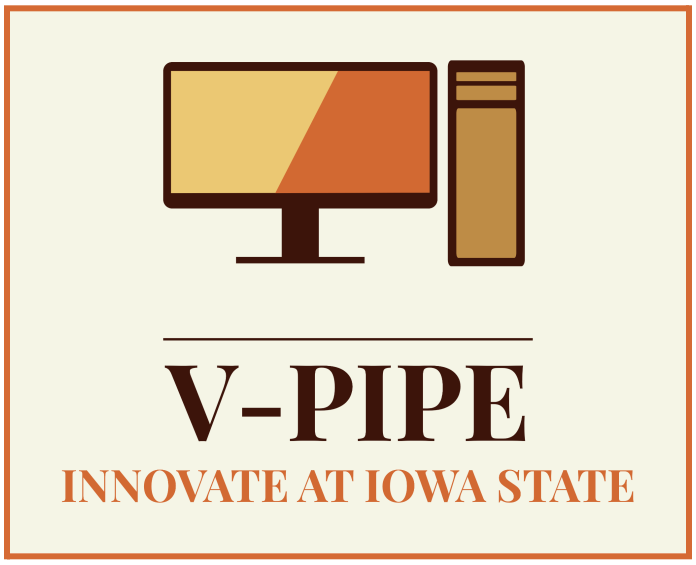
Video Pipeline for Machine Vision



Final Report

Team Number: 6
Adviser: Dr. Phillip Jones

Team Members:
Liam Janda • Taylor Johnson • Ritwesh Kumar • Deniz Tazegul

Team Email: sddec24-06@iastate.edu

Team Website: https://sddec24-06.sd.ece.iastate.edu/

Table of Contents

1. Introduction

1.1 Problem Statement

There are numerous applications for a computer vision pipeline that makes use of Machine Learning algorithms: from medical imaging to self-driving cars to surveillance drones. This project aims to create a proof-of-concept video pipeline prototype using common off-the-shelf materials and software. The video pipeline will take video data from an image sensor, route it through an FPGA board, and output the video stream to a DisplayPort monitor.

The project is part of a history of current and past senior design teams across several universities, with the inspiration being to help disabled or injured individuals increase their quality of life by tracking their eye movements in real-time. These individuals can have inhibited fine motor skills that make acute navigation challenging, and individuals confined to a wheelchair often require a helper to perform daily living tasks. While measuring eye movements in real-time is critical for the broader project, it is not a primary concern for this team. However, the requirements and design are set up to facilitate those capabilities in the future. Thus, having low latency is also not a primary consideration. Furthermore, while it is expected that the final product will need to operate under a variety of lighting conditions, for the scope of this project, it is assumed that the face of the user is well-lit by a light source.

Capturing eye movements in real-time for these individuals will be vital in assisting them with their communication and navigational needs. By combining an eye-tracking vision system with other computerized parts, these individuals can interact with their environment in a way that reduces the risk of injury, while gaining a higher sense of independence. This is just one application for this product which can be used in a broader range. For example, this product could be used as a ready set of components for a programmer to explore an interest in machine learning.

1.2 Intended Users and Uses

The primary user for this project is a disabled individual who is confined to a motorized wheelchair and has limited navigation. The video pipeline in combination with machine-learning algorithms will be used to capture a user's eye movements to aid in navigation.

1.3 Context and Implementation

Our project design is a video pipeline system designed for computer vision with the intent of helping users with illnesses or disabilities that limit their independence.

2. Revised Design

2.1 Requirements

During the development phase, our requirements evolved over time. There were certain hardware limitations (such as frame rate) that led to changes in our functional and non-functional requirements. Below is a list of our updated requirements.

Functional

- Output video data through DisplayPort monitor
- Linux image
- Ultra96-v2 FPGA board and daughter card
- Sony IMX219QH5-C image sensor

Non-Functional

- Minimum of 640x480p resolution at 15fps

Assumptions

- The user's face is well-lit by a light source

2.2 Engineering Standards

Engineering Standards	Relevance
IEEE Std 2977 2021	Video stream from the image sensor will be sent to a MIPI controller and must adhere to the MIPI-APHY Standard for conversion before being sent to the Ultra96-v2 FPGA board.
I2C Protocol	I2C is the communication system that will be used to transmit incoming data from the camera's image sensor to the Ultra96-v2 FPGA.
AMBA AXI Protocol	The AMBA AXI bus will be used to transport data within the Ulta96-v2 FPGA board. AMBA consists of both the AXI4-Lite bus used to transmit data from the program memory to configure IP registers and the AXI4-Stream bus used to transmit video data between individual IPs within the FPGA including the MIPI controller, VDMA, TPG, and DisplayPort.

Table 1: Used development standards and practices

2.3 Evolution of Design

The following changes were made to our initial design:

- The goals for resolution and framerate (which are limited by the hardware) were changed from 1920x1080p at 30 fps to 640x480p at 15 fps.
- We changed the number of CSI-2 data lanes from 4 lanes to 2 lanes. This is limited by the IMX219 image sensor.
- The requirement for considering lighting conditions was removed and replaced with an assumption that the user's face is well-lit by a light source.
- The requirement that the pipeline should operate in real-time was removed. While considering latency is expected to be a requirement for the final project, it is not a concern for this team. Our focus is on building a video pipeline prototype.
- A clarification was made to the design by breaking it up into two video pipelines that help to build the overall design modularly. The first pipeline implementation is for testing the TPG to the DisplayPort monitor (Figure 1), and the second is for sending live video data from the IMX219 to the DisplayPort monitor (Figure 2).

3. Implementation Details

3.1 Detailed Design

TPG to VDMA Video Pipeline

The first video pipeline, shown below in Figure 1, from the TPG to the VDMA was used for testing the backend modules of the full video pipeline, namely the VDMA and the DisplayPort monitor. The TPG sends a static image to the VDMA. Before sending video data to the monitor, the VDMA writes that data to frame buffers in DDR memory. Finally, the data is read from DDR memory and displayed on the monitor.

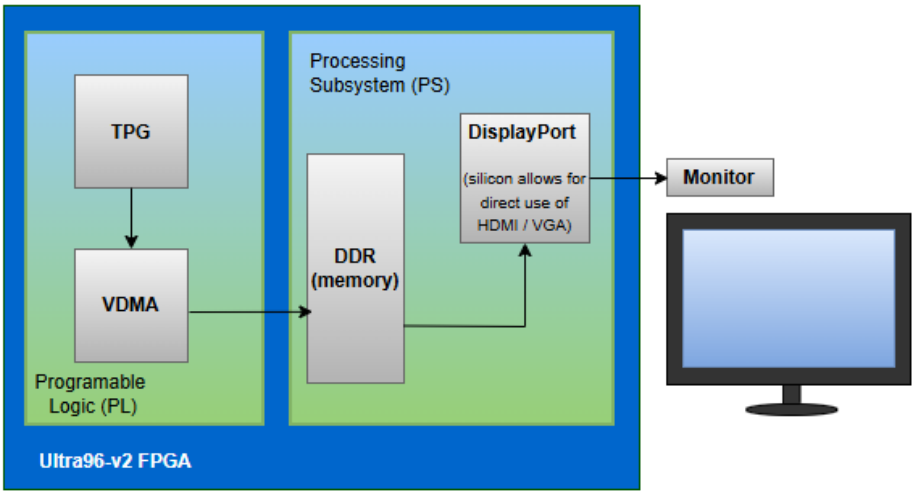


Figure 1: TPG to VDMA Block Diagram

IMX219 to DisplayPort Full Video Pipeline

Once we knew how to properly configure the VDMA and DisplayPort monitor, we worked on designing the full video pipeline, shown below in Figure 2, which includes the IMX219 image sensor and MIPI controller instead of the TPG. Raw video data is captured by the IMX219 and sent to the VDMA via the MIPI controller. The MIPI controller passes the data through the MIPI D-PHY and CSI-2 blocks. The D-PHY acts as the physical communication layer for interfacing between the IMX219 and the Ultra96-v2 FPGA, and the CSI-2 block creates a high-speed camera serial interface link between the IMX219 and VDMA. Once the MIPI controller sends the video data to the VDMA, the VDMA then writes the video data into DDR memory. Finally, the video data is read from DDR memory and output to the DisplayPort monitor.

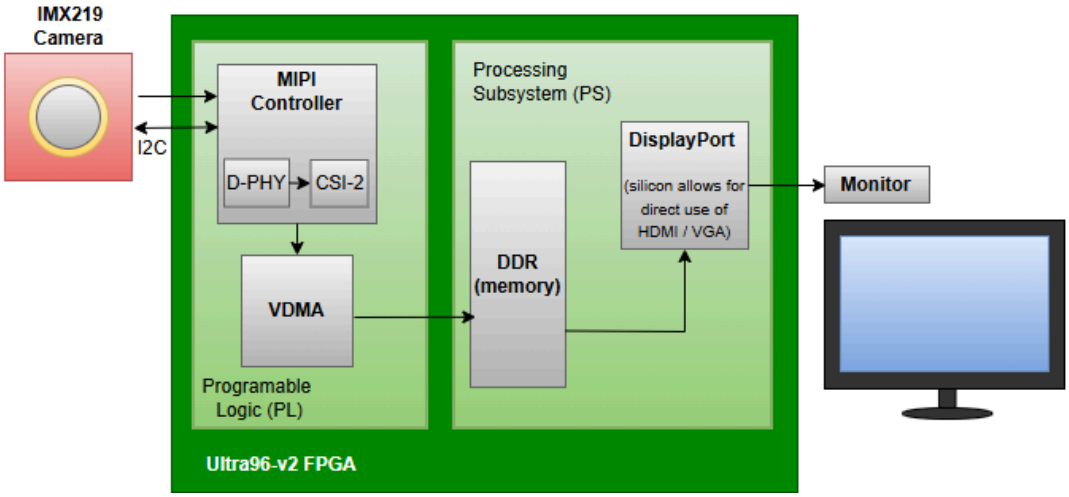


Figure 2: IMX219 Image Sensor to DisplayPort Block Diagram

3.2 Description of Functionality

The product will be utilized by the following users and in the following real-world contexts:

Disabled users can utilize the product to track their eye movements in real-time for seizure detection and communication and navigation assistance.

Programmers can utilize this product to implement Machine Learning algorithms for different tasks.

- Improving the navigational abilities of individuals restricted to motorized wheelchairs.
- Facial recognition.
- Object detection and avoidance for self-driving vehicles.
- Surveillance technology for use in traffic control or intelligence agency applications.

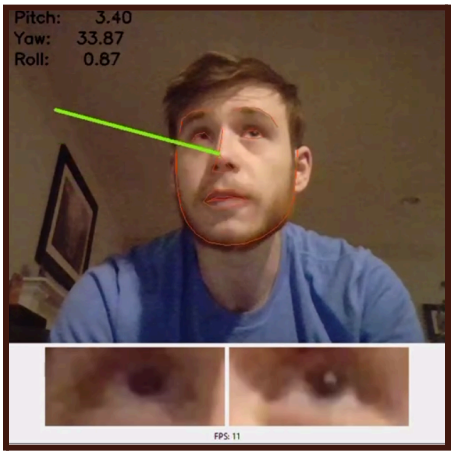


Figure 3: Eye-Tracking Algorithm by a previous Senior Design Team

3.3 Notes on Implementation

All of the project milestones from 491 have been accomplished. The video pipeline takes in video data from the IMX219 image sensor, routes it through the Ultra96-v2 FPGA, and the data is output to a DisplayPort monitor at a minimum frame rate of 15 frames per second and resolution of 640x480p. In addition, we have created a custom configuration code for setting the coordinates of a rectangular bounding box with a desired framerate, and the width and height of video data from the IMX219, if possible. The bounding box will be useful for future design teams working on integrating the video pipeline with the eye-tracking software. The bounding box will focus on capturing the eyes as shown in the bottom panel of Figure 3 above. The code defaults to the maximum calculated framerate if the desired framerate exceeds the maximum framerate for a given resolution. The main challenge we face is translating our code from Python to C to increase the speed of our code and gain a more accurate measurement reading of the framerate.

4. Testing

4.1 Process

Unit Testing

Individual components were tested as per the project description and requirements design. A unit in this project is one component, for example, the IMX219 image sensor would be one component/unit. Testing the configuration for each component primarily consists of writing to and reading from important register values. This was completed using terminal commands and shell scripts using TeraTerm. For the IMX219, for example, we successfully read from read-only registers with known values and verified that the values matched with the datasheet values for the IMX219.

Integration Testing

Integration testing was completed between connecting components of the system, such as the image sensor connecting to the MIPI controller. This testing was completed by reading appropriate register values using TeraTerm terminal commands. For the IMX219 and MIPI controller interface, for example, we monitored the MIPI D-PHY and CSI-2 registers and verified that the datatype of RAW8 we sent from the IMX219 was observed, the number of packets continuously increased up to 0xFFFF then reset back to 0x0000, and the frame received bit was observed.

System Testing

System testing was done after all unit tests and integration tests were completed and passed. System testing entailed reading register values to ensure video data was being transmitted effectively throughout the system. This was done by successfully observing the frame received bit at the MIPI controller from the IMX219 and the frame count interrupt being asserted by the VDMA as expected.

In addition, we did visual testing by comparing the video data from the IMX219 on the DisplayPort monitor with a known image that the IMX219 was pointed at to verify the video data outputted as expected. For example, Figure 4 below shows the rainbow image that was used to compare specific colors and image orientation, showing the video pipeline output on the right when the IMX219 is pointed at the original rainbow image on the left.

Figure 4: 640x480p at 20 fps: test image (left), frame from IMX219 (right)

Figure 4 shows that the orientation of the video data and colors match the expected orientation of the letters and colors from left to right and top to bottom, along with the expected colors of the rainbow from red to violet.

Regression Testing

Regression testing was handled by the unit, integration, and system testing. This was done, for instance, by creating a custom configuration code for the IMX219 to send a desired rectangular bounding box, framerate, width, and height while maintaining integral configuration constant across different resolutions and framerates. The code for configuring the DisplayPort monitor, VDMA, MIPI D-PHY, and MIPI CSI-2, along with most of the IMX219 registers was kept the same regardless of custom configuration. This was necessary to ensure that the video pipeline functions regardless of changes to the IMX219 registers relevant to configuring a desired resolution and framerate.

Acceptance Testing

Acceptance testing was handled by regular discussions and progress reports with the client and advisor. The progress of the unit, integration, system, and regression testing was shared with the client and advisor over time for feedback and verification of the completion of the project's functional and non-functional requirements.

4.1 Results

In addition to the results shown in Figure 4 for 640x480p video data at 20 frames per second from the IMX219, the following patterns from the TPG are given below in Figure 5 at 640x480p. These tests were useful for verifying and debugging the configuration of the VDMA and DisplayPort monitor using the video pipeline from Figure 1.

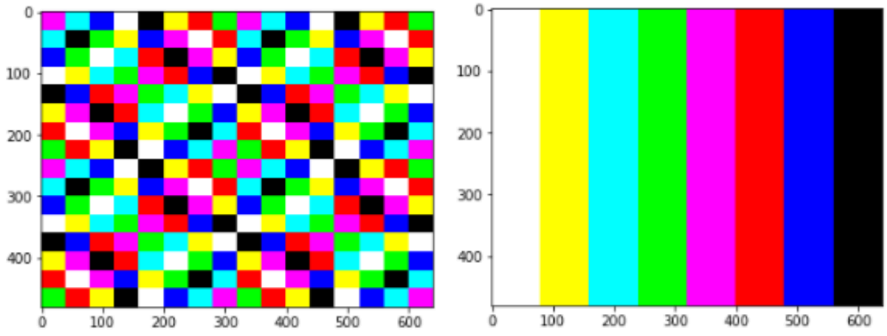


Figure 5: Test Patterns from TPG IP block at 640x480p

The majority of the work in debugging the video pipeline was done on the Vivado hardware overlays, which are used to configure the hardware. Common issues encountered during testing included incorrect video synchronization, timing, and a noticeable shift in colors. Figure 6 below gives an example of a noticeable shift in colors on the left for the TPG pattern on the right.

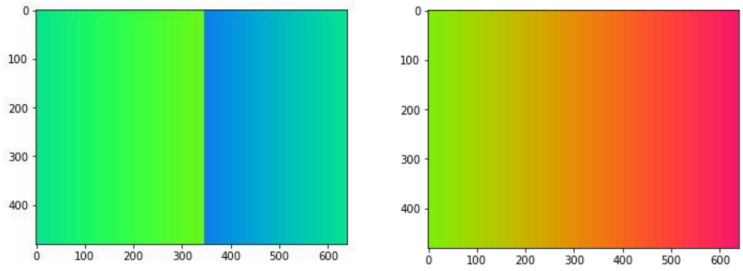


Figure 6: Incorrect noticeable shift TPG (left) and correct TPG 640x480p patterns

Figure 7 below shows an incorrect TPG video synchronization and timing output on the left and the correct output on the right at 640x480p.

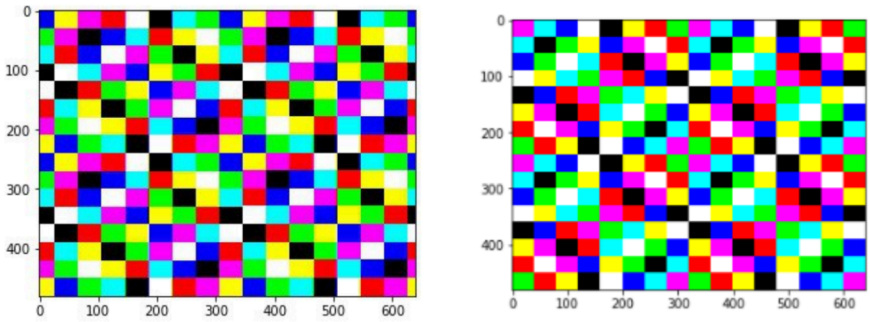


Figure 7: Incorrect video synchronization and timing TPG (left) and correct TPG 640x480p patterns

After identifying the above errors from Figures 6 and 7 and similar video synchronization, timing, and noticeable color shifts, we made corrections to the hardware design in Vivado for both the TPG to VDMA block diagram in Figure 1 and the full video pipeline in Figure 2. These changes resulted in the correct expected outputs shown in Figures 4 and 5.

5. Broader Context

Below is a table with relevant considerations for the broader context in which our project exists. We examine the impact our design will have on the communities we're designing for and what societal needs our project addresses.

Area	Description	Examples
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., the solution is implemented in their communities)	Product design is intended to increase safety for users with illnesses or disabilities by assisting in their navigation and communication needs.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Not applicable to the project design as it would not noticeably change our design if we did.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	The product will use commercially available components with a consideration of limiting the amount of power consumption.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	The product needs to remain affordable for target users which is why the design uses commonly available components contrary to custom components which require a more expensive design process.

Table 1: Broader Context Analysis

Our project design is a video pipeline system designed for computer vision. The most relevant broader context issues that relate to our project are public health, safety, and well-being. Our product aims to aid primarily in navigation and communication for individuals with illness or disabilities that limit their independence therefore, the design centers around improving their quality of life. There are also considerations being made to limit infractions on the personal data and privacy of users. Our team is not taking economic factors into huge consideration, our primary focus is functionality and performance, however, commonly available component choice and power consumption are areas that are being taken into account and have an effect on the overall cost of the system. This contrasts some of the products already on the market that use custom components that require a more involved and expensive design process. Our team is not considering the global, cultural, and social aspects of the broader context and is not convinced the project design would change noticeably if we did.

6. Conclusions

6.1 Progress Review

Our project successfully completed the creation of a prototype video pipeline that has a base configuration of 640x480p at 20fps. The code for this was done using Python in a Jupyter Notebook. The current code also contains a custom IMX219 image sensor configuration that allows future design teams to use the video pipeline to focus on a specific region of a frame, such as the eyes for eye-tracking.

6.2 Design Value

Overall, the design broke significant ground in the broader context of this project. The pipeline is successfully able to send video data through for a custom size and frame rate, which can be updated as the user needs. The testing framework implemented through Jupyter Notebook and PYNQ commands allows future teams to easily update and test the design as needed. Throughout the course of this project, the team has created a large amount of documentation on the individual components as well as code and testing processes that will greatly help for the further development of this project in future steps. For this purpose, we met with the next team and shared our code with them to help them understand and potentially build on our work and implement machine learning algorithms such as for eye-tracking.

6.3 Future Steps

This project is part of a larger project to create a navigational tool for people confined to motorized wheelchairs. There are a variety of future steps needed before that vision can be realized. The first of those steps will be to configure a machine-learning eye-tracking algorithm

to run on the designed pipeline. The eye-tracking algorithm is being developed by a different team, and the pipeline and algorithm will need to be updated to work together. A type of application will be developed to control the pipeline and customize the video data depending on variables such as user orientation, eye location, processing speed, etc. Before this process, the initialization and configuration scripts for the pipeline will need to be written in a faster, more efficient programming language, like C. This is because the application will be able to request a new configuration multiple times a second, and the current PYNQ implementation will not be fast enough to handle that. Another team will also need to construct a container to house this pipeline, as well as a mount that will attach this to a motorized wheelchair.

7. Appendices

7.1 Appendix 1 - Operation Manual

To send video data in the correct orientation from top to bottom and left to right, the data must first be flipped vertically and horizontally. After this, the top left and right and bottom left and right corners of the IMX219 are labeled in Figure 8 below, where TL stands for top left, TR stands for top right, BL stands for bottom left, and BR stands for bottom right.

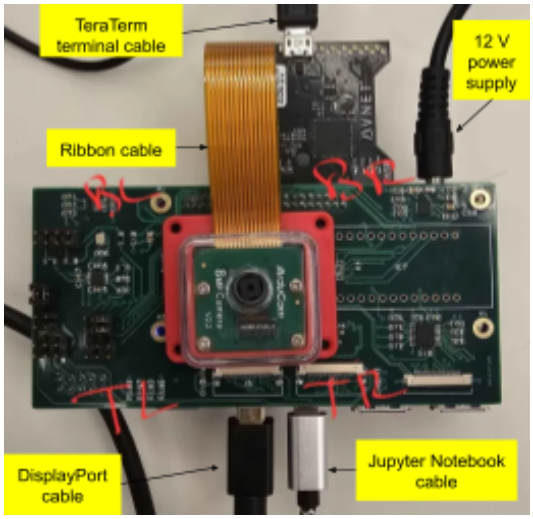


Figure 8: Orientation of the IMX219 image sensor after vertical and horizontal flipping

There are four cables used to set up the Ultra96-v2 FPGA and IMX219 image sensor, all shown in Figure 8. These include the 12 V power supply in the top right of the image, a micro-USB cable to connect to the TeraTerm terminal to the left of the power supply cable, a mini-DisplayPort cable on the bottom of the image to the left, and a micro-USB cable to connect to the Jupyter Notebook environment to the right of the mini-DisplayPort cable. In addition to these cables, the IMX219 must be connected to the MIPI connectors via its gold-colored ribbon cable. It is important to note that the IMX219 ribbon cable must be connected to the gold side facing up.

7.2 Appendix 2 - Design Iterations

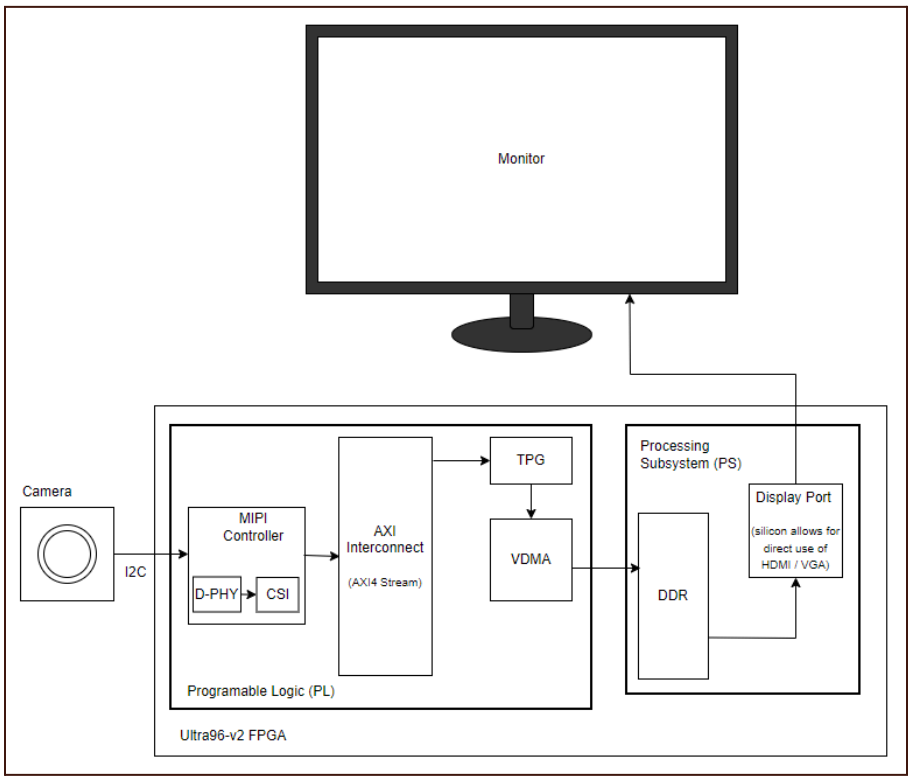


Figure 9: Original Video Pipeline Design

The figure above shows the original design and datapath for the video data. This design includes the TPG IP block as that block can be configured to directly pass video data. The idea was to be able to use the TPG to debug the VDMA and DisplayPort. After confirming that the TPG to DisplayPort was working, the TPG block could be kept and configured to pass data directly while focusing on the proper configuration of the camera and MIPI blocks. This approach was so that the video pipeline could be built modularly. However, based on the feedback we received for our faculty presentation, the inclusion of the TPG in the full video pipeline is unnecessary as the video data could be passed directly from the MIPI block to the VDMA for a more efficient data handoff. Therefore, the design was broken into two separate pipelines (see Figures 1 and 2).

7.3 Appendix 3 - Other Considerations

Throughout the design process, we have learned how to break down a relatively complicated engineering problem and handle it in smaller, more manageable chunks. We have learned how to think of potential issues in our design and come up with high-probability theories to test and either prove or disprove our initial hypotheses to overcome bottlenecks in our design. We used this method of thinking extensively, for example, during our testing process for sending a TPG

image to the DisplayPort monitor. During this challenge, we learned to record any unexpected results we were observing during our debug process, such as those seen in Figures 6 and 7, and then come up with theories of ways to change either our software code or the hardware design in Vivado to accomplish our project objectives and send video data to the monitor.

7.4 Appendix 4 - Definitions

IMX219 image sensor: camera

MIPI: mobile industry processor interface

CSI-2: high-speed camera serial interface

D-PHY: physical communication layer

VDMA: video direct memory access

DDR: double data rate (memory)

FPGA: field programmable gate array

PYNQ: python productivity for Zynq (python embedded systems developers)

ML: machine learning

Vivado: custom hardware overlays