The Chinese University of Hong Kong
Department of Computer Science and Engineering

CENG3430 Rapid Prototyping of Digital System

Final Project - FPGA-based Computer Vision Accelerator for Train Active Safety System

Report

Project Demo Deadline: 16 May 2025 23:59

1155193237 - Yu Ching Hei (chyu2@cse.cuhk.edu.hk)
1155206044 - Cheng Chung Hei Sennett(1155206044@link.cuhk.edu.hk)

Source code and project history is available on GitHub:
`https://github.com/Jellyfish227/FATASS_FPGA-Accelerated_Computer_Vision_on_Train_Active_System.git`

Under the kind support and guidance of
Prof. Ming-Chang YANG

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivs 3.0 Unported" license.

Contents

Division of Work

Task	Person In Charge
Project Proposal	Cheng Chung Hei Sennett
Data Pre-processing	Yu Ching Hei, Cheng Chung Hei Sennett
AI-Inferencing	Yu Ching Hei
Post-processing	Cheng Chung Hei Sennett
Report	Yu Ching Hei, Cheng Chung Hei Sennett

1 Background

It has been observed that the number of accidents involving trains colliding with objects/personnel has increased. As a result, this project aims to create a hardware acceleration for training active safety systems using a similar Advanced Driver Assistance System(ADAS) concept, enabling the trainee to gain situational awareness. The project aims to deploy Edge AI into the FPGA, which is becoming popular in ADAS due to its advantages of low latency and proximity data handling with limited resources such as power and memory, where one of the strengths of modern FPGAS is being an Accelerator for computer vision projects.

2 Introduction

The CENG3430 final project involved implementing a computer vision acceleration platform on the Zynq UltraScale+ MPSoC [2] that can be used to implement a train active safety system.
The workflow consists of two phases:

1. **Data Pre-processing:** We collect and process the data from the camera through the PS USB port, then port the video stream to the PL DPU.

2. **Data AI-Inferencing:** The DPU contains a set of pre-trained models that will perform lane detection, object detection, depth-of-field estimation and drivable area prediction.

3. **Data Post-processing:** Ideally, the PS should release PWM signals to control the motors to stop the rotation. But in our actual implementation, we did not have sufficient time to implement the remaining actuators.

This report outlines our challenges faced and insights gained during implementation, as we aim to deliver a fast and efficient computer vision acceleration platform for train active safety systems.

3 Implementation

The following section will describe the implementation of the ZCU104 on the train active safety system using computer vision.

3.1 Choice of board Due to some hardware limiation of Zedboard. The project required a transition of FPGA from Zedboard to Zynq UltraScale+ MPSoC ZCU102 and ZCU104 [2] as the development board for this project due to the following reasons:

- **AI Inference Acceleration**: To utilize AI inference acceleration effectively, we will feed the data into the DPU (Deep Learning Processing Unit), which will generate results for post-processing. The project will implement the Xilinx DPU IP to achieve this acceleration.
- **Support for DPU IP**: According to Xilinx documentation, the DPU IP is supported only on selected boards from the Zynq™ 7000 SoC and Zynq UltraScale™+ MPSoC families, necessitating a change to ensure compatibility.
- **Ultra RAM Requirements**: This project requires low-latency caching and high input/output throughput. The ZCU104 offers higher Ultra RAM capabilities, enabling the necessary high-speed read and write operations essential for our application.
- **Real-Time Processing Needs**: A real-time processor is critical in order to manage the high input and output throughput effectively. The ZCU104 has the necessary real-time processing capabilities that the Zedboard lacks.

3.2 Comparison of FPGA Options Xilinx vs. Intel, Several key factors led to the preference for Xilinx over Intel:

- **Cost**: Xilinx boards offered a more economical option compared to Altera (Intel) boards with similar performance specifications.
- **IP Resource Availability**: Xilinx provides a larger library of IP resources, which were particularly beneficial for our computer vision project.
- **Development Environment Familiarity**: Our team has experience working within the Xilinx development environment used for the Zedboard, making it easier to transition skills to the ZCU104 within the Zynq SoC family.

Considering the requirements for enhanced AI acceleration and the need for greater processing capabilities, replacing the Zedboard with the Xilinx ZCU104 Evaluation Board was warranted. The ZCU104 will support the necessary DPU IP, enabling effective implementation of hardware acceleration for our computer vision algorithm.

The Zynq UltraScale+ platform provides the following features:

- More logic resources
- More powerful processing unit, including the Real-Time Processing Unit (RPU)
- More support for AI, including the Vitis AI library

3.3 Phase 1: Data Pre-processing In this stage, in order to prototype the input in the most efficient way, we made use of the given USB protocol from the PS side. We tried to utilize the RPU to do the continuous video fetch, but we were unable to find a way to achieve this. In the end, we just used the normal approach that fetched the video frame from the USB port, then passed the frame to the DPU with the AXI HP inteconnect. This allowed for transmission bandwidth up to 128bits. This allowed us to be unaffected by the complex protocol of USB data transmission. The settings for the IO ports are shown below:

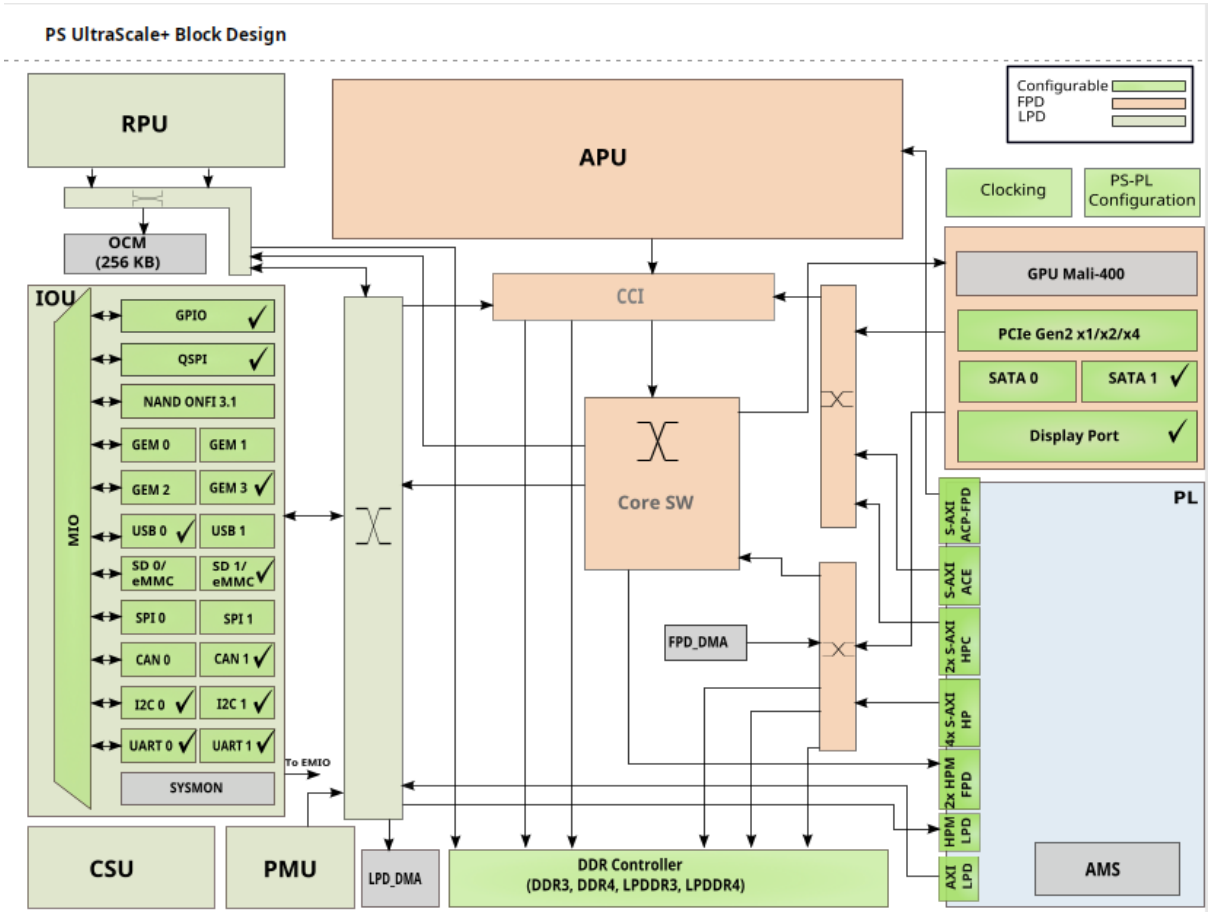

Figure 1: SoC Module Configuration

3.4 Phase 2: AI Data Inferencing This is the main part of this project. Unfortunately, due to the time constraint, we were not able to implement a dataflow-based accelerator implementation, which would take a lot of time to analyse the architecture of the model and break it down into smaller architecture. Therefore, we decided to leverage the Vitis AI library together with the soft DPU IP to implement the accelerator and deploying the pre-trained models.

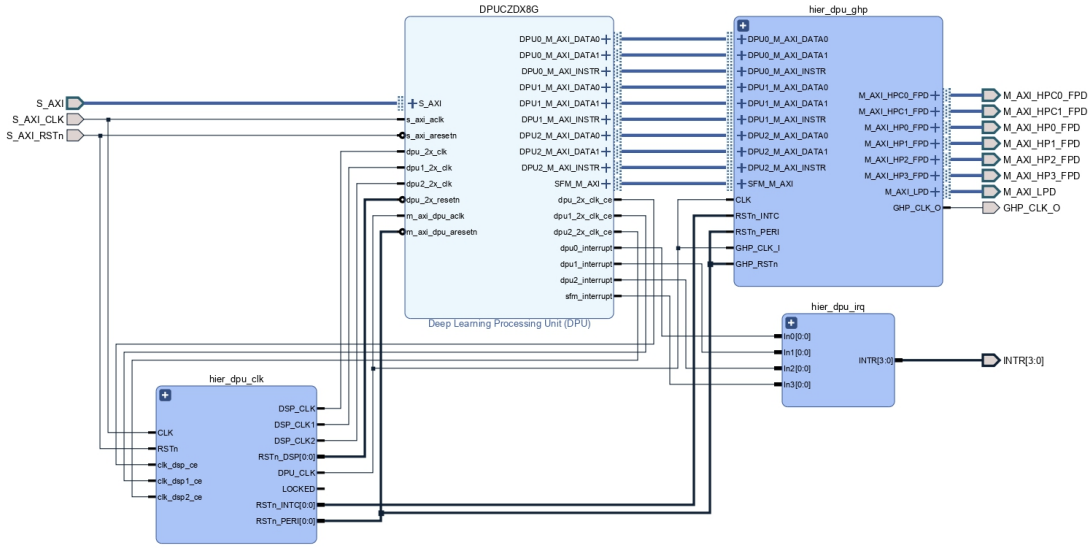

Figure 2: DPU Block Design

Thanks to the existing DPU IP provided by AMD Xilinx, we were able to prototype the DPU application platform effortlessly. However, even though we didn't have to implement a DPU ourselves, we were still struggling to follow the Targeted Reference Design(TRD) as the TRD documents provided had a lot of discrepancies with the resources available. We will discuss this further in the section focusing on difficulties during implementation. In the end, we managed to deploy the DPU and the CV application on the target and successfully ran it.

The DPU subsystem is comprised of several key components, as shown in Figure 3. These include the DPU subsystem, which is integrated into the complete system design using Vivado, the Zynq UltraScale+ Processing System, memory interface, and peripheral subsystems. The AXI interconnect fabric (shown in Figure **??**) plays a critical role in routing data between the DPU and memory.

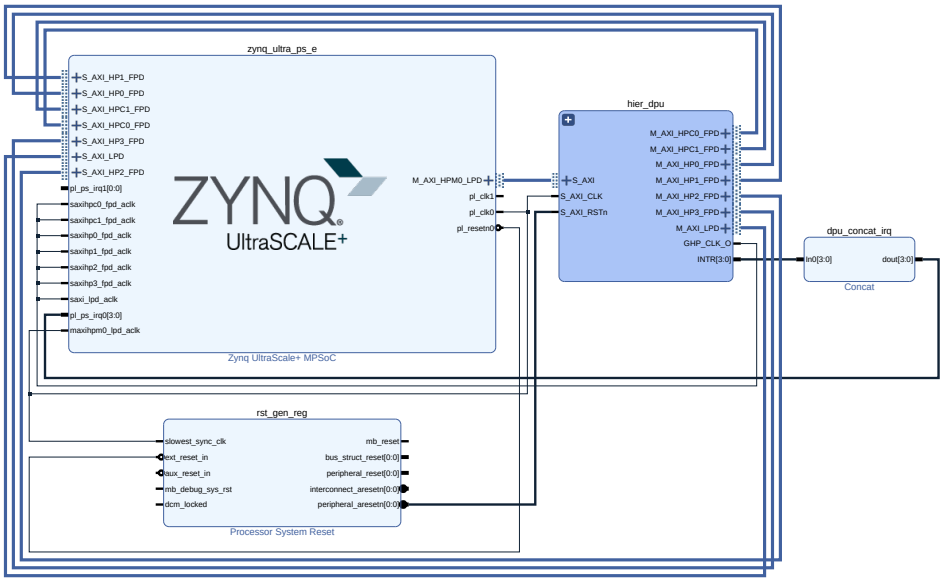

Figure 3: Block Diagram of DPU Integration with PS-PL AXI Interfaces

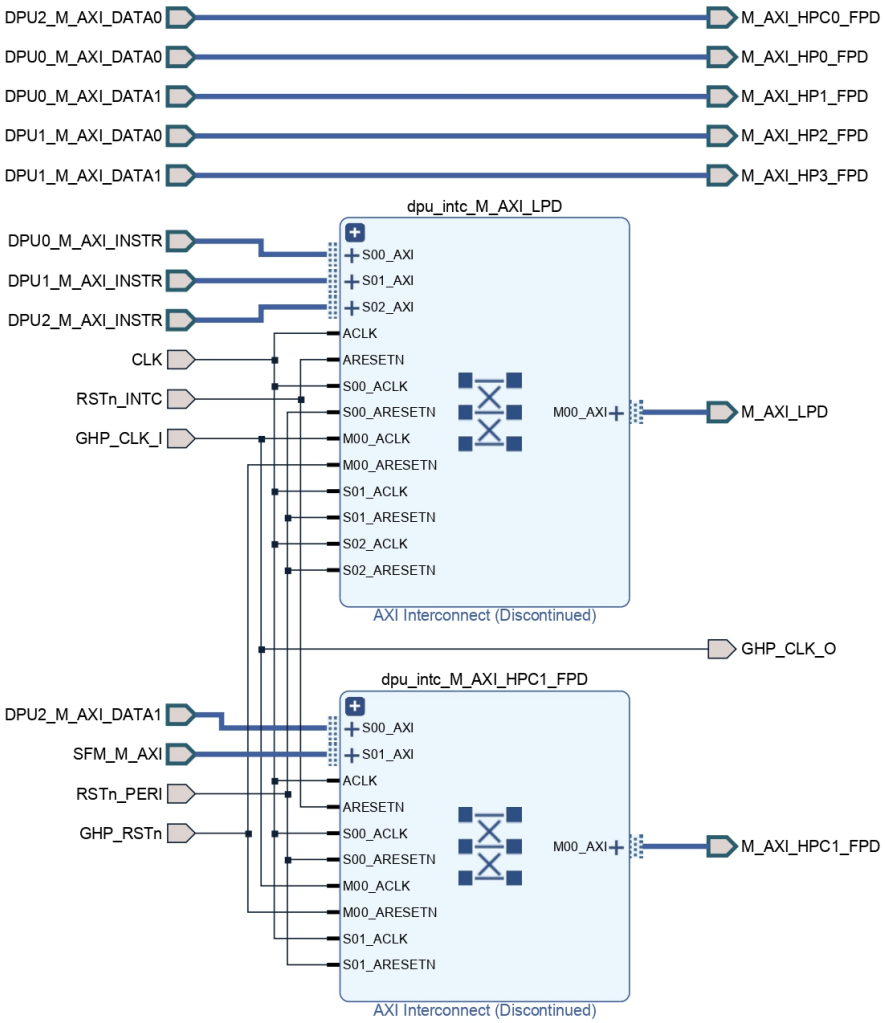

Figure 4: Block Diagram of AXI Interfaces

Furthermore, with the partially DPU Block Design diagram. It includes the DPU core, a clocking wizard for generating domain-specific clocks, interrupt logic, and a set of AXI interconnects that enable high-throughput communication between the DPU and the PS-side DDR memory.

Overall Block Diagram

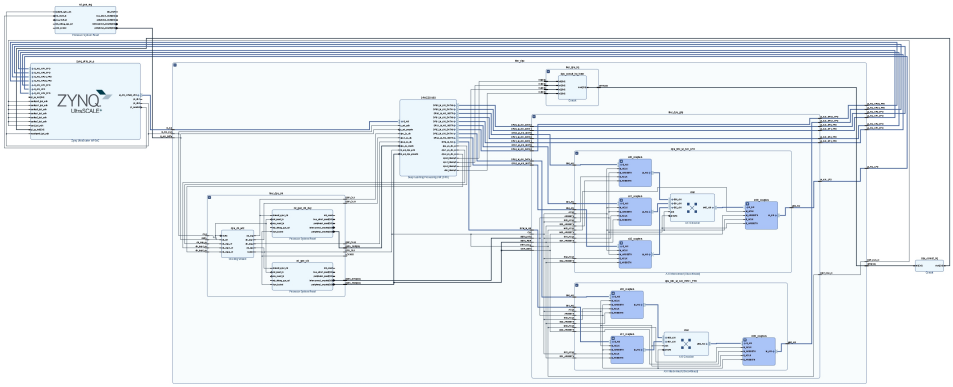

Figure 5: Overall Block Diagram

Figure 5 illustrates the complete hardware architecture implemented in Vivado, representing the final and most detailed instantiation of the AI-accelerated embedded system. This top-level design integrates multiple subsystems including the Zynq UltraScale+ Processing System (PS), the Deep Learning Processing Unit (DPU) subsystem, AXI interconnects, clocking infrastructure, and reset logic.

At the center of the programmable logic (PL) is the `DPUCZDX8G` core, encapsulated within a hierarchical block (`hier_dpu`). This block interfaces with the PS through multiple high-performance AXI ports (HP, HPC, and LPD), enabling high-speed memory access for inference tasks. Each DPU channel (instruction and data ports) is connected through dedicated AXI interconnects such as `dpu_intc_M_AXI_HPC1_FPD` and `dpu_intc_M_AXI_LPD`, which are responsible for routing traffic between the DPU and the PS-side DDR memory.

The design also includes a set of AXI crossbars and couplers (e.g., `xbar`, `s00_couplers`) that facilitate the communication between multiple master and slave AXI interfaces. These components ensure proper data path arbitration and synchronization across different subsystems.

Clocking is handled using the `dpu_clk_wiz` module, which generates multiple clock domains including `DPU_CLK`, `DSP_CLK`, and `DSP_CLK2`, required by the DPU and its supporting logic. Reset logic is managed by multiple instances of the `Processor System Reset` block (e.g., `rst_gen_clk`, `rst_gen_reg`) that ensure reliable startup and glitch-free operation across clock domains.

Interrupts from the DPU cores and supporting logic are aggregated using the `dpu_concat_irq` and `hier_dpu_irq` blocks. These are routed back to the PS through the `pl_ps_irq0[3:0]`

lines, enabling the software to respond to asynchronous events such as inference completion.

Overall, this block design reflects a modular and scalable architecture that leverages the reconfigurable logic of the Zynq UltraScale+ MPSoC to accelerate AI inference while maintaining tight integration with the software stack running on the ARM Cortex-A53 application cores.

3.5 Phase 3: Data Post-processing We were originally planning to utilize the DPU output to do some decision making and actuation. However, troubleshoot during the pre-development phase (including but not limited to setting up the OS environment and toolchains, and configuring the tools to match the latest version of the toolchain like fetching upgraded bsp, shared-state cache (sscache) and all the related PetaLinux build artifects) took too much time. These steps were extremely time consuming and we were unable to further develop on the post-processing system. Yet, we still wanted to visualize our DPU effort so we are exporting the video processed directly to the screen through the display port protocol. Which can also be see in Figure 1

4 Difficulties during implementation

4.1 Development environment differences During the initial implementation we intended to apply the Target Reference Design(TRD) to initialize the board. In order to do so we would require tools that support Linux, such as PetaLinux, Vitis AI and also Vivado Linux version. Since the development PC in the current lab and the personal PC only has windows installed, therefore we would need to reinstall a Linux development environment such as Ubuntu 24.04 LTS to run the tools that run the TRD.

4.2 New tools, new problems During the implementation process, we decided to use the latest version of Vivado design suite and Vitis AI library to keep up to date with the current technology. But the newer version of Vivado design suite is significantly different from the 2016 version which is used in the lab, therefore the interface and processes were significantly different and we were required to re-learn the entire development process on our own. Furthermore, when trying to make different TRD with the newer tool chains since AMD does not provide backward compatibility in Vivado and Petalinux it would not be able to run TRD version from an older version so therefore often we would need to install the older version of Vivado which takes minimum of 6 hours for setting up and restarting the entire progress, or find a workaround such as downloading the package required to successfully build the TRD and also petalinux that can be used in ZCU104.

4.3 Significantly long time for implementation and routing Due to the increase in complexity of the project for the implementation and routing of the project particularly for Vitus AI it would take a minimum of 2 hours particularly for minor changes, making it very difficult considering the little amount of time that we have for the current project. It would have increased the difficulty for our group to conduct even more testing with the system.

5 Results

After the Implementation of the design we have successfully deploy a hardware accelerated computer vision model as shown in the implementation design below.

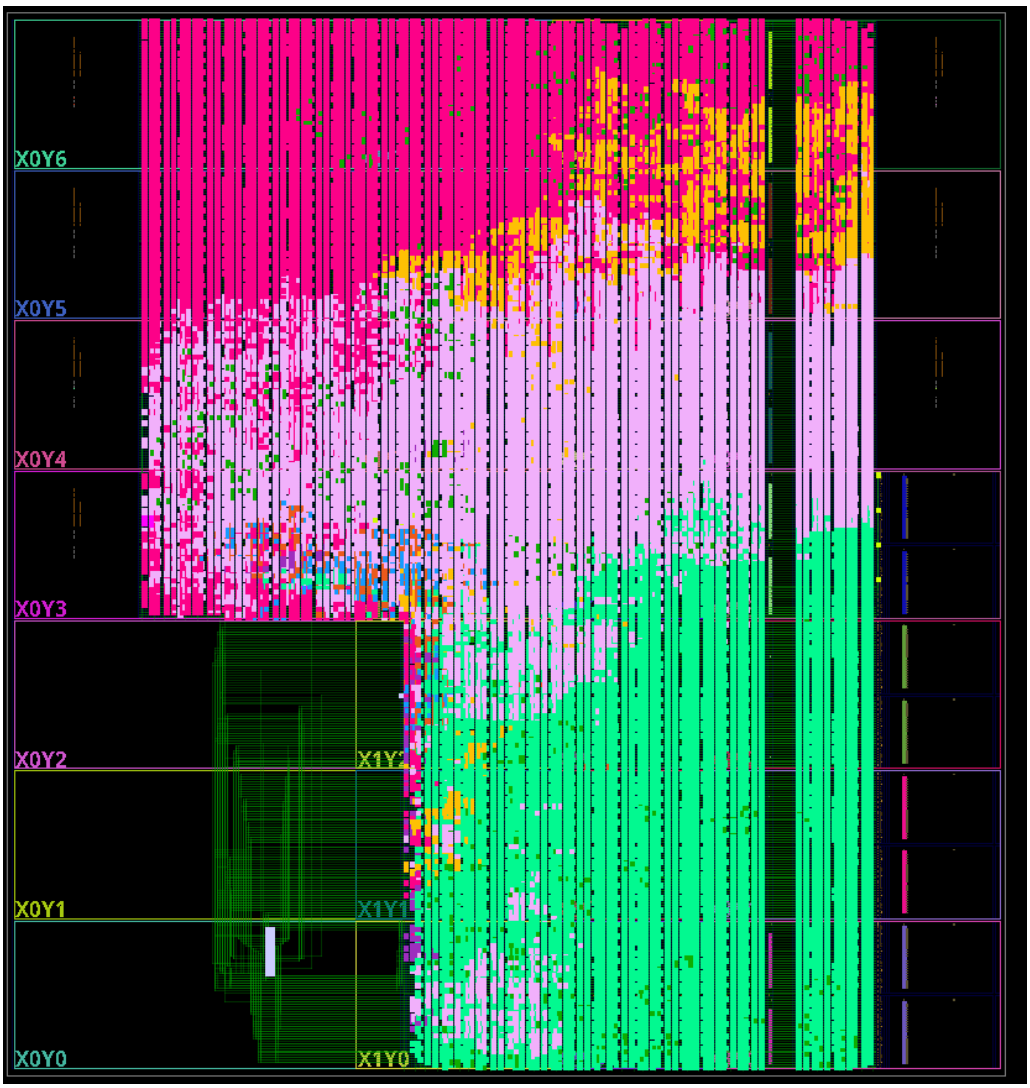

Figure 6: Implemented Design

From Figure 6 the resources from the ZCU104 have almost been fully utilized successfully. The 3 major Coloured areas in the chip floor plan correspond to the 3 cores of our DPU. Furthermore, in this design, the power usage is shown below:

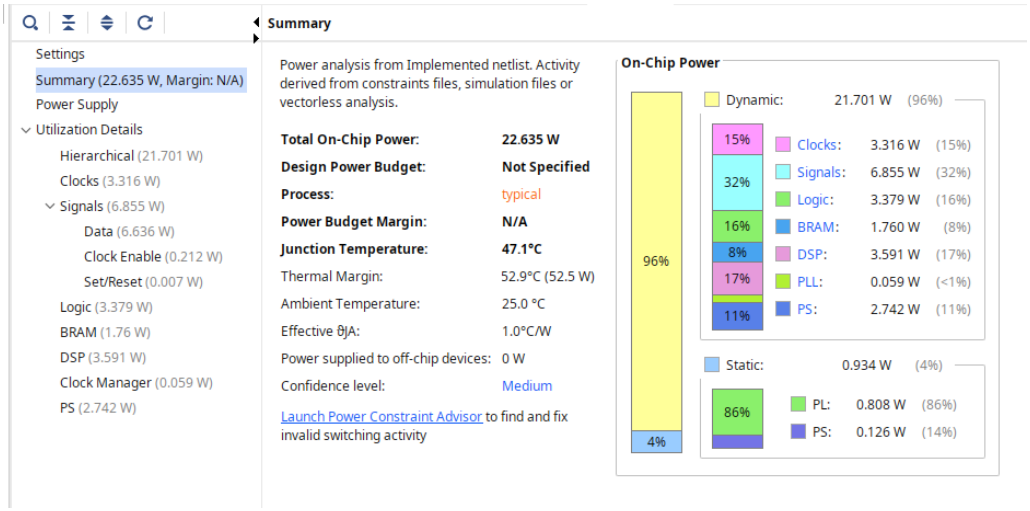

Figure 7: The power of the ZCU104 on running hardware acceleration model

From the following figure 7 it can be seen that the power of this project is 22.635 W compared to a RTX4090 that uses around 450 W and a RTX4060 that uses 115 W suggesting the power efficiency of the ZCU104.

5.1 Result of the computer vision model To test the accuracy of the model, our group synthesized a train track video [1] with foreign objects such as human and vehicles which were placed near or on the track to test the performance of the following deployed model.

5.2 Track detection For the current model, we have tested on track detection with the following reference video of train track in Switzerland. [**AKSense-Zurich˙2023**] and also with a more simple track with less visual cluster [1]. The result of the detection is as follows:

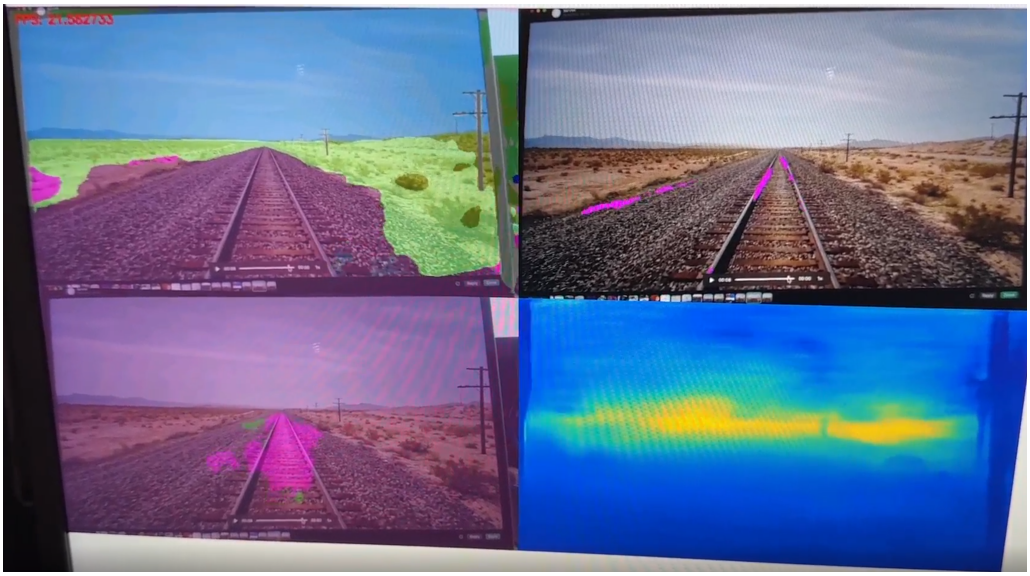

Figure 8: Simple Train Track testing result

The detection model can gain a stable 20 fps+ for most situation. From the figure 8 in the bottom left showing the drivable area in pink area it successfully detected the drivable area in the track tack and furthermore on the bottom right showing the depth detection we can see that the further away it is it will appear more yellow and when it is closer it will be a deeper blue color. Furthermore, the top right video is an edge detection for the rail way which is also successfully shown with a lower accuracy, and lastly the top left video it combines the 2 detections and on the top right image it is a detection for foreign object which is shown as follows:

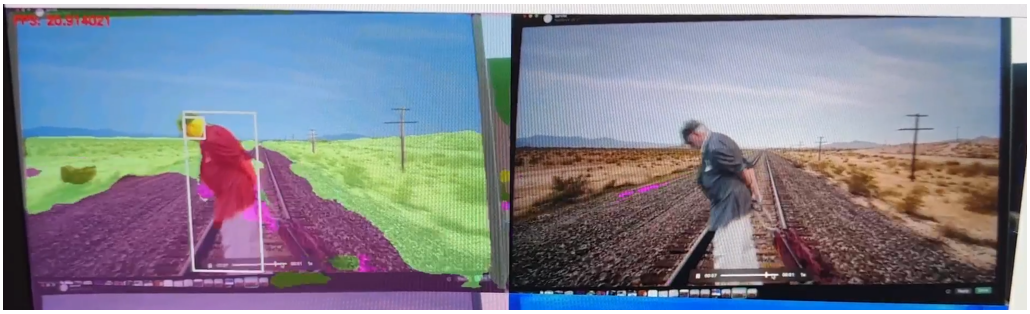

Figure 9: Train Track Testing with Foreign Object result
xd

By looking at the following figures 9 the person on the train track is successfully being detected and it is marked by the border.

Additionally, when we are testing on a more complex environment [3] the model also shows us some further insight for tract detection. In a more complex environment in a standard environment the model is still able to identify the track and drivable area successfully with the background environment being successfully identified.

Figure 10: Train Track Testing on a more complex environment

Additionally, when in another setting such as the complex environment where there is a train platform, the model can also successfully identify the people on the platform. While the object is being successfully identified the drivable area is not successfully

identified due to the lack of training on the model which would be explained further in the later part about the trained model.

Figure 11: object detection on a more complex environment

Furthermore, when testing in low light environment such as in a tunnel the model would still be able to identify the track area while over-estimating the drivable area, causing the edge or the railway to not be easily identified.

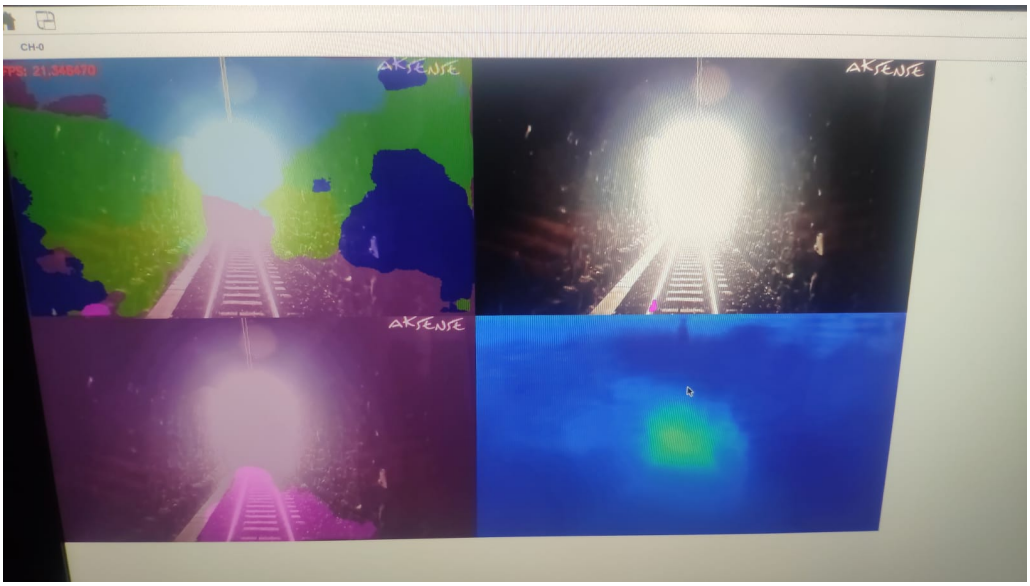

Figure 12: Object Detection on a Low Light Environment

5.3 Road detection with the current model The reason for a low accuracy of the rail way detection is due to how the model was being trained. Due to the limited time, the model was trained with roads instead of railway, explaining some inaccuracy with the model in the railway environment. By looking at the figure 13 the result for the object detection has a much higher accuracy.

Figure 13: Object Detection on a road environment

From the figure 13 the road edge is being identified in a much higher accuracy with the drivable area being shown much more clearly and also other foreign objects or cars are have also been clearly identified.

6 Recommendation and improvement

6.1 Transform learning Since the model was being trained in a road condition and it can collect a reasonable result with the current model, on a train tack condition we can conduct a transform learning to obtain a much higher accuracy in the model for object detection.

6.2 Further optimize to increase CPU core count To increase the DPU core count and efficiency with the model, apart from optimizing the model itself, we can also further optimize the current usage of resources and increase the CPU core count which enable us to have more pipelining for more DPU core to perform the hardware laceration.

6.3 Stability of the System Furthermore, the current system may often have crashes due to buffer issues, particularly when the RAM size is not enough. Therefore, for further development we would need to help optimise the buffer size to reduce the amount of crashes with the current development or increase the RAM size ensuring for a more stable system.

7 Conclusion

This project successfully demonstrates the implementation of a hardware-accelerated computer vision system for train active safety using the Zynq UltraScale+ MPSoC platform. By leveraging the Xilinx DPU IP and Vitis AI library, the team managed to deploy a functional AI inference system capable of detecting train tracks, foreign objects, and drivable areas with reasonable accuracy. The project showcases the advantages of FPGA-based acceleration, such as low latency, power efficiency, and real-time processing capabilities, which are critical for safety-critical systems like active train monitoring.

Despite these achievements, several challenges were encountered during development. These included difficulties in setting up the Linux-based development environment, adapting to new tools, and addressing the long implementation and routing times. Furthermore, the limitations in model training, such as using road data instead of railway-specific data, led to reduced accuracy in certain scenarios. Post-processing and actuator integration were also constrained by time, leaving room for further development.

8 References

[1] Frontman Media. *4K Train Tracks #1 - Front POV using Drone - Free Stock Footage Frontman Media*. YouTube. 2017-06. URL: https://www.youtube.com/watch?v=elgEX7hu8Xc&ab_channel=FrontmanMedia.

[2] AMD Xilinx. *Zynq UltraScale+ MPSoC Software Developer Guide*. UG1137 (v2022.2). AMD Xilinx. 2022. URL: https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug1137-zynq-ultrascale-mpsoc-swdev.pdf (visited on 2024-03-25).

[3] Andrea S and Amit K S. *4K Train Driver view - Montreux to Montbovon - Goldenpass Panoramic MOB Train Switzerland — Cab ride*. YouTube. 2023-09. URL: https://youtu.be/kUDxhIyaCEA.