



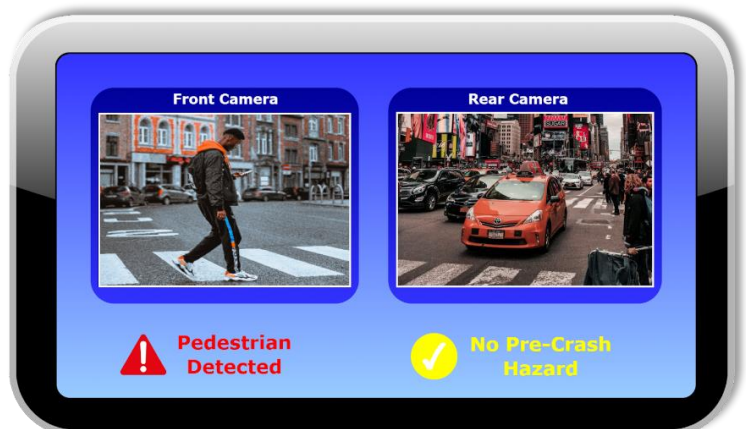
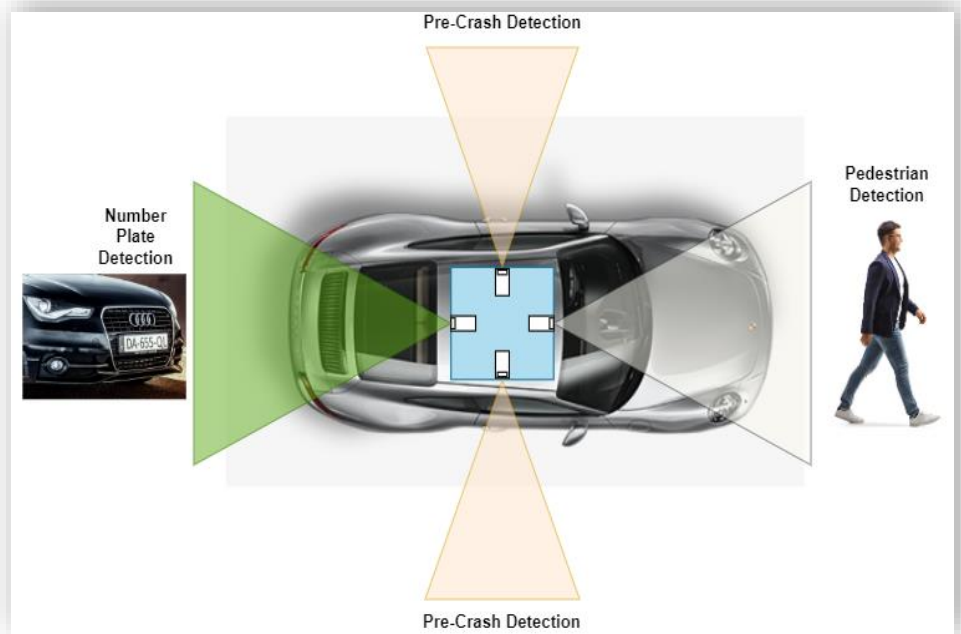
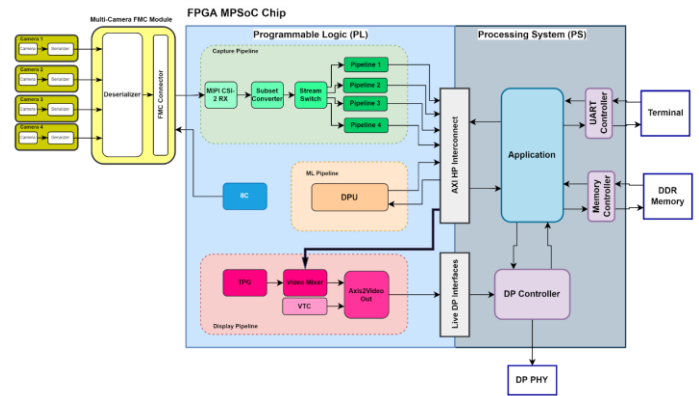
logictronix

ADAS Reference Design

V1.0

LRFD031

Nikil Thapa



Objectives

- To give brief ADAS Overview
- To discuss ADAS FPGA Hardware Design Flow
- To provide ADAS Reference Solution

Overview

By following this document you will have the information about a brief ML-based reference design for the ADAS solutions. This design solution has been made taking the **Zynq UltraScale+ MPSoC** device. The necessary hardware has been listed down below, which was taken from the AVNET.

- Quad AR0231AT Camera FMC Bundle
- UltraZed-EV Carrier Card
- UltraZed-EG SOM

ADAS, as we all know, stands for Advanced Driving Assistance System. This system is highly popular in the Automobile Industry. The system has major functionality in that it assists drivers in order to avoid accidents and provides road safety.

The general block diagrammatic view of the ADAS system is shown below.

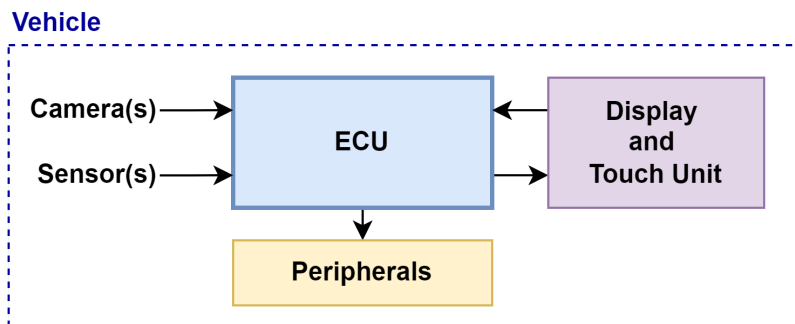


Figure 1. Conventional ADAS Hardware

The ADAS system can have multiple sets of cameras and sensors interfaced to grab the surrounding information of the vehicle and that information is supplied to ECU(Electronic Control Unit). The ECU acts as the main processing unit here, which is further interfaced with Peripherals and Display & Touch Unit to control peripherals or to show the status through Display Unit or to take user Touch input to do user-oriented control.

In nutshell, the ADAS is the group of electronics technologies creating a human-machine interface in the field of the automobile industry for road safety.

The conventional ADAS system was only the first step. Now, it can be taken to the next level by introducing **Machine Learning (ML)** capability. It could become the ultimate gateway towards the futuristic **Artificially Intelligent (AI) Driving System**, where the vehicle would drive itself without the requirements of the driver.

This document will focus on **ML-based ADAS Reference Design** by processing **Camera-Feeds** on the **Zynq UltraScale+ MPSoC** device.

The tools used for this design are

- Vivado 2020.2 or Later
- Petalinux 2020.2 or Later
- Vitis AI 1.3.1

The following picture shows the block diagrammatic view of the design.

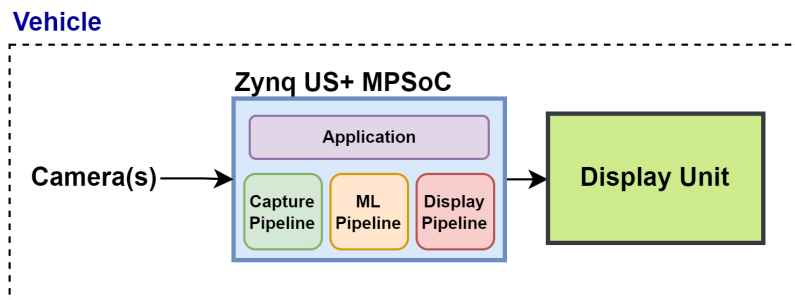


Figure 2. ML-Based FPGA ADAS Hardware

This time the design is only considering the **Camera(s)** input and the information will be shown in the **Display Unit**. In brief, the design can have a set of processing pipelines, such as a capture pipeline, ML pipeline, and Display Pipeline.

The camera feeds can be captured and done the post-video processing by the Capture Pipeline.

The ML pipeline can implement the AI to detect and recognize, for example, vehicles, road markings, traffic markings, vehicle number plates, obstacles, etc...

The display pipeline can utilize the ML pipeline information and do the video and audio composition, such as emphasizing the recognized objects and showing other scenery details in the display unit.

To host and extend the functionality and operability of such a system, the application can be written.

Hardware Design

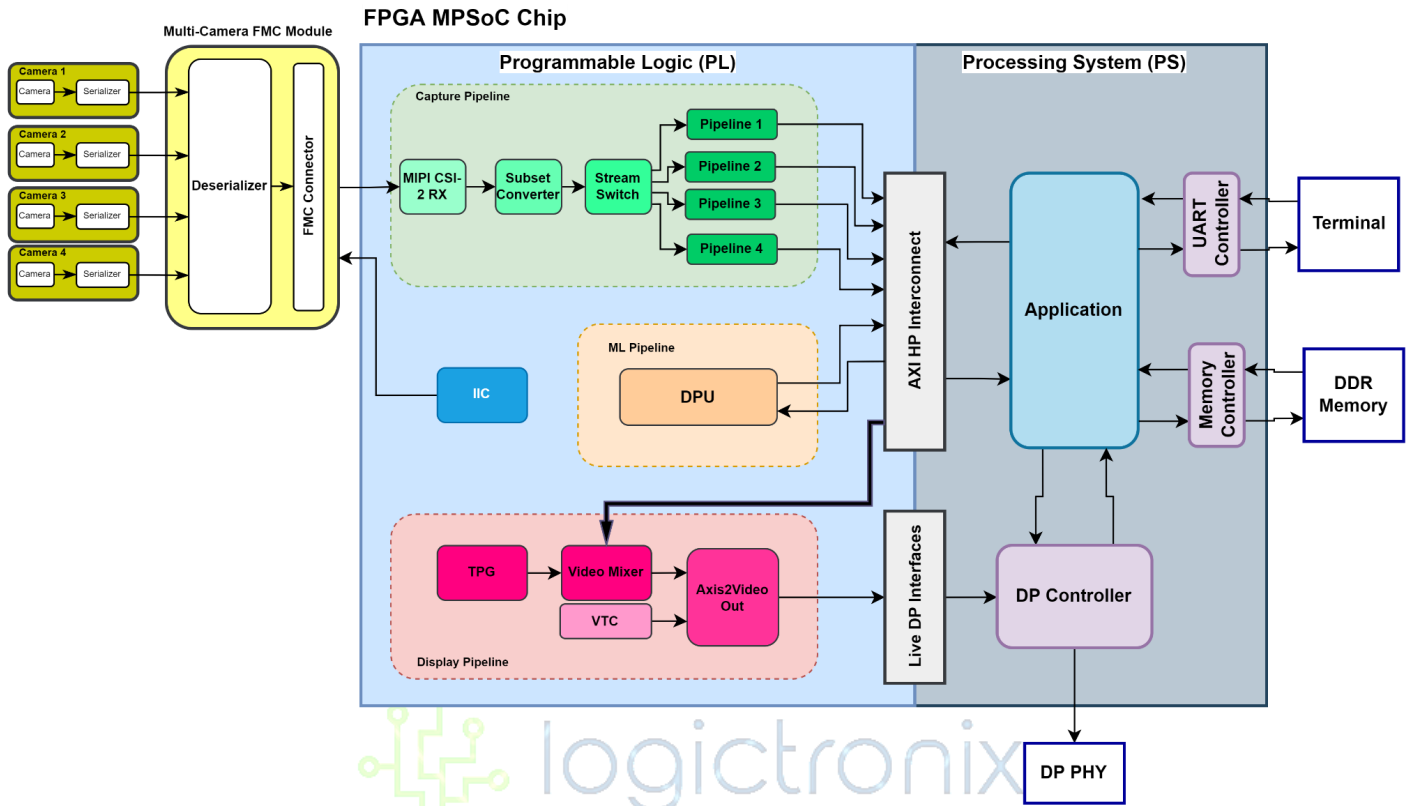


Figure 3. Functional Diagram of ML-Based FPGA ADAS Hardware

The above picture depicts the block diagrammatic view of ADAS FPGA hardware design.

Avent Quad Cam FMC Module has a quad-cam deserializer IC that is capable of receiving up to 4 camera GMSL links. These links are deserialized and all four camera data are obtained by a common MIPI CSI-2 Interface. The data is transferred to FPGA by an FMC connector.

The PL side will have three pipelines design. These are expounded in the following sections:

- **Capture Pipeline**

This pipeline is responsible for receiving the camera data, pre-processing the data, and writing data into memory. The pipeline consists of a series of blocks, namely, MIPI CSI-2 RX, Subset Converter, Stream Switch, and pre-processing pipeline block.

The camera data are obtained into FPGA by MIPI CSI-2 data format. These data are processed by the MIPI CSI-2 RX block, which then generates AXI stream data. The stream data width conversion is performed by Subset Converter Block. This block generates proper stream width for upcoming processing blocks. The Stream Switch block is used to demultiplex a single stream into four individual streams. This is so because the quad-cam deserializer is multiplexing four-camera data to a single MIPI CSI-2 interface. That's why it is needed to be done demultiplexing in FPGA so as to access and process individual camera data further. Now each stream of data goes to four pre-processing pipeline blocks respectively.

Each pre-processing pipeline channel consists of the following sub-blocks:

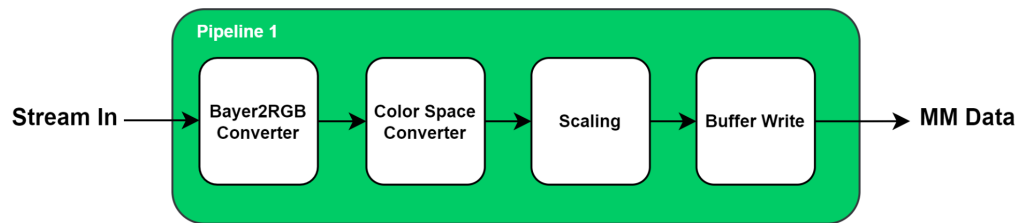


Figure 4. Pre-processing Pipeline

The pipeline consists of a Bayer2RGB converter, Color Space Converter(CSC), scaling and Buffer Write blocks.

The Bayer2RGB Converter converts the RAW format of sensor data into RGB format.

The CSC block can convert RGB format into other formats, such as YUV.

The Scaling block can do the scaling incoming video resolution into different output resolutions. The scaled resolutions will be utilized in the ML pipeline as well as the display pipeline.

Buffer write block is used to write the data into four separate locations of DDR memory, with a specific color format and resolution value. These memory locations will be accessed by the ML pipeline to retrieve the camera data and do the ML task.

• ML Pipeline

This design is following **Vivado Flow**, which means the **DPU** IP block will already be present in the Vivado Hardware Block Design.

This pipeline generally comes in between the capture pipeline and the display pipeline. This all depends on the type of design. This pipeline uses DPU (Deep Processing Unit) core upon which the Convolutional Neural Network (CNN) will be run. This core is the programmable engine for deep neural networks. It is pre-implemented on the hardware with no place and route required. It allows the efficient implementation of many deep learning networks. The DPU IP core has Slave AXI Port so that the core can be programmed from the PS side. The core also has Master AXI Ports, which are used to access the memory locations for input images as well as temporary and output data.

In this reference design, the DPU core has the following major configurations

- Numbers of DPU Core: 1
- Architecture of DPU: B4096
- RAM Usage: LOW
- Channel Augmentation: Enabled
- DepthWiseConv: Enabled
- ElementWise Multiply: Disabled
- AveragePool: Enabled
- ReLU Type: ReLU + LeakyReLU + ReLU6

- Number of SFM cores: 0

In this design, the DPU core accesses first input stream frame from the memory. However, DPU can only take 640x480 resolution of frame. So, the resolution is downscaled by the scaling IP block. The DPU core then does the task, such as detection of specific objects in the frame, and that information are utilized by the application, running on APU, to show the markings in the front and rear camera frame. This frame will become the final output, which is again stored in another memory location.

The multitude of detection models can be implemented upon the DPU core. From the ADAS reference design point of view, the design will be able to do the following things:

- Vehicle License Plate Recognition
- Vehicle to vehicle distancing
- Face detection
- Pedestrian Detection
- Traffic Sign Detection
- Animal Detection
- Lane Detection

● Display Pipeline

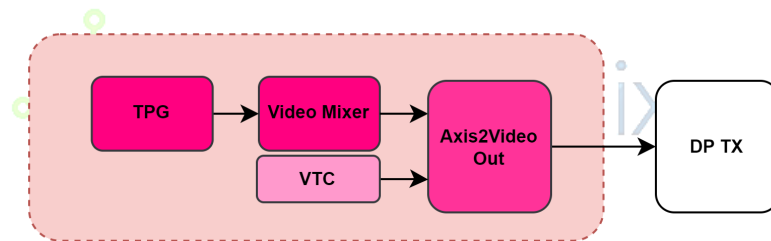


Figure 5. Display Pipeline

This pipeline contributes to showing off the processed camera data on the display monitor. The video mixing block is the major block of this pipeline, which will read the ML processed data from the memory and show them as overlay layers. The resulting video data is converted into parallel video data by Axis2Video out the block. This block takes stream data and video timing as input and generates parallel video data. This data will be fed to Live Display Port(DP) interfaces.

The following picture shows the Vivado Hardware Block Design for the reference design. The block design has Processing System Blocks, Capture Pipeline Blocks, DPU Pipeline(ML Pipeline) Blocks, and Display Pipeline Blocks. The FMC module is programmed by the AXI IIC block.

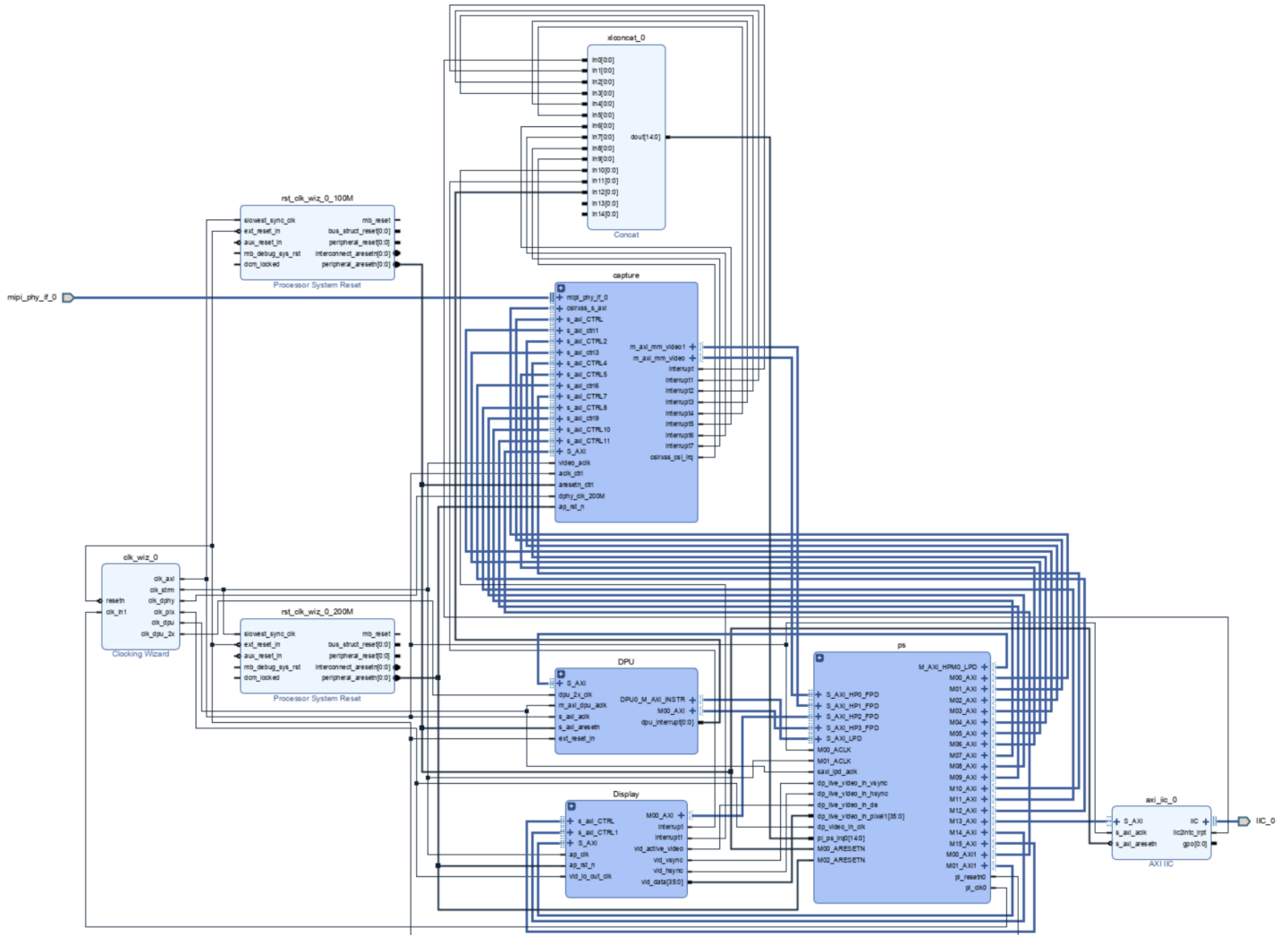


Figure 6. Vivado HW Block Design

Application Design

Before starting application design, the hardware file is generated from vivado and will be used for petalinux application. It will be used following steps to complete the application design.

- Ultrazed BSP, provided by AVNET
- Configure Device Tree
- Build petalinux: img, wic
- Burn SD card img.
- Petalinux User Application: gst pipeline

Running Petalinux Application

The four detections networks will be run upon four camera streams using following GStreamer pipeline.

```
gst-launch-1.0 \
  v4l2src device=/dev/video2 io-mode=4 ! \
  video/x-raw, width=640, height=360, format=BGR, framerate=30/1 ! \
  queue ! vaipersondetect ! queue ! \
  fpsdisplaysink video-sink="kmssink bus-id=b0050000.v_mix plane-id=38 \
  render-rectangle="\<0,0,640,360>" sync=false fullscreen-overlay=true \
  \
  v4l2src device=/dev/video3 io-mode=4 ! \
  video/x-raw, width=640, height=360, format=BGR, framerate=30/1 ! \
  queue ! vailpdetect ! queue ! \
  fpsdisplaysink video-sink="kmssink bus-id=b0050000.v_mix plane-id=39 \
  render-rectangle="\<640,0,640,360>" sync=false fullscreen-overlay=true \
  \
  v4l2src device=/dev/video4 io-mode=4 ! \
  video/x-raw, width=640, height=360, format=BGR, framerate=30/1 ! \
  queue ! vaioobjectdetect ! queue ! \
  fpsdisplaysink video-sink="kmssink bus-id=b0050000.v_mix plane-id=40 \
  render-rectangle="\<0,360,640,360>" sync=false fullscreen-overlay=true \
  \
  v4l2src device=/dev/video5 io-mode=4 ! \
  video/x-raw, width=640, height=360, format=BGR, framerate=30/1 ! \
  queue ! vaioobjectdetect ! queue ! \
  fpsdisplaysink video-sink="kmssink bus-id=b0050000.v_mix plane-id=41 \
  render-rectangle="\<640,360,640,360>" sync=false fullscreen-overlay=true \
  \
  -v
```

Camera Hardware Setup

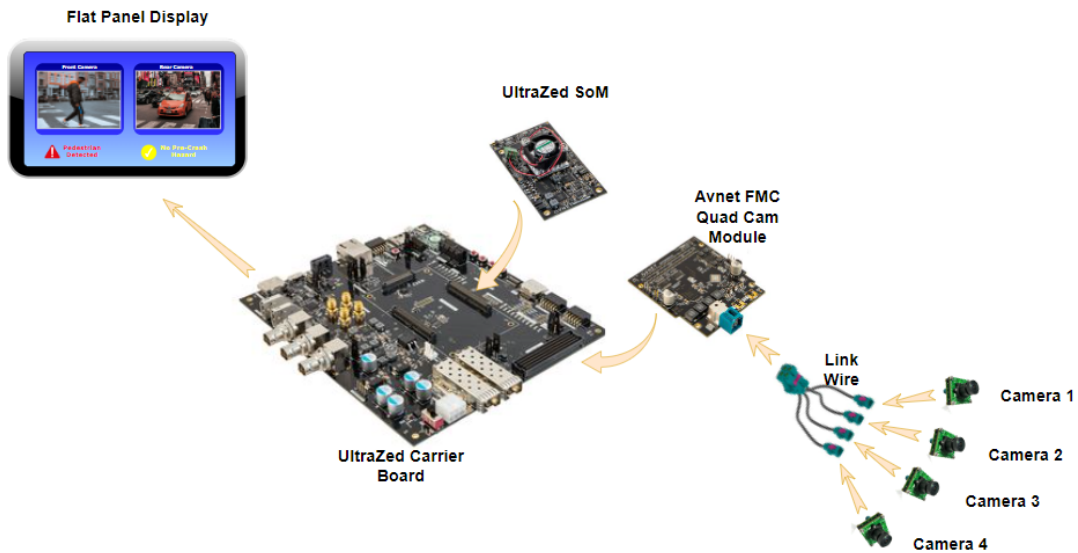


Figure 7. Camera-FPGA-FPD Connection

The above picture shows the hardware connection. The FPD will be interfaced with UltraZed Carrier Board.

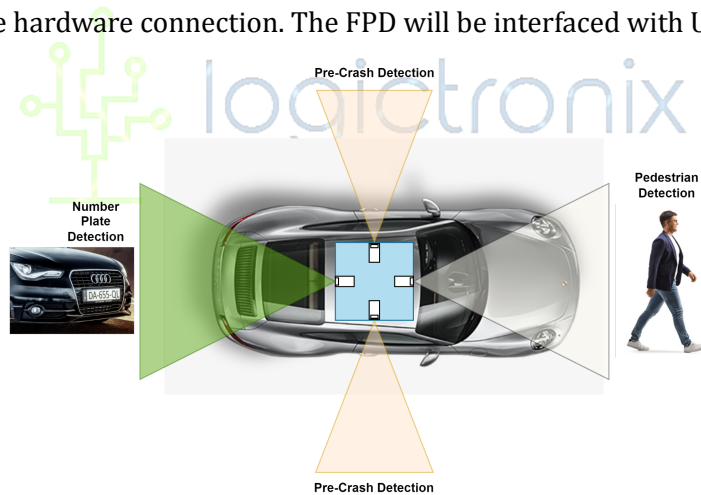


Figure 8. ADAS Camera System on Vehicle

The above figure shows the four-camera setup and installation on the vehicle. The cameras are installed in such a way that the front camera is used specifically to detect Pedestrian Detection, the back camera is used for back vehicle number plate detection. Left-right cameras are used for pre-crash detection.

User/Driver Interface

After the data outcome from the PL processing pipeline, the data are utilized to display in the display unit, such as Flat Panel Display (FPD). The light-weight GUI, developed on QT framework, will be displayed in the FPD, where all four camera data, with detection information, will be displayed in a display unit using GST pipeline. The FPD creates the Human Machine Interface(HMI), from which users can get information as well as user can enter the information. On the other hand, the detection information can also be delivered to the user by visual color signs and text signaling. If detected, red-colored warnings and text can be displayed, and so on.

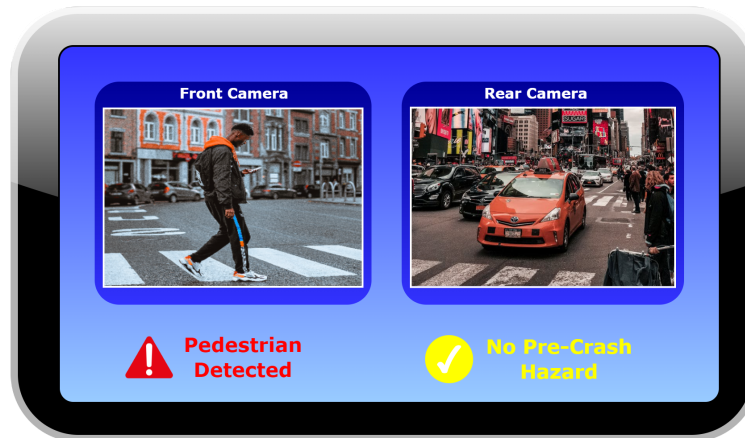


Figure 9. ADAS Flat Panel Display GUI-ADAS Viewer

The above picture shows the information to be displayed over the GUI on FPD. Currently, this GUI has four sections Front Camera Feed, Back Camera Feed, Information Section, and Pre-crash Detection Info Section.

Pedestrian Detecting Front camera feed will be shown in the Front Camera Feed section.

The vehicle number plate detecting the Back Camera feed will be shown in Back Camera Feed Section.

Based on side cameras information, the crash information will be shown in the Pre-Crash Detection Info Section.

Front and Rear camera detection-related information will be shown in Information Section.

What's Next?

The current version, ADAS 1.0 is only based on four-camera inputs. The next version, **ADAS 2.0** will have sensor fusion, such as LiDAR, and UltraSonic Sensor along with eight numbers of cameras to expand ADAS system capability and features to a greater extent.

References

1. [Quad AR0231AT Camera FMC Bundle](#)
2. [UltraZed-EV Carrier Card](#)
3. [UltraZed-EG SOM, Industrial Grade](#)
4. [Ultrazed EV and Multi Cam FMC](#)
5. [ZCU104 Multi-Cam FMC](#)



Revision History

Document Version:	1.0		
Document Date:	01 April 2022		
Document Author(s):	Nikil Thapa		
Document Classification:	Public-Release		
Version History:	Version	Date	Comment
	1.0	04/01/2022	First Version

About Logictronix

LogicTronix provides Turnkey Solutions, Design Services, and Intellectual Property (IP) to customers on FPGA Design, Computer/Machine Vision, Machine Learning Acceleration on FPGA [Edge or Cloud] for various applications as ADAS, Surveillance, Computer Vision, FinTech, etc.

LogicTronix also offers solutions on “Real-Time Traffic Video Analytics Solution (TVAS) - including Automatic vehicle Number-Plate Recognition (ANPR) Solution”, “Enhancing Financial Trading Algorithms with AI/ML” and “High-Frequency Trading (HFT) based Infrastructure”.

For Design Support: **Contact**

LogicTronix Technologies Pvt. Ltd.

Xilinx Certified Partner

Email for design support: info@logictronix.com

Email for IP-Cores: ip-sales@logictronix.com

Web: www.LogicTronix.com