CHALMERS
UNIVERSITY OF TECHNOLOGY

Design and Implementation of a Cascaded Camera Switching System for Drone-Based Multi-Camera Perception

Master's thesis in Embedded Electronic System Design

Gege Yang

Department of Microtechnology and Nanoscience
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Master's thesis 2026

Design and Implementation of a Cascaded Camera Switching System for Drone-Based Multi-Camera Perception

Gege Yang

Department of Microtechnology and Nanoscience
Chalmers University of Technology
Gothenburg, Sweden 2026

Design and Implementation of a Cascaded Camera Switching System for Drone-Based Multi-Camera Perception
Gege Yang

Supervisor: Lena Peterson, Department of Microtechnology and Nanoscience
Company advisor: Fredrik Falkman, Remote Aero
Examiner: Per Larsson-Edefors, Department of Microtechnology and Nanoscience

Master's Thesis 2026
Department of Microtechnology and Nanoscience
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Design and Implementation of a Cascaded Camera Switching System for Drone-Based Multi-Camera Perception
Gege Yang
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

This project introduces the design and implementation of a multi-camera expansion system based on Raspberry Pi. The goal is to extend the Raspberry Pi's CSI interface, enabling multiple camera modules to share a single CSI interface. Compared to existing commercial solutions, the core contribution of this research is the design of a lighter and more compact expansion board to meet the miniaturization requirements of UAV platforms.
The system achieves multiplexing switching through an I^2C multiplexer and GPIO control of MIPI. On the hardware side, a custom four-layer PCB was designed and fabricated. On the software side, a dedicated device tree overlay layer was defined to describe its hardware topology. During the verification phase, software tools such as libcamera and media-ctl, as well as custom control scripts, were used for system integration and verification.
Experimental results show that the system successfully implemented I^2C communication, GPIO-based channel selection, device driver registration, camera enumeration, and media pipeline creation. However, in system-level testing, stable image stream transmission failed to be established. Further testing revealed that the fault lies in the high-speed MIPI CSI-2 transmission path. Although complete image streaming was not achieved, the project successfully validated the control architecture and software integration of the proposed multi-camera system, and the troubleshooting process provided a reference for subsequent development and optimization.

Keywords: Raspberry Pi, MIPI CSI-2, Multi-Camera Expansion, IMX219, I2C Multiplexer, Device Tree Overlay, Embedded Vision System

Acknowledgements

I would like to express my sincere gratitude to everyone who contributed to and supported this project.
My thanks go to my company supervisor, Fredrik Falkman, for his guidance, valuable advice, and willingness to answer questions throughout the project.
I am also grateful to my academic supervisor, Lena Peterson, for monitoring the project progress on a weekly basis and for providing constructive feedback on the thesis. In addition, I would like to thank my examiner, Per Larsson-Edefors, for his valuable comments and suggestions regarding thesis and report.
Finally, I would like to acknowledge the encouragement and support of my friends throughout my studies and during the completion of this thesis.

Gege Yang, Gothenburg, July 2026

Contents

1

Introduction

In recent years, small unmanned aerial vehicles (UAVs) have been widely used in intelligent monitoring, environmental perception, agricultural inspection, and autonomous navigation [1]. With the continuous expansion of application scenarios, UAV systems are placing higher demands on environmental perception capabilities. Compared to a single camera, a multi-camera system can provide a wider field of view and richer environmental information, thereby improving the system's perception capabilities and decision-making reliability in complex environments [2].

However, small UAVs typically use resource-constrained embedded computing platforms, such as common single-board computers. These platforms have limitations in terms of power consumption, size, and interface resources, especially the limited number of MIPI CSI-2 interfaces [3, 4]. This limitation makes it difficult for the system to connect multiple high-resolution cameras simultaneously, thus restricting the realization of multi-view perception capabilities. Therefore, achieving the trade-off between multi-camera access, power consumption, and system complexity on resource-constrained embedded vision platforms has become a key research topic.

1.1 Related Work

To address the limited number of MIPI CSI-2 interfaces on resource-constrained embedded platforms, academia and industry have proposed various solutions. These methods can be mainly divided into three categories: image data aggregation schemes based on FPGAs or ASICs, transmission extension schemes based on high-speed serial links, and schemes based on channel multiplexing or switching. These methods attempt to overcome the limitations of interface quantity from different angles, alleviating the bottleneck problem of multi-camera access to some extent.

However, these methods still have certain limitations in small UAV applications. First, FPGA-based aggregation offers high flexibility and parallel support, but its significant power, size, and weight overhead make it unsuitable for small UAVs [5, 6]. Second, serial-link technologies (e.g., GMSL/FPD-Link) provide robust long-distance transmission for the automotive industry, yet introduce hardware complexity and latency that can jeopardize UAV flight stability [7]. The third approach, represented by Arducam, utilizes high-speed analog multiplexers for physical switching [8]. While this achieves an ideal balance between power and latency, current commercial boards often lack size optimization for small UAV carriers and comprehensive high-speed signal integrity data.

To address these issues, this thesis proposes a cascaded MIPI CSI-2 camera switching system for resource-constrained UAV platforms. This system employs a multi-level hardware switching architecture, achieving dynamic selection of multiple camera inputs through high-speed signal switching at the physical layer. During the design process, the focus is on the transmission characteristics of high-speed differential signals in the cascaded path, and signal integrity is ensured through optimization of PCB structure, impedance matching, and routing strategies. The system is implemented under compact size constraints, and its performance is verified by experimental methods.

1.2 Purpose and Goal

This research aims to enable resource-constrained drones to perceive their surroundings from multiple perspectives without increasing processor load or power consumption. This allows drones to perform more complex tasks, such as obstacle avoidance, panoramic environmental scanning, or multi-angle inspection, which are difficult to achieve with a single camera. Furthermore, the research on the signal integrity of high-speed switching circuits provides a reference for other compact embedded vision systems.

The core objective of this project is to design, implement, and verify a MIPI switching system supporting Gbps-level data rates. This system must ensure signal integrity while achieving software integration with existing embedded Linux systems (such as Raspberry Pi OS). To achieve the above objectives, this study is divided into the following specific tasks. First, a signal transmission model conforming to the MIPI D-PHY physical layer specification is established, and the 100 Ω differential impedance constraint under the four-layer PCB architecture is calculated [9]. Second, the high-speed differential line layout and routing on the PCB must be completed within the physical size constraints. Third, hardware-software integration and driver development are carried out, including a camera switching driver logic that enables stable video stream capture and dynamic switching. Finally, system-level testing experimentally verifies the proposed hardware and software architecture, while also identifying potential limitations that may affect the stability of CSI-2 data transmission.

1.3 Thesis Outline

This report aims to design and implement a cascaded multi-camera switching system for a resource-constrained UAV platform. Therefore, this report first discusses the MIPI CSI-2 protocol and differential signaling theory, which are crucial to this work. This background section also introduces the software stack used in the project, with a focus on the embedded Linux V4L2 framework and device tree mechanism.

Following the background introduction, the project's methodology will be discussed. This chapter covers hardware design methods and software development strategies. Furthermore, the system verification process will also be discussed.

Next, the design will be discussed. This chapter will demonstrate the rationale behind the design choices and illustrate the overall system architecture using block diagrams. It will also explain component selection, PCB stack-up structure, and signal integrity optimization strategies.

Finally, before the conclusion, a results section and a discussion section will be presented. The results section will showcase all experimental results from the project, including I²C communication testing, GPIO functional testing, and system integration testing. The discussion section will discuss all results. In addition to discussing the project's achievements, this section also focuses on lessons learned from this project for future projects and areas for future work.

2

Technical Background

The following chapter will explore the theoretical concepts crucial to this project. First, the MIPI CSI-2 protocol and the D-PHY physical layer are introduced, followed by the fundamentals of differential signaling and signal integrity. Finally, the embedded Linux camera software stack and device tree are outlined. These parts together provide the theoretical foundation for subsequent design work.

2.1 MIPI CSI-2 Protocol

The MIPI Camera Serial Interface 2 (CSI-2) is a standardized protocol defined by the MIPI Alliance for high-speed image data transmission in embedded vision systems [10]. It is widely used to connect image sensors to application processors in mobile and embedded platforms. CSI-2 is designed as a layered architecture, separating data transport functions from the underlying physical signaling mechanism, which is typically implemented using the MIPI D-PHY specification. The MIPI transmission system architecture is shown in Figure 2.1.

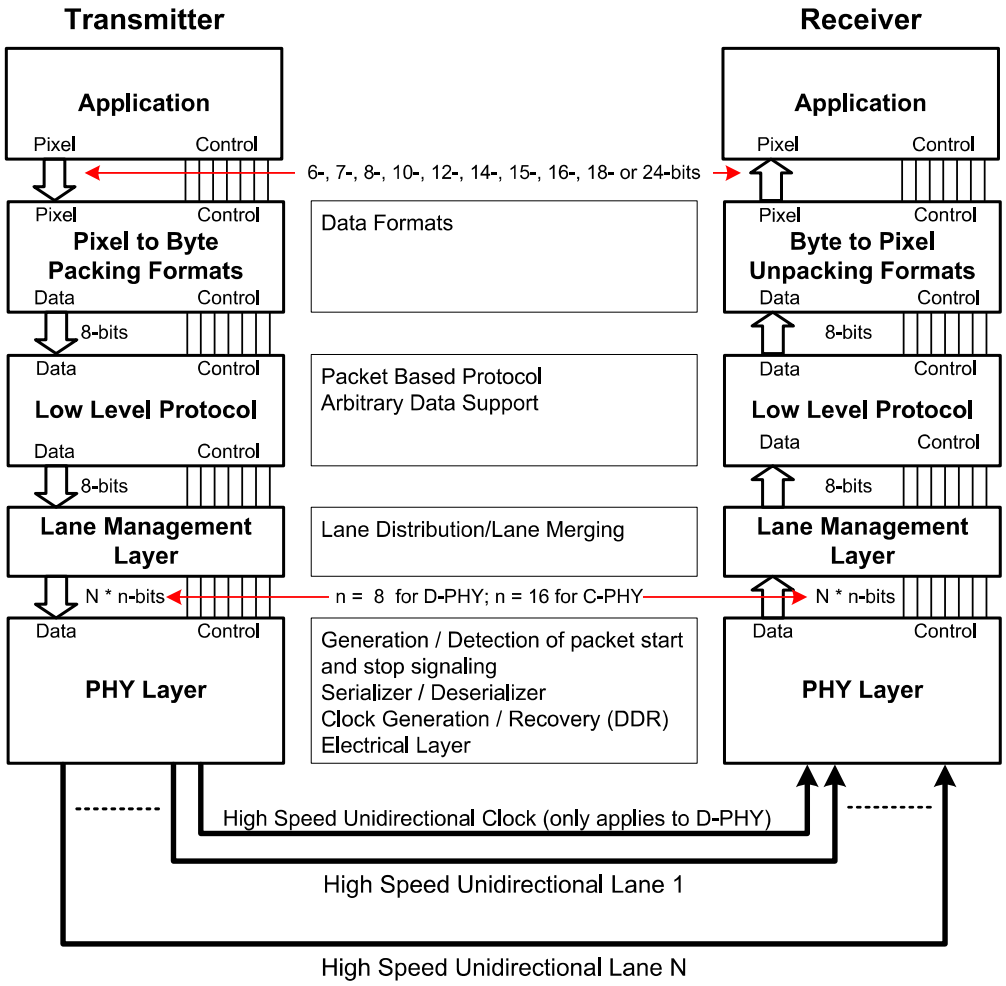


Figure 2.1: CSI-2 layer definitions. Reproduced from [11]

At the architectural level, CSI-2 operates as a packet-based protocol that defines how image data is structured and transmitted. The protocol encapsulates image information into structured packets, enabling reliable and ordered delivery of data streams between the camera sensor and the host processor. This abstraction allows CSI-2 to support different image formats while maintaining a unified transport mechanism. A CSI-2 frame is transmitted as a sequence of packets [11]. Each frame typically begins with a Frame Start (FS) short packet, followed by multiple long packets carrying line-based image data, and ends with a Frame End (FE) packet. This packetized structure enables synchronization between the transmitter and receiver, ensuring correct reconstruction of image frames at the receiving end. To support multiple image formats, CSI-2 defines a pixel-to-byte conversion layer that converts sensor-specific pixel formats into a unified byte stream [11]. This allows different formats such as RGB and YUV to be transmitted using a consistent data representation, simplifying the transport mechanism at the protocol level. CSI-2 also supports scalable bandwidth through multiple data lanes. Image data packets can be distributed across one to four data lanes, enabling higher throughput without changing the higher-level protocol structure. The lane configuration is handled transparently to the protocol layer, ensuring that packet integrity is preserved

during transmission.

The underlying physical transmission is implemented using the MIPI D-PHY standard. D-PHY defines the electrical signaling characteristics and timing behavior of the interface. It operates in two modes: high-speed (HS) mode and low-power (LP) mode. HS mode uses low-voltage differential signaling for high-speed data transmission, while LP mode uses single-ended signaling for control and initialization purposes. In HS mode, data is transmitted over differential pairs consisting of clock and data lanes. The clock lane provides a source-synchronous timing reference, allowing the receiver to sample data accurately from the data lanes. Each transmission begins with a start of transmission (SoT) and ends with an end of transmission (EoT), ensuring clear packet boundary detection at the physical layer [9]. The separation between CSI-2 and D-PHY enables a clear abstraction between protocol-level data organization and physical-level signal transmission, allowing the CSI-2 architecture to remain flexible and adaptable across different hardware implementations.

2.2 Signal Integrity of Differential MIPI Links

2.2.1 Differential Line Theory

A differential line consists of two coupled, parallel, and equal-length transmission lines. These two lines carry signals that are 180 degrees out of phase. Compared to single-ended signals referenced to the ground plane, differential signals reference each other. External interference (common-mode noise) acts simultaneously on both lines and is canceled out at the receiving end through subtraction, thus providing strong anti-interference capability [12].

In addition to transmitting differential signals, differential lines also transmit common-mode signals. The former consists of driving voltages of equal magnitude but opposite direction, while the latter consists of driving voltages of equal magnitude and the same direction. Define V_1 and V_2 as single-ended signal voltages. The peak value of a differential signal is theoretically twice that of a single-ended signal, thus maintaining a high signal-to-noise ratio even under low voltage swing. In contrast, the common-mode signal is the average voltage of the two lines relative to the reference plane, which manifests as voltage fluctuations with the same phase, as shown in Equations 2.1, 2.2 and Figure 2.2.

$$V_{\mathrm{DM}} = V_1 - V_2 \tag{2.1}$$

$$V_{\mathrm{CM}} = \frac{V_1 + V_2}{2} \tag{2.2}$$

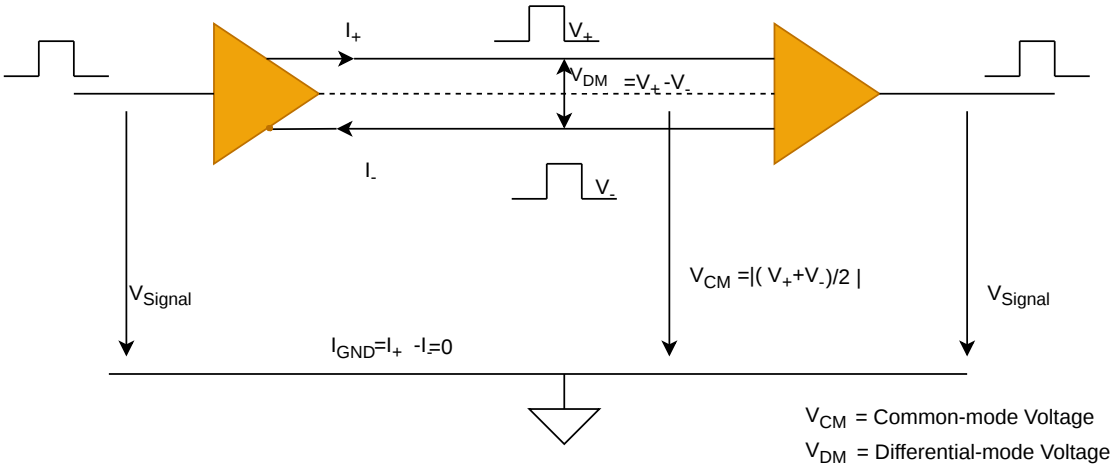


Figure 2.2: Voltage of a differential pair

The impedance characteristics of a differential pair exhibit two distinct physical states under different modes. when the system transmits differential signals, a strong mutual attractive electric field is generated between the two conductors. The characteristic impedance exhibited by a single transmission line at this time is called the odd-mode impedance. Due to electromagnetic coupling, the odd-mode impedance Z_{odd} is slightly smaller than the single-ended independent impedance Z_0. The differential impedance Z_{diff} is physically represented as the series connection of two odd-mode impedance branches:

$$Z_{\text{diff}} = \frac{V_{\text{diff}}}{I} = \frac{2 \times V}{I} = 2Z_{\text{odd}} \tag{2.3}$$

When a system propagates a common-mode signal, the electric field lines between conductors repel each other. The single-wire impedance at this time is called the even-mode impedance. The common-mode impedance Z_{comm} is represented by the parallel connection of two even-mode impedance branches:

$$Z_{\text{comm}} = Z_{\text{equiv}} = \frac{Z_{\text{even}} \times Z_{\text{even}}}{Z_{\text{even}} + Z_{\text{even}}} = \frac{Z_{\text{even}}}{2} \tag{2.4}$$

The physical relationship between Z_{odd} and the geometric parameters (trace width, spacing, and dielectric height) dictates the signal integrity of the link. Any deviation from the target differential impedance—caused by changes in the cross-sectional geometry—will lead to energy reflection and mode conversion, where differential energy is parasitically converted into common-mode noise.

2.2.2 Signal Integrity Fundamentals

Signal integrity (SI) represents the fidelity of an electrical signal as it propagates from a transmitter to a receiver. In high-speed digital systems, a signal is considered to have high integrity when it maintains the required timing margins and waveform

quality to be correctly interpreted by the receiver. As data rates increase into the gigahertz range, as seen in MIPI D-PHY links, the non-ideal physical characteristics of PCB traces begin to dominate signal behavior [12].

The fundamental mechanism behind signal distortion is the variation in transient impedance. According to transmission line theory, any abrupt change in the geometry of the signal path—such as the transition through a via, a connector pad, or an integrated circuit pin—creates an impedance discontinuity. When an electromagnetic wave encounters such a discontinuity, a portion of the energy is reflected toward the source. The relationship between the incident and reflected wave is governed by the reflection coefficient Γ [13]:

$$\Gamma = \frac{Z_{\text{load}} - Z_0}{Z_{\text{load}} + Z_0} \tag{2.5}$$

The superposition of reflected waves and forward waves can cause ringing, overshoot, and edge degradation. If a signal passes through chip pins and vias multiple times, precise impedance matching is necessary to suppress the increase in bit error rate caused by reflection. From a theoretical perspective, minimizing Γ requires that the characteristic impedance (Z_0) of the transmission line be matched to the load impedance (Z_{load}) throughout the entire channel. This necessitates the careful design of PCB stack-ups and the compensation of parasitic capacitances and inductances introduced by physical transitions (e.g., vias), providing the theoretical justification for the impedance control techniques employed in the design phase.

2.3 Crosstalk and Timing Skew

Crosstalk exists in high-frequency circuit design and becomes increasingly significant as signal edge rates and routing density increase [12, 14]. It is an electromagnetic coupling phenomenon where the electromagnetic field of a signal propagating along a transmission line affects neighboring conductors, causing unexpected signal changes. The edge field of a high-speed signal can penetrate the dielectric and affect adjacent traces. In high-speed PCB design, this coupling occurs through mutual capacitance (C_m) and mutual inductance (L_m) between traces. The intensity of crosstalk is primarily governed by the signal rise time (t_r), the parallel coupling length, and the edge-to-edge separation between conductors. In a multi-channel system, near-end crosstalk (NEXT) and far-end crosstalk (FEXT) can severely degrade the signal-to-noise ratio. From a field theory perspective, the electromagnetic field strength decays rapidly with increasing distance from the source. Therefore, maintaining adequate isolation—often defined in engineering practice by the spacing relative to the trace width—is a fundamental requirement to minimize deterministic jitter and maintain signal integrity in dense routing environments.

The internal delay of a differential pair refers to the time difference between the arrival of the inverted signal (N) and the non-inverted signal (P) within the pair [15]. It is a key factor in balanced transmission systems. Any physical length mismatch between the two lines shifts the relative phase of the signals. In the frequency domain, this phase inconsistency causes a portion of the differential-mode energy

to be converted into a common-mode component, a process known as mode conversion. Unlike differential signals, the converted common-mode component does not cancel at the receiver. Instead, it returns through the reference plane, potentially leading to increased electromagnetic interference (EMI) and reduced timing margins. Consequently, ensuring phase alignment through precise length matching is a physical prerequisite for suppressing common-mode radiation and ensuring protocol-compliant timing in high-speed links.

2.4 Embedded Linux Camera Software Stack

Linux V4L2 refers to the video device processing architecture for Linux systems and provides a standardized interface for video capture devices in Linux-based embedded platforms [16]. The overall framework diagram is shown in Figure 2.3.

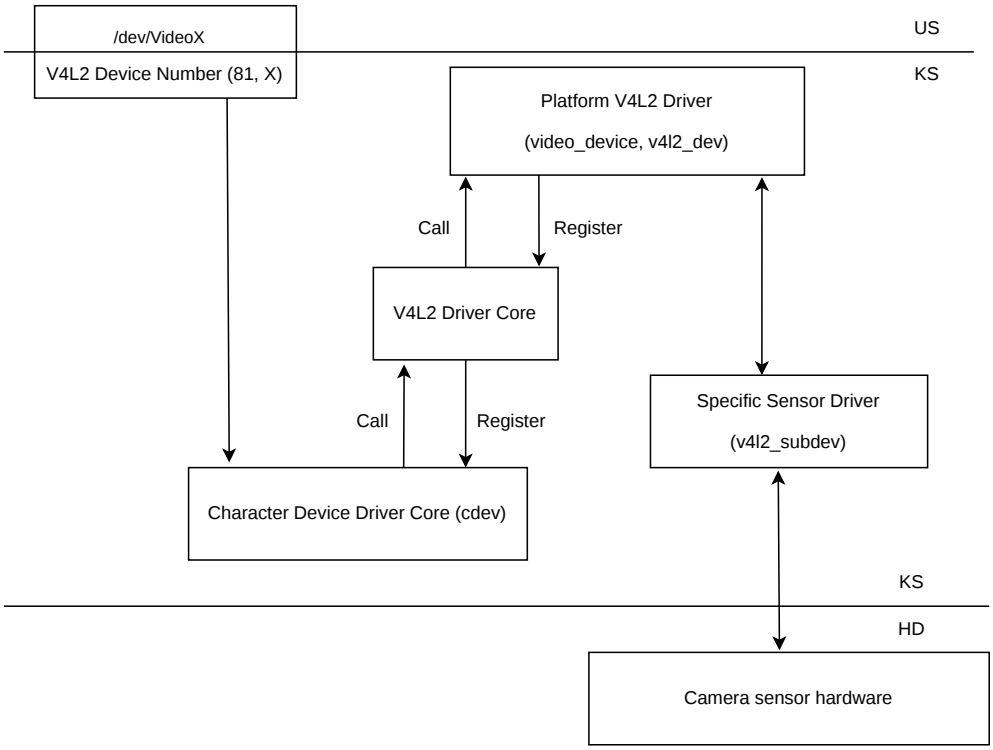


Figure 2.3: Linux V4L2 architecture. Adapted from [17]

As illustrated in the figure, the V4L2 framework consists of several layered components. The character device driver core exposes interfaces to user space, creating device nodes such as /dev/videoX. The V4L2 driver core provides a unified interface for video operations within the kernel. Beneath it, the platform V4L2 device driver implements platform-specific components based on the underlying hardware

characteristics. Finally, the sensor driver implements device control methods and registers as a v4l2_subdev with the V4L2 framework.

Modern embedded camera systems typically employ the Linux Media Controller framework to describe relationships between sensors, CSI receivers, and processing components within a video pipeline [17].

2.5 Device Tree

The Device Tree (DT) is a standardized mechanism for describing hardware devices and their relationships in embedded Linux systems [18, 19]. It uses a tree structure to describe the configuration and characteristics of hardware resources. As shown in Figure 2.4, its syntax resembles a tree, with the system bus as the trunk and I^2C controllers, GPIO controllers, SPI controllers, etc., as branches connected to the system bus. Each device is a node, called a device node. Each node is described by certain attribute information. The file describing the device tree is called the DTS (Device Tree Source). This DTS file uses a tree structure to describe the device information on the development board, such as the number of CPUs, memory base addresses, which devices are connected to the I^2C interface, which devices are connected to the SPI interface, and so on.

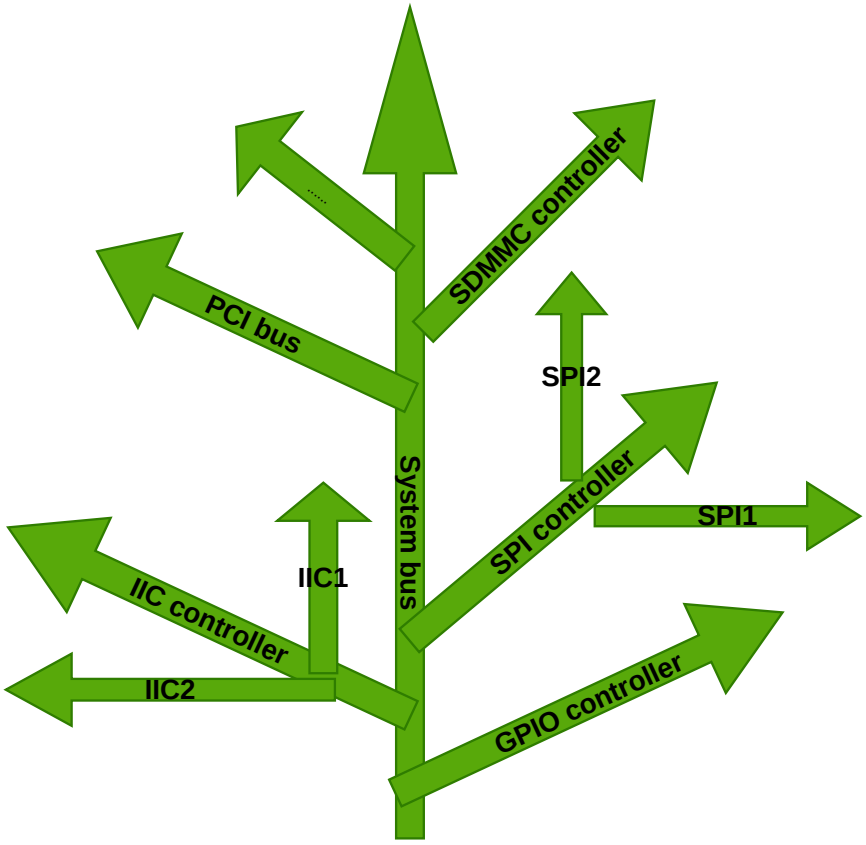


Figure 2.4: Device Tree

On the Raspberry Pi platform, the default hardware description can be extended using the Device Tree Overlay mechanism. For camera devices, the Device Tree typically describes their connection to the system and the hardware resources required for operation, including I^2C bus connections, clock sources, power supply configurations, MIPI CSI-2 interfaces, and module-related information. By configuring this information, new hardware configurations and device topologies can be added to the system without modifying the kernel source code. During system startup, the bootloader loads the corresponding Overlay file, merges it with the base Device Tree, and then the Linux kernel initializes the device based on the final generated hardware description. This mechanism provides a flexible implementation method for hardware integration in embedded systems.

3

Methodology

This chapter explains in detail the methods adopted during the project implementation. It first introduces the overall research methodology, then explains the design strategies from both hardware design and software development perspectives, and finally introduces experimental evaluation methods based on physical measurements.

3.1 Research Approach

This study employs a combined approach of engineering prototype construction and experimental verification. The goal is to evaluate how introducing a high-speed physical switching path affects the MIPI CSI-2 link on a compact embedded platform. The focus is on whether the MIPI CSI-2 link can maintain basic functionality after integrating an expansion board. The core idea is to first design and implement a compact PCB prototype supporting high-speed differential signal switching. Then, system-level testing, debugging, and fault localization are conducted to verify the normal operation of the control link and identify the fault points in the high-speed data path. This method emphasizes practical evaluation under actual layout constraints, rather than relying solely on theoretical modeling or simulation analysis.

3.2 Hardware Design Method

To achieve physical multiplexing of multiple camera signals, this research selects a high-speed differential analog switch chip as the core switching device. The selection is guided by two key criteria: sufficient analog bandwidth and low on-capacitance, in order to minimize insertion loss and signal distortion in the high-speed path. During the layout phase, chip placement is optimized to keep the length of new signal paths within a controllable range, and ground vias enhance the continuity of return paths. To ensure stable transmission of high-speed differential signals, a four-layer PCB stack-up with a dedicated ground reference plane is adopted. During the routing stage, a set of layout constraints is applied, including:

- Differential pair length matching
- Avoid unnecessary via jumps
- Reduce sharp-angle bends

- Maintain consistent differential spacing
- Adjust trace width and spacing to constrain the target differential impedance within the 100 Ω design range.

These constraints are derived from signal integrity considerations discussed in the technical background and are used to reduce reflections, impedance discontinuities, and common-mode noise.

3.3 Software Development Strategy

The software development portion of this study can be divided into two verification strategies: using Python scripts to verify functionality and using a kernel driver integration scheme to ensure subsequent stability.

In the first strategy, the SEL pin of the MIPI switch chip is controlled via the RPI.GPIO library for channel switching, and the I^2C MUX chip's channel switching is operated via the smbus2 library. This method allows for rapid functional verification, checking the correctness of the GPIO logic and I^2C MUX channel switching. After verifying that the functionality is normal, to prevent resource contention between the user-space program and the Raspberry Pi system's built-in driver, a device tree overlay scheme is adopted. This transfers the I^2C MUX topology and GPIO control information to the kernel, allowing the kernel to manage camera switching. This transfers hardware control from user space to kernel space, fundamentally eliminating potential contention.

This strategy of first verifying in user space and then integrating in kernel space allows for rapid verification of the path in the early stages of hardware design. After the hardware is finalized, the control logic is then solidified into the kernel's device tree configuration, ensuring the stability and maintainability of the subsequent system.

3.4 Experimental Evaluation Method

This study employs a physical measurement-based approach to validate the proposed hardware scheme.

During the design phase, impedance calculation tools were used to calculate trace dimensions to achieve the target 100 Ω differential impedance. Ideally, after hardware implementation, a high-bandwidth oscilloscope should be used to acquire high-speed differential signal waveforms, and eye diagram analysis should be applied to evaluate changes in signal aperture. However, due to a lack of high-speed measurement equipment, this quantitative analysis was not performed in this study. At the system level, successfully identifying and stably receiving camera data streams is the primary criterion for validating the effectiveness of the hardware design in practical applications. A series of tests and fault localizations were conducted at this level, such as I^2C probing and GPIO verification.

4

Design

This chapter details the specific design scheme of the multi-camera switching system. First, it compares the advantages and disadvantages of different architectural schemes, demonstrating the feasibility of the cascaded MIPI CSI-2 switching scheme. Then, the overall system architecture and the basis for component selection is introduced. Next, the implementation details of the hardware design are presented from three aspects: PCB stack-up structure, layout strategy, and signal path design. Finally, it describes the design and implementation of the software part, including the user-mode verification scheme and the kernel-mode integration scheme.

4.1 Alternatives Comparison

To achieve fast switching between multiple cameras, different architectural approaches were evaluated at the system level, based on the goal that only one camera is active and outputting data to the processing unit at any given time. Several possible solutions were analyzed, including MIPI physical-layer switching, sensor-level control, and higher-level data aggregation methods.

First, a single-stage MIPI CSI-2 multiplexer is the most straightforward approach. In this scheme, a high-speed MIPI switch supporting multiple inputs (e.g., 4:1 or 8:1) is used to route one of several camera signals to the processor, controlled via registers or GPIO. This approach has a simple structure and a short signal path, resulting in low switching latency and relatively straightforward control logic. However, in practice, high-port-count MIPI switch chips are difficult to source, partly due to signal integrity challenges, higher design complexity, and limited market demand.

To address this, a cascaded MIPI CSI-2 switching scheme can be used. Cameras are grouped and switched in multiple stages until a single output is selected. This hierarchical design improves scalability and reduces the load on any single device, making high-speed routing more manageable. It also allows flexible grouping based on physical placement (e.g., front, bottom, rear cameras). In addition, 2:1 MIPI switch chips are widely available and relatively mature in design. The trade-off is increased system complexity, more components, and more complicated control between stages, as well as an acceptable increase in latency. Overall, this approach offers a good balance between scalability and complexity.

Another option is to control camera power or enable signals so that only one sensor is active at a time, avoiding conflicts on the shared interface. However, this is

not suitable for MIPI CSI-2 systems. The interface is point-to-point and does not support multiple transmitters on the same physical link. In addition, each sensor requires re-initialization after power cycling, including register configuration and clock stabilization, which introduces significant switching delay.

At the protocol level, CSI-2 itself supports the Virtual Channel mechanism, which theoretically allows the transmission of multiple image data streams over the same physical link using different channel IDs. However, in practical systems, this approach is limited by platform support. For example, on the platform used in this project, although the CSI-2 controller supports the standard at the hardware level, the upper-layer software stack (such as libcamera and ISP processing flow) is typically designed for a single video stream and lacks full support for parallel processing of multiple virtual channels. Therefore, this approach is difficult to implement in the current system environment.

Finally, a more flexible but complex option is to use an FPGA or dedicated video aggregator. Such a system can receive multiple MIPI inputs, buffer and process the data, and then output a selected stream to the processor. This enables advanced features such as multi-stream processing and image stitching [20]. However, it significantly increases cost and design complexity, as implementing MIPI D-PHY/CSI-2 requires substantial hardware and logic development effort, as well as additional memory and high-speed interfaces. For a system focused only on fast switching, this solution is unnecessary and overengineered.

In summary, these approaches involve clear trade-offs between complexity, scalability, and implementation effort. High-port-count MIPI switches are hard to source, virtual channels are limited by software support, and FPGA-based solutions are overly complex. Among these, physical-layer MIPI switching provides the most balanced trade-off for this project and was therefore selected as the main implementation approach.

4.2 Overall System Architecture

This system is designed for applications with multiple camera inputs and a single CSI-2 output. Its core constraint is that the CSI-2 interface uses a point-to-point transmission structure, meaning only one camera can be the data source at any given time. Based on this constraint, the system employs a hierarchical cascaded MIPI switch architecture to achieve step-by-step convergence of multiple signals. The overall structure consists of two stages of switches, enabling signal selection from four inputs to a single output. The system logic architecture is shown in Figure 4.1. In the first stage, two 2:1 MIPI switches perform initial grouping selection for the four cameras, reducing the input signals to two intermediate signals. In the second stage, a third 2:1 switch performs the final selection on these two signals and outputs a single CSI-2 data stream to the main control unit.

The main advantage of this cascaded structure is that it distributes the selection process across multiple physical layers, thus avoiding the complex routing and dense wiring problems associated with single-stage multi-input structures in high-speed

differential distribution lines. Simultaneously, each stage of the switch chip maintains independent control, allowing the system to complete path selection using simple GPIO signals without affecting the high-speed data path itself. In addition, the camera configuration is completed through an independent I^2C bus, which is completely isolated from the CSI-2 data link at the physical layer, thereby avoiding interference from control signals to the high-speed transmission channel.

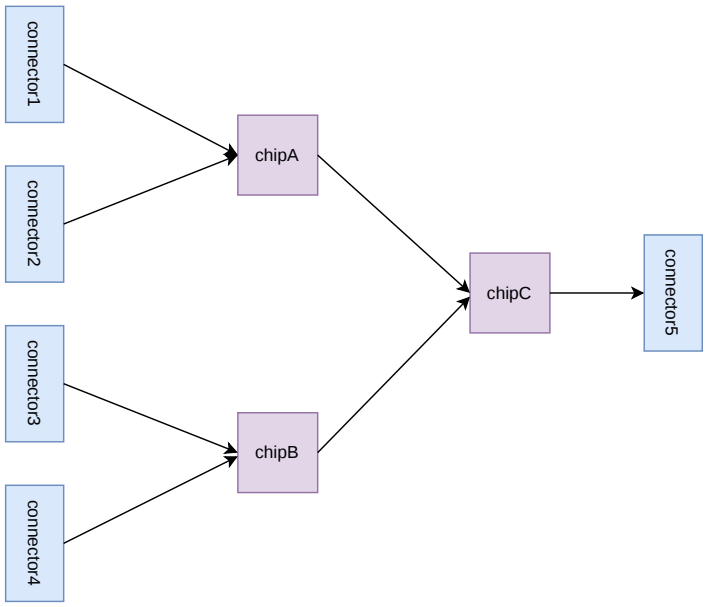


Figure 4.1: Overall system architecture

4.3 Device Selection

The key selection constraints of this system primarily come from three aspects: the integrity requirements of high-speed CSI-2 differential signals, the routability feasibility under limited PCB space constraints, and the system stability requirements in multi-camera switching scenarios. Therefore, component selection must not only meet the basic bandwidth specifications but also comprehensively consider the impact of the switch device's input/output capacitance on differential impedance continuity, as well as the package size limitations on high-speed differential routing layout.

Under the above constraints, to achieve physical-layer switching of multiple MIPI CSI-2 camera signals, a high-speed differential analog switch is required as the core switching device. Such devices must satisfy the following two key design requirements:

Bandwidth: The fundamental frequency of a MIPI 1.5 Gbps signal is 750 MHz (calculated as $f_0 = \frac{\text{Data Rate}}{2}$). According to Fourier transform, a periodic square wave is formed by the superposition of the fundamental frequency and multiple harmonics. If the switching bandwidth is insufficient, higher harmonics will be filtered out, and the originally square signal waveform will become a sine wave [12].

The rising edge of a sine wave is relatively slow, which can cause clock jitter and bit errors in high-speed transmission. To ensure that the rising edge of the waveform is steep enough, the chip bandwidth is typically chosen to be at least three times the fundamental frequency, i.e., 2.25 Gbps.

On-Capacitance: In high-speed MIPI links, on-capacitance (C_{on}) locally reduces the characteristic impedance of the transmission line, causing impedance discontinuities and leading to signal reflection and return loss. To ensure that the link impedance fluctuation is maintained within 10% of 100 Ω, this study sets the selection criterion as $C_{\text{on}} < 3$ pF.

After preliminary research and parameter comparison, the performance indicators of the candidate chips are summarized in Table 4.1. The data in the table shows that all three chips meet the transmission requirements of a 1.5 Gbps MIPI link in terms of key electrical performance indicators (bandwidth and on-capacitance). Especially in terms of signal integrity, the PI3WVR626 and TS3DV642 perform best. However, considering that the core application scenario of this study is a resource-constrained compact UAV platform, PCB space utilization is a key consideration besides signal integrity. Although the TS3DV642 provides high bandwidth margin, its 31.5 mm^2 package area is significantly larger than the PI3WVR626's 6.25 mm^2. The PI3WVR626 can reduce the occupied area by approximately 80% while ensuring high-performance signal switching, leaving more physical space for equal-length routing of high-speed differential pairs. While the AW35645 offers a slightly smaller footprint than the PI3WVR626, its FCBGA package introduces greater challenges in manufacturing and subsequent debugging. In conclusion, this study ultimately selected the PI3WVR626 as the core high-speed switching chip.

Table 4.1: Technical Specifications Comparison of Candidate Chips

Parameter	**PI3WVR626**	**AW35645**	**TS3DV642**
Bandwidth	6.0 GHz	3.5 GHz	6.9 GHz
On-Cap (C_{ON})	1.3 pF	1.5 pF	1.3 pF
Package	X1QFN-24	FCBGA-36	WQFN-42
Area ($mm \times mm$)	2.5×2.5	2.4×2.4	9.0×3.5
Total Area (mm^2)	6.25	5.76	31.5

4.4 PCB Stack-up

While two-layer PCBs offer advantages in cost control, they consist of only two signal layers (top and bottom) and lack a continuous reference plane, resulting in poor electromagnetic compatibility (EMC) at high frequencies. Furthermore, the lack of a stable electromagnetic reference environment causes characteristic impedance to fluctuate drastically along the trace path, making impedance matching difficult.

In contrast, four-layer PCB architectures offer significant advantages in routing space and signal integrity. First, a complete ground plane provides a defined and controlled

characteristic impedance environment for high-speed signals, a physical prerequisite for achieving 100 Ω differential matching. Second, the four-layer structure allows for the establishment of a minimum return path, effectively reducing current loop area and thus lowering electromagnetic radiation and magnetic field coupling [21].

The stack-up sequence designed in this study follows a symmetrical "signal-plane-plane-signal" principle, specifically defined as follows:

L1 (Top Signal Layer): Deploys the main MIPI differential pairs and core MIPI switching chip components. This layer is adjacent to the L2 ground plane, enabling high-precision impedance control and minimizing loop inductance.

L2 (GND Plane): Serves as the core reference plane. Provides the shortest return path for L1 layer signals and also acts as a shielding layer against high-frequency interference, isolating switching noise that may be generated by the L3 power plane.

L3 (Power Plane): Responsible for low-impedance power distribution to the voltage rails. L3 is tightly coupled to the adjacent L2, forming an interlayer capacitance, assisting in high-frequency decoupling and improving power supply stability.

L4 (Bottom Signal Layer): Carries the remaining MIPI traces that cannot be completed on the top layer due to space constraints, as well as non-sensitive low-speed control signals (GPIO/I^2C). When routing on this layer, the signal will be referenced to the L3 power plane. Extra attention needs to be paid to the return continuity when switching between layers in the design.

4.5 Layout and Signal Path Design

4.5.1 Layout Considerations

This design uses a 4-layer PCB with components placed on both sides. This approach achieves high density and compactness while optimizing high-speed signal links. As discussed in Section 4.4 on the PCB stack-up, a continuous ground reference is maintained in the inner layers, which is essential for controlled impedance routing. To improve routing efficiency, differential pairs are distributed across both top and bottom layers in a staggered manner. This helps shorten total trace length and reduces unnecessary layer transitions, which would otherwise increase insertion loss and routing complexity. Decoupling capacitors are placed on the back side of the corresponding ICs close to power pins, in order to minimize loop inductance. This improves the high-frequency decoupling performance and reduces local power supply noise during fast switching.

To accommodate multiple camera inputs within a limited PCB area, three switching chips are arranged along a vertical axis on the top layer. Flexible printed circuit (FPC) connectors are used because they are compatible with the Raspberry Pi CSI interface and the IMX219 camera modules, while also providing a lightweight and low-profile connection suitable for compact UAV hardware. At the first stage, two switching chips (Chip A and Chip B) are placed in series along this axis. Each first-stage chip connects to two FPC camera connectors located at the left and right edges of the PCB. To alleviate routing congestion and avoid excessive crossings

on a single layer, signals are distributed across both top and bottom layers in an interleaved manner. For Chip A, the left-side FPC connector is routed on the top layer, while the right-side FPC connector is routed through vias to the bottom layer. For Chip B, the routing scheme is reversed: the left-side FPC connector is connected through the bottom layer, and the right-side FPC connector is routed on the top layer. This complementary arrangement balances routing density between layers and reduces local congestion. At the second stage, Chip C is placed below the first-stage chips along the same vertical axis and serves as the aggregation node, receiving outputs from both Chip A and Chip B. Signal transitions between stages involve layer changes, where ground vias are placed adjacent to signal vias to maintain return path continuity and reduce loop inductance. This vertical cascaded and layer-interleaved structure improves routing compactness while maintaining signal integrity in a high-density design.

In this 4-layer stack-up architecture, the complete ground/power plane of the intermediate layers (Layer 2/3) is used as an electromagnetic shielding layer. Low-speed control signals such as I^2C and GPIO are deployed on the bottom layer, which physically decouples them from the high-speed MIPI differential lines on the top layer in the vertical direction, preventing steep edge noise of low-speed signals from inductively coupling the differential pairs.

4.5.2 Signal Path Design

As introduced in the signal integrity section, maintaining controlled differential impedance is critical for high-speed CSI-2 transmission. In this design, the differential impedance is modeled using a standard approximation [22]:

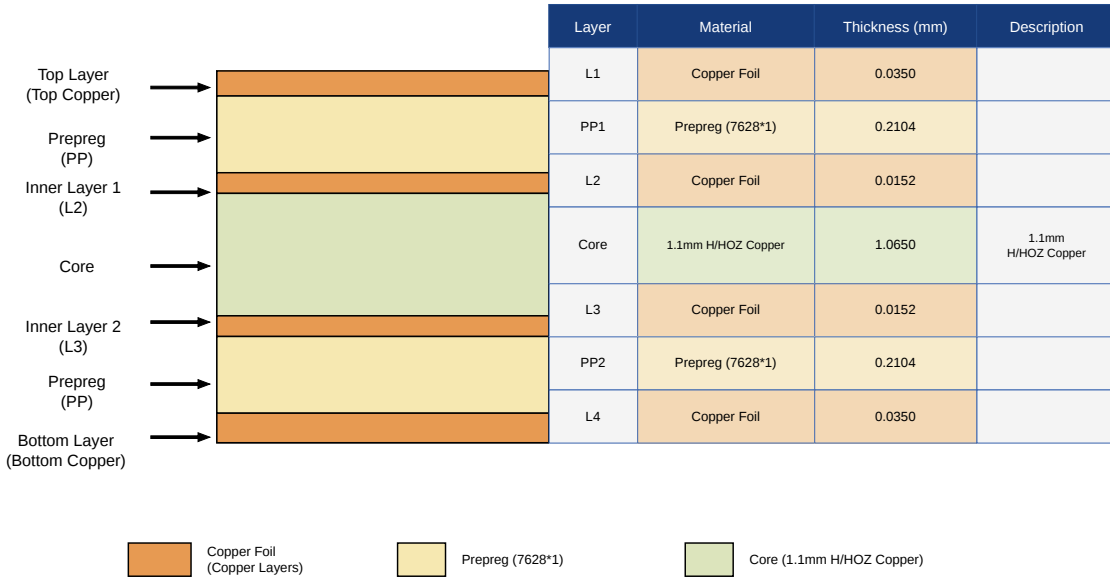
$$Z_{\text{diff}} \approx 2Z_0 \left(1 - 0.48 \cdot e^{-0.96 \cdot S/H}\right) \tag{4.1}$$

Based on this model, routing parameters were selected using the Saturn PCB Toolkit to meet the target impedance of 100 Ω [23]. A tightly coupled routing strategy with equal width and spacing (W = S = 0.15 mm) was adopted. This configuration improves field coupling between differential pairs and enhances common-mode noise rejection. It also reduces dependence on the reference plane during layer transitions.

Simulation results show that the differential impedance remains within 100.1 Ω to 100.9 Ω across the expected variation of dielectric constant, ensuring stable transmission performance.

- Target impedance: 100 Ω
- Trace width / spacing: 0.15 mm / 0.15 mm
- Dielectric height: 0.2 mm
- Dielectric constant: $\epsilon_r \approx 4.5$

In actual production, according to the manufacturer's process requirements, the final four-layer lamination structure is shown in Figure 4.2, with the total impedance within the range of 100 Ω ±10%.



Layer	Material	Thickness (mm)	Description
L1	Copper Foil	0.0350	
PP1	Prepreg (7628*1)	0.2104	
L2	Copper Foil	0.0152	
Core	1.1mm H/HOZ Copper	1.0650	1.1mm H/HOZ Copper
L3	Copper Foil	0.0152	
PP2	Prepreg (7628*1)	0.2104	
L4	Copper Foil	0.0350	

Figure 4.2: Four-layer lamination structure

Based on the impedance design described above and the signal integrity considerations discussed in Section Technical Background, additional routing constraints are applied to ensure timing consistency and minimize electromagnetic interference.

Within each differential pair, length mismatch is controlled within 5 mil to reduce intra-pair phase imbalance. For unavoidable mismatches introduced by vias or component placement, localized serpentine compensation is applied near the source end to preserve symmetry. Between clock and data lanes, the total length mismatch is kept within 100 mil to ensure correct source-synchronous sampling alignment at the receiver. To reduce crosstalk, a 3W spacing rule is applied between high-speed traces, where the minimum spacing is at least three times the trace width [24]. This helps limit electromagnetic coupling between adjacent signals and improves noise immunity in dense routing areas. Routing corners are implemented using smooth or angled transitions instead of sharp bends to avoid abrupt impedance discontinuities and local reflections.

In addition to trace-level constraints, discontinuities introduced by vertical interconnects must also be carefully managed. In double-sided high-density cabling, vias generated during signal layer switching are essential for maintaining interconnectivity. However, the parasitic capacitance introduced by vias often causes signal edge degradation (rounding) and local impedance reduction. This design employs differentiated via modeling and compensation measures to address the different electrical requirements of high-speed and low-speed signals. In high-speed signal transmission, the via parasitic capacitance can be expressed as [25]

$$C \approx \frac{1.41\epsilon_r T D_1}{D_2 - D_1}, \tag{4.2}$$

where D_1 is the pad diameter and D_2 is the antipad aperture diameter on the reference plane. As the formula shows, smaller pads help maintain the via impedance

the same as that of the high-speed line. Simultaneously, considering the extreme values of current mainstream four-layer PCB manufacturing processes, and without increasing special process costs, this design uniformly uses 0.45 mm (diameter) / 0.2 mm (hole) vias on high-speed links to achieve higher signal integrity. For low-speed control signals such as I^2C and GPIO, general-purpose 0.5 mm/0.3 mm vias are used. In these non-sensitive signal paths, larger via diameters offer better mechanical strength and processing yield.

When high-speed signals pass through vias to change layers, the reference plane also changes. If the return current cannot quickly transition between layers, a large loop inductance will be generated. To ensure that the return signal can closely follow the signal path to cross layers, ground vias are symmetrically arranged in the vicinity of the high-speed differential vias. This design ensures the electromagnetic compatibility of the system by suppressing common-mode radiation caused by reference plane discontinuity.

As shown in full-layer Figure 4.3, the overall PCB size is 38.6 mm × 33.6 mm, meeting the requirements of a compact embedded system. Three high-speed switching chips (Chip A, B, C) are vertically centered, with four MIPI input ports symmetrically distributed on both sides of the board. This symmetrical design reduces the physical path differences between each input channel and the central cascade node. Partial routing shows that all differential traces maintain a constant line width and spacing, strictly matching the 100 Ω differential impedance requirement, reducing reflection losses caused by sudden changes in link impedance.

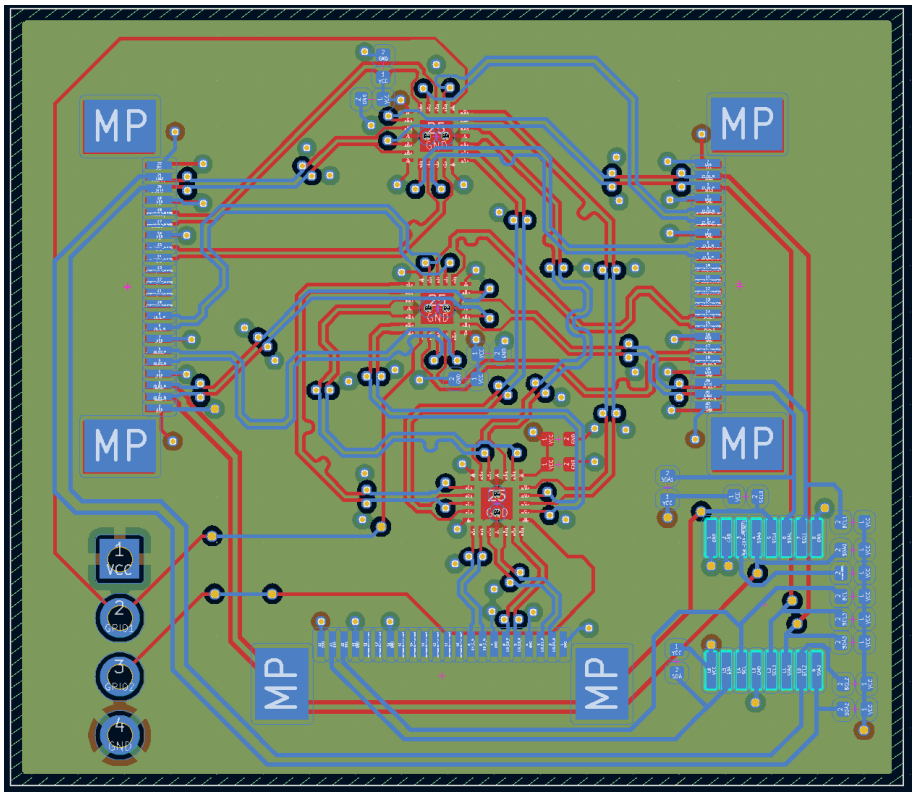
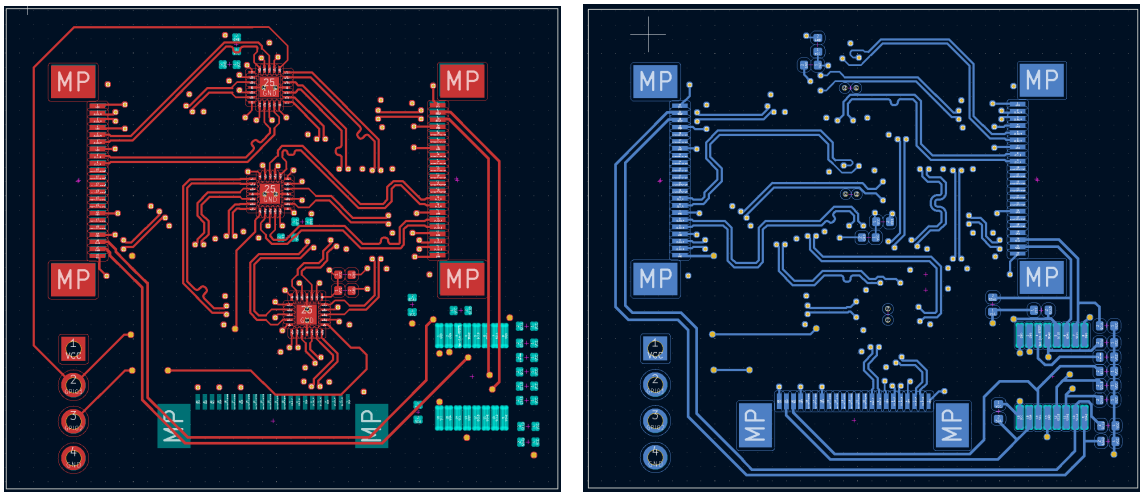


Figure 4.3: Full layer

In addition to the full-layer view, Figure 4.4 also shows the top and bottom signal layers separately. The top layer mainly contains the MIPI switching chips and part of the high-speed differential traces, while the bottom layer carries the remaining high-speed differential traces and low-speed control signals. The high-speed traces on the two signal layers are arranged in an interleaved manner, improving space utilization and reducing local routing congestion.



(a) Top layer **(b)** Bottom layer

Figure 4.4: Top and bottom layers

4.6 Software Design and Implementation

4.6.1 Software Overall Architecture

This design employs two software development approaches, with the architecture shown in Figure 4.5. The left side of the diagram represents the Python approach. The application layer sends GPIO and I^2C commands via the RPi.GPIO and smbus2 libraries, and the kernel layer transparently transmits these commands to the hardware. This approach offers fast development speed and is suitable for prototype verification. The right side represents the device tree approach. The application layer calls the standard 'rpicam-hello' interface, and the IMX219 and PCA9546 drivers in the kernel automatically handle I^2C MUX channel switching and MIPI routing control based on the hardware topology described by the device tree. This approach eliminates resource contention between user space and kernel drivers.

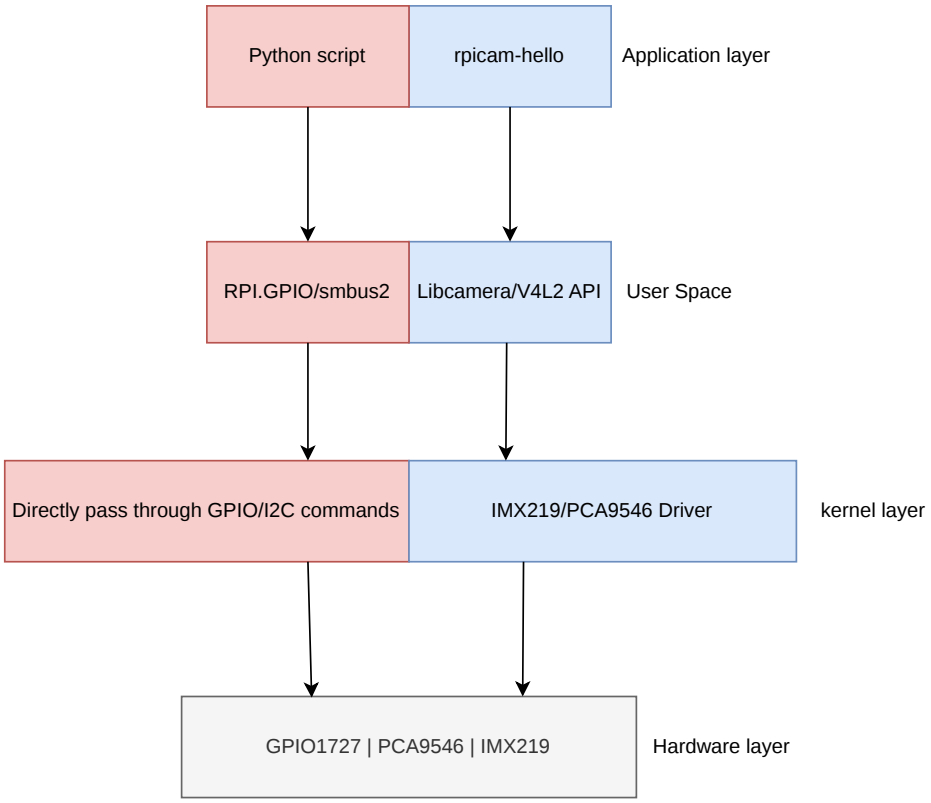


Figure 4.5: Software Overall Architecture

4.6.2 User-mode control scheme

The software component mainly consists of a channel mapping module, a GPIO control module, an I^2C control module, and a switching process module.

In the channel mapping module, a dictionary is used to define the correspondence between camera ID, GPIO level combinations, and I^2C MUX channel values. The mapping relationship is as follows:

```
CAMERA_MAP = {
    0: {"gpio": (0, 0), "mux_ch": 0x01},
    1: {"gpio": (1, 0), "mux_ch": 0x02},
    2: {"gpio": (0, 1), "mux_ch": 0x04},
    3: {"gpio": (1, 1), "mux_ch": 0x08},
}

I2C_MUX_ADDR = 0x70
```

In the GPIO control module, the SEL pin of the MIPI chip is controlled via Raspberry Pi GPIO17 and GPIO27, and the RPI.GPIO library is used to control the levels of these two GPIOs. During switching, the corresponding level combination is output according to the channel mapping relationship, with a 50 ms delay inserted

to wait for the switch to stabilize.

```
def set_gpio(cam_id):
    s0, s1 = CAMERA_MAP[cam_id]["gpio"]
    GPIO.output(SEL0, s0)
    GPIO.output(SEL1, s1)
    time.sleep(0.05)
```

In the I²C control module, the PCA9546 chip (address 0x70) is connected to bus 10, and the smbus2 library is used to operate the I²C lines. The write_byte() function writes the channel selection byte to the chip, with a 100ms delay inserted to ensure I²C bus stability.

```
def select_i2c_channel(cam_id):
    ch = CAMERA_MAP[cam_id]["mux_ch"]
    bus.write_byte(0x70, ch)
    time.sleep(0.1)
```

In the switching module, GPIO switching, I²C MUX switching, and preview startup are called sequentially to complete the complete camera switching process.

```
def switch_camera(self, cam_id):
    set_gpio(cam_id)
    select_i2c_channel(cam_id)
    restart_pipeline(cam_id)
```

In the preview startup module, restart_pipeline calls the system command 'rpicam-hello -t 2000' to start a 2-second preview to verify the image acquisition function after switching.

4.6.3 Kernel-mode control scheme

To avoid conflicts between user programs and the kernel during subsequent system integration, after system functionality verification, a device tree can be added to pass hardware information to the kernel, allowing the kernel to centrally manage GPIO switching, I²C switching, etc. To reduce debugging complexity, this study first implemented a single-channel minimal device tree, supporting the camera on channel 0. The device tree overlay file for this version contains four parts: I²C MUX node definition, CSI receiver endpoint binding, clock source definition, and GPIO control pin configuration. The core structure of the device tree is as follows:

```
/ {
    fragment@0 {
        target = <&i2c_csi_dsi>;
        __overlay__ {
            pca9546@70 {
                compatible = "ti,pca9546";
                reg = <0x70>;

                i2c@0 {
                    reg = <0>;
```

```
                    camera@10 {
                        compatible = "sony,imx219";
                        reg = <0x10>;

                    };
                };

            };
        };
    };

    fragment@1 {
        target = <&csi0>;
        __overlay__ {
            port {
                csi0_ep: endpoint {
                    remote-endpoint = <&imx219_0_ep>;
                };
            };
        };
    };

    fragment@3 {
        target = <&gpio>;
        __overlay__ {
            mipi_switch_pins: mipi_switch_pins {
                brcm,pins = <17 27>;
                brcm,function = <1 1>;
                brcm,value = <0 0>;
            };
        };
    };
};
```

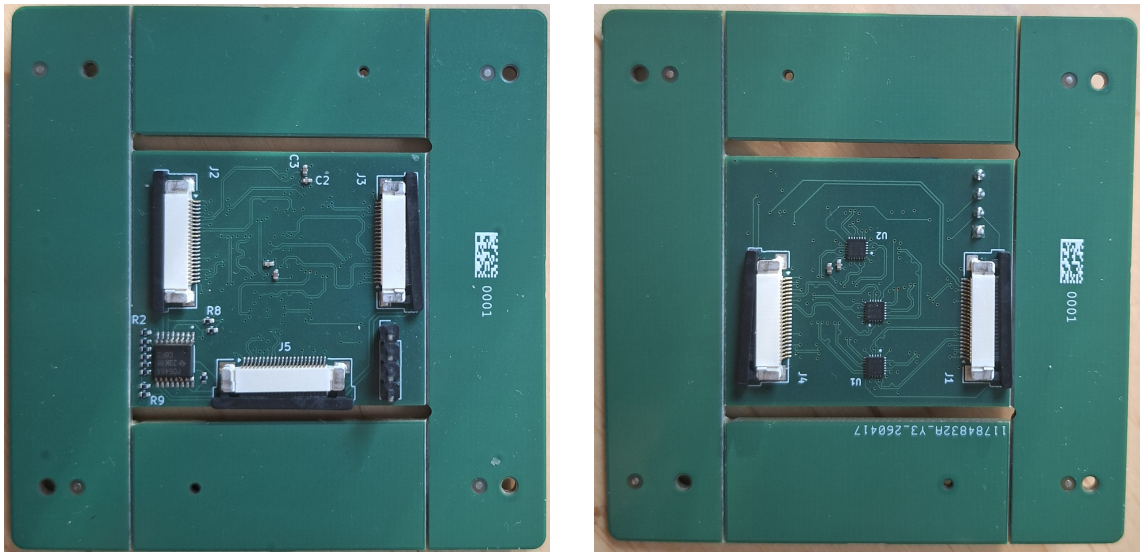
5

Results

This chapter details the experimental verification results of the system. The physical prototype and test environment configuration are introduced first. Through I^2C communication testing and GPIO function testing, the correctness of the control link is verified, with the troubleshooting process recorded in the I^2C testing section. Finally, system integration testing is conducted to present the experimental results, including direct connection testing, system testing after expansion board connection, video stream testing, and fault location analysis.

5.1 Physical Prototype and Test Environment

Figure 5.1 shows the fabricated PCB physical prototype, which has a mass of 0.12 kg. The dimensions of this expansion board are 38.6 mm × 33.6 mm, representing a size reduction of approximately 45% compared to current mainstream commercial counterparts (56 mm × 42 mm).



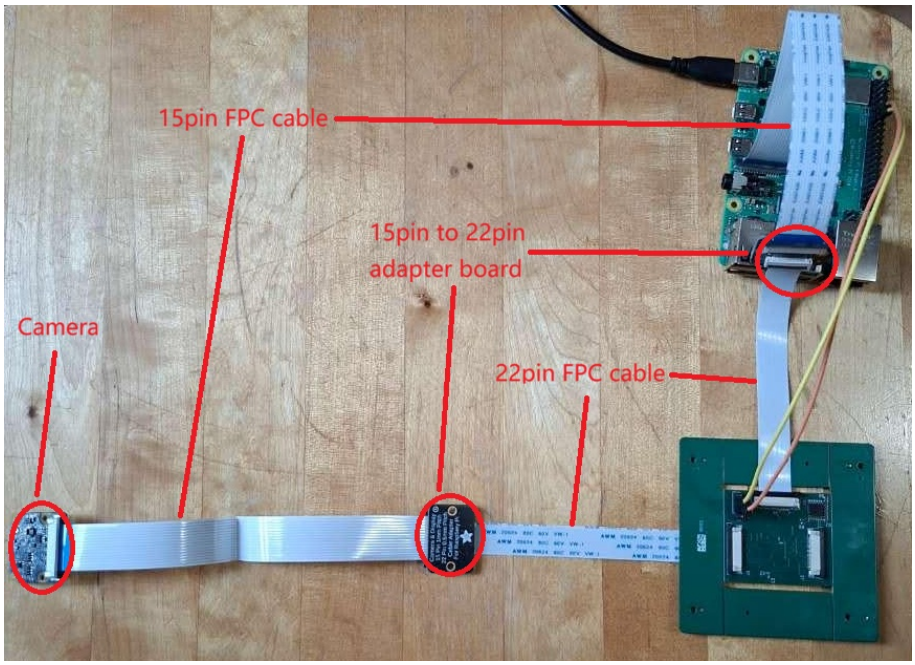
(a) Front view of the prototype

(b) Back view of the prototype

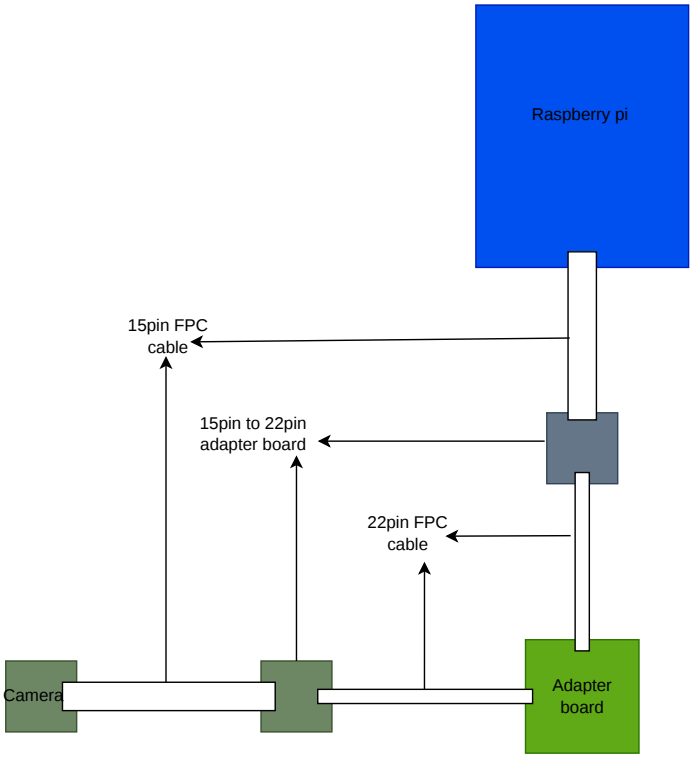
Figure 5.1: Prototype photograph

The test environment setup is shown in Figure 5.2. Because the FPC connector on the expansion board is 22 pin, it needs to be connected to the camera via a 15 pin

to 22 pin adapter board, and the same applies to the Raspberry Pi end.



(a) Experimental setup photograph



(b) System connection schematic

Figure 5.2: Experimental setup

5.2 I²C Communication Test

5.2.1 Direct Connection Test

We connect the camera directly to the CSI interface of the Raspberry Pi via an FPC cable. Then we execute the address detection command 'i2cdetect -y 10' in the terminal. The displayed addresses indicate that 10 and 64 are mounted on bus 10. These two are the addresses of the IMX219 chip of camera module v2 and the identity chip, respectively. Successfully and stably detecting these two addresses indicates that the I²C path of camera module v2 is normal.

5.2.2 I²C MUX Chip Testing

Now we connect the expansion board to the Raspberry Pi separately via an FPC cable, and execute the address probe command again. The address 70 of the I²C MUX chip on the expansion board was successfully detected, indicating that the I²C MUX path on the board is normal.

5.2.3 Camera Address Test

Building upon the previous step, connecting one camera to a channel on the expansion board and executing the same address probe command successfully displays the addresses of the I²C MUX chip and the camera's identity chip. The camera sensor module's address is displayed as UU, indicating successful binding to the driver. This demonstrates that the I²C link of the entire system is functioning correctly. However, the address probe results for the camera module's sensor chip are unstable, fluctuating between UU, 10, and no address detected. Troubleshooting will be discussed in next section.

5.2.4 Troubleshooting Process

After system integration, the entire link includes expansion board, additional FPC cables, and adapter boards. This increases the length of the signal lines and the parasitic capacitance, potentially causing a slower rise time for the I²C signal, leading to missed data reads by the main controller. Therefore, an attempt was made to reduce the I²C rate from 400 kHz to 10 kHz in the configuration file to allow more time for signal charging and discharging. However, the operation logs show that reducing the I²C rate did not solve or alleviate the problem, indicating that the issue is not due to insufficient timing setup/hold time.

The Raspberry Pi CSI I²C interface has a 1.8 k pull-up resistor, and the camera module has a 2.2 k pull-up resistor. On the expansion board's I²C MUX, each SCL and SDA data line is equipped with a 2.2 k pull-up resistor. When one channel is active, the equivalent pull-up resistor R_{eq} on each of these SCL and SDA data lines is as follows:

$$R_{\text{eq}} = \left(\frac{1}{1.8\,\text{k}\Omega} + \frac{1}{2.2\,\text{k}\Omega} + \frac{1}{2.2\,\text{k}\Omega} + \frac{1}{2.2\,\text{k}\Omega}\right)^{-1} \approx 521\,\Omega \tag{5.1}$$

The smaller the pull-up resistor, the faster the signal rises to a high level, enabling higher communication rates. However, when the bus is pulled low, a relatively large current is generated. If the device's pull-down capability is insufficient to withstand this current, it cannot effectively pull the level low, which may lead to communication errors such as ACK failure, data misjudgment, or even bus abnormalities. Therefore, we tried gradually removing the pull-up resistors on the expansion board to troubleshoot the problem. First, we removed the pull-up resistor on the side where the expansion board connects to the Raspberry Pi. After the channel was turned on, the equivalent resistance became:

$$R_{\text{eq1}} = \left(\frac{1}{1.8\,\text{k}\Omega} + \frac{1}{2.2\,\text{k}\Omega} + \frac{1}{2.2\,\text{k}\Omega}\right)^{-1} \approx 683\,\Omega \tag{5.2}$$

and there was no significant change when running commands. Then, we removed the pull-up resistor on the side where the expansion board connects to the camera. At this point, the equivalent resistance was:

$$R_{\text{eq2}} = \left(\frac{1}{1.8\,\text{k}\Omega} + \frac{1}{2.2\,\text{k}\Omega}\right)^{-1} \approx 990\,\Omega \tag{5.3}$$

and there was still no significant change. Therefore, we can conclude that the problem was not caused by excessively strong I²C pull-up resistors.

Upon examining the schematic design, it was discovered that pin 11 of the CSI interface, the power enable pin, was floating. The schematic of camera module v2 is shown in Figure 5.3, this enable pin is responsible for activating the LDO module in the camera module, generating three different voltages: 1.8 V, 2.8 V, and 1.2 V. This floating pin prevented the camera module from successfully generating the required voltage, while the reference voltage for the I²C communication pull-up resistor is 1.8 V, leading to I²C communication failure.

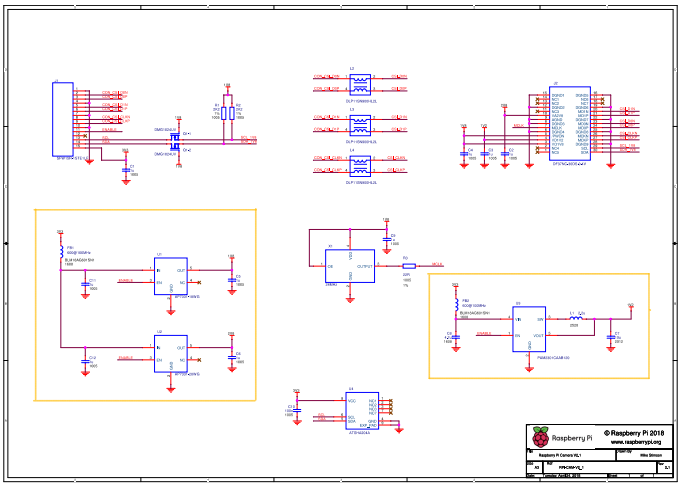


Figure 5.3: Camera schematic

The jump between the camera module address (UU,10,no address) and undetectable address during address detection is likely due to the 'antenna effect' caused by

the floating pin. A floating pin can randomly absorb electromagnetic noise in the environment, proximity of a hand or distance between different components can cause the camera chip's internal state machine to randomly switch between 'awake' and 'hard reset' states. Because FPC cables are inconvenient for jumper wires, and the pin pitch on the expansion board is too small, the 15-pin FPC connector on the adapter board has a recommended pin pitch of 1mm, which can be used as a temporary jumper wire location. On the adapter board connecting the camera, jumper wires are used to connect pin 11, the power enable pin, to GPIO22 on the Raspberry Pi. After powering on, before probing the camera on the Raspberry Pi, first pull GPIO22 low to reset, then pull it high to release the reset signal and execute the I^2C detect command. It can seen that the camera's identity chip address 64 can be reliably detected, and the I^2C MUX chip and the camera's sensor chip are successfully driven and occupied.

5.3 GPIO Functionality Testing

This design uses GPIO17 and GPIO27 of the Raspberry Pi to control the SEL pin of each chip stage. A Python script is used to control the GPIO to generate corresponding high and low levels to complete the channel switching of the MIPI switching chip. The SEL pin voltages of each chip are measured using a multimeter, and the results are summarized in Table 5.1. The results show that the SEL pin voltage is consistent with expectations, and the Python script can successfully control the SEL pin of the MIPI switching chip to complete the chip's channel switching.

SEL0 (GPIO17)	SEL1 (GPIO27)	Expected Channel	Measured SEL0 Voltage	Measured SEL1 Voltage
0	0	Channel 0	0.02 V	0.01 V
1	0	Channel 1	3.29 V	0.01 V
0	1	Channel 2	0.02 V	3.28 V
1	1	Channel 3	3.29 V	3.28 V

Table 5.1: SEL pin voltages and GPIO voltages

5.4 System Integration and Final Results

5.4.1 Direct connection test

Connecting the camera directly to the Raspberry Pi and executing the 'rpicam-hello' command result in a successful image preview window, indicating that the image transmission link between the camera module and the Raspberry Pi motherboard is functioning correctly.

5.4.2 System testing after connecting the expansion board

After verifying that the direct camera connection was working correctly, the expansion board was integrated into the Raspberry Pi camera system. To support the

expansion board's hardware architecture, a dedicated device tree was developed to describe the PCA9546 I²C multiplexer, the GPIO-controlled MIPI switching network, and the IMX219 image sensor. By executing the command 'rpicam-hello –list-cameras', the system successfully enumerated the IMX219 camera devices, indicating that the Linux camera framework could correctly detect the image sensor and complete driver registration. This verified the correctness of the I²C communication link, GPIO control logic, and Device Tree configuration. Subsequently, 'media-ctl' was used to check the media controller topology. The generated media graph showed a valid connection established between the IMX219 image sensor and the Raspberry Pi's Unicam receiver, and the sensor was successfully exposed as a V4L2 video device. This indicates that the system software stack was correctly configured, and the camera media pipeline could be successfully built. The above results show that the system's control link is working properly after the expansion board is integrated, providing a foundation for subsequent video stream transmission tests.

5.4.3 Streaming Test

After the above verification was completed, video data streaming was initiated via 'rpicam-hello', and the sensor initialization process and data reception were observed. The test results and operation logs show that the system successfully initialized and selected the corresponding working mode and data format. Simultaneously, the Unicam receiver also completed the corresponding format configuration. However, during the video streaming phase, the system continuously reported the 'Camera frontend has timed out' error and failed to receive valid image frames. Although the camera had completed initialization and entered streaming mode, the Raspberry Pi failed to successfully acquire the data stream from the image sensor. This phenomenon indicates that the system's software control link is working normally, but the video data link failed to establish effective data transmission.

5.4.4 Failure Localization

The preceding tests have demonstrated that I²C communication, GPIO control, device tree configuration, driver registration, and media pipe creation were all successfully completed. To further pinpoint the problem, we checked whether Unicam was receiving interrupts to determine if the Raspberry Pi was receiving data. We executed 'watch -n 1 "grep -i unicam /proc/interrupts"' and simultaneously observed the interrupt count while running 'rpicam-hello'. The results show that the Unicam interrupt count remained at zero, with no interrupt activity observed. This indicates that the Raspberry Pi's CSI receiver was not receiving valid CSI-2 image frames.

To verify whether insufficient power was causing the camera to fail to output MIPI data, additional power supply tests were performed on both the camera and expansion board. First, a 3.3 V and GND jumper wire was connected from the Raspberry Pi's 40-pin connector to the camera adapter board, making the camera's power supply path similar to that of a direct connection to the Raspberry Pi. Subsequently, the VCC and GND pins on the expansion board were directly connected to the 3.3 V and GND of the Raspberry Pi's 40-pin connector to provide an additional

low-impedance power supply path for the expansion board and the MIPI switching chip. However, in both cases, the test results did not improve, and video streaming still failed.

In conclusion, the fault is unlikely to be primarily caused by insufficient 3.3 V power supply to the camera module or expansion board. Based on the aforementioned verification results, the fault is more likely located in the MIPI CSI-2 high-speed data transmission path. The overall test results are shown in Table 5.2.

Table 5.2: Summary of system verification results

Verification Item	**Result**	**Evidence**
Direct camera connection	PASS	The IMX219 camera operated normally when directly connected to the Raspberry Pi CSI interface.
I^2C MUX detection	PASS	The PCA9546 I^2C MUX was detected on the I^2C bus.
Camera I^2C communication	PASS	Stable I^2C communication was achieved after connecting the camera ENABLE pin to GPIO22.
GPIO switching logic	PASS	The expected GPIO level combinations were generated for different camera channels.
Device Tree configuration	PASS	The IMX219 sensor node and PCA9546 topology were successfully described by the device tree overlay.
Camera enumeration	PASS	The camera was visible using `rpicam-hello -list-cameras`.
Media graph creation	PASS	A valid connection between the IMX219 sensor and the Unicam receiver was shown by `media-ctl`.
Video streaming	FAIL	`rpicam-hello` reported repeated frontend timeout errors during streaming.
Additional power supply verification	PASS	Additional 3.3 V and GND connections did not change the system behaviour, reducing the likelihood of a power supply issue.
Unicam interrupt activity	FAIL	The Unicam interrupt counter remained zero during streaming attempts.
Final localization	PARTIAL	The remaining fault was narrowed down to the MIPI CSI-2 high-speed transmission path.

6

Discussion

This chapter is divided into four parts. First, the results presented in the previous chapter are analyzed and discussed. Next, the limitations of the current design and possible improvements are examined. Then, directions for future work are outlined. Finally, the use of Large Language Models in this thesis is described.

6.1 Interpretation of Results

The experimental results in Section 5 show that the system control link functions correctly, while the fault lies in the MIPI CSI-2 high-speed data transmission link. I^2C communication, GPIO control, device tree configuration, driver registration, and media pipe creation all completed successfully, but the Raspberry Pi's CSI receiver consistently failed to receive valid image data.

The reasons for this phenomenon may include the following: First, the system uses multiple FPC cables and adapter boards, increasing the complexity of the high-speed signal transmission path. Although the expansion board is theoretically designed to withstand 100 Ω, actual manufacturing errors and via structures may cause the actual manufactured expansion board to deviate from the design value. Furthermore, some of the FPC cables used in the system are not dedicated CSI cables, and their impedance control cannot be guaranteed. Therefore, the high-speed differential signal may be affected by impedance discontinuities, reflections, and insertion loss in this long link, leading to a decrease in signal integrity and preventing the receiver from correctly recovering the clock and data [26].

Second, after initialization, the MIPI CSI-2 interface needs to switch from Low-Power (LP) mode to High-Speed (HS) mode for data transmission. If the clock signal is not received or locked correctly, this switch may not complete properly, resulting in the inability to establish a normal data transmission link. Furthermore, the fault may also originate from physical layer connection problems, such as open circuits in differential signal lines, poor connector contact, soldering defects, or malfunctioning MIPI switching chips. Due to the lack of high-speed signal measurement equipment, it is impossible to further confirm the specific fault point; therefore, the fault can currently only be narrowed down to the MIPI CSI-2 high-speed transmission path.

6.2 Potential Improvements

Based on the experience gained from this debugging, the following improvements can be made to the next PCB revision. First, reduce the number of intermediate layers. In this design, a 22 pin FPC connector board was chosen instead of the commonly used 15 pin board to reduce the expansion board area. However, due to pin order issues, using the official 15 pin to 22 pin FPC cable resulted in pin reversal, requiring an adapter board for conversion between 15 pin and 22 pin during final testing. In future optimizations, the order of the FPC connectors on the expansion board could be reversed, avoiding the need for additional adapter boards and FPC cables. Alternatively, a custom 15 pin to 22 pin Opposite-sided FPC cable could achieve the same effect, although this would incur additional costs. From a cost perspective, the first method is simpler and more efficient.

Second, increase the number of test points. This design uses only FPC cables, and the small pin pitch on the expansion board lacks suitable test points, increasing the difficulty of the debugging process. Future versions can add test pads or test interfaces to the camera input, MIPI switch output, and Raspberry Pi connection to enable continuous testing, voltage measurement, and high-speed signal analysis. This will significantly improve system debuggability and shorten fault location time.

Furthermore, the current design is powered via the CSI interface's VCC. While this works fine with a single camera, insufficient power may occur when all four cameras are connected to the expansion board. Therefore, future optimizations could add an LDO power module to the expansion board, providing a stable 5 V power supply via the Raspberry Pi, instead of powering the entire expansion board and all four cameras via the CSI interface's VCC.

6.3 Future Work

Future work could initially focus on optimizing and manufacturing the next PCB version. The current PCB has added more components due to pinout issues with some FPC connectors. Future work could correct these errors, shorten data paths, remove redundant components, and test the data link for proper functioning.

After resolving the faults and successfully establishing video stream data transmission, the overall system performance could be tested. All four cameras may then be connected to the expansion board to test the switching function, switching latency, and stability. The camera performance at different resolutions and frame rates could also be assessed. These tests would help verify the system's feasibility and stability. Based on the system's ability to stably transmit image data, a more in-depth performance evaluation could be conducted. This includes measuring the system's bit error rate, signal-to-noise ratio, maximum supported frame rate, and link reliability under different operating conditions. Simultaneously, the performance differences between the expansion board solution and the direct connection solution, as well as existing commercial solutions on the market, could be compared to quantify the impact of multi-stage switching structures, area, and layout on high-speed signal transmission.

6.4 Usage of Large Language Models

This thesis makes use of Large Language Models (LLMs). Their main uses include eliminating errors in code, correcting technical terminology and language expressions, and explaining difficult-to-understand knowledge and concepts. The research content, design, and experimental results in this thesis were all completed by the author.

7

Conclusion

This project designed and implemented a multi-camera expansion system based on Raspberry Pi, enabling multiple IMX219 image sensors to share a single CSI interface through an I^2C multiplexer and a MIPI switch network.

The project completed hardware design, PCB fabrication, driver configuration, device tree development, and software control framework implementation. Experimental results show that I^2C communication, GPIO control, device tree configuration, driver registration, and media pipeline creation all function correctly. The IMX219 image sensors can be correctly identified and initialized by the system, and the control architecture can be successfully integrated into the Raspberry Pi camera subsystem. Although a stable video stream transmission was ultimately not established, through a systematic testing and fault localization process, the fault was narrowed down to the MIPI CSI-2 high-speed data transmission link. The results indicate that the system control path design is feasible, and the remaining issues mainly focus on the implementation of the high-speed physical layer.

Overall, this project completed the design, implementation, and verification of the multi-camera expansion architecture, laying the foundation for subsequent high-speed link optimization, multi-camera switching verification, and system performance evaluation.

Bibliography

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, May 2015.

[2] J. Xiao, R. Zhang, Y. Zhang, and M. Feroskhan, "Vision-based learning for drones: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 9, pp. 15 601–15 621, Sep. 2025, doi: 10.1109/TNNLS.2025.3564184.

[3] J. Carroll, "MIPI CSI-2 interface provides flexibility for next-generation embedded vision systems," Vision Systems Design, May. 7, 2020, [Online]. Available: https://www.vision-systems.com/boards-software/article/14173598/mipi-csi-2-interfaces-for-embedded-vision-systems.

[4] Raspberry Pi Ltd, "Raspberry Pi Compute Module 4 Datasheet," 2021, Accessed: Jun. 17, 2026. [Online]. Available: https://datasheets.raspberrypi.com/cm4/cm4-datasheet.pdf.

[5] Embedded.com, "FPGA accelerator for embedded vision MIPI cameras," *Embedded.com*, Sep. 30, 2021, [Online]. Available: https://www.embedded.com/fpga-accelerator-for-embedded-vision-mipi-cameras/.

[6] D. Bhowmik and K. Appiah, "Embedded Vision Systems: A Review of the Literature," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, ser. Lecture Notes in Computer Science, vol. 10824. Cham, Switzerland: Springer, 2018, pp. 204–216, doi: 10.1007/978-3-319-78890-6_17.

[7] Allied Vision Technologies GmbH, *Alvium GM2-030 VSWIR Datasheet*, 2024, version 2.0.0, Sep. 9, 2024.

[8] Arducam, "Arducam Multi-Camera Adapter Module for Raspberry Pi," *Arducam*, 2023, Accessed: Jun. 17, 2026. [Online]. Available: https://docs.arducam.com/Raspberry-Pi-Camera/Multi-Camera-CamArray/Quick-Start-Guide-for-Multi-Adapter-Board/.

[9] MIPI Alliance, "MIPI D-PHY Specification," Standard specification, 2022.

[10] *MIPI CSI-2 and D-PHY Overview*, MIPI Alliance, 2022, technical overview document.

[11] MIPI Alliance, "MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2)," Standard specification, 2012.

[12] E. Bogatin, *Signal and Power Integrity—Simplified*, 3rd ed. Boston, MA, USA: Pearson, 2018.

[13] H. W. Johnson and M. Graham, *High-Speed Digital Design: A Handbook of Black Magic*. Englewood Cliffs, NJ, USA: Prentice Hall, 1993.

[14] X. Jing and R. Zhou, "Crosstalk Analysis and Simulation in High-Speed PCB Design," in *Proc. 2007 8th Int. Conf. Electronic Measurement and Instruments (ICEMI)*, Xi'an, China, Aug. 2007, pp. 437–440, doi: 10.1109/ICEMI.2007.4350711.

[15] H. D. Dsilva *et al.*, "Comparing the Different Metrics of Intra-Pair Skew in Tracking Channel Performance," in *Proc. DesignCon 2024*, Santa Clara, CA, USA, Jan. 2024.

[16] Linux Kernel Documentation, "Video for Linux API," 2025, Accessed: Jun. 17, 2026. [Online]. Available: https://docs.kernel.org/userspace-api/media/v4l/v4l2.html.

[17] ——, "Media subsystem kernel internal API," 2025, Accessed: Jun. 17, 2026. [Online]. Available: https://docs.kernel.org/driver-api/media/index.html.

[18] J. Corbet, A. Rubini, and G. Kroah-Hartman, *Linux Device Drivers*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2005.

[19] Devicetree Specification Working Group, "Devicetree Specification Version 0.4," Standard specification, 2023, Accessed: Jun. 17, 2026. [Online]. Available: https://www.devicetree.org/specifications/.

[20] Z. Wan, Y. Zhang, A. Raychowdhury, B. Yu, Y. Zhang, and S. Liu, "An Energy-Efficient Quad-Camera Visual System for Autonomous Machines on FPGA Platform," in *Proc. 2021 IEEE 3rd Int. Conf. Artificial Intelligence Circuits and Systems (AICAS)*, Washington, DC, USA, Jun. 2021, pp. 1–4, doi: 10.1109/AICAS51828.2021.9458486.

[21] I. Novak, "Reducing Simultaneous Switching Noise and EMI on Ground/Power Planes by Dissipative Edge Termination," *IEEE Transactions on Advanced Packaging*, vol. 22, no. 3, pp. 274–283, Aug. 1999, doi: 10.1109/6040.784475.

[22] WPCalc, "Microstrip Differential Impedance Calculator," 2025, Accessed: Jun. 17, 2026. [Online]. Available: https://wpcalc.com/en/engineering/microstrip-differential-impedance/.

[23] Saturn PCB Design, Inc., "Saturn PCB Design Toolkit, Version 8.44," Software, Accessed: Mar. 2026. [Online]. Available: https://saturnpcb.com/saturn-pcb-toolkit/.

[24] S. H. Hall and H. L. Heck, *Advanced Signal Integrity for High-Speed Digital Designs*. Wiley, 2009.

[25] IBE Electronics, "PCB Vias," *PCBAAA*, n.d., Accessed: Jun. 17, 2026. [Online]. Available: https://www.pcbaaa.com/pcb-vias/.

[26] R. J. Allred and C. M. Furse, "Reflection Budgeting Methodology for High-Speed Serial Link Signal Integrity Design," *Progress In Electromagnetics Research B*, vol. 91, pp. 59–77, 2021, doi: 10.2528/PIERB20102108.

A

Appendix 1

The single channel device tree overlay used in the kernel-mode validation is provided below.

```
/dts-v1/;
/plugin/;

/ {
    compatible = "brcm,bcm2711";

    fragment@0 {
        target-path = "/";
        __overlay__ {
            imx219_clk: imx219_clk {
                compatible = "fixed-clock";
                #clock-cells = <0>;
                clock-frequency = <24000000>;
            };
        };
    };

    fragment@1 {
        target = <&gpio>;
        __overlay__ {
            cam_enable_hog {
                gpio-hog;
                gpios = <22 0>;
                output-high;
                line-name = "cam-enable";
            };

            cam_sel0_hog {
                gpio-hog;
                gpios = <17 0>;
                output-low;
                line-name = "cam-sel0";
            };
```

```
            cam_sel1_hog {
                gpio-hog;
                gpios = <27 0>;
                output-low;
                line-name = "cam-sel1";
            };
        };
    };

    fragment@2 {
        target-path = "/soc/i2c0mux/i2c@1";
        __overlay__ {
            #address-cells = <1>;
            #size-cells = <0>;
            status = "okay";

            i2c-mux@70 {
                compatible = "nxp,pca9546";
                reg = <0x70>;
                #address-cells = <1>;
                #size-cells = <0>;

                i2c@0 {
                    reg = <0>;
                    #address-cells = <1>;
                    #size-cells = <0>;

                    imx219_0: camera@10 {
                        compatible = "sony,imx219";
                        reg = <0x10>;
                        status = "okay";

                        clocks = <&imx219_clk>;
                        clock-names = "xclk";

                        port {
                            imx219_0_ep: endpoint {
                                remote-endpoint = <&csi_ep>;
                                clock-lanes = <0>;
                                data-lanes = <1 2>;
                                clock-noncontinuous;
                                link-frequencies = /bits/ 64 <456000000>;
                            };
                        };
                    };
                };
```

```
            };
        };
    };

    fragment@3 {
        target-path = "/soc/csi@7e800000";
        __overlay__ {
            status = "okay";

            port {
                csi_ep: endpoint {
                    remote-endpoint = <&imx219_0_ep>;
                    clock-lanes = <0>;
                    data-lanes = <1 2>;
                };
            };
        };
    };
};
```