board. The sub-layers in the diagram are discussed below:

Pixel/Byte Packing/Unpacking Layer Several image applications with varying pixel formats from 6 to 24 bits per pixel (bpp) are supported. 8 bpp are transferred unchanged by this layer.

Low-Level Protocol (LLP) Layer LLP includes means of establishing bit-level and byte-level synchronization for serial data transferred between SoT and End of Transmission (EoT) events. It also includes assignment of bit-value interpretation, such as Endian assignment. The minimum data granularity is 1 byte.

Lane Management Layer This layer manages number of data lanes (1,2, 3 or 4) depending on the bandwidth requirements of the application. The transmitting side of the interface distributes (Distributor function) bytes from outgoing data stream to one or more Lanes. on the receiving side, the interface collects bytes from the Lanes and merges (Merger function) them into a recombined data stream.

2.2.2 D-PHY

The physical layer is the transmission medium comprising of the electrical conductors, input/output circuitry, and the clocking mechanism that captures ones and zeroes from the serial bit stream. This medium needs to be fast in order to transmit the ever growing mobile display resolution sizes serially. There are 3 primary MIPI PHYs used: D-PHY, C-PHY and M-PHY. Of these, D-PHY is the oldest and the only one with viable open source IP alternatives for public use, confining the project to only that medium. Since D-PHY has industry wide vendor adoption and is backward compatible it has 3 versions to suit client needs as shown in table below.

D-PHY is quite prominent in devices where low-power and low-cost requirements are crucial due to their low-latency transitions between HS and LP modes, high noise immunity and jitter tolerances. They are often integrated with CSI-2 and DSI protocols. Traditionally, interfaces between components on a printed circuit board (PCB) are based on single-ended parallel buses at low bit rates using Low-Voltage

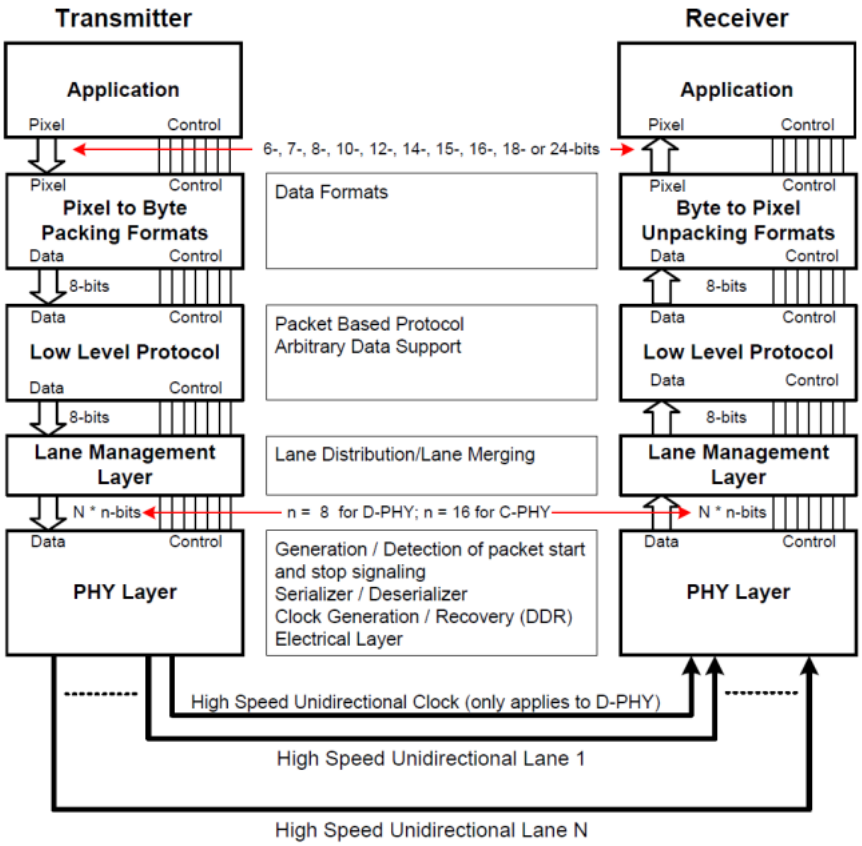


Figure 2.2.3: Various CSI-2 sub-layers comprising data transmission between a transmitter (image sensor) and a receiver (host processor)[3]. Note the pixel is only assembled at the top most layer just before application usage.

CMOS (LVCMOS), differential high-speed serial buses, or single differential channels. D-PHY provides an extension to this structure by turning the low-speed, low-power interface to the serial format of the high-speed, differential interface where both are combined into a ***single*** interface. This means that between transceiver sessions, the differential data and clock lane or lanes can switch to and from a LP state. Interfaces should be in the idle state when they are not actively transmitting or receiving high-speed data. To make efficient use of the few pins used in the D-PHY interface, there is a lack of traditional bus handshaking signals (for example, data enable, line select). Therefore, the transition between the various LP and HS modes is handled through a series of handshaking sequences. In terms of IO circuitry, when in LP mode, all wires are operated single-ended and, are not terminated while in HS mode, each lane is terminated and driven by a low-swing, differential signal[4].

D-PHY uses a source-synchronous, Double Data Rate (DDR) and differential clocking mechanism operating in two modes: HS (for data) and LP (idle state). A four-lane interface would consist of four differential pairs (eight pins) and one differential clock pair (two pins) totaling 10 signal pins. Typically, traffic flows uni-directionally but can be reversed.

Source-synchronous clocking scheme allows receivers to recover data without a clock recovery circuit

Feature	D-PHY v1.2	D-PHY v2.0	D-PHY v3.0
Max Speed per Lane	2.5 Gbps	4.5 Gbps	9 Gbps
Total Throughput (4 Lanes)	10 Gbps	18 Gbps	36 Gbps
Primary Use Cases	1080p/4K Cameras	4k/8K Cameras, 120Hz displays	8k+ Cameras, VR/AR, AI & Automotive Vision
Backward Compatible?	Yes	Yes	Yes
Release Year	2014	2016	2021

Table 2.2.1: Comparison of different D-PHY versions. Gbps is Gigabits/second

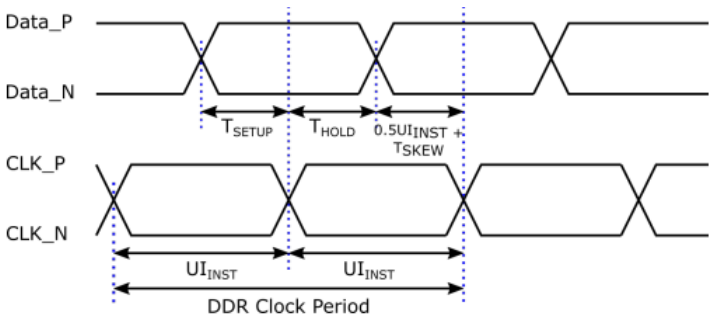


Figure 2.2.4: Timing alignment between data and clock lanes. Since the clock has both a rising and falling edge, D-PHY can transmit 2 bits per cycle (one on each edge) therefore making the clock run at half the data rate. UI is Unit Interval[3]

resulting in LP consumption. However, the skew between data and clock signals limits maximum data rate due to signal integrity issues where compensation and skew calibration circuits (deskewing) are used to eliminate the static skew between data and clock signals[5]. Due to the differential nature of the clock signal, it can transmit data at double the rate at which the rising and falling transitions meet (dotted vertical lines) from the positive (CLK_P) and negative pins (CLK_N) in Figure 2.2.4.

Fortunately, as will be shown in a later section, deskewing circuits are not required for data rates below 1.5 Gbps[6] which is ample bandwidth for the purposes of this project where a maximum of 30 Frames per second (FPS) is used. If the maximum data rate for D-PHY v1.2 (the version used in this project) is 2.5 Gbps per lane from Table 2.2.1 then the maximum clock period is 1.25 GHz. Now you have traffic streaming from ten differential pins and D-PHY needs to deserialize it to a parallel bus for use by a CSI-2 module (Figure 2.2.3) through the PHY-Protocol Interface (PPI). PPI will be discussed in the block diagram section. As mentioned earlier there are two modes of operation and their voltage levels are shown in Figure 2.2.5. The FPGA used in this project does not have native support for the D-PHY IO standard as the latter relies on Scalable Low-Voltage Signaling (SLVS) levels, and therefore the voltages in Figure 2.2.5 need to be adapted.

For a deeper treatment of the dynamic switching of the different modes in the same lane with a

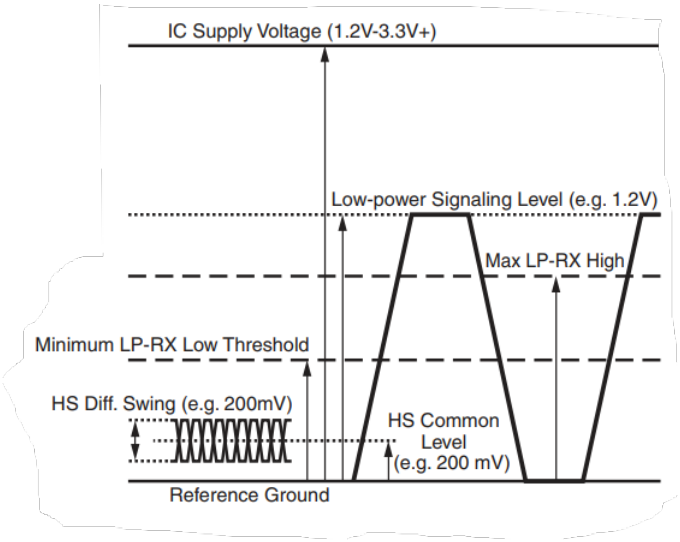


Figure 2.2.5: D-PHY voltage levels for HS and LP modes[4]

focus on IO circuitry of D-PHY, please refer to Appendix A.

2.3 Image Sensor Basics

An image sensor contains an array of photosensitive elements called pixels, with Active CMOS architecture rather than the older Passive CMOS types. Active means that it uses transistors. Each element converts light into a voltage using a photodiode and 3 transistors, in the most basic form. For further details please refer to Appendix B. Not all pixels are used for image capture and only the active columns and active lines (rows) are factored in for processing data. This is the active pixel or effective pixels often quoted as 5 megapixels etc. The Analog-to-Digital Converter (ADC) attached (not shown in figure) to the end of the pixel array samples the voltage of each pixel one at a time (raster)and outputs a digital value onto a data bus until the entire frame is read out. The bus width is equal to the ADC resolution bit-depth. The unprocessed value is in RAW format which is 10 bits wide (RAW10) for the image sensors used in this project. The frame of RAW10 image data goes through a series of post-processing basic steps such as demosaicing (Bayer filtering), colour correction, and white balance before it can be properly displayed. These steps are handled on the image sensor itself (like with sensor OV5640 using ISP functions executed using a microcontroller) or during post-processing using software (like with the IMX219 sensor).

There are three clocks that are crucial to be used with MIPI image sensors. The master clock (MCLK) is usually generated by an external clock source, such as a crystal oscillator or a clock generator for a stable and precise timing reference for camera sensor's internal operations. The MCLK is used to synchronize the timing of the Pixel Clock (PCLK) and other signals and in Figure 2.3.1, indicated as

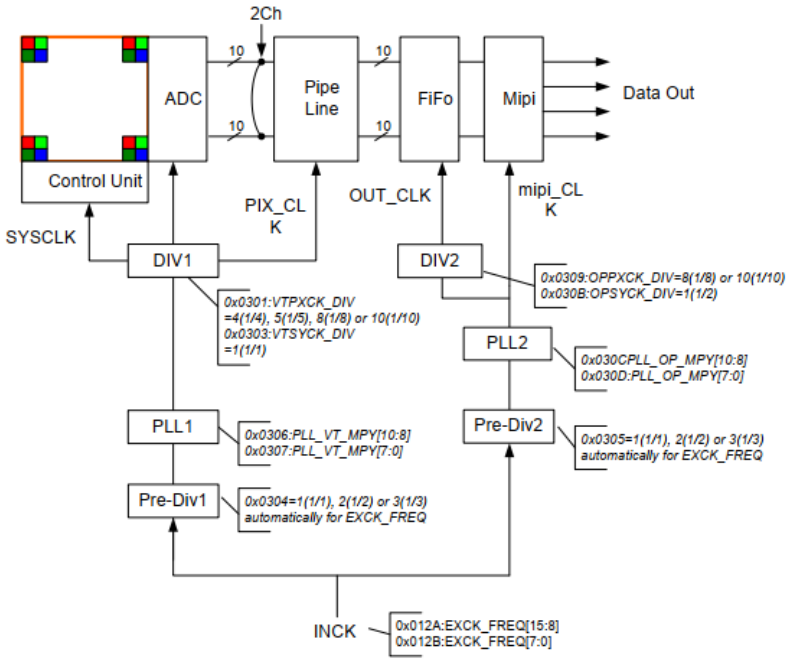


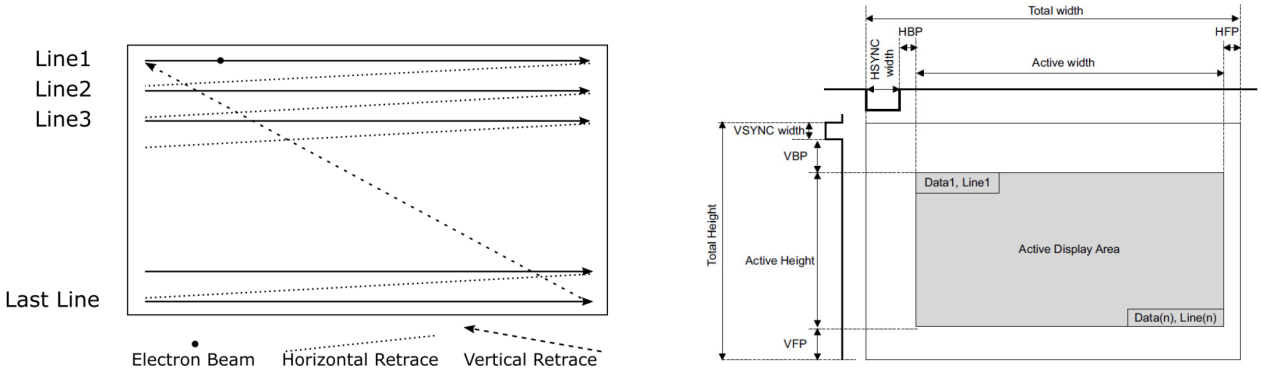
Figure 2.3.1: An example system showing the different clocks and register addresses for timing configuration of an image sensor, in this case its IMX219[7]

INCK. The Phased Lock Loop (PLL) in the image sensor take this signal and generate other signals that make the image sensor ICs operate. Next, the PCLK and MIPI_CLK, generated by the 2 PLLs that are of interest on the receiver side.

2.3.1 Video Timing

Modern computer displays use raster displays as mentioned above, meaning the image is created by writing one line at a time, pixel by pixel from left to right and from top to bottom of display (Figure 2.3.2) This is the frame. A video stream is composed of a series of frames displayed at a fixed timing interval (FPS). In a frame, there are many horizontal retraces but only one vertical retrace as seen in Figure 2.3.2a. These retraces have roots in the old Cathode Ray Tube (CRT) type displays where time was needed to steer the electron beam. However, now that time (called Blanking Interval) is used for like image processing and embedding additional data (audio in case of HDMI or embedded image data for CSI-2). Therefore many synchronization signals are used to relay the information between retraces as shown in Figure 2.3.2b.

A single horizontal line of image data is signaled by the assertion of an Horizontal Sync Pulse (HSYNC) pulse. That pulse is followed by the Horizontal Back Porch (HBP), the active pixel data (H_{REF}), and finally the Horizontal Front Porch (HFP). The HSYNC pulse, HBP, and HFP all use pixels [p] for their unit of measurement. These values amount to Horizontal Blanking Interval (HBI) in



(a) How a raster scan is achieved in modern displays **(b)** Data display synchronization and its signals

Figure 2.3.2: Raster scan and its corresponding synchronization for CSI-2

(1), shown with its unit of measurement.

$$HBI = HFP + HSYNC + HBP \ [p] \tag{1}$$

$$H_{TOT} = H_{REF} + HBI \ [p] \tag{2}$$

Vertical Sync Pulse (VSYNC) pulse signals the start of a new frame. That pulse is followed by the Vertical Back Porch (VBP) then the active image lines and finally the Vertical Front Porch (VFP). This process repeats for the next image frame. The VSYNC, VBP, and VFP all use horizontal image lines $[l]$ as their unit of measurement. These values amount to Vertical Blanking Interval (VBI) in (3). For a system configured to capture an image of resolution 1920x1080, the active lines is 1080.

$$VBI = VFP + VSYNC + VBP \ [l] \tag{3}$$

$$V_{TOT} = Active\ lines + VBI \ [l] \tag{4}$$

The system bandwidth must be calculated to ensure that the various clocks support the desired system. The basis of all image system design begins with the PCLK. The total bandwidth, BW_{sys} of the system (bits per seconds (bps)) and Data Rate Per Lane (DRPL) are calculated after depending on the number of data lanes which is two in this project.

$$PCLK = H_{TOT} \times V_{TOT} \times FPS \ [Hz] \tag{5}$$

$$BW_{sys} = PCLK \times bpp \ [bps] \tag{6}$$

$$DRPL = \frac{BW_{sys}}{\#\ Data\ Lanes} \ [bps] \tag{7}$$

It may seem that the PCLK is transmitted on the differential clock pins (CLK_P and CLK_N in Figure 2.2.4) from the image sensor, but the MIPI_CLK is, at half the data rate per lane as discussed

in section D-PHY. It is calculated as,

$$MIPI\_CLK = \frac{DRPL}{2} \; [Hz] \tag{8}$$

2.3.2 Transmitter CSI-2 Video Timing, Packing and Packet Details

CSI -2 hardware block in the transmitter takes the image data from the parallel bus in the sensor and packs it into a serial byte stream which can be transmitted over D-PHY as shown in Figure 2.3.3. The blue arrows and numbers to the left of it depict the relationship between CSI-2 and D-PHY. The

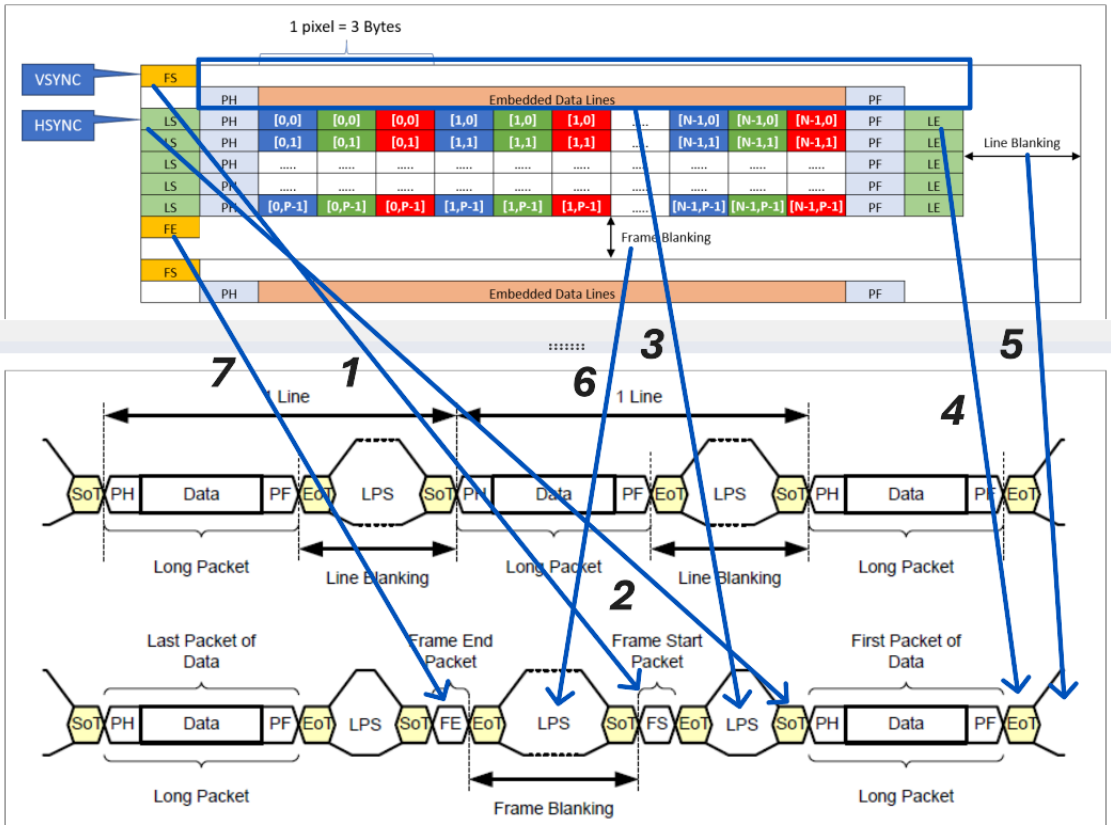


Figure 2.3.3: CSI-2 packetized frame structure for a RGB888 (3 bytes) format and conversion into a serial byte stream for transmission over a D-PHY lane. N is number of columns (pixels) and P is number of rows (lines)

above diagram is a fusion of two figures in [3]. FS, FE, LS, LE and LPS are Frame Start, Frame End, Line Start, Line End, and LP State. respectively. The size of the data packet depends on the output format of the video image sensor. There are no embedded data lines being used here so it will be blank. As can be seen, the D-PHY has fewer signals overall and EoT always initiates an LPS for power conservation, which is the most crucial aspect of this design. The line and frame blanking interval sizes change depending on the resolution selected for the image sensor as indicated by the dashed lines surrounding LPS states in the figure. The corresponding signals across different layers shown with the

Arrow No.	Video Standard	CSI-2	D-PHY
1	VSYNC	FS	FS
2	HSYNC	LS	SoT
3	blank	blank	LP
4	HFP	LE	EoT
5	horizontal retrace	line blanking	LP
6	vertical retrace	frame blanking	LP
7	VFP	FE	FE

Table 2.3.1: The correspondence of signals across MIPI layers shown in Figure 2.3.3

arrows are captured in Table 2.3.1. Since only RAW10 output is used in this project the data packet length is more complicated and will be discussed in the next section.

2.3.2.1 RAW10 Data Packetization

CSI-2 stipulates a minimum data width of 1 byte from the image sensor. RAW10 does not fit in a byte and partial pixel data cannot be transferred, CSI-2 has various pixel packing standards based on pixel format used specified in Table 18 in [3]. with RAW10 bpp is 10 bits and the length required to make it integer divisible by a byte (8 bits) is 40 bits long. Therefore, four RAW10 pixels are arranged into a packet, where the packet length is now 5 bytes long $10 \times 4 = \frac{40\ bits}{packet} = 5\ byte\ packets$. This way ensures CSI-2's 8-bit boundary conditions are met and makes better use of bandwidth in MIPI lanes. This operation is shown in Figure 2.3.4. *P1* contains the first 8 bits (Most Significant Bit (MSB))

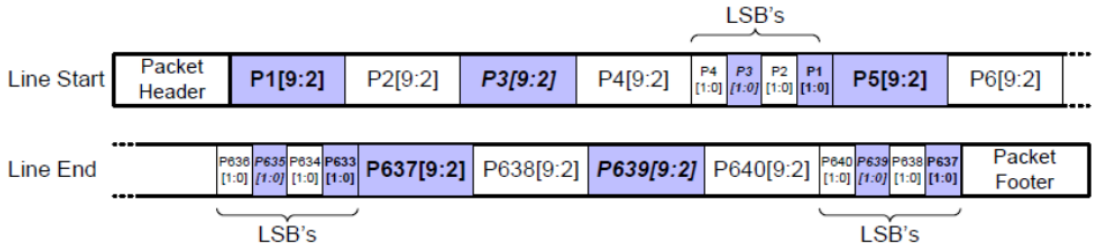


Figure 2.3.4: RAW10 packetization for CSI-2 supported formats packet data size constraints[3]

of a RAW10 10-bit stream from the ADC, *P2* contains the 8 MSBs of the next RAW10 stream and so on till *P4* is packed with the 8 MSBs of the fourth RAW10 stream. There are still 2 lower bits (Least Significant Bit (LSB)) from each of the 4 RAW10 streams that are collected and packed into another, final byte as indicated by the LSB's bracket in figure above. The packing process is not as straightforward as the calculation for 5 bytes above. Slices of 4 consecutive streams from ADC are rearranged into 5 bytes by the CSI-2. This is then decoded by the receiver and processed in the FPGA.

2.4 Related Works

A major shift in the image sensor industry was the adoption of serial interfaces over parallel. Parallel interfaces were the traditional transfer mechanism preferred by published literature on stereo vision. However, due to their high pin count, inefficiency at higher resolutions, limited bandwidth, and susceptibility to signal integrity issues, serial interfaces became more popular but at the cost of circuit complexity. MIPI is a serial interface but not an entirely open standard like USB or Ethernet. It is not free to the public and requires full membership and adherence to legal agreements. This limitation restricts widespread academic adoption and research, restricting sufficient output of peer-reviewed literature on stereo vision, MIPI on FPGAs. Hence, only works relevant or analogous to some aspects of this project that further the overall understanding will be discussed including Masters level literature. Although not common to include this type of literature, the practical aspects discussed were useful for this project and hence referenced.

The closest analogue to the system presented here is a senior year engineering thesis by M. Feldmieier [8] in German. A machine translated version was used to understand the thesis focused on stereomicroscopy. This project was done entirely using an open-source toolchain, incorporating the ULX3S FPGA board from Radiona with a Lattice ECP5 PL chip. Two MIPI image sensors (IMX219) were used to combine the camera streams in the board and output it via HDMI. In the report, it is noted that due to hardware limitations, only a passthrough of the a video stream achieved but no snapshot of the stream is provided. However, the details served as a good introduction with MIPI-based stereo vision. More details of this project are shown in Figure 3.4.4.

Next, Yang et al[9] devise a similar system but with a single camera, Sony IMX377, to acquire Ultra High Definition (UHD) - 3840×2160 resolution images, for endoscope applications. This is a full FPGA system (Artix 7) with no PS utilizing the same resistor network as in this project using all 4 data lanes. Due to the source-synchronous nature of the D-PHY, IDELAY and ISERDES primitives are used to dynamically phase align the each data lane and clock lane. Similar methods at the physical layers are integrated within the IPs used in this project.

In [10], Jang et al, designed a receiver bridge chip that can decode MIPI CSI-2 protocol supporting up to 2.5Gbps per lane. However, the cost of a self-designed chip is very high. The bridge chip converts 4 lane HS data of SLVS of CSI-2 into 32 lower speed data fit for a parallel LVCMOS interface for the FPGA chip. This solution required custom chip and board design and did not use commercially available products. Following the success of this module, Jang et al devised a 5 Gbps per lane receiver bridge chip[6] that demonstrated an auto-skew calibration architecture. The architecture was insensitive to dynamic noise owing to the use of multiple bis supplied from the deserializer as a result of the

phase detector for skew calibration. However, this setup was not designed for a stereo vision system, in addition to being custom manufactured.

For literature with stereo vision components, [11] D. Qendri discusses an image stitching system in real time for semi-panoramic video synthesis at 30 FPS using a Zynq SoC. The developed system was compared with OpenCV implementation running on a single core CPU where it outperformed it. In [12], J. Hippolyte discusses a more accurate object tracking and ranging algorithm compared to a previous system. Images are captured using a dual camera setup and focused on the window of interest on the suspect object area and rechecked at camera pixel resolution for a moving object. Following along the similar lines, a tracking system specifically aimed at HFRs is discussed by O. Samarin[13] where the coordinates of the object are used by the fuzzy logic control system to provide rotation and focus control for object tracking.

Another HFR architecture for the purposes of autonomous satellite grasping, is demonstrated by J. Islam[14] with the stereo matching algorithm used, described in thesis by S. Sabihuddin[15] for fast estimation of depth using an algorithm called Dynamic Programming Maximum Likelihood (DPML). DPML is very hardware friendly. Even though all of these systems use customized parallel buses for image transfer often at lower resolution, their content is very useful for future applications of this project.

Chapter 3

Proposed Approach

One of the main tenets of this project is to utilize COTS devices for stereo vision. The heart of the system is the FPGA and an overview of that chip is discussed in section 3.1 followed by the initial connection layout in section 3.4. The devices in the layout are explored including the pitfalls encountered. Next, the final design and its respective solution finding method are detailed in 3.5.

3.1 Zynq-7020 Overview

The prototyping system consists of the Digilent Zybo Z7-20 development board which uses a Xilinx (now AMD) Zynq-7000 Series SoC architecture, particularly the Zynq-7020 (Z-7020) FPGA chip in the board. This chip has 2 parts: A dual-core PS and a 28nm PL. The chip's designation is XC7Z020-1CLG400C meaning it has 400 pins with a speed grade of -1 (Max frequency of 667 MHz). CLG stands for Ceramic Land Grid Array. CLG packages are non-flip chip, utilizing bonding techniques, where the chip is physically connected to the package pins through wire or other conductive material. Table below details relevant details regarding the chip in Table 3.1.1.

PS		PL	
Feature	**Value**	**Feature**	**Value**
Processor Core	Dual-Core ARM Cortex-A9	AMD FPGA Equivalent	Artix-7
Max Frequency	667 MHz	Cells	85,000
L1 Cache	32 KB Instruction; 32 KB data per processor	Look-Up Tables (LUT)	53,200; True 6-input LUTs
L2 Cache	512 KB	Flip-Flops (FF)	106,400
On-Chip Memory (OCM)	256 KB	Block RAM (BRAM) (# of 36 Kb Blocks)	4.9 Mb (140); $\leq$ 36b wide; True dual ports
DMA	8 (4 dedicated to PL)	DSP Slices	220
GPIO	4 32b banks = 128; PS uses 54 (1 32b+1 22b banks); PL uses 64 (2 32b banks);	Programmable IO blocks	Supports LVCMOS, Low-Voltage Differential Signaling (LVDS), SSTL; 1.2-3.3V IO; Programmable IO delay and SerDes
Static RAM (SRAM)	8b SRAM data bus, upto 64 MB support; 8b parallel and serial NOR, NAND Flash support	Configurable Logic Blocks (CLBs)	8 LUTs/CLB, 16 FFs/CLB, 2x4b cascadeable adders; LUTRAMs: 64x1 or 32x2 bit RAM or shift register (SRL)

Table 3.1.1: The features of the Zynq-7020 SoC. Note that b is bits and B is bytes[16]

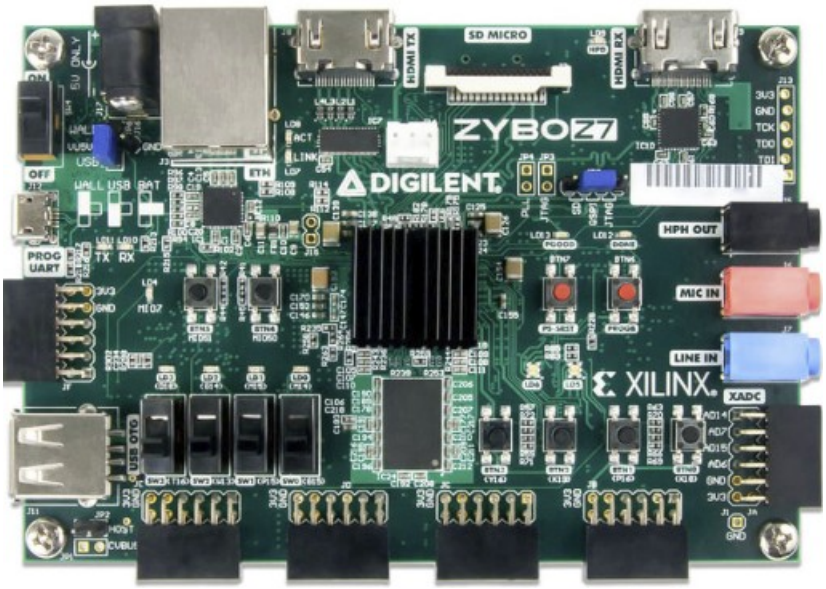


Figure 3.2.1: Top view of Zybo Z7-20

The PS consists of various peripheral controllers with their inputs and outputs multiplexed to 54 dedicated pins called MIO. These occupy the same pin locations as GPIO where the desired pins can be selected for the particular use-case whether a GPIO or peripheral controller. Peripheral controllers that do not have their inputs and outputs connected to MIO pins or if additional IO pins beyond 54 are required, it is possible instead to route their IO through the PL, via the Extended MIO (EMIO) interface.

3.2 Digilent Zybo Z7-20 FPGA/SoC Development Board

This development board surrounds the the Z-7020 chip, discussed above, with a rich set of multimedia and connectivity peripherals to create a powerful single-board computer with the FPGA. It contains 6 pushbuttons (2 being connected to PS), 4 slide switches, 5 LEDs (1 PS connected) and 2 RGB LEDs. The Zybo Z7's CSI-2 compatible Pcam connector is the main reason this particular board was chosen (in Figure 3.2.1 it is white connector above the label "ZYBO Z7").

Since the Z-7020 chip does not have native support for D-PHY, the Zybo Z7 boards equip the Pcam MIPI lanes with a passive resistor solution detailed in Appendix A. This is called the D-PHY Compatible solution detailed in [4]. Zybo Z7-20 was chosen because The Z-7020 has more FPGA resources for dual image processing than the one used in Zybo Z7-10. The features are reviewed in 3.2.1

IO Peripheral	Specification
External reference clock for PL	125 MHz; pin K17
USB-UART Bridge	• FTDI FT2232HQ Bridge IC • Independent USB-UART and USB-JTAG controllers
Memory	1 GB DDR3L with 32b bus @ 1066 MHz; 16 MB Quad-SPI Flash
PCAM Port	• Supports 2-Lane CSI-2 (6 wires), I2C bus (2 wires), 2 GPIO, 4 grounds, 1 3.3V signals • Designed to have same pinout as RPi supported cameras • Uses standard 15-pin, 1 mm pitch, Zero Insertion Force FFC
HDMI	• 1 unbuffered source port (**HDMI TX**) • 1 buffered sink port (**HDMI RX**) • Both ports use HDMI type-A receptacles with data and clock signals terminated and connected only to the PL
Pmod Interface	• 6 ports in total: 1 analog, 3 high-speed, 1 standard, 1 MIO • MIO and standard Pmods are connected to PS and PL, respectively, via 200 Ω series resistors • The series resistors prevent short circuits that may occur when an input is accidentally driven as output • Series resistor limits maximum switching speed of data signals due to added resistor that increases the RC constant affecting charging and discharging times

Table 3.2.1: A quick reference to the features of the IO peripherals and memory features of interest in the Digilent Zybo Z7-20[17]

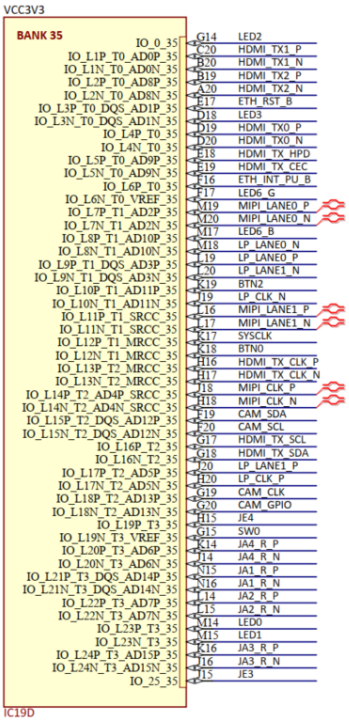
3.2.1 CSI-2 Pcam Port and Slide Switch Schematics

The D-PHY compatible solution using a resistor network shown in Figure 3.2.2. This network provides the high-speed and low-power MIPI interfaces. However, it is important to note that this does not comply fully with MIPI IO standards. The HS signals are adapted from SLVS levels to LVDS in the Zybo Z7-20. For the full compliant solution, active components are used and packaged in another IC like from Meticom.

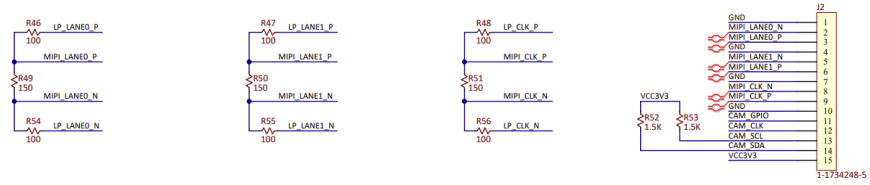
3.2.2 Standard and MIO Pmod Ports

The Pmod ports are used to communicate with the MCA in the final design for both the softwaare and hardware switching functions. The general expectation of Pmod interfaces is that a 3.3V logic power supply will be used and the signals will conform to LVCMOS 3.3V or LVTTL 3.3V logic conventions. The standard Pmod port is used for hardware switching and MIO Pmod port is used for software switching. These ports are not intended for high-speed control due to the 200 Ω resistors connected to all pins of the headers shown in Figures 3.2.3 (Standard) and 3.2.4 (MIO). There are also Zener diodes connected to every pin to suppress voltage spikes and protect the input to FPGA.

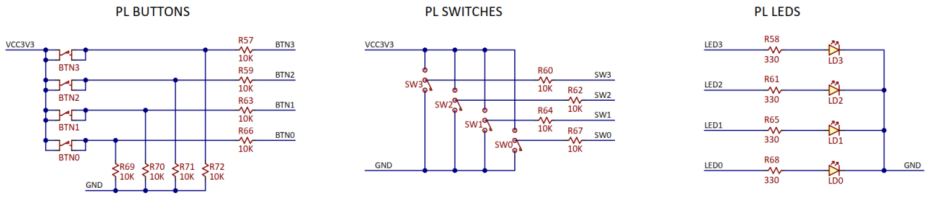
The pinouts for the 6 Pmod ports are listed in Figure 3.2.5 below. They are categorized into 4



(a) The FPGA Bank 35 connecting all MIPI signals



(b) The resistive network as wired on the Zybo Z7-20 from the pcam header J2. Both the clock and data lanes have resistors ond HS and LP lines. The 2 I2C lines are connected to 3.3V using 2 pull-up resistors, 1.5 $K\Omega$ each



(c) The wiring diagrams in the board for the slide switches, pushbuttons and LEDs (not RGB LEDs)

Figure 3.2.2: The internal wiring diagram. The resistive connections are wired from (b) to (a) to 4 different pins from differential points per lane to support HS, LP modes. The red icons indicate differential signals[18]. I2C and GPIO signals are also wired to FPGA. The pin designations shown are used in the constraints file

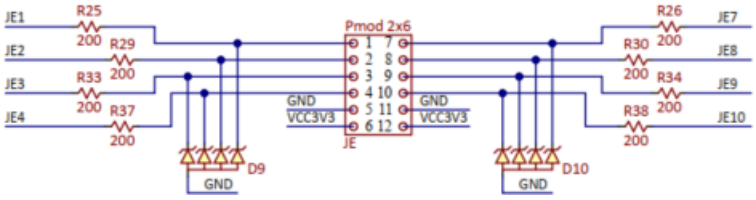


Figure 3.2.3: Standard Pmod port used for hardware switching. 200 Ω is used as limiters for short circuit currents

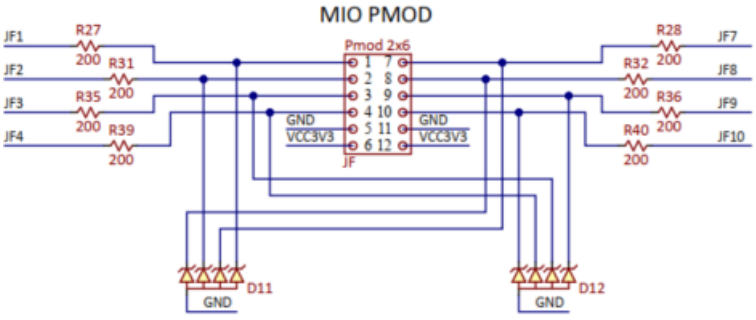


Figure 3.2.4: MIO Pmod port used for software switching. 200 Ω is used as limiters for short circuit currents

categories. These pins are used in the constraints file for switching operations.

	Pmod JA	Pmod JB*	Pmod JC	Pmod JD	Pmod JE	Pmod JF
Pmod Type	XADC	High-Speed	High-Speed	High-Speed	Standard	MIO
Pin 1	N15	V8	V15	T14	V12	MIO-13
Pin 2	L14	W8	W15	T15	W16	MIO-10
Pin 3	K16	U7	T11	P14	J15	MIO-11
Pin 4	K14	V7	T10	R14	H15	MIO-12
Pin 7	N16	Y7	W14	U14	V13	MIO-0
Pin 8	L15	Y6	Y14	U15	U17	MIO-9
Pin 9	J16	V6	T12	V17	T17	MIO-14
Pin 10	J14	W6	U12	V18	Y17	MIO-15

*Pmod JB is not available on the Zybo Z7-10

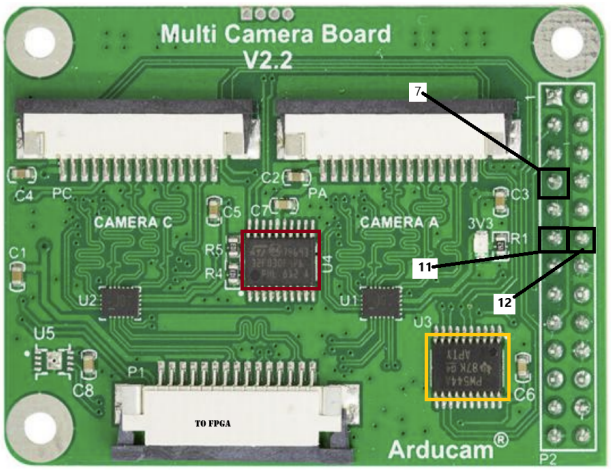
Figure 3.2.5: Zybo Z7 Pmod pinouts[17]

3.3 Arducam Multi-Camera Adapter Module V2.2

The next common component in all the designs is the MCA where it can accommodate up to 4 camera modules with the purpose of quickly switching between them. It is a common component in all the designs. It is intended to connect 4 Arducam MIPI cameras like the 5 and 8 Megapixel ones to a single CSI camera port on the RPi board. Due to CSI-2's signal integrity sensitivity to long cable connection, this board does not support stacking and can only connect 4 cameras at maximum. Some important features:

- No mixing of different sets of cameras allowed (ie 5 Megapixel with an 8 Megapixel)
- 3 GPIOs required for multiplexing for switching between cameras
- Cameras work sequentially, not simultaneously

The PCB of MCA is shown in Figure 3.3.1. After repeated attempts to the manufacturer, Arducam, the schematics of this device was not shared. Therefore, the flow of operations and details of the components were unavailable. Thus investigative engineering was performed by carefully tracing out the lines on the PCB, front and back, to see where connections went to what components. The designation of the ICs can be gleaned by the markings and there are two main ones: 20-pin embedded microcontroller (STM32F030F4P6[19]) and 20-pin I2C Multiplexer (TCA9544A[20]) indicated as red and yellow boxes by figures in 3.3.1. The way cameras are selected is through 3 GPIO pins 7, 11 and 12 labeled as **Selection**, **Enable 1** and **Enable 2**, respectively[21]. These signals were originally intended to be connected to the RPi board with a 40-pin header as shown in Figure 3.3.2. How is the camera port



(a) The front view of the multi-camera adapter

(b) The back view of the MCA. The boxes over the ICs are estimations

Figure 3.3.1: The board views of the multi-camera adapter. The GPIO signals are shown in labeled 7, 11 and 12. The I2C bus master IC (yellow box) and microcontroller (red box) on both figures. The dot in the boxes signifies the start of the numbering of the pins [21]

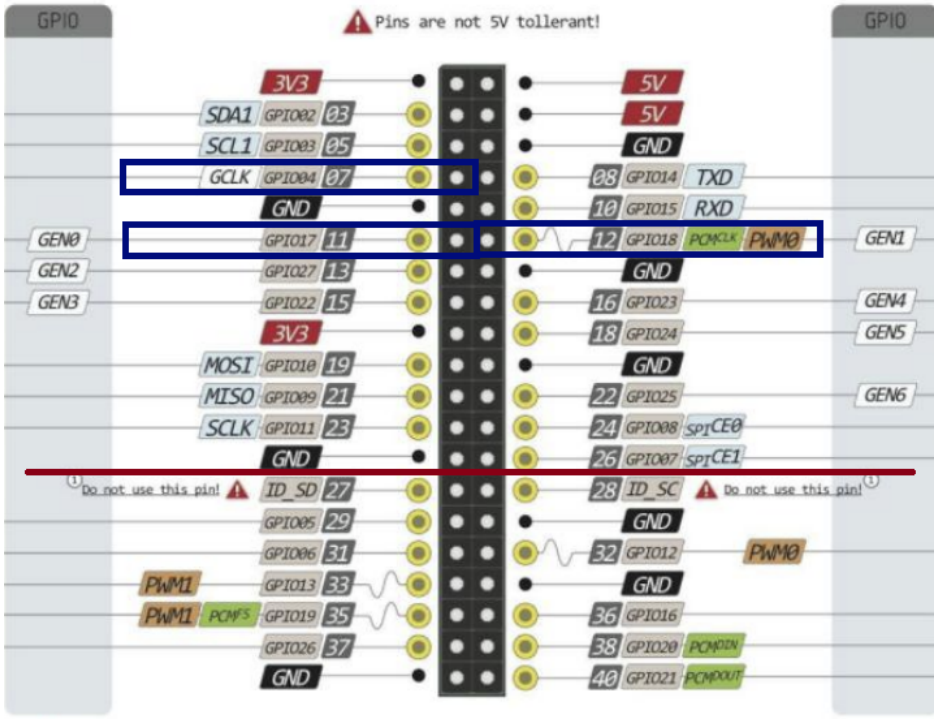


Figure 3.3.2: Standard RPi 40-pin header connections. Pins 7, 11 and 12 are shown in blue boxes. The header in MCA is only 26 pins (Figure 3.3.1b) and the boundary from the standard 40-pin header is indicated by the red line

actually selected in the MCA? GPIO pins 7 (Selection), 11 (Enable 1) and 12 (Enable 2) are modified as desired where they drive the power up signal to the image sensors. There is conflicting information with documentation on pin selection. The enlightening moment came when the I2C command set in Quick Start Guide[22], written in Python, was found and it detailed how to address and configure different

cameras attached to the camera ports in MCA. The cameras **need** to be configured **before** they can be selected. That command set also detailed the address of the device (The annotated black box in Figure 3.3.3a) that performs the configuration. furthermore, in Figure 3.3.3a notice how cameras A, B, C, D are sent the hexadecimal information 4, 5, 6, 7 respectively.

```
adapter_info = {
    "A" : {
        "i2c_cmd":"i2cset -y 10 0x70 0x00 0x04",
        "gpio_sta":[0,0,1],
    }, "B" : {
        "i2c_cmd":"i2cset -y 10 0x70 0x00 0x05",
        "gpio_sta":[1,0,1],
    }, "C" : {
        "i2c_cmd":"i2cset -y 10 0x70 0x00 0x06",
        "gpio_sta":[0,1,0],
    },"D" : {
        "i2c_cmd":"i2cset -y 10 0x70 0x00 0x07",
        "gpio_sta":[1,1,0],
    }
}
```

(a) I2C Command Set in Python used to configure cameras connected to MCA[22]

Camera Selection Configuration	Selection	Enable 1	Enable 2
A	0	0	1
B	1	0	1
C	0	1	0
D	1	1	0
No Camera	X	1	1
Error	X	0	0

(b) GPIO Camera Selection Table[21]

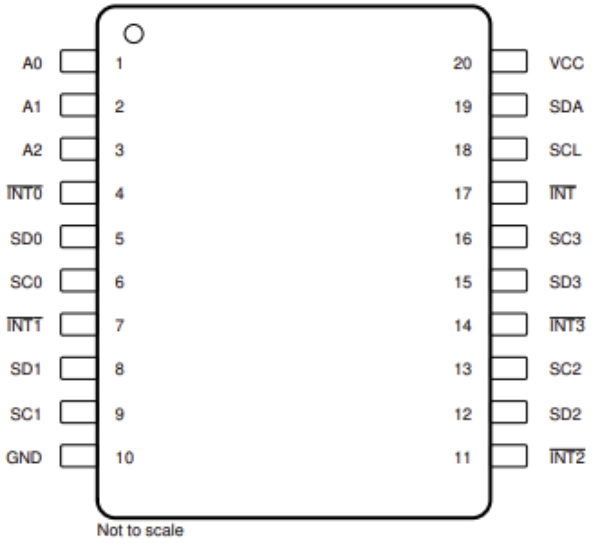
Figure 3.3.3: All the camera ports (A, B, C, D) are first configured by device on address 0x70. The GPIO information is also specified here for camera selection.

However, of the two ICs in the MCA it is unclear which does what. Moving along with the investigation, there is a black header in the back view (Figure 3.3.1b). Initially, this was left unconnected and was not given much thought because it was assumed that the 15-pin FFC port labeled "To FPGA" (Figure 3.3.1a) would control all 4 camera feeds. However, after further analysis following many failed attempts, these cameras were enabled via a selection process controlled by the I2C Multiplexer.

3.3.1 I2C Multiplexer

The reason this multiplexer is required is that all cameras connected to MCA will be factory installed with the same I2C address that is usually not user modifiable. Therefore, a switching operation is required to independently configure each camera[11]. The I2C multiplexer is a 4-channel device with Interrupt Logic as can be seen in Figure 3.3.4a. The dot in the figure corresponds to the dots in yellow boxes in Figure 3.3.1 for easy identification of the pins. The master I2C enters through pins 18 and 19 (in Figure 3.3.1b below the yellow dot the 2 holes) from header pins 3 and 5 on the other side (Figure 3.3.1a). Pins 3 and 5 are shown as I2C signals in Figure 3.3.2 that is supposed to be controlled by a host like RPi. But also the same line goes to the 15-pin FFC connecting to the FPGA on lines 13, 14 which are coincidentally the I2C signals!

Confirmation that the device used to configure the cameras is this multiplexer was obtained by

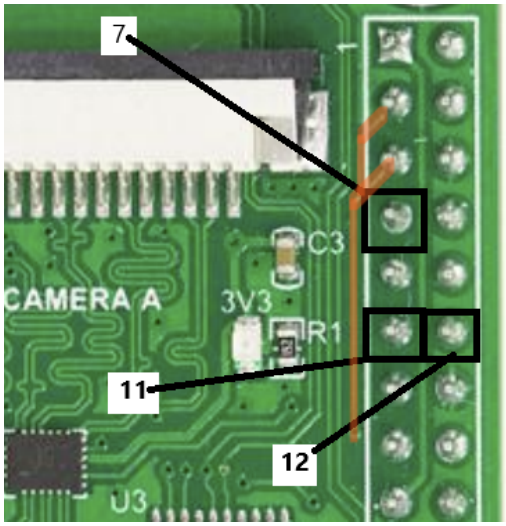


(a) The pinouts for the I2C Multiplexer. It's a 4-Channel controller with Interrupt logic.

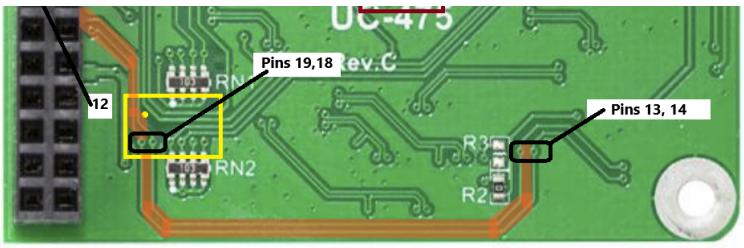
B2	B1	B0	COMMAND
0	X	X	No channel selected
1	0	0	Channel 0 enabled
1	0	1	Channel 1 enabled
1	1	0	Channel 2 enabled
1	1	1	Channel 3 enabled
0	0	0	No channel selected, power-up default state

(b) Channel Selection Table

Figure 3.3.4: The low level details regarding I2C Multiplexer[20]



(a) A cropped version of Figure 3.3.1a displaying the I2C signals connecting to I2C Multiplexer from pins 3, 5 (not shown) in header pin going to the underside, shown in brown highlight.



(b) Cropped view of Figure 3.3.1b. The brown highlight depicting how I2C master line can either come from RPi or FPGA through pins 13 and 14 (confirm with Figure 3.2.2b). Both master lines are going to pins 19 and 18 of I2C Multiplexer

Figure 3.3.5: Detailed view showing the lines of I2C master control coming from different hosts to the I2C Multiplexer

looking at Figure 3.3.4b. The hexadecimal numbers used to enable the 4 channels perfectly match those for Camera A, B, C and D in Figure 3.3.3a, where they are both 4, 5, 6 and 7. Futhermore, one can see the camera ports B and D are fed by I2C signals from I2C Multiplexer from the pull up resistors RN1 and RN2 to pins 13 and 14 of the same ports B and D in Figure 3.3.1b.

3.4 Initial Design

The initial setup of the project was conceived with the layout shown in Figure **??** below using Flat Flexible Cable (FFC) for image capture. The binocular vision module came assembled on a PCB, ready for computer vision applications, replete with 2 Sony IMX219 sensors. To switch the stream, the Arducam MCA is employed to usher the selected stream onto the Digilent Zybo Z7-20 board. The stream is processed in the FPGA and sent to the monitor via HDMI. It was assumed, before the troubleshooting began, that it would be possible to switch the streams via GPIO from the FPGA smoothly without connecting to the header in MCA.

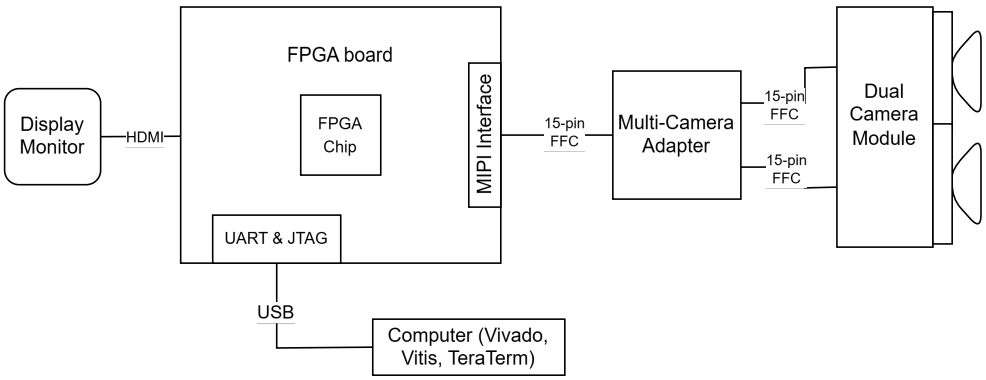


Figure 3.4.1: The initial design layout

3.4.1 Waveshare Binocular Camera Module

The binocular dual camera module housing 2 Sony Exmor R Series Image Sensors (IMX219), comes readily assembled on a PCB, specifically designed for stereo vision. It is sold by Waveshare. It is an 8 Megapixel camera and does use the same 15-pin FFC but it is compressed for 0.5 mm pitch. Like the MCA described above, it is intended for the RPi and untested for hardware platforms. This module also boasts an inertial measurement unit IC (ICM20948) for motion tracking but is not necessary for this project. There is no literature regarding the usage of this particular module for vision applications but the image sensor itself, IMX219, is quite popular and it is used by RPi to make a version of their camera module as well. Called the Camera Module 2, its a single camera, not a binocular module. The block diagram of IMX219 depicting internal on-chip operations is shown in Figure 3.5.2b.

3.4.2 System Bring-Up

Now that the initial design is complete a housing was made to hold all the elements in a self-contained unit. The hope was to make this system portable. In Figure 3.4.3, a plastic case with cover was drilled for the riser screws to secure the FPGA board, MCA and binocular module (not shown). The Zybo

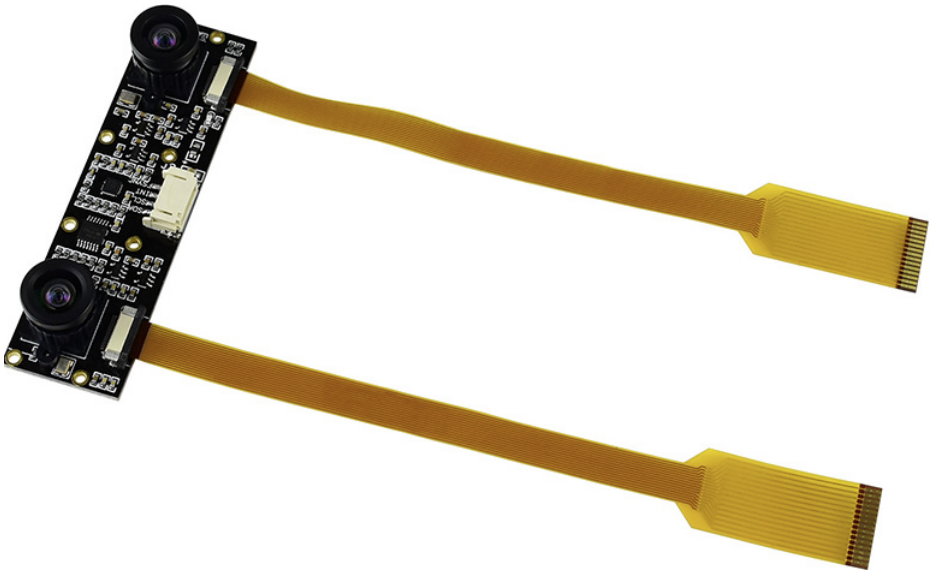


Figure 3.4.2: 8 Megapixel Binocular Camera module from Waveshare. It also uses a compressed 15-pin FFCs

Feature	Specification
CMOS Size	1/4 in.
Pixels	8 Megapixels
Aperture (F)	2.4
Resolution	3280 x 2464 (per Camera)
Focal Length	2.6 mm
Field of View (FOV)	$83^\circ\ (D), 73^\circ\ (H), 50^\circ\ (V)$
Distortion	$< 1\%$
Baseline Length	60 mm

Table 3.4.1: Table of specification for Waveshare Binocular Camera[23]

Z7-20 had to be elevated to accommodate the HDMI cables over the MCA as shown in the figure. Note the black header on MCA is pointing down and any connections to it need to come out from the bottom of the unit, which at this point provisions were not made for that. Once the system was sufficiently housed and openings made for the cables to run through, next action was to perform system bring-up. The software and hardware designs were ported from various sources, listed in Figure 3.4.3. The initial plan was to configure one of the cameras, capture and display the images coming through. However, configuration was unsuccessful. At no point was there any partial image, blurred or snowed in image. The entire was screen was blank and in fact the screen was not getting signal. After debugging, using ILA, it was found out there was no timing signal generated for the HDMI block to output a video stream. No timing signal was generated because there was no valid video input from CSI-2 hardware block which meant that camera was not configured properly. After many versions were attempted and

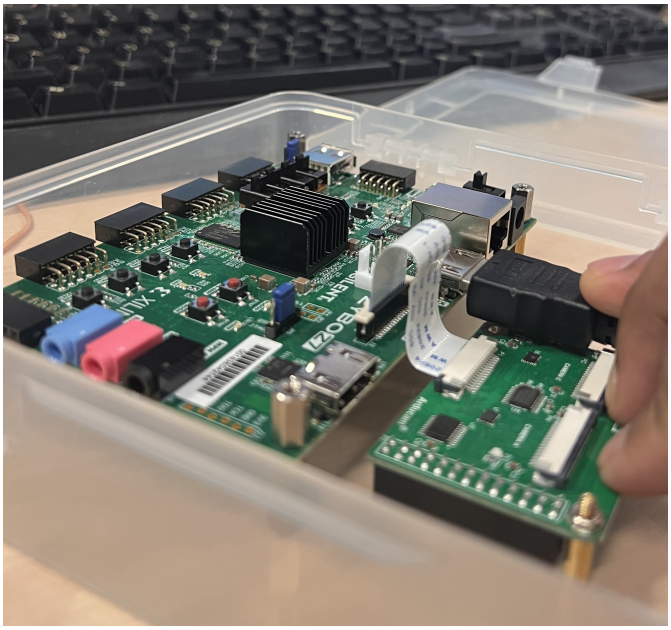


Figure 3.4.3: The assembly of the housing to contain all components of the initial design

failed, the RPi Camera Module 2[24] was purchased and experimented similar to the design by Greg Taylor (bottom row) in Figure 3.4.3. That implementation used a Zynq Ultrascale+ and displayport which were modified to Zynq-7020 and HDMI but the configuration settings for IMX219 was kept the same. Unfortunately that did not yield the expected results and the same issue with the configuration of IMX219 persisted. The clock configuration settings in Figure 2.3.1 were checked quite a few times for accuracy, among others, but no obvious mismatch was found. The decision to shift to another image sensor, OV5640, was made because it has hardware support from Digilent Inc. However, this sensor came packaged as a single camera module called PCAM-5C. The plastic case was abandoned due to the different form factor of the Pcam-5C.

In pursuit of a working system with at least one camera, many publicly available systems were modified and attempted. The list in Figure 3.4.4 discusses the public projects that were attempted along with setbacks faced. Since some of them were in foreign languages, the machine translated versions were relied upon for understanding and execution. A few of these were discussed in Related Works.

Project Name	Author	Vivado Version Created (Cr) Used(U)	Board Used	Chip Used	MIPI Hardware	Main HDL	Software Used	Status	Notes
Zybo-Z7-20-Pcam-5C-hw	Digilent	Cr:2022.1	Zybo Z7	Xilinx Zynq 7000	Digilent IP: MIPI_D_PHY_RX MIPI_CSI_2_RX	VHDL	Vitis:C++	-Able to generate bitstream -Hardware Error because this project uses OV5640	-Bitstream generated -uses OV5640 Camera -github
Zybo-Z7-20-HDMI-hw	Digilent	Cr:2022.1	Zybo Z7	Xilinx Zynq 7000	N/A	VHDL	Vitis:C	-Able to generate bitstream and see the HDMI passthrough operation	-Board is a passthrough from HDMI -github
IMX219-83-Stereo-Demo-V2	Waveshare	N/A	Raspberry Pi	N/A	Microprocesor	N/A	C++		-This folder has the driver files provided by Waveshare via Google Drive link
StereoNinjaFPGA	Michael Feldmeier	N/A	ULX3S	Lattice ECP5	Custom MIPI RX using Lattice Prmitives	Verilog	C++	-have not attempted bitstream generation -Need open source tools like verilator and tools to do makefiles.	-Only project here that has stereovision -Uses 2 Raspberry Pi Cams (Sony IMX219 Sensors) -Documentation is in German and working on a translated version
Sobel_zybo_z7	Adam Taylor	Cr:2018.3 U:2017.4,2022.1	Zybo Z7	Xilinx Zynq 7000	N/A	VHDL	Vitis,SDK:C	-implementation failed in image_filter block in 2022.1	-using an Apeman 1080P HDMI Camera applies Sobel Filter via HLS and outputs to monitor -HLS Sobel is in Hackster folder in zipped format in \Sobel\solution1\impl\ip -article title: "FPGA-Based Edge Detection Using HLS"
MIPI_HDMI_IMAGER	Adam Taylor	Cr:2018.3 U:2017.4,2022.1	Zybo Z7	Xilinx Zynq 7000	Digilent IP: MIPI_D_PHY_RX MIPI_CSI_2_RX	VHDL	Vitis:C	-Synthesis failed in MIPI_CSI_2_RX	-Article:"Building a Camera/Imager Test Platform"
zybo_z7_imx219_hdmi zybo_z7_imx219_hdmi_lpf	Ryuji Fuchikami	Cr:2019.2 U:2019.2	Zybo Z7	Xilinx Zynq 7000	Xilinx: MIPI_D-PHY Custom: jelly_mipi_csi2_rx	Verilog	C++	-Successful bitstream generation -Have not carried out SW part.	-This is not baremetal. Requires a linux (Debian) distribution where makefiles need to be executed -Youtube channel is Ryuz88 and his github https://github.com/ryuz/jelly/tree/v2.0.2 -His documentation is all in Japanese and need to be translated
ultra96v2_imx219_to_displayport	Greg Taylor	Cr:2021.2 U:2022.1	Ultra96	Xilinx Zynq UltraScale+	Xilinx: MIPI_CSI-2 Rx Subsystem	Verilog	Vitis:C	-Successful bitstream generation. -Do not have hardware to run it.	-github handle: gtaylormb -This project is meant for Ultra96 or KV260 boards -Does not use HDMI as output but displayport via the Zynq Ultrascale processor.

Figure 3.4.4: A list of public projects that were attempted with descriptions, results and issues faced

3.5 Final Design

Two Digilent Pcam-5C camera modules[25] were purchased to test the switching operation first. However, the connections were modified for both hardware switching and software-based switching. for the former, 1 slide switch (PL switch) and LEDs are used. Their internal wiring is shown in Figure 3.2.2c. The single slide switch is used to select cameras A or D and LEDs illuminate their 3 digit GPIO designation from the camera selection table in Figure 3.3.3b. For example when camera A is selected, the code, in binary, is "100" (mirrored version of corresponding value in the table), so only LED 3, the Enable 2 (MSB) bit, will be lit. Camera D was chosen because the bracket for camera B was damaged and it was easier to wire to port D and test. When the system with MCA in the middle was run before any configuration of the cameras, camera A always worked by default. It seemed that Channel 0 in Figure 3.3.4b was the initial value the I2C multiplexer was set to.

The Pcam-5C is a bigger module than RPi Camera Module 2 because the image sensor, OV5640, includes a micocontroller and on-chip ISP functions integrated in the sensor chip itself. There is also an option to output the video as a parallel (DVP) stream rather than a MIPI serial stream. However, this module is 5 Megapixels and smaller than the IMX219.

3.5.1 Digilent Pcam-5C Camera Module

The Pcam-5C is an imaging module meant for use with FPGA development boards. This sensor includes various internal processing functions that can improve image quality, including Auto White Balance (AWB), automatic black level calibration, and controls for adjusting saturation, hue, gamma and sharpness. Data is transferred over a 2 Lane CSI-2 interface, which provides enough data bandwidth to support common video streaming formats such as 1080p (at 30 FPS) and 720p (at 60 FPS). The module is connected to the FPGA development board via a 15-pin FFC. The Pcam-5C comes with a 10 cm flat-flexible cable and a factory-installed fixed focus lens with M12 lens mount, so it is ready to use out of the box.

Similar to previous camera module, the specifications for Pcam-5C are listed below in Table 3.5.1.

3.5.1.1 Omnivision OV5640 Image Sensor

It is helpful to understand the image sensor used in this project. The block diagram of the OV5640 is presented in Figure 3.5.2a. The OV5640 sensor core generates streaming pixel data at a constant frame rate, indicated by H_{REF} (Equation 2) and VSYNC. The timing generator outputs signals to access the rows of the image array, pre-charging and sampling the rows of the array in series. In the

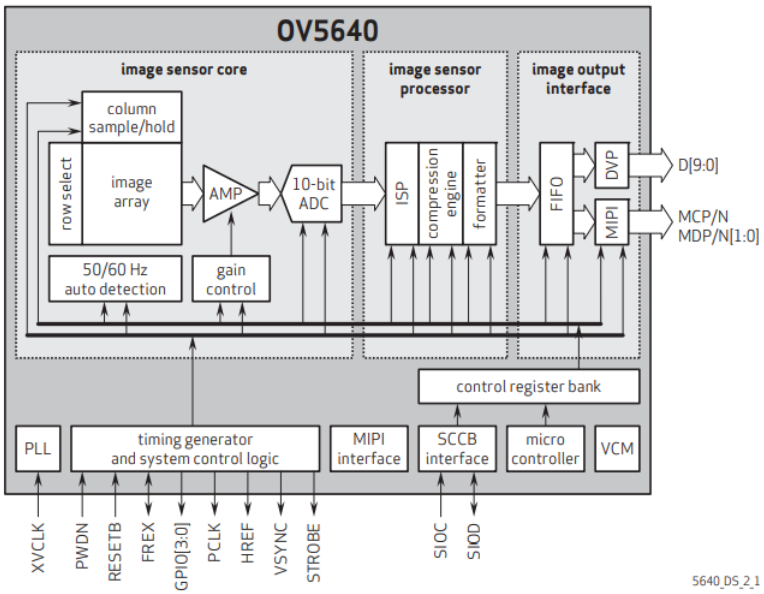
Figure 3.5.1: The Digilent Pcam-5c Camera Module

Feature	Specification
CMOS Size	1/4 in.
Pixels	5 Megapixels
Output Format	RAW10, RGB565, CCIR656, YUV422/420, YCbCr422, JPEG compression
Resolution	2592 x 1944 (Active pixels)
Input Clock Frequency	6 $\sim$ 27 MHz
Maximum image transfer rate	1080p: 30 FPS, 720p: 60 FPS
Sensitivity	600 mV/Lux-sec
Image Area	3673.6 x 2738.4 μm^2
Dark Current	8 mV/s @ 60° C junction temperature

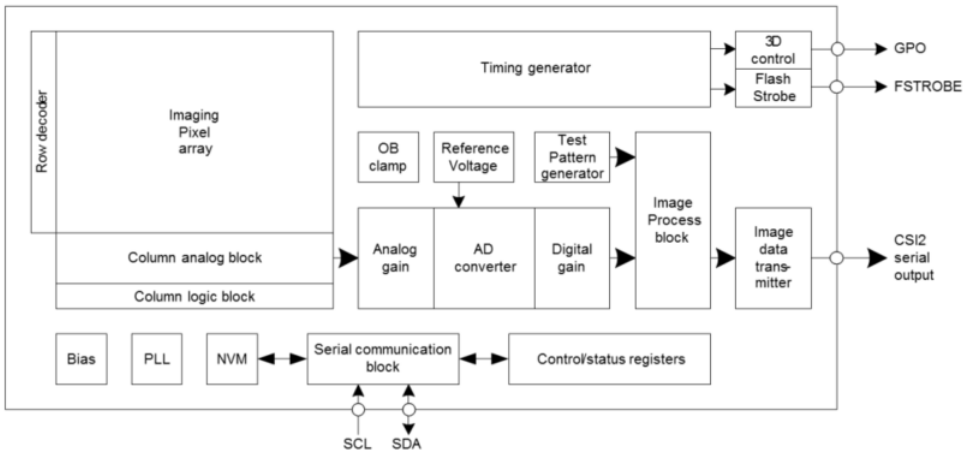
Table 3.5.1: Specification for Digilent Pcam-5C Camera module

time between pre-charging and sampling a row, the charge in the pixels decreases with the time exposed to the incident light. This is known as exposure time.

The exposure time is controlled by adjusting the time interval between pre-charging and sampling. After the data of the pixels in the row has been sampled, it is processed through analog circuitry to correct the offset and multiply the data with corresponding gain. This rolling shutter readout process is explored further in Appendix B. Following analog processing is the ADC which outputs 10-bit data for each pixel in the array. OV5640 provides full-frame, sub-sampled, windowed or arbitrarily scaled 8-bit/10-bit images in various formats via the control of the Serial Camera Control Bus (SCCB) interface. SCCB is functionally equivalent to I2C. The ISP module provides lens correction, gamma, de-noise, sharpen, auto focus, etc. The embedded microcontroller, which can be combined with an internal autofocus engine and programmable GPIO for external autofocus control. The microcontroller will not be used in this application.



(a) The block diagram for OV5640[26]



(b) The block diagram of IMX219[7]

Figure 3.5.2: Comparison of block diagrams of popular image sensors OV5640 and IMX219

In Figure 3.5.2, a side by side comparison of both the image sensors discussed here reveals the simpler architecture of the IMX219 with limited ISP functions and far less output signals from the timing generator but with more pixels than OV5640. There is no microcontroller on the IMX219.

3.5.2 Hardware Switching Design

With this design the Pmod was used to communicate with MCA. The pins in the MCA that are connected are 7, 11 and 12 as shown in Figure 3.3.2. One slide switch is used to switch between cameras A and D while the LEDs aid in identifying from which port the video stream is coming from as explained in section 3.5. Two Pcam-5C cameras are attached to the MCA as shown. The successful demonstration's implementation is discussed in the next chapter.

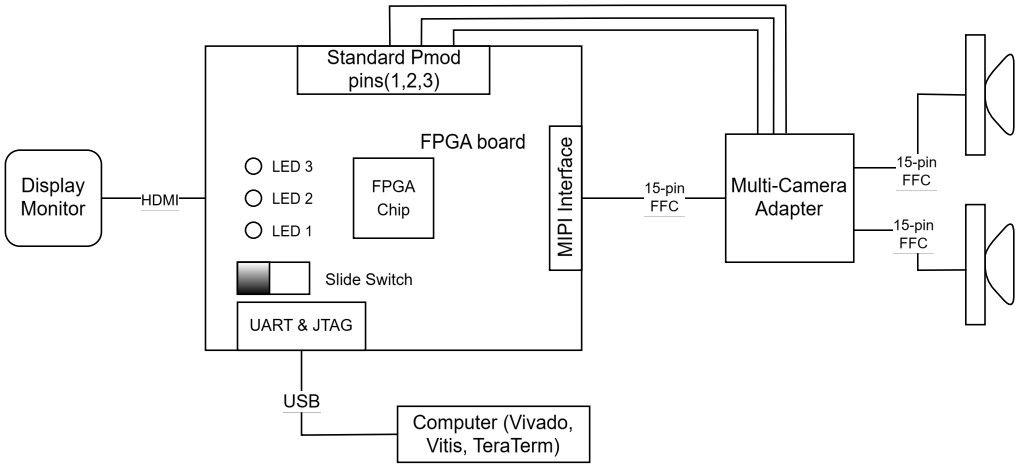


Figure 3.5.3: The final design layout for hardware switching. One slide switch and 3 LEDs were employed for selection of camera and confirmation of the selection

3.5.3 Software Switching Design

Once hardware switching is completed the next step was to migrate the platform to software switching for a better user experience as this was achieved on the terminal of the host PC. TeraTerm was used to select which camera was selected. The slide switch and LEDs were no longer needed and that information was transmitted to the terminal program, (TeraTerm). The software application, written in C++, was modified for the operation. The successful implementation of this setup is discussed in the following chapter.

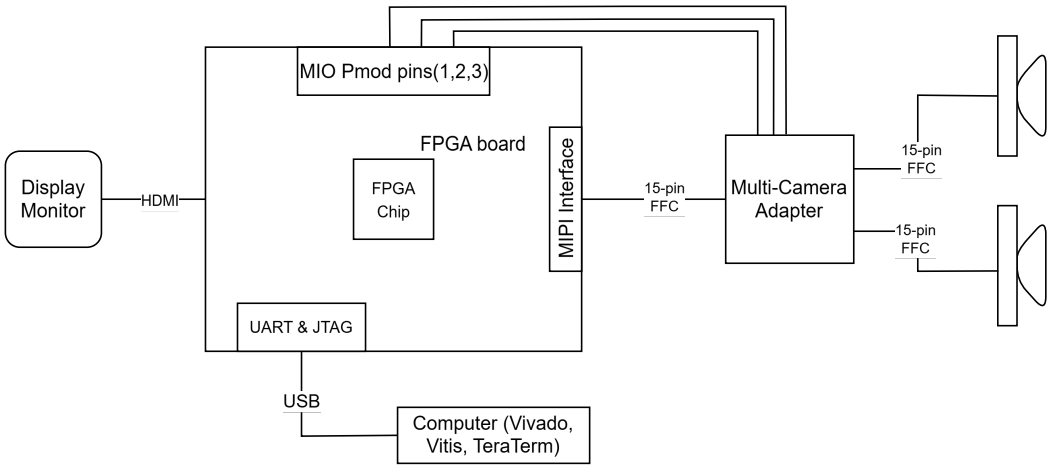


Figure 3.5.4: The final design layout for software switching. The slide switch and LEDs are replaced by an interactive prompt running on the TeraTerm program where selection and confirmation are displayed

Chapter 4

Implementation

With the Pcam-5C cameras selected, Digilent provided software and hardware support for this module that greatly reduced development time. Now all that was left to do was to integrate the MCA into the image capture system, configure the cameras and demonstrate switching. Executing this operation, entailed understanding the Digilent implementations in both software and hardware domains and COTS devices (explored in Chapter 3) in order to surgically modify the platform for demonstration. This work, navigating through these complex systems, forms the crux of this project.

The implementation was chosen to be done in Vivado 2022.1 and Vitis 2022.1. The hardware and software implementations are in zip files, Zybo-Z7-20-Pcam-5C-hw.xpr.zip and Zybo-Z7-20-Pcam-5C-sw.ide.zip, respectively and were downloaded from Digilent[3].

4.1 Hardware Implementation

The hardware implementation is done using the block diagrams or IPI in Vivado. The full diagram with ILAs added is shown in Figure 4.1.1. The diagram is rather compressed and to identify individual blocks more clearly a numbering scheme is introduced. The blocks are naturally arranged from one column to another in total of 10 columns with 1 on the left and 10 at the far right. The Zynq PS, for example, is located at 10th column and 4th block down, that will be given a coordinate of (10,4). The coordinate pair will be used to address the blocks in the figure. This FPGA system is going to have 1 MIPI input path and a common video output path (HDMI).

MIPI stream IN - Connected to J2 in 3.2.2b

HDMI Video Out - Connected to header J8 (page 5 in [18])

Internally, the FPGA architecture will use the following IP blocks from Digilent Video Library[1]. The blocks from AMD will be specified and interconnects and ILAs will not be included in list below for brevity.

1. **RGB2DVI** (10,3) - Conversion from the parallel output video format to HDMI
2. **PS** (10,4) - The software interface around the Zynq®-7000 platform Processing System. The Zynq-7000 family consists of SoC style integrated PS and PL, providing an extensible and flexible SoC solution on a single die. The Processing System 7 core acts as a logic connection between the PS and PL.

[3] https://digilent.com/reference/programmable-logic/zybo-z7/demos/pcam-5c?srsltid=AfmBOoo_yOq-Wqn4vCFesMwYojcyP59-j3UlXokr_5gzP2OY-umSUqc3

[1] https://github.com/Digilent/vivado-library

3. **AXI Stream to Video Out** (9,1) - An AMD IP Block, this is the top level module for the AXI4-Stream to Video-Out bridge. The bridge is used to convert AXI4-Stream input to parallel video by synchronizing to the Video Timing Generator (VTG) input signals and outputting to the RGB2DVI block above.

4. **Concat** (9.2) - AMD block that concatenates the 3 Interrupt signals: 1 from VTG, 2 from AXI Video DMA (VDMA).

5. **Video Timing Controller/VTG** (8,1) - AMD block Used to detect and generate timing.

6. **AXI VDMA** (8,2) - AMD block stores the video frames in the PS DDR memory.

7. **Processor System Reset** (7,1) - AMD block soft IP that handles reset conditions for the dynamic clocking wizard at (4,1).

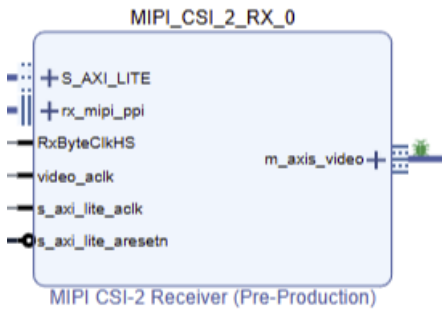
8. **AXI Gamma Correction** (7,2) - This component performs image Gamma correction. The gamma correction factor is specified in the "StoredGammaCoefs" subcomponent header comments. The component receives 1 pixel at a time, through its AXI Stream Slave interface. It outputs one pixel at a time, on the AXI Stream Master interface.

9. **Processor System Reset** (7,3) - AMD block soft IP that handles reset conditions for the clock wizard generating 150 MHz.

10. **DVIClocking** (6,1) - A VHDL module generating Pixel and Serial clocks.

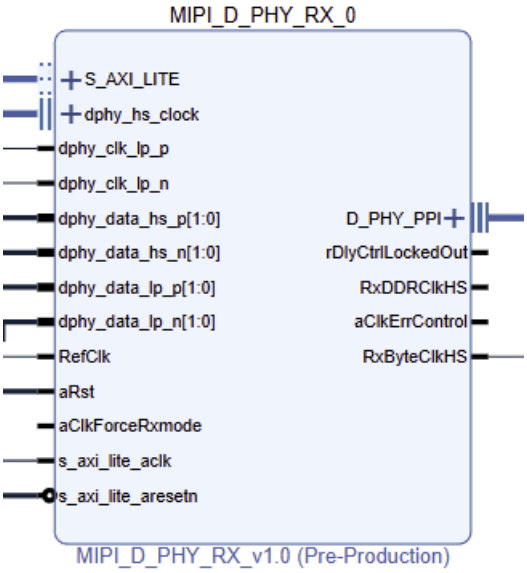
11. **AXI Bayer to RGB** (6,2) - Converts raw image to a full colour image.

12. **MIPI CSI-2 Receiver** (5,1) - This IP is compatible with CSI-2 1.0 specifications and supports decoding selected pixel formats and packing data into an AXI-Stream. It pairs up with a D-PHY Receiver IP over the standard PPI to source a video subsystem. The IP has been tested in dual-lane configuration with an 84 MHz PPI high-speed byte clock (RxByteClkHS) and 150 MHz AXI-Stream clock (video_aclk) - see diagram below. Has single or dual lane support for RAW10, and 4 pixels per beat AXI-Stream output for high bandwidth applications [27].



13. **AXI Dynamic Clock** (4,1) - AMD block configured as AXI slave for dynamic clock generation.

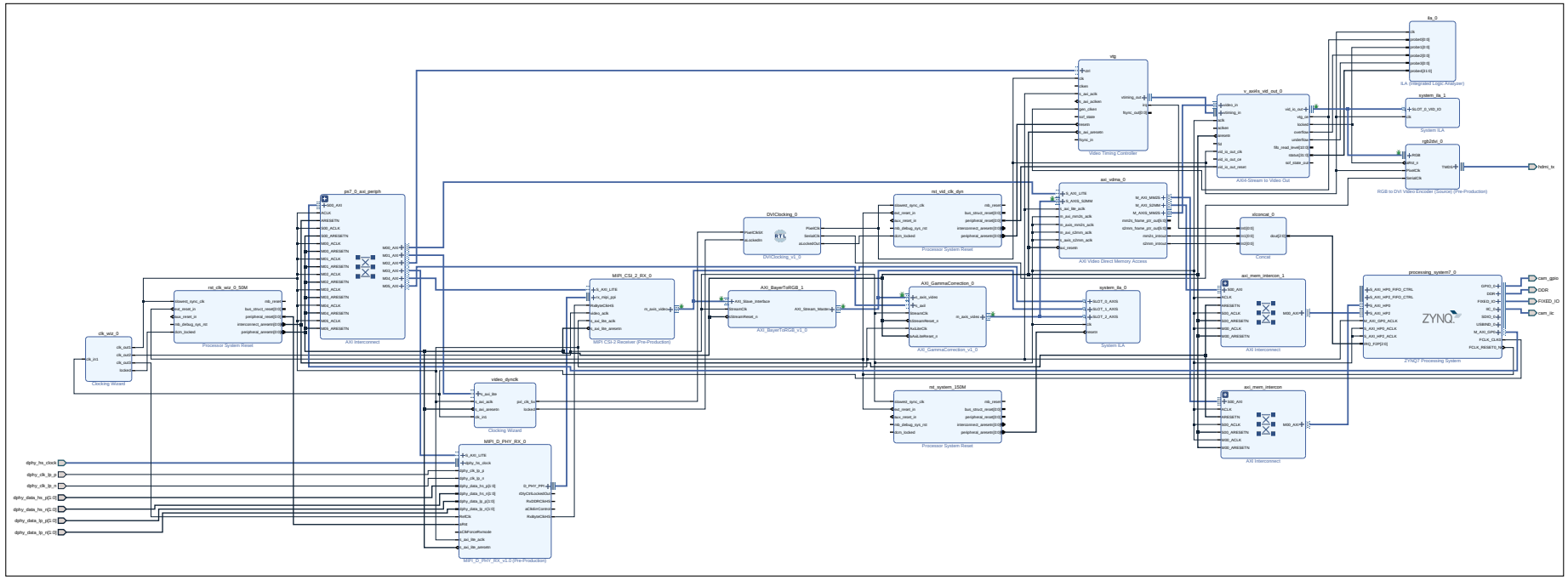
14. **MIPI D-PHY Receiver** (4,2) - This IP is an implementation of a subset of the PHY-level protocol of the D-PHY 1.0 specification. It bundles a SCNN (negative side of differential clock pair) clock lane and one or more SFEN (negative side of differential data pair) data lanes to implement the receiver end of a Lane Interconnect. In an FPGA implementation of the communication stack it occupies the lowest level. On top of it, over the PPI connects the protocol layer specific to the application, like the CSI. IP has been tested in dual-lane configuration with 1344 Mbps total data rate, resulting in $(\frac{1344}{8\ bits\ per\ byte} \div 2\ edges\ per\ cycle =)$ 84 MHz D-PHY high-speed byte clock (RxByteClkHS) - see close-up diagram below[28].



15. **Processor System Reset** (2,1) - AMD block soft IP that handles reset conditions for the clock wizard generating 50 MHz.

16. **Clocking Wizard** (1,1) - AMD block that generates 3 clock signals from the 100 MHz coming from PS: 150, 50 and 200 MHz.

Figure 4.1.1: The full block diagram of the PL design in the IPI section of Vivado



4.1.1 Bandwidth Calculations

Since Pcam-5C was chosen as the camera of choice, its timing parameters are used to calculate the bandwidth required by the system. The maximum image transfer rate of 30 FPS is used for a resolution of 1920 x 1080p (Table 3.5.1). The timing specifications are shown in Figure 4.1.2. The unit tp is pixels, t**p**.

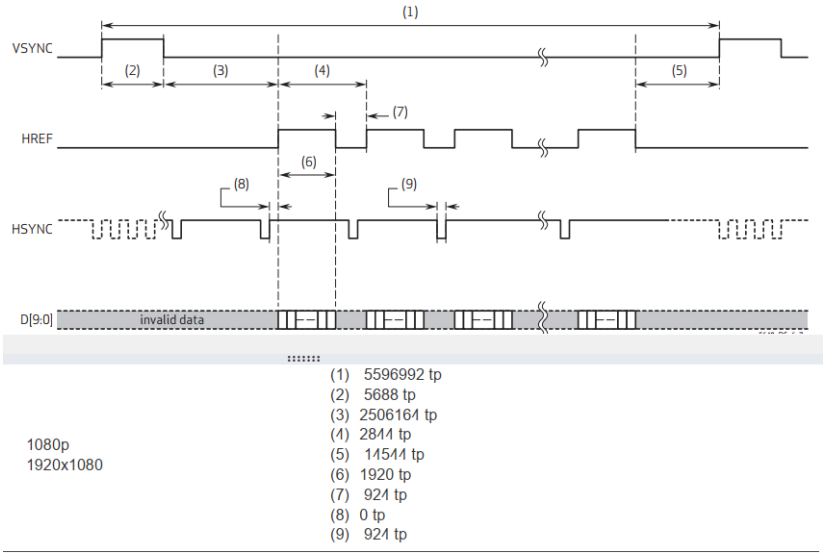


Figure 4.1.2: Timing specification for 1080p resolution on the Pcam-5C[26]

From equations (1) and (2) (not to be confused with the equations, the values in parenthesis in calculation blocks below represent their respective identifiers shown at bottom half of figure above),

$$HBI = (8) + (9) + 0 \ (assumed\ no\ HFP) = 0 + 921 = 921 \tag{1a}$$

$$H_{TOT} = 1920 + HBI = 1920 + 921 = 2841\ p \tag{2a}$$

Next, equations (3) and (4) are in units of lines, while the units in Figure 4.1.2 in pixels. Therefore to get number of lines, l, VSYNC, VBP and VFP are divided by H_{TOT}. However to calculate VBP + VFP term in (3), HSYNC is subtracted as follows[3]:

$$\begin{aligned} VBI &= \frac{(VFP + VBP) - HSYNC + VSYNC}{H_{TOT}} \\ &= \frac{(5) + (3) - 921 + (2)}{2841} \\ &= \frac{14544 + 2506164 - 921 + 5688}{2841} \approx 889 \end{aligned} \tag{3a}$$

$$V_{TOT} = 1080 + VBI = 1080 + 889 = 1969\ l \tag{4a}$$

Now plugging in the values to get PCLK in equation (5),

$$PCLK = 2841 \times 1969 \times 30 \approx 168MHz \tag{5a}$$

leads to

$$Bandwidth\ (BW) = 168 \times 10^6 \times 10\ bpp = 1680\ Mbps \tag{6a}$$

in RAW10 format. And since 2 D-PHY lanes are used, according to equation (7)

$$DRPL = \frac{1680 \times 10^6}{2} = 840\ Mbps \tag{7a}$$

which finally gives a MIPI_CLK of, according to equation (8)

$$MIPI\_CLK = \frac{840 \times 10^6}{2} = 420\ MHz \tag{8a}$$

because this clock is always half the data rate due DDR operation. This MIPI_CLK along with PCLK are configured onto the image sensor for proper operation of image capture at the receiver. Notice how the MIPI_CLK is bigger than the PCLK and this always the case since the image data is serialized and sent over differential lanes. The former is known by various names in different implementations such as CLKA, DPHY_ClkHS in Figures A.1.4 and 5.1.1, respectively. Due to the source-synchronous nature of the D-PHY there is no need for recovery circuits at the receiver end, simplifying circuit design. In the full implementation shown in Figure 4.1.1, there are many clocks used to process and display the images. The main clock is supplied by the Zynq PS block which is used by the clocking wizard to generate 3 more clocks for D-PHY, image processing, AXI Lite commands. A dynamic clock generator is used for the HDMI output subsystem to display the images for different frame rates as desired by the user. These clocks and their connections are specified in Figure 4.1.3. A glitch filter module is used on all inputs to the clock or data lanes at the receiver to filter out any unnecessary pulses caused to transitions as explained in section A.2.3.

4.1.2 Hardware Switching

Once the path of configuration was established with the I2C Multiplexer and its address gleaned from related websites (encircled in Figure 3.3.3a), the switching operation using hardware slide switches connected to the PL was attempted with two Pcam-5Cs connected to port A and D in the MCA as shown in Figure 3.5.3.

The hardware switching wiring is shown in Figure 4.1.4. This makeshift model is a drastic shift from the plastic, self-contained unit that was devised earlier so that the black header pins can be accessed on

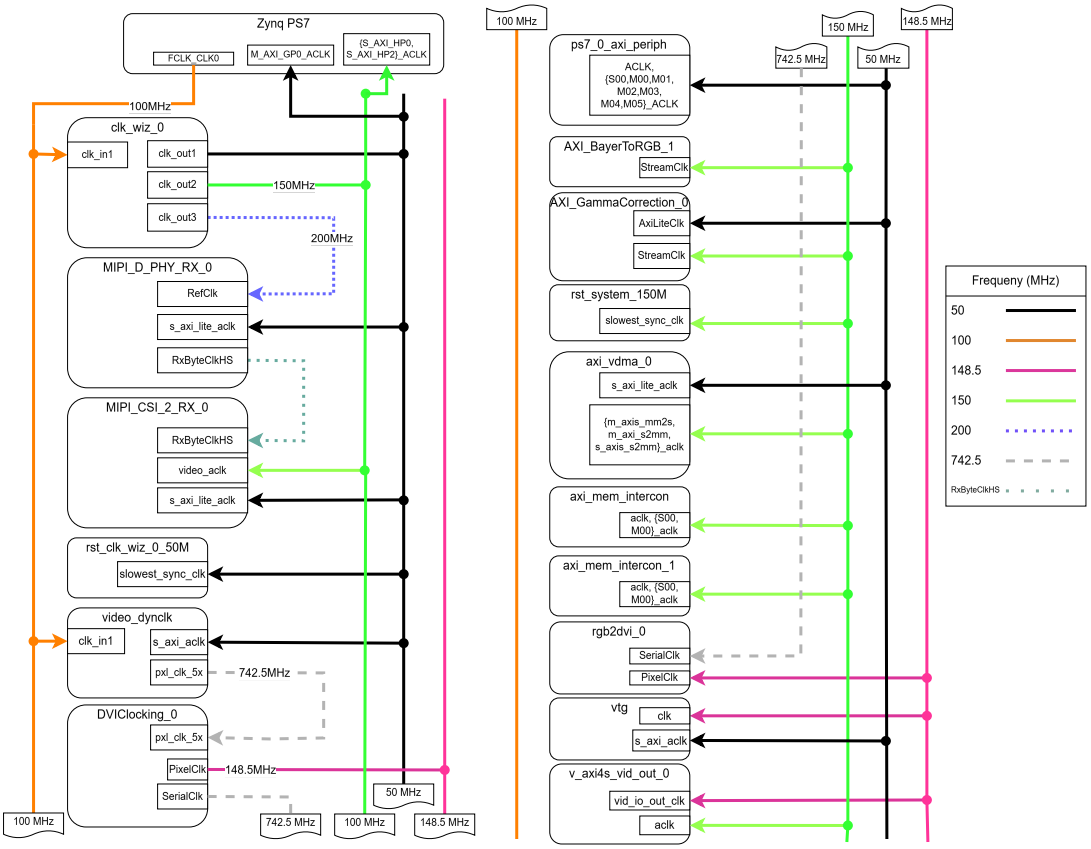


Figure 4.1.3: Different clocks and their connections to IPs in Figure 4.1.1

the back side of the MCA. The GPIO signals are sourced from the standard Pmod interface (Figure 3.2.5) as indicated by the yellow arrow in the diagram. The slide switch used to control the camera selection is the first of the four from the right in Figure 3.2.1. The LEDs are above slide switches.

The original hardware code was modified to include the changes shown in Listing 4.1.1. The snippet describes a combinational circuit that is sensitive to the signals in the its parameter list, where the LEDs are lit according to the camera selected from the signals that leave the standard Pmod interface. The changes to the original software, written in C++, are visualized in Figure 4.1.5, where the entire system is mapped beginning with *main.cc* file in the middle. The diagram is complex but it has been coloured to indicate the changes made to the specific submodules relevant to make hardware switching work.

The files and classes that contain the modifications are shown in red and the functions and other additions within are shown in blue. The bold, black arrows indicate the relationship of interest between the modified components. The address of I2C Multiplexer is incorporated in the private portion of class OV5640, shown in Listing 4.1.2. *pw544a* is another designation for I2C IP chip TCA9544A. The next step is create a function that can use this address to select desired channel for the camera. Hence, *write_mux* was created (Listing 4.1.3). The right argument given to this function is the I2C channel

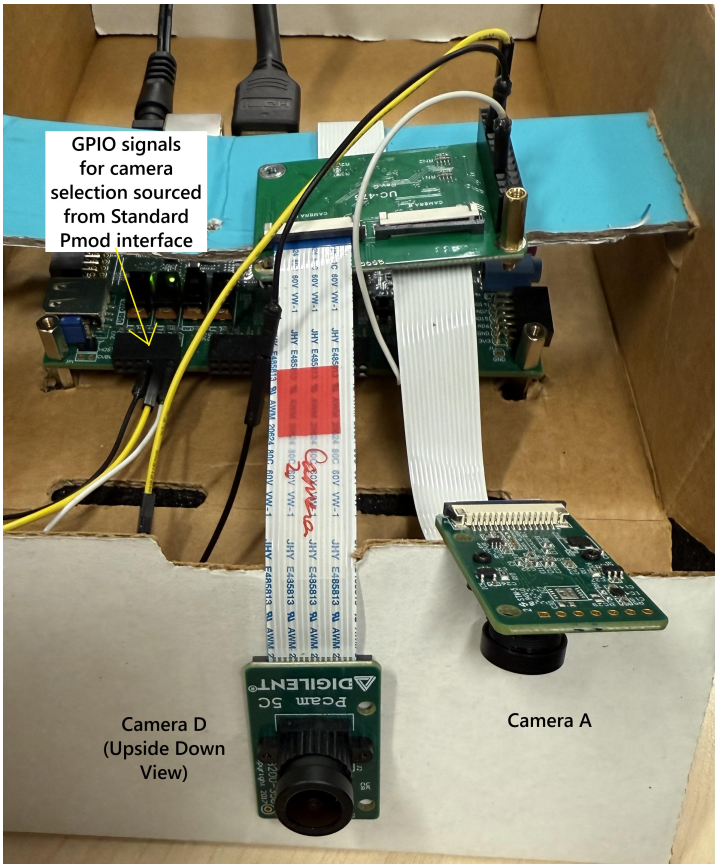


Figure 4.1.4: A snapshot showing the hardware architecture wired to the MIO Pmod interface in the FPGA board

designation shown in Figure 3.3.4b and should not be confused with the GPIO configuration in Figure 3.3.3b. There is no need to specifically call *write_mux* for Camera A since it is the default choice for the MCA. Hence only the one for port D is called in line 2 of Listing 4.1.4. This hexadecimal value to set port D (0x7) should also not be confused with the I2C Multiplexer address of 0x70.

```
        signal modeA : std_logic_vector(2 downto 0) := "100";   --Camera A code mirrored
        signal modeD : std_logic_vector(2 downto 0) := "011";   --Camera D code mirrored
    begin

        process(switch, modeA, modeD)
        begin
            if (switch(0) = '0') then
                led <= modeA;
                pmodCtrl <= modeA;
            else
                led <= modeD;
                pmodCtrl <= modeD;
            end if;
        end process;
```

Listing 4.1.1: Snippet of code modified from the declarative region of the top VHDL file, *system_wrapper.vhd*

```
    uint8_t pw544a_slv_addr_ = 0x70;
```

Listing 4.1.2: The I2C address for the Multiplexer is added here in private portion of class OV5640 for configuration via FPGA

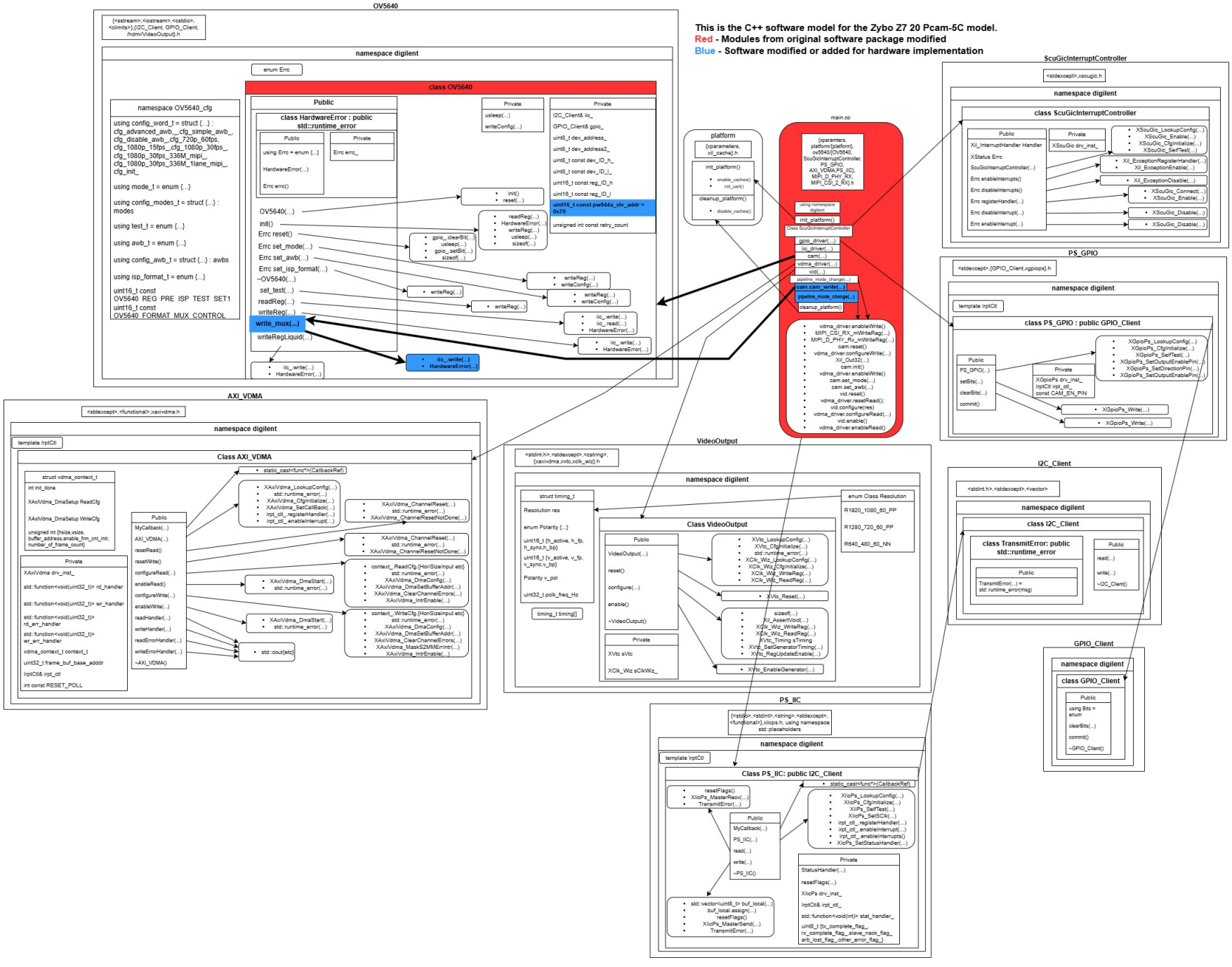
```
                void write_mux(uint16_t reg_addr, uint8_t const reg_data)
        {
                for(auto retry_count = retry_count_; retry_count > 0; --retry_count)
                {
                        try
                        {
                                auto buf = std::vector<uint8_t>{(uint8_t)reg_addr,reg_data};
                                iic_.write(pw544a_slv_addr_, buf.data(), buf.size());
                                break; //If no exceptions, no mo retries
                        }
                        catch (I2C_Client::TransmitError const& e)
                        {
                                if (retry_count > 0) continue;
                                else throw HardwareError(HardwareError::IIC_NACK, e.what());
                        }
                }
        }
```

Listing 4.1.3: Void function *write_mux* added to class OV5640 to select Camera address for configuration

```
        pipeline_mode_change(vdma_driver, cam, vid, Resolution::R1920_1080_60_PP,
        ↪  OV5640_cfg::mode_t::MODE_1080P_1920_1080_30fps);
        cam.write_mux(0x0, 0x7);
        pipeline_mode_change(vdma_driver, cam, vid, Resolution::R1920_1080_60_PP,
        ↪  OV5640_cfg::mode_t::MODE_1080P_1920_1080_30fps);
```

Listing 4.1.4: *write_mux* function of object 'cam' called to set the value to Camera D (0x7) as shown in binary in Figure 3.3.4b. 0x0 is chosen as placeholder value for the first argument. The original function *pipeline_mode_change(...)* configures camera port set by *write_mux*

Figure 4.1.5: The modifications done to the original software model to make hardware switching work



4.1.2.1 Demonstration

The results of the experimental setup for hardware switching are presented here. The screen starts to display the images by the camera set by the slide switch unlike for software switching which always begins at Camera D. The results for Camera A are shown in Figure 4.1.6 where a white highlighter (not shown) is imaged and a magnified version is displayed.

Figure 4.1.6: Demonstration of Camera D selection. A magnified video stream of a white highlighter. The highlighter is not visible but for a view of the highlighter please consult Figure 4.1.12. Note the 2 LEDs lit up

Next, the camera selection for Camera D is shown. It is imaging the calculator shown upside down due to the orientation of Camera. Two LEDs light up due to the GPIO selector code of "011" and is indicated in Figure 4.1.7.

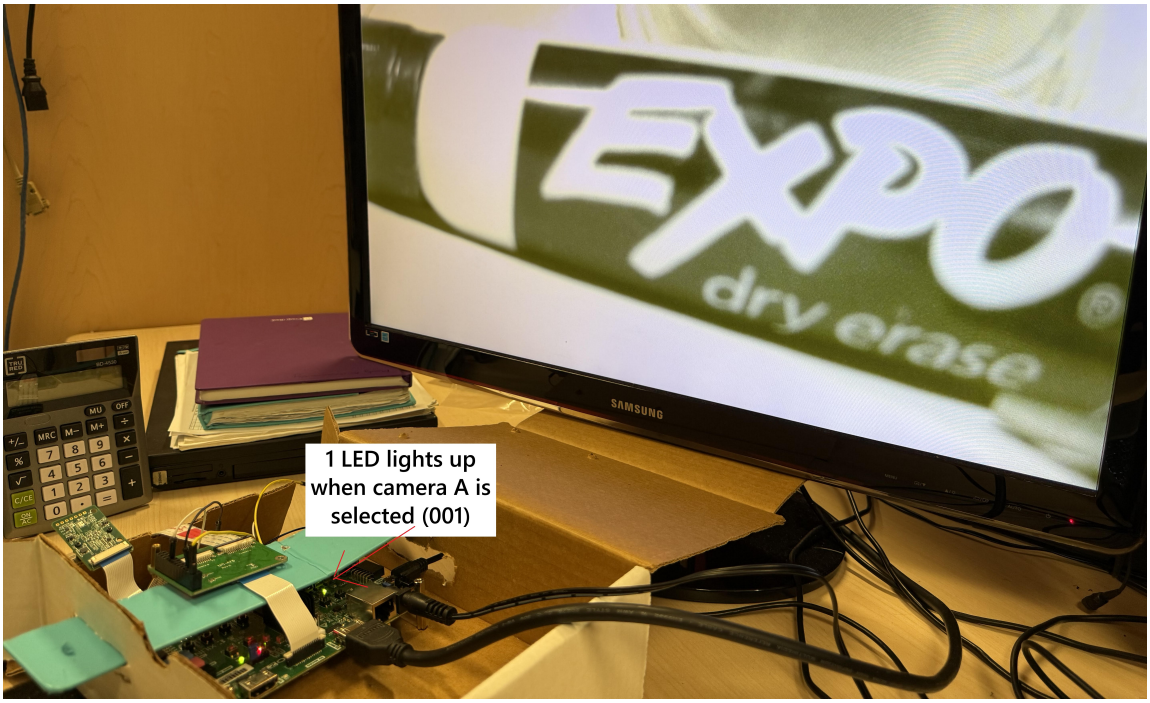


Figure 4.1.7: Demonstration of Camera A selection. A magnified video stream of white highlighter (not shown but for a view of highlighter consult Figure 4.1.12). Note the single LED lit up

4.1.3 Software Switching

The hardware architecture is shown in Figure 4.1.8. This is a makeshift prototype where the MCA is inverted and supported by a piece of cardboard on top of the Zybo Z7-20 board. It is inverted to access connections from the black header pin of the MCA. The GPIO signals are wired to Pmod interface (JF header left side in Figure 3.2.1). This interface is wired to MIO pins connected to the PS because the selection preference originates from software program running on TeraTerm.

For software switching, the original hardware code is used and GPIO control is migrated to the software program for a better control and use case. According to the connection diagram (Figure 3.5.4), MIO Pmod is used to control the camera selection. The cameras are selected via a program running on a terminal application like TeraTerm connected by USB to the FPGA board. The software modifications and connections are shown in Figure 4.1.9. The GPIO client is modified to address the Selection, Enable 1 and Enable 2 bits and called in class OV5640. More functions are added for proper display in the terminal running on the host computer where user selects desired operation.

The terminal menu, already part of the original software, is updated with items 'h' and 'i' as displayed in Listing 4.1.5. There are more changes than for hardware switching, therefore not all of them are presented except for the important ones. To differentiate, blue coloured changes for hardware switching, changes for software switching are shown in green which covers are wider spectrum of modifications. The *select_cam* function is shown in Listing 4.1.6. Other functions depicted in green are

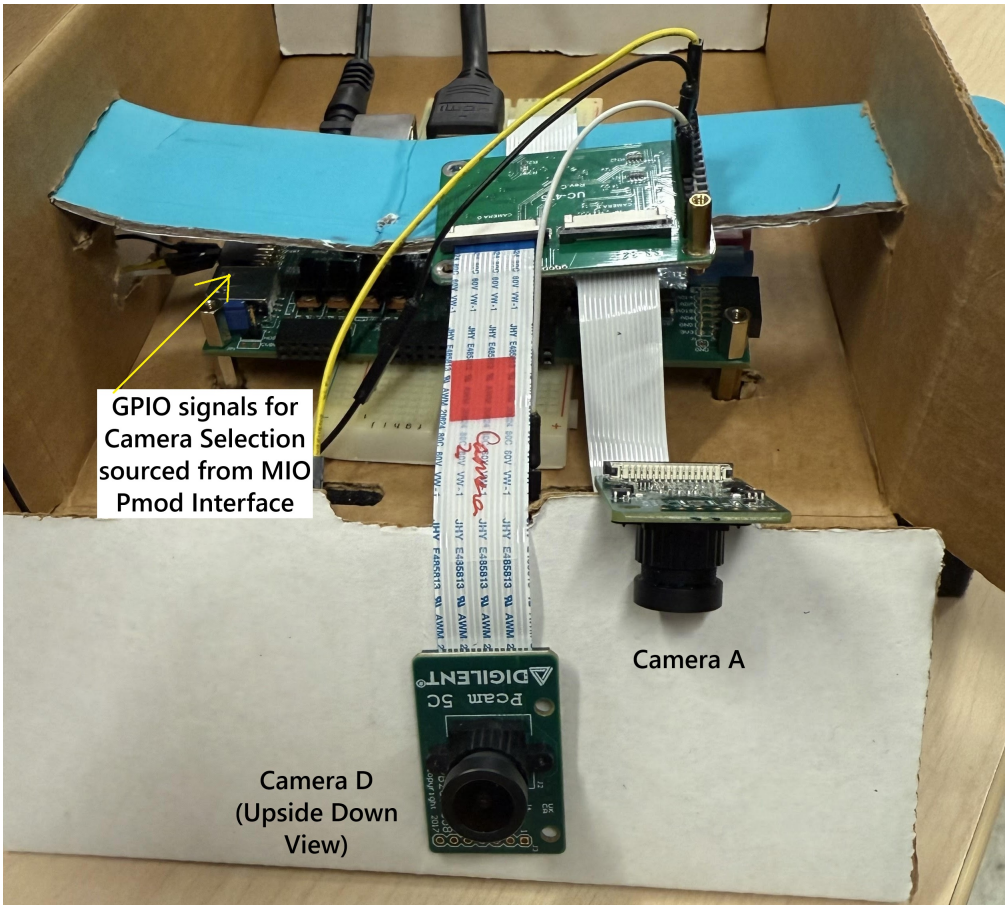


Figure 4.1.8: A snapshot showing the software architecture wired to the MIO Pmod interface in the FPGA board

not included for brevity.

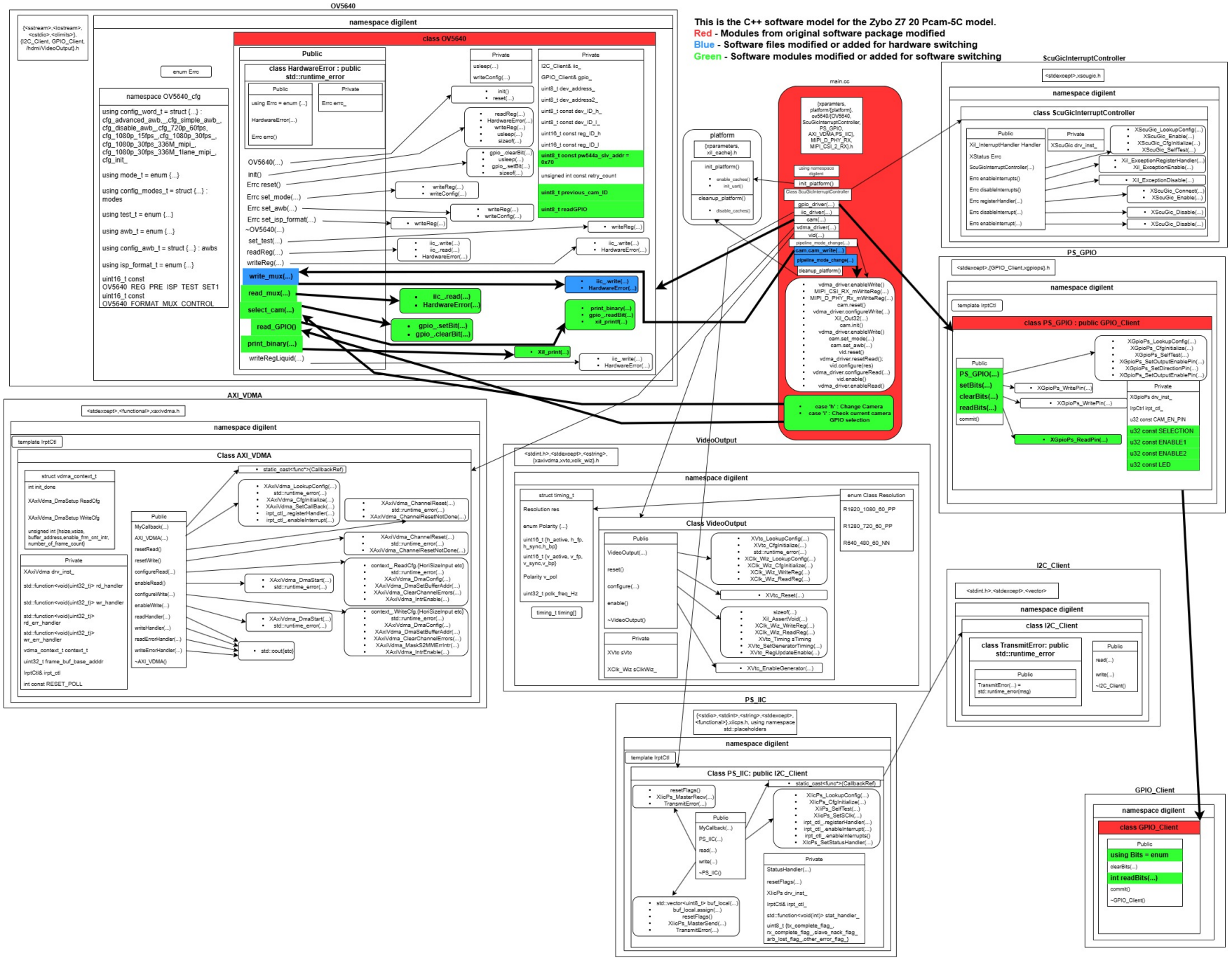
```
      case 'h':
        xil_printf("  Please press the letter (lower case) corresponding to the desired
        ↪  camera:\r\n");
        xil_printf("    a. Camera A\r\n");
        xil_printf("    b. Camera B\r\n");
        xil_printf("    c. Camera C\r\n");
        xil_printf("    d. Camera D\r\n");
        read_char3 = getchar();
        getchar();
        xil_printf("Read: %d\r\n", read_char3);
        switch(read_char3) {
        case 'a':
          cam.select_cam(read_char3);
          xil_printf("Enabled Camera A\r\n");
          break;
        case 'b':
          cam.select_cam(read_char3);
          xil_printf("No Camera B. Switching to Camera D.\r\n");
          break;
        case 'c':
          cam.select_cam(read_char3);
          xil_printf("No Camera C. Switching to Camera A.\r\n");
          break;
        case 'd':
          cam.select_cam(read_char3);
          xil_printf("Enabled Camera D\r\n");
          break;
        default:
          xil_printf("  Selection is outside the available options! Please retry...\r\n");
        }
        break;
      case 'i':
        cam.read_GPIO();
        break;
```

Listing 4.1.5: Terminal menu addition of two items, cases 'h' and 'i' for selecting the camera and displaying the confirmation about selection.

```
void select_cam(uint8_t cam_ID)
  {
    switch(cam_ID) {
    case 'a':
     if (previous_cam_ID != '\0') {
          xil_printf("Switching from Camera %C \r\n", to_uppercase(previous_cam_ID));
        }
      gpio_.clearBit(gpio_.Bits::SELECTION);
      gpio_.clearBit(gpio_.Bits::LED); //if SELECTION is 0 then clear LED
      gpio_.clearBit(gpio_.Bits::ENABLE1);
      gpio_.setBit(gpio_.Bits::ENABLE2);
      usleep(1000000);
      break;
    case 'b':  //      -- B is turned on by "101"
     if (previous_cam_ID != '\0') {
          xil_printf("Switching from Camera %C \r\n", to_uppercase(previous_cam_ID));
        }
      gpio_.setBit(gpio_.Bits::SELECTION);
      gpio_.setBit(gpio_.Bits::LED); //if SELECTION is 1 then set LED
      //No Camera is "X11"
      gpio_.setBit(gpio_.Bits::ENABLE1);
      gpio_.setBit(gpio_.Bits::ENABLE2);
      usleep(1000000);
      break;
    case 'c': //      -- C is turned on by  "010"
     if (previous_cam_ID != '\0') {
          xil_printf("Switching from Camera %C \r\n", to_uppercase(previous_cam_ID));
        }
      gpio_.clearBit(gpio_.Bits::SELECTION);
      gpio_.clearBit(gpio_.Bits::LED); //if SELECTION is 0 then clear LED
      //No Camera is "X11"
      gpio_.setBit(gpio_.Bits::ENABLE1);
      gpio_.setBit(gpio_.Bits::ENABLE2);
      usleep(1000000);
      break;
    case 'd': //      -- D is turned on by "110"
       if (previous_cam_ID != '\0') {
         xil_printf("Switching from Camera %C \r\n", to_uppercase(previous_cam_ID));
          }
      gpio_.setBit(gpio_.Bits::SELECTION);
      gpio_.setBit(gpio_.Bits::LED); //if SELECTION is 1 then set LED
      gpio_.setBit(gpio_.Bits::ENABLE1);
      gpio_.clearBit(gpio_.Bits::ENABLE2);
      usleep(1000000);
      break;
    }
    previous_cam_ID = cam_ID;
  };
```

Listing 4.1.6: Terminal menu addition of two items, case 'h' and 'i' for selecting the camera and displaying the confirmation about selection

Figure 4.1.9: The modifications done to the original software model to make software switching work



The snapshot of the improved user menu is shown in Figure 4.1.10. The top red arrow illustrates the response of case 'i' is selected after switching cameras from D to A. The value for A is read back as "001" which corresponds to the Selection, Enable 1 and Enable 2 bits discussed in Figure 3.3.3b.

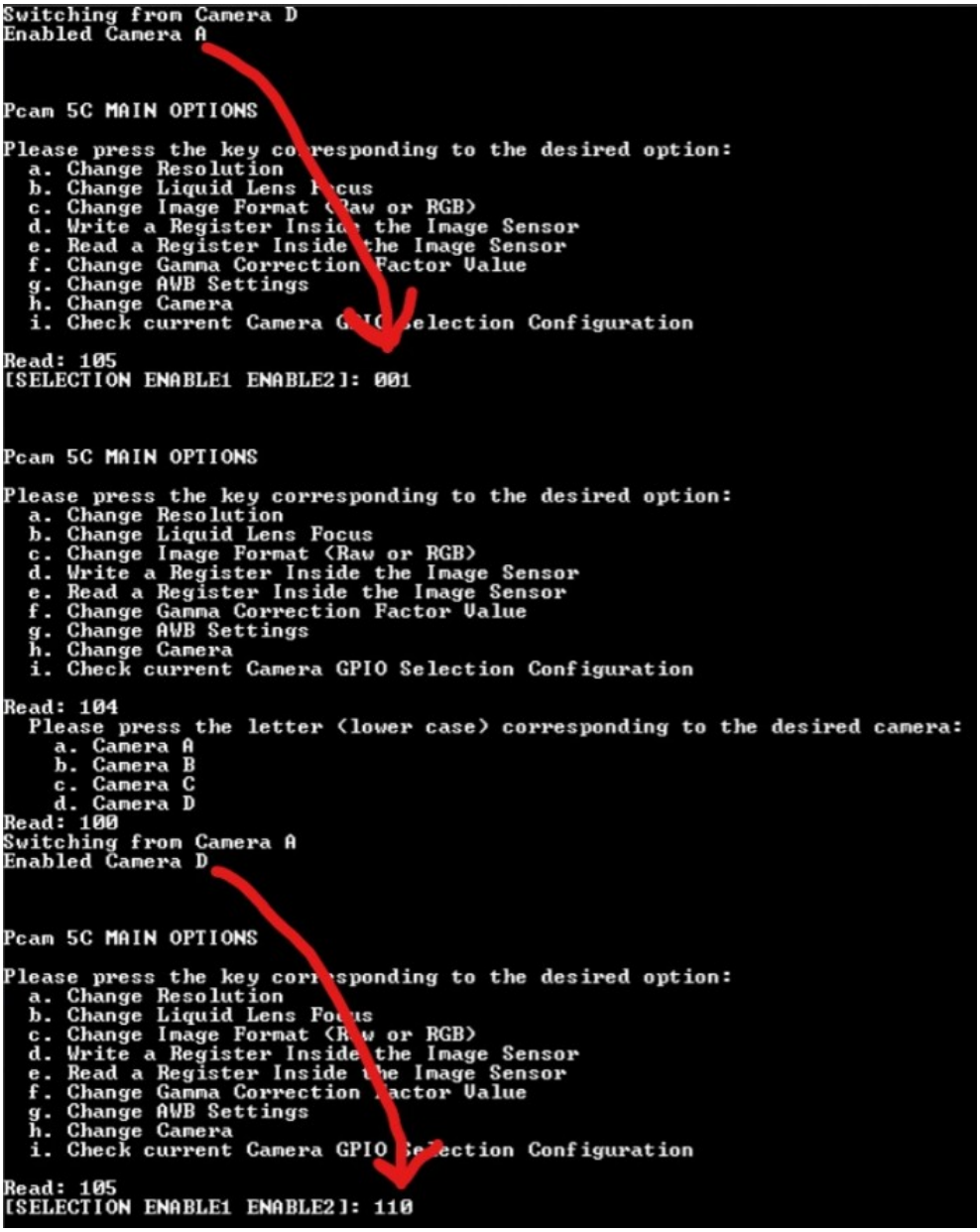


Figure 4.1.10: The improved user menu in action. Top red arrow depicts response after case 'i' is selected, while the bottom red arrow does the same when case 'h' is selected

4.1.3.1 Demonstration

The results of the switching of cameras using a menu begins with the streaming of Camera D by default because it configured last as shown in line 2 of Listing 4.1.4. A snapshot of the Camera D working is shown in Figure 4.1.11 where the upside down captured video of camera is streamed to the

screen in real-time. Its upside down due to the placement of Camera D as shown in Figure 4.1.8. The menu item selection (in very small lettering) is shown in the laptop screen on the bottom right.

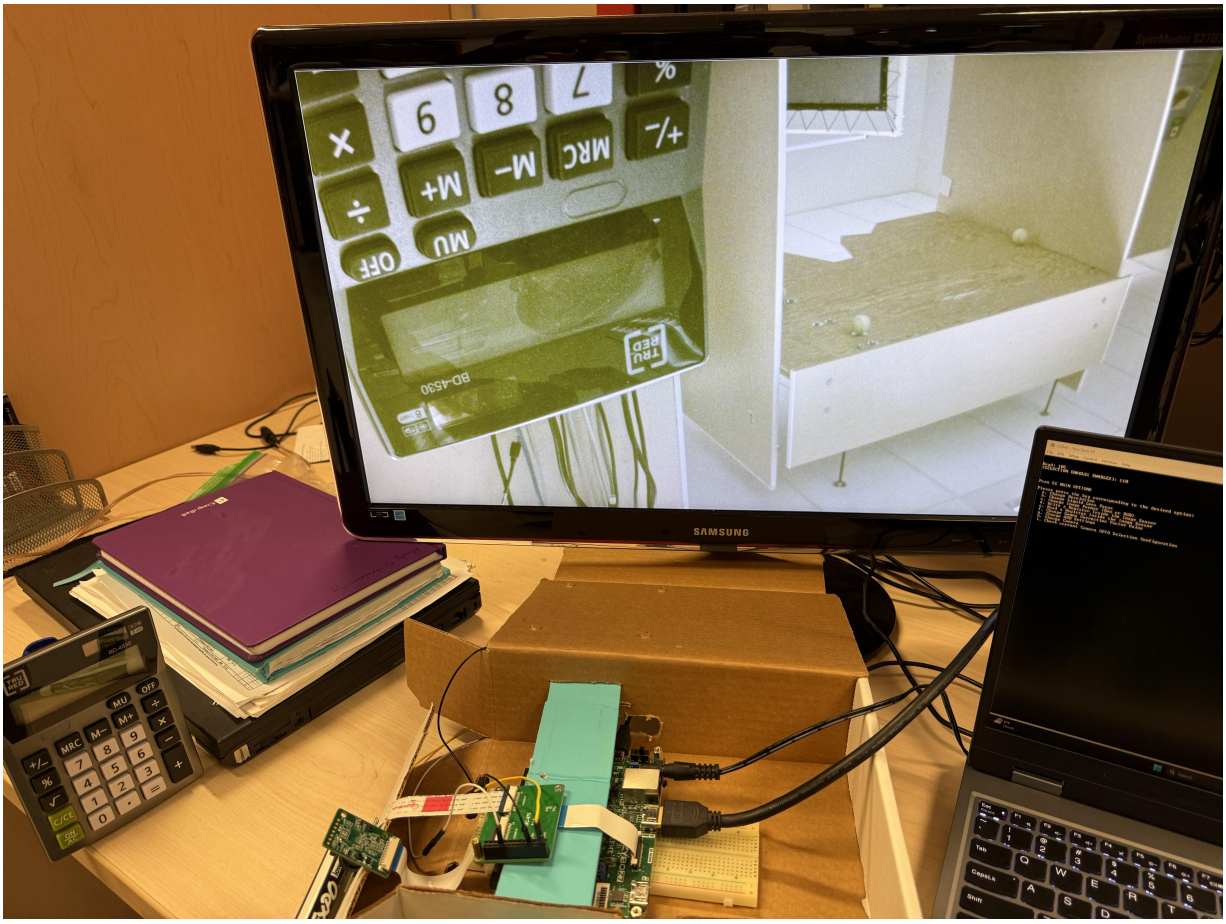


Figure 4.1.11: Demonstration of Camera D selection. What is displayed on the screen is the upside view of the calculator captured by Camera D (not visible). The case 'i' is selected confirming which camera is chosen on the right laptop screen (needs to be zoomed in)

Next demonstration (Figure 4.1.12) is switching to Camera A from Camera D that starts streaming by default when FPGA board is programmed from Vitis. The video displayed is a non-inverted, magnified white highlighter placed in front of Camera A.

Figure 4.1.12: Demonstration of Camera A selection. What is displayed on the screen is the magnified video stream of a white highlighter placed in front of Camera A at the bottom The back of Camera A is visible. Since Camera D is enabled first, the menu is used to switch to Camera A as indicated with more lettering on the laptop screen compared to Figure 4.1.11

Chapter 5

Analysis

5.1 Simulation Analysis of D-PHY

A VHDL testbench was used to simulate and verify the proper operation of the D-PHY at the receiver side. The D-PHY IP is set as the Device-Under-Test (DUT) and data vectors of 16 bits length are sent to the DUT 17 times to capture the moment it arrives out of PPI interface of D-PHY. The byte synchronizer to begin HS transfer is `0xb8` (also referred as Leader Sequence in reverse in Figure A.2.2). The first test vector is 0x78cc, which in bits, forward and reverse including byte synchronizer, translate to,

$$\texttt{0xb8} = 1011\ 1000 \mid \textit{Reverse}\ \texttt{0x1d} = 0001\ 1101$$
$$\texttt{0x78} = 0111\ 1000 \mid \textit{Reverse}\ \texttt{0x1e} = 0001\ 1110$$
$$\texttt{0xcc} = 1100\ 1100 \mid \textit{Reverse}\ \texttt{0x33} = 0011\ 0011$$

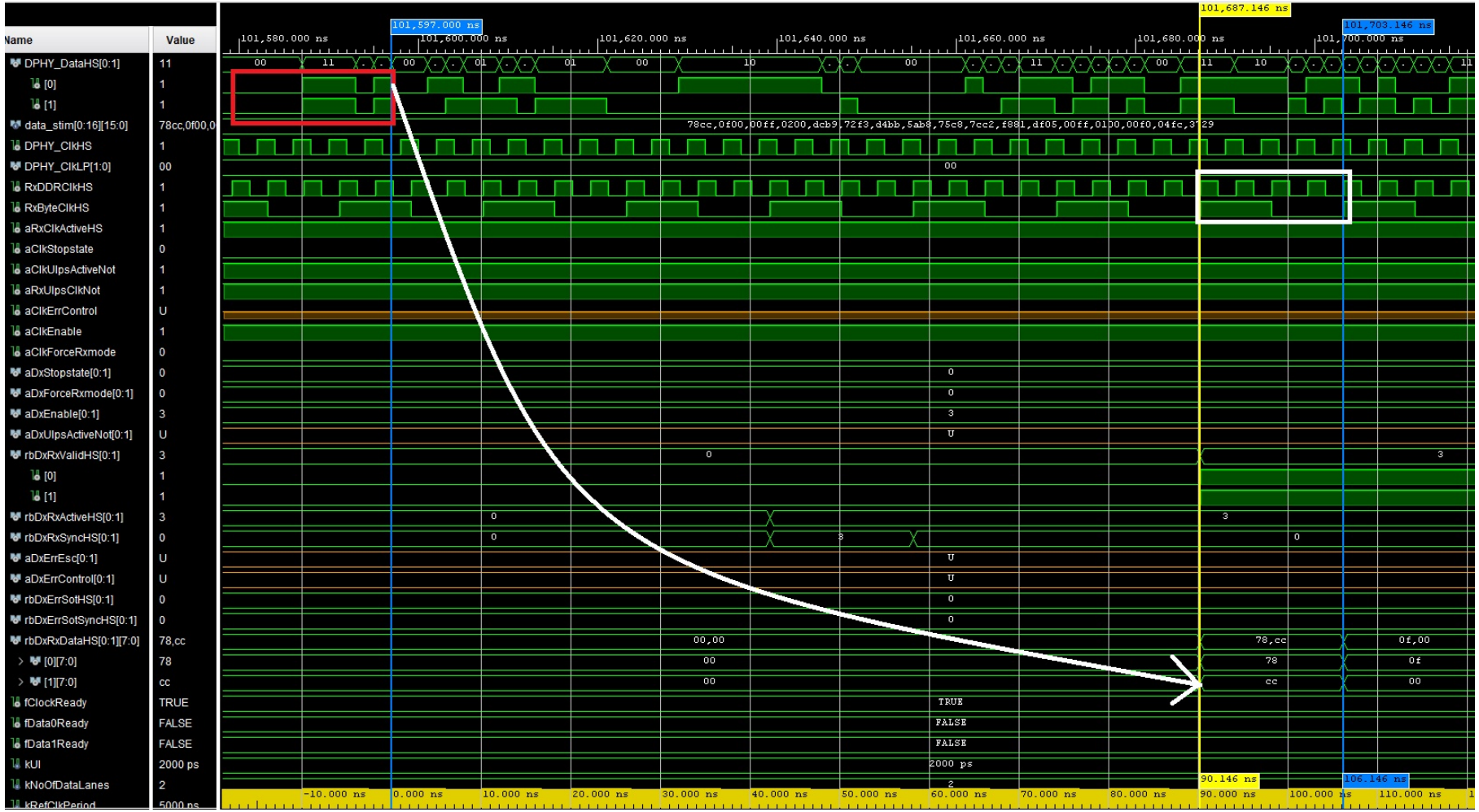
These values are annotated in Figure 5.1.1. Data is sent with a 90° phase shift with respect to all transition, rising and falling, of DPHY_ClkHS (3rd item in the figure). The red box in the top left corner is the byte synchronizer in reverse. Since 2 lanes are used the signal DPHY_DataHS is 2 bits wide. The first bit, DPHY_DataHS(0), serializes `0xcc` with LSB first and it can be seen in the stream after the red box in the first line as 0011 0011 (`0x33`). DPHY_DataHS(1) does the same for `0x78` and this can be verified by looking at that second line next to the red box.

The blue vertical line indicates the start of the data transfer in both lanes right after the byte synchronizer. The white, curved arrow points to when that data is deserialized by the D-PHY module and available on the PPI. The yellow vertical line is the time stamp when that happens and difference is about 90 ns for the data output to appear (shown at end of yellow line). That is the latency of this module.

The white box indicates the timing at which the output bus changes. Notice deserialized data is output as bytes, where `0xcc` is reconstructed as 8 bit parallel bus in the second bit of vector rbDxRxDataHS. The signal RxByteClkHS changes the value of the bus every 4 cycles DPHY_ClkHS as indicated by the white box.

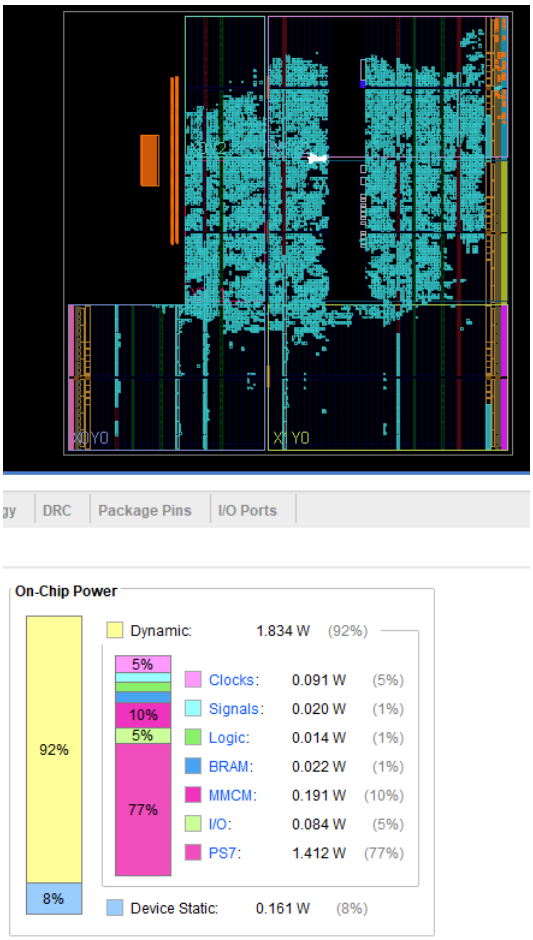
The blue vertical line to the right of the yellow indicates the cycle of RxByteClkHS which is about 16 ns (= 106.146 - 90.146). These results are in congruence with the conceptual deserializer diagram shown in Figure A.1.4b.

Figure 5.1.1: The timing diagram of the simulation for D-PHY module

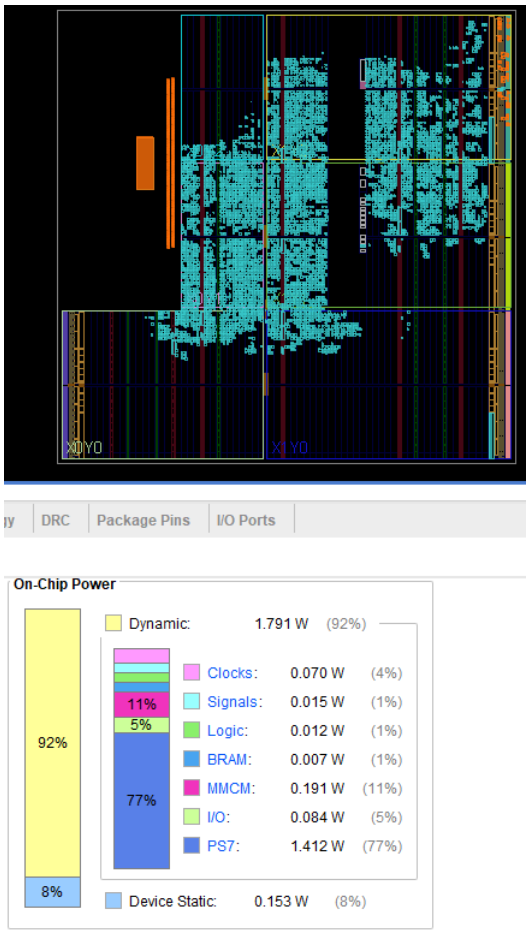


5.2 Resource Utilization

The resource utilization takes up a significant portion of the available capacity. When using ILAs the usage increases. However, there is some room left over to do other ISP related functions, like image stitching, for future needs. In Figure 5.2.1 the resource map and estimated power consumption is shown with ILAs and without (on the right).



(a) The resource map and power consumption estimate for the switching system with ILAs



(b) The resource map and power consumption estimate for the switching system with no ILAs

Figure 5.2.1: The resource usage map and estimated power consumption for the switching system with and without ILAs

The total power consumption is slightly more with ILAs. In Figure 5.2.1a the total on-chip power consumption, dynamic and static, is 1.995 W (= 1.834 + 0.161). This is not the actual power measured when the system is running. For system without ILAs the total power consumption amounts to 1.944 W (= 1.791 + 0.153). It is interesting to see that in both cases the PS has most dynamic power usage, even though this system does not use an OS but a bare metal design.

The post-implementation FPGA logic usage without ILAs is displayed in Figure 5.2.2. The PLL

type logic, MMCM, has most usage due to the various clock systems used in this implementation.

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	9001	53200	16.92
LUTRAM	427	17400	2.45
FF	13050	106400	12.27
BRAM	12	140	8.57
IO	23	125	18.40
BUFG	5	32	15.63
MMCM	2	4	50.00

Figure 5.2.2: Post-implementation table detailing logic usage of system without ILAs

Chapter 6

Summary

This project began with the idea of developing a stereo vision system using the MIPI standard using a low-cost FPGA development board. The purpose was to implement using an industry-proven standard that is readily available in many of today's electronic devices. An innovative idea in itself because it uses serial interface for a hardware-based stereo vision, unlike the traditional parallel buses used for image transfer, the progress was beset with lack of hardware documentation from vendors. Consequently, the project was scaled back to integrating a system of commercially available products for the switching of CMOS camera images plugged into adapters.

A major portion of the work was dedicated to investigating the control paths for the configuration of the camera. Once the path was found, then other cameras can be configured in a similar fashion and then switching can take place, seamlessly. This is part of the problem discussed in MCA section in Objectives section. The resolutions tested with are maximum 1080p but that is limited by the image sensor resolution and speed of the transfer interface. Initially, a Sony image sensor was used but attempts to configure even one directly without a switching adapter in the middle, were futile. Existing solutions were ported and ILAs were added for debugging and it was found there was no video signal coming from CSI-2 module due to incorrect configuration via hardware. That camera only had software support from the manufacturer and it was intended to be used for those purposes. Therefore, a camera with hardware support was found with an Omnivision image sensor from Digilent. This configured well with one camera and to scale to more, the switching adapter (MCA) needed to be used and this proved quite challenging.

The MCA PCB was carefully studied to see where the lines were headed to and from. Schematics were not shared by the vendor, nor available online. Two main ICs were identified and investigated further to find out the control path for configuration. After the correct I2C address was found and successfully target the I2C Mulitplexer from the FPGA, next step was to configure the cameras. In this system the cameras are configured via software application running on the PS.

After successful configuration of both cameras, the switching was tested. In order to control the selection it was decided to use low-speed Pmod interfaces for the GPIO command signals to be transmitted from FPGA to the MCA. First, to test this out, a slide switch and 3 LEDs in the development board were employed. The slide switch was sufficient to switch between 2 cameras, with their corresponding selection codes lighting up the LEDs. For this to work the hardware and software files had to be modified.

Next, software switching was attempted to see if the user can switch between cameras via a terminal program connected to the development board. There was no need for hardware components so the

LEDs and slide switches were removed. Significant changes were made to the complex software files. This time around another low-speed Pmod interface connected to the MIO part of the PS was used since the selection was going to be user controllable via a computer. It was successfully tested.

Due to investigative aspect of system integration, copious time was consumed. All the tasks and problems to be solved listed in Objective section were accomplished. It is hoped that this system can serve as a stepping stone for future projects that can implement a stereo vision system. Furthermore, this system is not optimized in its current setup where more cameras with higher resolutions and frame rates should be tested. This system uses a passive resistor network for image transfer at D-PHY level, therefore is bandwidth limited due to the higher charging and discharging times. This interface can be improved for higher performance if skew calibration is added.

In summary, the main goal of the project was implemented and constraints satisfied. Original contributions include successful design and integration of an experimental serial image transfer system for switching between two cameras on an FPGA SoC for stereo vision applications.

Bibliography

[1] P. W. Chun, "Architecture synthesis methodology of run-time reconfigurable multi-task and multi-mode systems with self-assembling micro-architecture," *PhD Thesis*, Jan. 2009. DOI: 10.32920/ryerson.14647689.v1.

[2] *Dual IMX219, 8 megapixels, applicable for Jetson Nano and Raspberry Pi, Stereo Vision, depth Vision*. [Online]. Available: https://www.waveshare.com/imx219-83-stereo-camera.htm.

[3] NXP Semiconductors NV, "I.mx 8/rt mipi dsi/csi-2," Application Note, 2024. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN13573.pdf.

[4] M. Defossez, "XAPP894 D-PHY Solutions," AMD Inc, Application Note, 2021. [Online]. Available: https://docs.amd.com/v/u/en-US/xapp894-d-phy-solutions.

[5] T.-J. Kim, J.-I. Hwang, S. Lee, *et al.*, "A 14-gb/s dual-mode receiver with mipi d-phy and c-phy interfaces for mobile display drivers," *Journal of the Society for Information Display*, vol. 28, no. 6, pp. 535–547, 2020. DOI: https://doi.org/10.1002/jsid.916.

[6] P.-H. Lee and Y.-C. Jang, "A 20-gb/s receiver bridge chip with auto-skew calibration for mipi d-phy interface," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 4, pp. 484–492, 2019. DOI: 10.1109/TCE.2019.2942503.

[7] Sony Corp, *Sony-Exmor-R-IMX219PQH5-C-Raspberry-Pi-Camaera-V2-CMOS, 8 Megapixel, Diagonal 4.60 mm Datasheet*. [Online]. Available: https://github.com/rellimmot/Sony-IMX219-Raspberry-Pi-V2-CMOS/blob/master/RASPBERRY%20PI%20CAMERA%20V2%20DATASHEET%20IMX219PQH5_7.0.0_Datasheet_XXX.PDF.

[8] M. Feldmeier, "Low-Latency Stereo Camera with FPGA-based Video Processing," Hochschule Munchen University of Applied Sciences, BEng Thesis, 2023. [Online]. Available: https://github.com/StereoNinja/StereoNinjaFPGA/blob/main/Bachelor%20Thesis.pdf.

[9] F. Liu, L. Wang, and Y. Yang, "A UHD MIPI CSI-2 image acquisition system based on FPGA," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 5668–5673. DOI: 10.23919/CCC52363.2021.9550077.

[10] P.-H. Lee, H.-Y. Lee, Y.-W. Kim, H.-Y. Hong, and Y.-C. Jang, "A 10-gbps receiver bridge chip with deserializer for fpga-based frame grabber supporting mipi csi-2," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 3, pp. 209–215, 2017. DOI: 10.1109/TCE.2017.014908.

[11] D. Qendri, "Real Time Video Stitching Implementation on a ZYNQ FPGA SoC," *MEng Report*, Nov. 2022. DOI: 10.32920/ryerson.14654502.v1.

[12] S. J. Hippolyte, "Development and implementation of the method for high resolution object tracking in 3D space," *MEng Report*, 2016. DOI: 10.32920/ryerson.14660313.v1.

[13] O. Samarin, "Reconfigurable fuzzy logic system for high-frame rate stereovision object tracking," Jan. 2008. DOI: 10.32920/ryerson.14654598.v1. [Online]. Available: https://rshare.library.torontomu.ca/articles/thesis/Reconfigurable_fuzzy_logic_system_for_high-frame_rate_stereovision_object_tracking/14654598.

[14] J. Islam, "Architecture and Implementation of a High Frame-Rate Stereo Vision System," *Master's Thesis*, Jan. 2008. DOI: 10.32920/ryerson.14648928.v1.

[15] S. Sabihuddin, "Dense Stereo Reconstruction in a Field Programmable Gate Array," *Master's Thesis*, Jan. 2008. DOI: 10.13140/RG.2.2.23207.15529.

[16] AMD Inc, *ZYnQ 7000 SOC Device Family • ZYNQ 7000 SOC Technical Reference Manual (UG585)*. [Online]. Available: https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM/Zynq-7000-SoC-Device-Family.

[17] Digilent Inc, *Zybo Z7 Board Reference Manual*, Feb. 2018. [Online]. Available: https://digilent.com/reference/_media/reference/programmable-logic/zybo-z7/zybo-z7_rm.pdf.

[18] Digilent Inc, *ZYBO Z7 Schematics Rev D.1*, Engineering, Feb. 2022. [Online]. Available: https://files.digilent.com/resources/programmable-logic/zybo-z7/zybo-z7-d1-sch.pdf.

[19] STMicroelectronics NV, *Mainstream Arm Cortex-M0 Value line MCU with 16KB of flash memory, 48 MHz CPU*, Webpage, 2021. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f030f4.html#overview.

[20] Texas Instruments Inc, "TCA9544A Low Voltage 4-Channel I2C and SMBus Multiplexer with Interrupt Logic," Product Datasheet, 2019. [Online]. Available: https://www.ti.com/lit/ds/symlink/tca9544a.pdf.

[21] ArduCAM, *Raspberry Pi Multi Camera Adapter Module v2.2 User Guide*, Nov. 2019. [Online]. Available: https://www.uctronics.com/download/Amazon/B0120.pdf?%20srsltid=AfmBOoovwpIPt_yerOYOIVEJx4wW1FvOVA96lHrIXYoUATgp-KIbmVyc.

[22] ArduCAM, *Quick Start Guide for Multi-Camera Adapter Boar - Arducam Wiki*. [Online]. Available: https://docs.arducam.com/Raspberry-Pi-Camera/Multi-Camera-CamArray/Quick-Start-Guide-for-Multi-Adapter-Board/#python-demos.

[23] Waveshare, *Dual IMX219, 8 megapixels, applicable for Jetson Nano and Raspberry Pi, Stereo Vision, depth Vision*. [Online]. Available: https://www.waveshare.com/imx219-83-stereo-camera.htm.

[24] R. P. Ltd, *Raspberry Pi Camera Module 2 – Raspberry PI*. [Online]. Available: https://www.raspberrypi.com/products/camera-module-v2/.

[25] S. Bobrowicz, *Pcam 5c - digilent reference*, Apr. 2025. [Online]. Available: https://digilent.com/reference/add-ons/pcam-5c/start.

[26] OmniVision Technologies Inc., *color CMOS QSXGA (5 megapixel) image sensor with OmniBSI technology datasheet (CSP3)*, datasheet, May 2011. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/OV5640_datasheet.pdf.

[27] E. Gyorgy, "Mipi csi-2 receiver 1.1 ip core user guide," Digilent Inc, IP Product Guide, 2018. [Online]. Available: https://github.com/Digilent/vivado-library/blob/master/ip/MIPI_CSI_2_RX/docs/mipi_csi_2_rx.pdf.

[28] E. Gyorgy, "MIPI D-PHY Receiver 1.3 IP Core User Guide," Digilent Inc, IP Product Guide, 2018. [Online]. Available: https://github.com/Digilent/vivado-library/blob/master/ip/MIPI_D_PHY_RX/docs/mipi_d_phy_rx.pdf.

[29] InterOperability Lab, "MIPI D-PHY Reference Termination Board (RTB) Overview and Datasheet," University of New Hampshire, Product Overview, 2016. [Online]. Available: https://www.iol.unh.edu/sites/default/files/test-services/mipi/UNH_IOL_RTB_Datasheet_20160713_-_Copy_-_Copy.pdf.

[30] Tektronix Inc, *Understanding and performing MIPI D-PHY physical layer, CSI and DSI protocol layer testing*, Webpage. [Online]. Available: https://digilent.com/reference/learn/programmable-logic/tutorials/zedboard-fmc-pcam-adapter-dual-camera-demo/start.

Appendices

Chapter A

Appendix

A.1 D-PHY IO circuitry

MIPI D-PHY is somewhat unique in that one of the key power-saving aspects of D-PHY is the mechanism of a dynamic switchable resistive termination at the receiver which is enabled during HS and disabled to present an open termination environment during LP. The HS and LP modes utilize different signaling schemes, with different voltage levels (Figure 2.2.5). The LP mode of operation is designed such that data can still be communicated between the two ends of the link, but at a much lower speed (∼10 Mbits/sec) than in HS mode, which is designed for operation between 80 and approximately 800 Mbps. This dual-mode functionality requires a different receiver architecture than that typically found in other high-speed serial technologies like SATA, SAS, PCI Express, XAUI, 10GBASE-CX4, etc even though both have the requirement of a 100 Ω differential termination[29].

In LP mode, all wires are operated single-ended, and are unterminated at the receiver. In this mode, minimal current flows between the two ends of the link (due to the lack of termination as shown in Figure A.1.1), reducing power consumption. In HS mode, the link utilizes differential signaling like LVDS, but with a lower amplitude level than most other high-speed serial technologies, and also with a common-mode DC offset (whereas most other high-speed serial technologies are AC coupled, and use DC-balanced signaling levels in conjunction with 8B/10B line encoding). D-PHY has common mode DC offset of about 200 mV in HS mode (Figure 2.2.5).

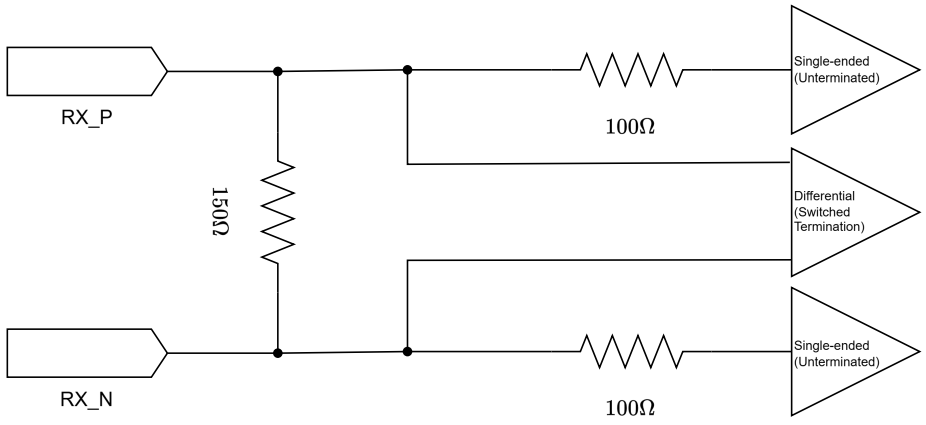


Figure A.1.1: FPGA Compatible D-PHY Receiver with a passive resistor network for dynamic termination enabling HS and LP modes, all on a single lane

A.1.1 D-PHY LP Receiver

In LP mode the signaling is CMOS Inverter-like single-ended and unterminated leading to high impedance at the receiver end. Therefore, the 100 Ω resistors can be neglected as no current can flow through. Both lines of the lane can be driven independently between 0V and 1.2V (Figure A.1.2). What that means is each line can swing rail-to-rail. However, due to the 150 Ω resistor there is a small current and hence a power dissipation during the LP01 transition in Figure A.2.1 explained in section D-PHY Operation below. The LP CMOS push-pull drivers have low impedance and therefore can easily overcome the current flow through the resistor. The signaling speed is low enough so that the impedance mismatches can be safely ignored.

The capacitor coupling in the transmitter helps reduce the power consumption by ensuring that the gate of the NMOS remains at an appropriate voltage level when transitioning. The signal from the Pre-driver controls the switching of the transistors.

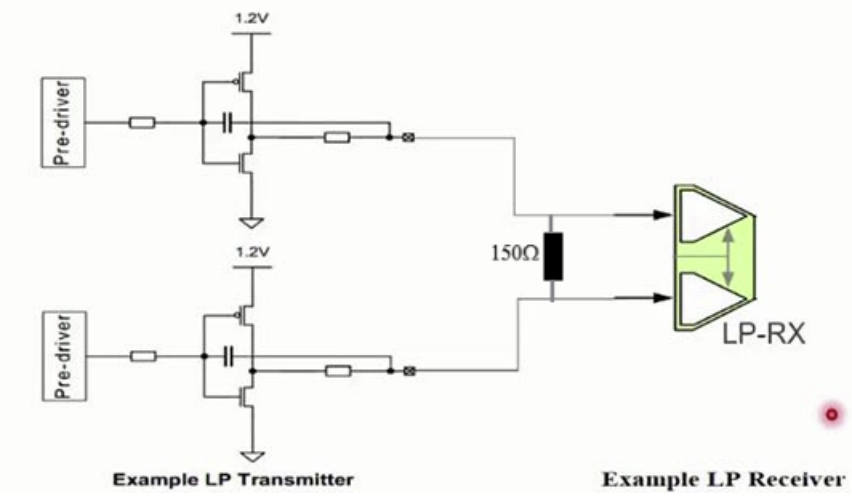


Figure A.1.2: Simplified version showing an unterminated LP transmitter and receiver with the 150 Ω resistor

A.1.2 D-PHY HS Receiver

The switched termination in Figure A.1.3 is a key feature commonly used in high-speed digital communication interfaces like LVDS. The components are:

1. **Termination Resistors ($Z_D/2$):** These resistors are connected between the differential signal pair (Dp and Dn) and the common-mode voltage (C_{CM}). They match the characteristic impedance of the transmission line, reducing reflections. From Figure A.1.1, $\frac{1}{Z_D} = \frac{1}{150} + \frac{1}{100} => Z_D = 60\ \Omega$. The internal differential termination resistance from [16] is 100 Ω. The effective total load of 60 Ω

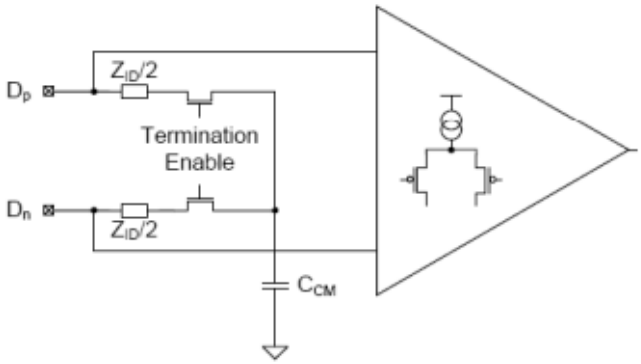


Figure A.1.3: Simplified version showing a D-PHY approved receiver implementation[29]

is less than 100 Ω required by MIPI standard but it appears to be working within the specification when tested and verified for the receiver eye diagrams[4].

2. **Termination Enable:** This receiver control can enable or disable termination, depending on the operational mode.

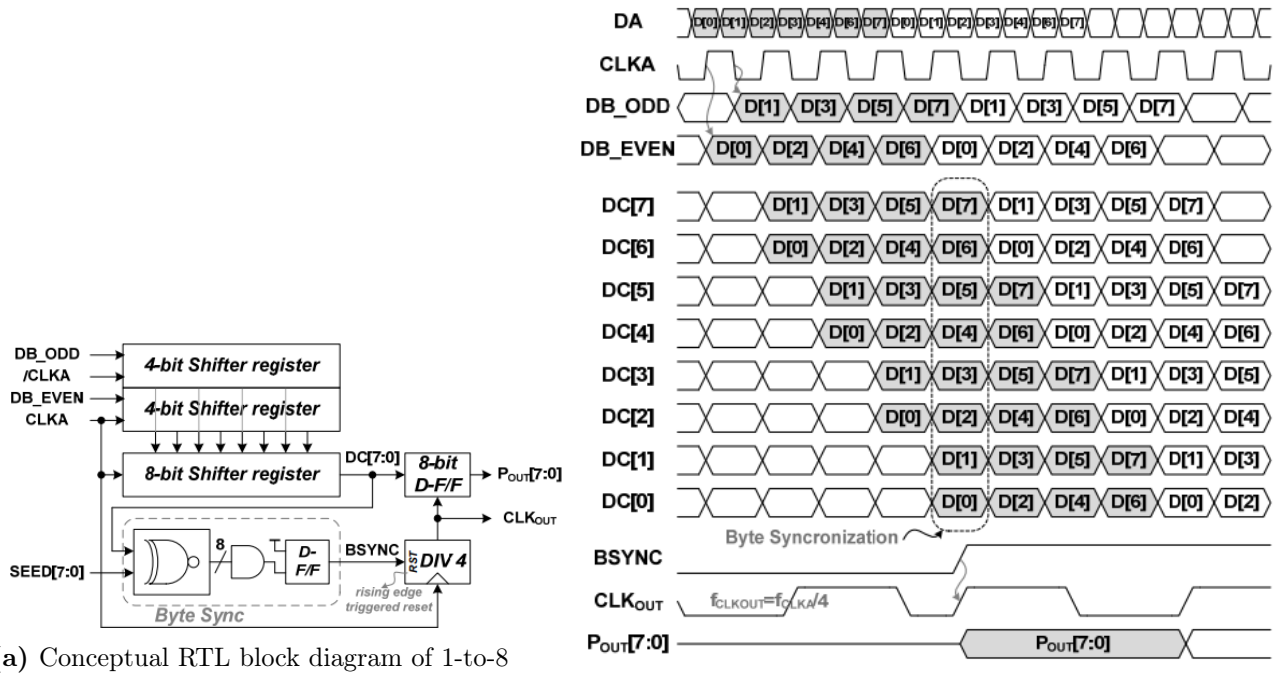
3. **Differential Amplifier:** This converts the small voltage difference between Dp and Dn into a logic-level output. It typically has a current source or biasing circuit, as shown on the right side.

A.1.3 D-PHY Deserializer

A receiver needs to convert the serial bit stream into a parallel bus via a Deserializer whenever the Leader Sequence (referred to as SEED in Figure) is detected. In addition, it needs to output a clock for the parallel bus to be read by a downstream module, often called a HS Byte Clock. A visual understanding of the Register-Transfer Level (RTL) architecture is presented below in Figure A.1.4. It is also apparent that the data (DA) is appearing before the clock line, CLKA as mentioned in main section D-PHY.

A.2 D-PHY Operation

D-PHY devices switch between HS and LP modes via communication of a special LP state sequence, which signals an intent to transition to HS mode through a series of LP transitions and HS zeros as in Figure A.2.1. When a receiver detects this mode it will enable its HS line termination, which prepares the line for HSoperation. The transmitter is then able to transmit data at the HS line rate for a period of time, after which a different control sequence is sent, which initiates a return to the LP mode of operation (whereby the receiver disconnects its HS termination at the appropriate time). This act of switching in and out of HS mode is typically called HS 'burst mode' operation.



(a) Conceptual RTL block diagram of 1-to-8 Deserializer

(b) Timing Diagram of RTL block diagram

Figure A.1.4: The conceptual operation of a Deserializer for a single lane. *CLKA* and */CLKA* are the differential clock pins, *DA* is the serial bitstream from one of the data lanes and *SEED* is the Leader Sequence bits needed to achieve Byte synchronization for Byte Clock, in this case CLK_{out}[10]

Depending on the number of data lanes to be used for data transmission, the image data is organized by the transmitter. The transmitter then serializes the data on each lane and transmits it to the corresponding receiving lanes.

For example, if two lanes are used, the first byte of payload data is sent over data lane 0 and second byte on data lane 1. Similarly, on the receiving side, the serial data from each data lane is converted into byte format with the help of deserializer present in each receiving lane of the D-PHY. After this, the deserialzed bytes from each lane are merged together by the CSI-2 controller. This is quite different from using parallel interfaces.

A.2.1 Procedure to initiate HS Clock transmission

As part of the initialization of D-PHY, all lanes are initially in the LP11 state (1.2V level) for a specified time. After this, for sending the image data, the transmitter drives a particular sequence on the receiver to enter the receiver lanes from the low power mode to high speed mode. The high-speed entry sequence consists of driving LP11->LP01->LP00 (LP->HS transition) on the receiver lanes as shown in the following figure. On successful reception of this sequence the high-speed receiver module enables its termination to receive the high-speed differential data. The transmitter sends HS zeros for a time before driving the high-speed clock signal and then the data lanes start up[3].

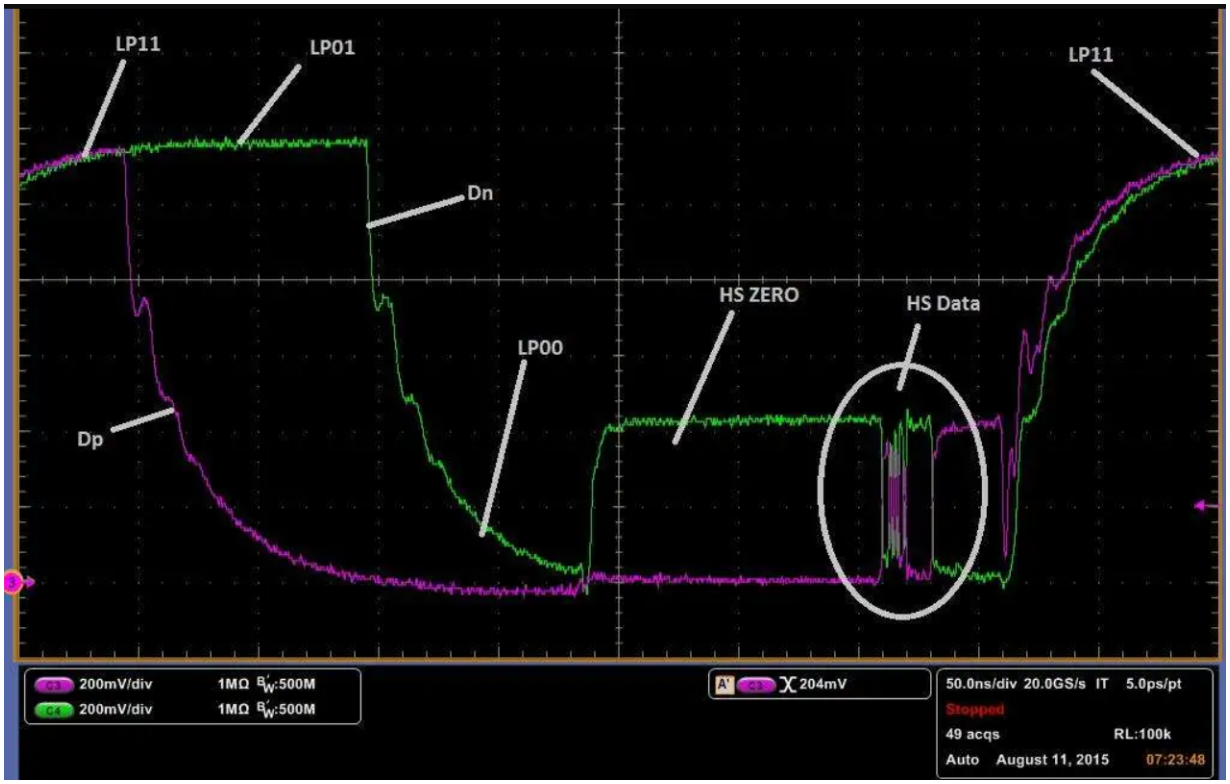


Figure A.2.1: HS Burst on Data Lane depicting mode transition. Dp and Dn are the differential signals, positive and negative respectively, of a lane. Note the voltage levels of both modes

A.2.2 Transition for SoT Procedure in Data Lanes

Now the high speed receiver termination has become active and the receiver starts to receive the high speed data from the transmitter. HS zeros are important because they signal to the receiver to be activated properly before any payload data is transmitted.

The payload data transmitted over the D-PHY data lane is in packet format. It could be either a long packet or a short packet. Long packet consists of 32 bit Packet header, payload data and 16 bit of packet footer. Short packet consists of 32 bit of packet header only. The procedure to wake up the data lanes is similar to HS clock transition but with a Leader-Sequence, in bits, 8b'00011101'is inserted by the transmitter. Once that is recognized by the receiver it is ready to start receiving payload data as shown in Figure A.2.2.

After every HS burst the data lanes go to LP11 state. A single HS burst represents the image data corresponding to one of the **horizontal line** of an image and the LP11 state in-between the HS bursts represents the blanking periods. Because low power commands require signals to be send at lower frequency, this intermittent movements of D-PHY in LP and HS modes helps in reducing the overall power consumption. The high speed payload data is transmitted on both the edges of the HS differential clock (DDR clock). The high speed differential clock and the data transmitted from the

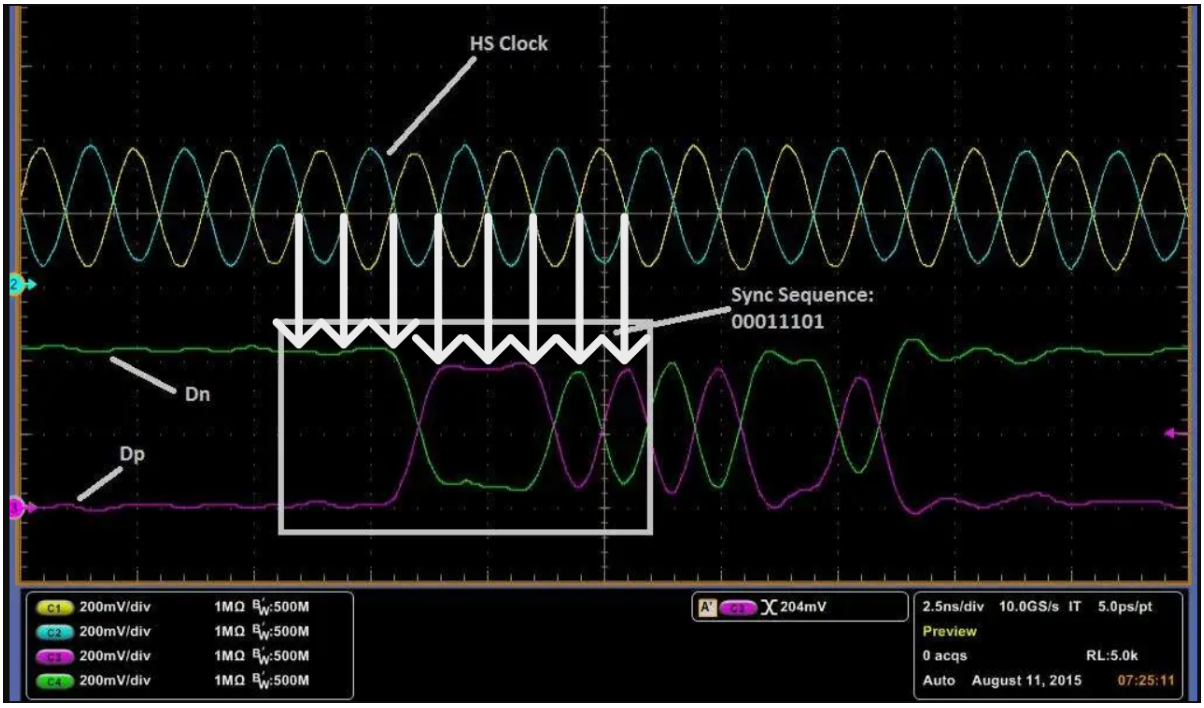


Figure A.2.2: Leader Sequence bits, 8b'00011101', for data lane synchronization detected by the receiver as shown by the white arrows from the HS clock differential crossings. When Dn > Dp, its a 0 and 1 otherwise.

transmitter are 90 degrees out of phase and with the data being transmitted first(Figure 2.2.4). This timing relationship between clock and data helps in achieving the setup and hold time requirements at the receiver data lanes.

A.2.3 Glitches

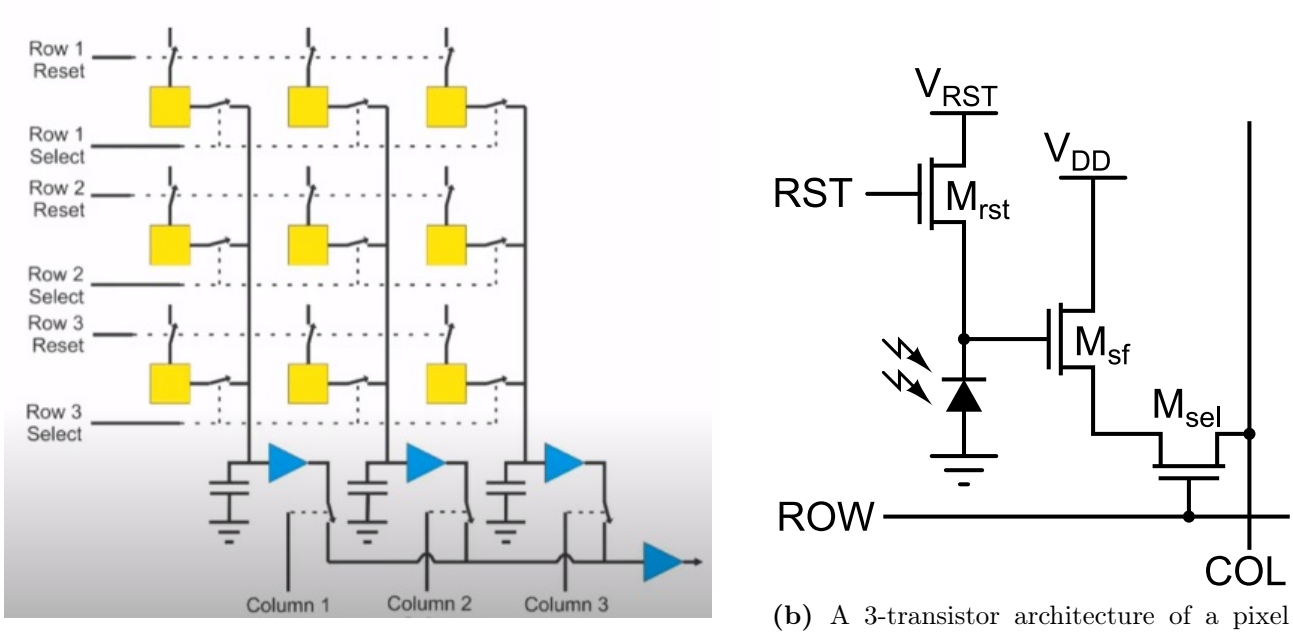
During the transition of signal from single-ended (LP) to differential HS, the impedance imbalance that occurs on the line can cause glitches. These may be due to non-synchronous termination transition between the transmitter and receiver[30]. In implementations, glitch filters help to remove them.

Chapter B

Appendix

B.1 CMOS Image Sensor Array and Circuitry

The circuitry is built around a photodiode, which consequently reduces the active area to collect light. Therefore, microlens are placed on top of the photodiode area to focus the light as much on it. Since the circuits are built on each photo-sensing element, there is more flexibility to target specific areas of the image for higher FPS. How is this possible? It becomes clearer with understanding the operation of the image sensor as shown in Figure B.1.1a.



(a) Simplified CMOS Image Sensor Array

(b) A 3-transistor architecture of a pixel element

Figure B.1.1: General description of an Active CMOS Image Sensor

1. "Row 1 Select" is activated where all the voltages from the first row (line) pixels are read out and stored in the capacitors at the end of each column. The dashed line is connected to all "Row 1 Select" switches as shown in Figure B.1.1a and that line is read simultaneously.

2. "Column 1 Select" is turned on where the charge from that column's capacitor is read out by the ADC and converted to a digital signal.

3. Continue reading the next columns sequentially, unlike the simultaneous nature of line readout, till the last column. Simultaneously, enable "Row 1 Reset" for that entire line to begin a new exposure.

4. Move to the next line, "Row 2 Select" until the last line is read out and repeat from 1.

In this manner, specific rows of interest in the image can be targeted for reading out the values. This feature adds more flexibility unlike previous image sensor technologies. However, each line is read at a different time, leading to skewed distortion of images, especially when they are moving fast like helicopter blades. This type of readout is called a Rolling Shutter and it is the most commonly used including in this project. Global Shutter eliminates this issue but with more electronics and higher cost. This also reduces the available area for light collection even more, leading to more noise compared to Rolling Shutter devices.

Figure B.1.1b shows how the switches in the adjacent figure look like at the transistor level. The reset transistor,M_{rst} (ie. "Row 1 Reset"), acts as a switch to reset the floating diffusion to V_{RST}, which in this case is represented as the gate of the M_{sf} transistor. When the reset transistor is turned on, the photodiode is effectively connected to the power supply, V_{RST}, clearing all integrated charge. Since the reset transistor is n-type, the pixel operates in soft reset. The read-out transistor, M_{sf}, acts as a buffer (specifically, a Source Follower), an amplifier which allows the pixel voltage to be observed without removing the accumulated charge. Its power supply, V_{DD}, is typically tied to the power supply of the reset transistor V_{RST}. The select transistor, M_{sel} (ie. "Row 1 Select"), allows a single row of the pixel array to be read by the readout electronics.