IMPLEMENTATION AND PERFORMANCE ANALYSIS OF DROWSINESS DETECTION USING HARDWARE ACCELERATION ON PYNQ-Z1 FPGA

OOI HAN YI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

IMPLEMENTATION AND PERFORMANCE ANALYSIS OF DROWSINESS DETECTION USING HARDWARE ACCELERATION ON PYNQ-Z1 FPGA

OOI HAN YI

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computer Engineering with Honours

Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka

2024

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek	:	Implementation and Performance Analysis of Drowsiness Detection using Hardware Acceleration on PYNQ-Z1 FPGA
Sesi Pengajian	:	2023/2024

Saya OOI HAN YI mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.
✓	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS)

(COP DAN TANDATANGAN PENYELIA)

DR. ANIS SUHAILA BINTI MOHD ZAIN
Pensyarah Kanan
Fakulti Teknologi Dan Kejuruteraan Elektronik Dan Komputer (FTKEK)
Universiti Teknikal Malaysia Melaka (UTeM)

Tarikh : 20 Jun 2024

Tarikh : 26 Jun 2024

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this report entitled “Implementation and Performance Analysis of Drowsiness Detection using Hardware Acceleration on PYNQ-Z1 FPGA” is the result of my own work except for quotes as cited in the references.

.

Signature	:	……………………………………
Author	:	Ooi Han Yi
Date	:	20 June 2024

APPROVAL

I hereby declare that I have read this thesis and in my opinion, this thesis is sufficient in terms of scope and quality for the award of Bachelor of Computer Engineering with Honours

Signature	:	…………………………………………
Supervisor Name	:	Dr. Anis Suhaila Binti Mohd Zain
Date	:	26 June 2024

DEDICATION

To my beloved parents, who have been my unwavering source of inspiration, motivation, and encouragement throughout my academic journey. Without your love, sacrifices, and steadfast support, I would not be where I am today. This paper is a testament to your unfailing faith in me, and I dedicate it to you with all my heart. To my supervisor, Dr. Anis Suhaila Binti Mohd Zain, and my co-supervisor, Ts. Dr. Sani Irwan Bin Salim, whose expertise, guidance, and mentorship have shaped my research and inspired me to push my limits. Your patience, constructive feedback, and valuable insights have helped me grow as a researcher and scholar. I am grateful for the opportunities you have given me to learn, collaborate, and contribute to the academic community. To Universiti Teknikal Malaysia Melaka, which has provided me with a rich academic environment, cutting-edge resources, and a platform to pursue my educational passion. My experience at this university has transformed me, allowing me to think critically, explore new ideas, and be exposed to different perspectives. I am honoured to be a part of this academic community, and I dedicate this paper to this university that has given me so much. Thank you.

ABSTRACT

Drowsiness detection algorithms implemented on general-purpose processors perform well but suffer from portability issues and high power consumption. This project aims to overcome these limitations by designing and developing a drowsiness detection system on the PYNQ-Z1 FPGA platform. The project transitions from a software-based model to an FPGA-optimized design using high-level synthesis (HLS) of the Xilinx FINN compiler. By leveraging the parallel processing capabilities of FPGAs, the drowsiness detection is optimized for latency, power consumption, and resource utilization. The system monitors yawning and blinking, ensuring high performance while improving computational efficiency and power consumption. The integration of convolutional neural networks with FPGA frameworks demonstrates the synergy between neural network architectures and reconfigurable hardware. The results show that switching from a 6-bit model to a 2-bit model significantly reduced memory usage by 45.24%. Additionally, the quantized model on the PYNQ-Z1 reduces power consumption by 95.52% compared to the CPU. This research not only advances FPGA-based deployment, but also lays the foundation for future innovations in hardware design, neural networks, and artificial intelligence, enhancing the visual perception capabilities of computer vision and autonomous systems.

ABSTRAK

Algoritma pengesanan rasa mengantuk yang dilaksanakan pada pemproses tujuan umum berfungsi dengan baik, namun mempunyai masalah isu mudah alih dan penggunaan kuasa yang tinggi. Projek ini bertujuan mengatasi kekurangan tersebut dengan membangunkan sistem pengesanan mengantuk pada platform FPGA PYNQ-Z1. Projek ini beralih daripada model berasaskan perisian kepada reka bentuk FPGA yang dioptimumkan dengan menggunakan pengkompil Xilinx FINN high-level synthesis (HLS). Dengan memanfaatkan keupayaan pemprosesan selari FPGA, pengesanan rasa mengantuk dioptimumkan untuk kependaman, penggunaan kuasa dan penggunaan sumber. Sistem akan memantau aktiviti menguap dan kelipan mata untuk membolehkan prestasi tinggi dapat dicapai sambil meningkatkan kecekapan pengiraan dan mengurangkan penggunaan kuasa. Penyepaduan Rangkaian Neural Konvolusi dengan rangka kerja FPGA menunjukkan sinergi di antara seni bina rangkaian neural dan perkakasan yang boleh dikonfigurasikan semula. Keputusan menunjukkan bahawa menukar daripada model 6-bit kepada model 2-bit mengurangkan penggunaan memori dengan ketara sebanyak 45.24%. Selain itu, model terkuantisasi pada PYNQ-Z1 mengurangkan penggunaan kuasa sebanyak 95.52% berbanding dengan CPU. Penyelidikan ini bukan sahaja memajukan *penggunaan berasaskan FPGA, tetapi juga meletakkan asas untuk inovasi masa hadapan dalam reka bentuk perkakasan, rangkaian saraf, dan kecerdasan buatan, meningkatkan keupayaan persepsi visual penglihatan komputer dan sistem autonomi.*

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who played an important role in the completion of this thesis. First, I would like to express my sincere gratitude to my supervisor, Dr. Anis Suhaila Binti Mohd Zain, and co-supervisor, Ts. Dr. Sani Irwan Bin Salim, for their unwavering guidance, valuable insights, and continuous support throughout the research process. Their expertise and encouragement were crucial in determining the direction of this work. Their expertise has greatly enriched the content and methodology of this thesis. Not to forget, special thanks to my beloved father and my beloved mother for their external support. I would also like to thank my good friends who always stayed by my side to cheer me up. I am lucky to have such a warm and loving family and friends. I would like to thank Universiti Teknikal Malaysia Melaka, Malaysia, for providing the necessary resources and research facilities to facilitate the implementation of this project. Finally, I would like to thank all my friends and supervisors who contributed in various ways to the completion of this thesis. Your support is indeed invaluable. This thesis is a testament to the spirit of cooperation and collective efforts of the people I met in my academic journey. Thank you all for your support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

FPGA	:	Field Programmable Gate Array
SoC	:	System on Chip
CNN	:	Convolutional Neural Network
ReLU	:	Rectified Linear Unit
ONNX	:	Open Neural Network Exchange
HLS	:	High-Level Synthesis
YawDD	:	Yawning Detection Dataset
EAR	:	Eye Aspect Ratio
MAR	:	Mouth Aspect Ratio
PERCLOS	:	Percentage of Eyelid Closure
ECD	:	Eye Closure Duration
YOLO	:	You Only Look Once
ECG	:	Electrocardiogram
EEG	:	Electroencephalogram
HRV	:	Heart Rate Variability
LDW	:	Lane Departure Warning
SWA	:	Steering Wheel Angle

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter is divided into 5 sections. Section 1.2 will discuss the background study for the project followed by the problem statement in Section 1.3. The objectives of the project are discussed in Section 1.4 and the scope of the work is discussed in Section 1.5. The thesis layout will be introduced in Section 1.6.

1.2 Background of Project

Drowsiness is a state of being drowsy, tired, exhausted, or mentally or physically weak. A sleepy person has a low concentration level, which may make it difficult for them to maintain a certain level of focus. If this happens to a driver, a mechanic operating a heavy machine, or a railway operator, it may lead to an accident. Car accidents are one of the fatal accidents and are considered a major problem in society.

In Malaysia, a study by the Malaysian Institute of Road Safety Research (MIROS) reported that fatigue is one of the biggest causes of car, truck, and bus accidents. In addition to this, another study conducted by MIROS on commuter accident victims in the Klang Valley reported that about 15% of them were involved in accidents due to drowsiness or fatigue [1].

Drowsiness detection analysis has become an interesting field, and many methods have been introduced based on different categories of invasive and non-invasive. With invasive methods, the subject needs to attach a sensory device (such as electrodes) to the body to measure signals from certain parts of the body (such as brain signals and heart signals). This can be uncomfortable for the subject, and any large movements can affect the signal. In contrast, non-invasive methods are more user-friendly, flexible, and more acceptable as they do not require any connection to the human body. But as of now, we know that deploying drowsiness detection on traditional computing platforms such as CPUs and GPUs often encounters difficulties, including excessive power consumption, high cost, and heat dissipation issues.

This project aims to implement drowsiness detection by analyzing the signs of drowsiness through eye and mouth activities using non-invasive techniques focusing on image processing methods. By the end of the project, it is expected that the developed algorithm will be able to detect blink and yawn with low latency within the limited resource constraints of the FPGA.

1.3 Problem Statement

The implementation of drowsiness detection is usually done on CPUs and GPUs because they have better speed and resources [2]. However, the energy and processing requirements of the drowsiness detection process remain high in pursuit of higher throughput. This project aims to address the power consumption issue by studying the implementation of drowsiness detection using FPGAs. This is because FPGAs have become a possible alternative for implementing drowsiness detection. After all, they consume less power and fewer resources than traditional computing platforms. FPGAs are known for their portability, reconfigurability, and power consumption levels. FPGA implementation also means the algorithm is embedded in the system while only a single board deployment is required.

The basic idea to solve the problems of traditional computing platforms is to reduce their high-power consumption and use the parallel processing and reconfigurable capabilities of FPGAs to provide a more economical and effective platform for implementing drowsiness detection. Energy efficiency is crucial to ensure drowsiness detection can operate for extended periods without draining the power source in the vehicle system. However, the challenges of doing so are also foreseeable. Due to the resource and power limitations and architecture of FPGAs, several efforts need to be made to implement drowsiness detection on FPGAs successfully [3]. Analysis is also required to validate the issues with traditional computing platforms and FPGA-based drowsiness detection solutions. The need for energy-efficient hardware acceleration of drowsiness detection on FPGAs must be emphasized [4].

1.4 Objectives

i. To design and implement drowsiness detection on FPGA.
ii. To explore optimization techniques for improving the drowsiness detection performance on FPGA.
iii. To analyze FPGA-based drowsiness detection performance in terms of latency, power consumption, and resource utilization.

The primary goal of this project is to design and implement drowsiness detection on a field programmable gate array (FPGA) and verify it on a board to ensure that it works properly on the board. Due to the resource limitations of the FPGA, the drowsiness detection needs to be optimized to ensure that it can be implemented on the actual FPGA board. Finally, analyzing the performance of the FPGA-based drowsiness detection is also critical to gaining a deeper understanding of the project.

1.5 Scope of Work

The scope of work for this project is to design and implement drowsiness detection on the Xilinx Zynq-7000 PYNQ-Z1 FPGA board using optimization techniques and to perform performance analysis of the drowsiness detection implemented on the board. The datasets used are the WIDER FACE dataset and the Yawning Detection Dataset (YawDD). The software and frameworks used are Docker, Brevitas, Jupyter Notebook, Vivado, and Vitis HLS.

1.6 Thesis Layout

Chapter 1 provides the overall concept of the project and covers the introduction of drowsiness detection, problem statement, objectives, scope of work, and thesis layout. This will help us understand more details and the specific reasons for undertaking this project.

Chapter 2 includes a literature review based on the architecture and platform that will be used in the project. For this project, face detection, drowsiness sign methods, and the FPGA platform are the core architectures for drowsiness detection. The corresponding literature review will refine and delve into the concepts that are closely related to the project through diagrams. The chapter ends with a discussion section where all the reviewed methods are discussed based on their pros and cons.

Chapter 3 presents the methodology of the implementation and analysis process of drowsiness detection architecture in the FPGA platform. This chapter will provide a flowchart of the project and describe each process in detail. In the last section, the techniques to obtain drowsiness sign analysis are described.

Chapter 4 presents the results obtained in this project and the performance analysis. The dataset will be validated on the FPGA to verify that drowsiness detection is well implemented. This chapter presents the results of accuracy, latency, power consumption, and resource utilization.

Chapter 5 is the last and will present the conclusion of this project after completing all the theories, results, and analysis. In addition, this chapter also involves suggestions for future work.

CHAPTER 2

BACKGROUND STUDY

2.1 Introduction

This chapter focuses on the background research and literature review of this project to investigate the latest progress in this field. The background research will cover drowsiness, face detection, signs of drowsiness, and field programmable gate arrays (FPGAs). In addition, a review of related projects and literature will be provided to illustrate the relevance and progress in these areas.

2.2 Drowsiness

Drowsiness is a state of feeling tired due to lack of sleep, which can affect a person's level of consciousness and is particularly fatal for drivers. A drowsy person lacks a certain level of consciousness or alertness, which triggers the desire or tendency to fall asleep. Unfortunately, many fatigued drivers often experience microsleep. This is a state in which people do not realize that they are falling asleep after driving for hours due to a poor mental state. For drivers who drive for a long time, drowsiness is often caused by various medications, lack of sleep, and boredom. Drowsy drivers may lose control of their vehicles, resulting in accidents. So far, an increasing number of traffic accidents are caused by low driver alertness, which has become a serious problem in society. When drivers are drowsy, accidents tend to be more serious because they cannot react and control the vehicle to avoid crashing.

Three technologies can detect signs of drowsiness, namely physiological-based measurements, vehicle-based measurements, and behavior-based measurements. Generally, vehicle-based measurements are used in drowsiness detection research by observing driving patterns. Steering wheel movements, braking patterns, lane changes, and speed are examples of driving patterns that can be observed to indicate drowsiness. However, this approach is limited to the type of vehicle and road conditions. Instead, the presence of camera technology has enabled researchers to apply behavior-based measurement techniques that use image processing methods to detect drowsiness through the driver's behavior [5]. This approach is more user-friendly and easier to implement compared to intrusive methods. Head rotations, blinking patterns, gaze estimation, and yawning activities are behaviors that have been used as indicators to detect drowsiness stages.

2.2.1 Physiological Based Approach

Physiological-based measurement is an invasive method to detect signs of drowsiness. This technique measures signs of drowsiness by acquiring signals from certain parts of the body, such as the heart rate, called electrocardiogram (ECG), or brain wave patterns, called electroencephalogram (EEG). A sensing device is required, or multiple electrodes are used to acquire the signals. The electrodes are used to analyze drowsiness and fatigue states through EEG data. A method for predicting driver drowsiness by evaluating heart rate variability (HRV) through ECG devices is introduced [6]. Users need to place electrodes on body parts, which is uncomfortable, inefficient, and dangerous to implement in real-time.

Therefore, as technology develops, researchers have found a way to develop new wireless sensing devices to record the rate of physiological signals. EEG channels are selected to record signals to detect sleepiness stages [6]. Certain mobile headphones are used to record EEG signals. Figure 2.1 shows a driver alertness monitoring system for drowsiness detection using wired wearable EEG, which is typically embedded in the driver's hat [7].

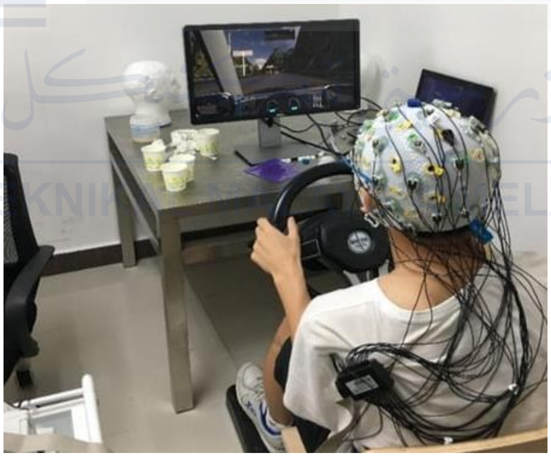


Figure 2.1: Electroencephalogram (EEG) Signals [7]

2.2.2 Vehicle Based Approach

In addition to measurements based on physiological signals, another approach to detecting signs of drowsiness is to analyze the driver's driving pattern, mainly recording lane changes, steering wheel movements, and vehicle speed changes. This approach is a non-invasive method and does not require the device to be worn or attached to the user.

Figure 2.2 shows the Lane Departure Warning (LDW) system as one of the vehicle-based measurement categories [8]. The system completely relies on the detection of lane markings on the road, which requires the lane markings to be always visible for the system to work properly [9]. Occlusion caused by the preceding vehicle or weather conditions during heavy rain may degrade the performance of the system. Moreover, road conditions may lead to false detection of signs of drowsiness. For example, if there are potholes on the road, some drivers may suddenly change lanes. Therefore, the system is preferably implemented only on straight roads or highways.

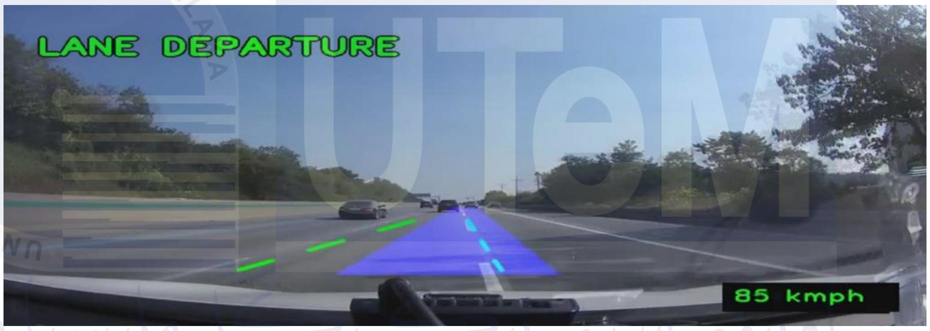


Figure 2.2: Lane Departure Warning [8]

Steering Wheel Angle (SWA) is another approach used to detect signs of drowsiness in vehicles, where triaxial measurement of SWA is implemented. Driver drowsiness is monitored by calculating SWA data obtained from a sensor mounted on the steering column [10]. Analyzing the data is challenging because it is acquired from a real-time environment and random vibrations may cause the retrieved data to vary slightly. The main reason is that the vibration of the wheel and suspension system interferes with the frequency range of the steering signal. It turns out that SWA is not a perfect indicator of signs of drowsiness because it requires complex calculations and pre-processing operations. In addition, drowsiness detection based on steering wheel angle is not reliable to implement on FPGA because it requires the use of many sensors and is computationally complex.

2.2.3 Behavioral Based Approach

The simplest way to detect whether a driver is drowsy is through his behavior because behavior shows the most obvious signs. For example, a drowsy driver tends to move his head frequently to avoid the feeling of falling asleep or a drowsy driver will show rapid blinking activities. Like vehicle-based measurements, this method is a non-invasive method and does not require the user to wear any equipment. Eye features are one of the common indicators for detecting signs of drowsiness. A system for detecting drowsy drivers is developed by combining three parameters of eye movement, namely, percentage of eye closure (PERCLOS), blinking frequency, and eye closure duration [11]. In addition to eye features, yawning is another method for measuring drowsiness based on driver behavior, which has been widely used. Moreover, the head pose is one of the obvious signs of sleepiness because sleepy drivers nod frequently. However, implementing this measure in real-time is dangerous because the sign of nodding indicates that the driver is already in the final stage of drowsiness.

2.3 Face Detection

Face detection has received much attention and is one of the most promising applications in the field of image analysis. Face detection is an important component of biometrics, video surveillance, and human-computer interaction. Many image face detection methods have been proposed, which gave all researchers more inspiration to improve the performance, speed, and accuracy of the algorithms. For this, they have managed to obtain various algorithms for detecting faces, each with their characteristics, which will be compared in more detail in this chapter. Face detection is a critical initial step in detecting signs of drowsiness based on facial features, as successfully detecting a face will provide accurate true positive results for detecting other facial features. There are three main techniques for detecting faces and their facial features, including eyes and mouth, namely the haar cascade algorithm, the convolutional neural network architecture, and the skin colour technique.

2.3.1 Haar Cascade Algorithm

The Viola-Jones algorithm is widely used in face detection algorithms because it is the first face detection system ever [12]. The proposal of this framework can process images very quickly while achieving high detection rates. According to the research, the algorithm consists of three parts that work simultaneously to achieve fast and accurate detection. First, the image is converted into an “integral image”, which allows for faster calculation of the features used by the detector. Second, the classifier used is an efficient and straightforward classifier built using the AdaBoost learning algorithm. Finally, the classifier is generated by combining weak classifiers into a “cascade”, which allows for the rapid elimination of background areas of the image while spending more computation to improve face-like areas. Figure 2.3 shows more details on how the Viola-Jones algorithm works in face detection [13].

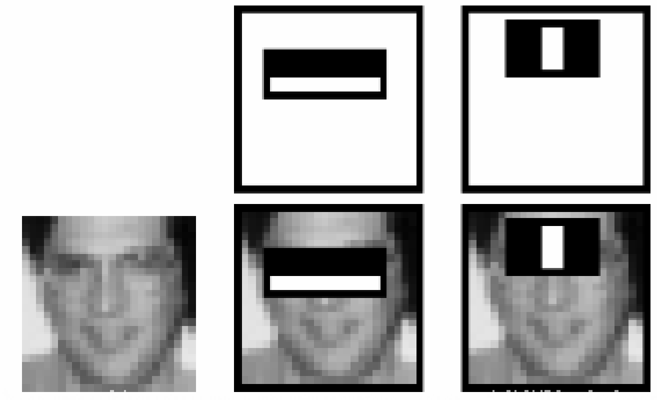


Figure 2.3: Haar Cascade Algorithm [13]

AdaBoost is a mechanism for cascading training of simple classifiers. By applying the AdaBoost learning algorithm, it can help reduce the number. The AdaBoost algorithm for feature selection and attention cascade can allocate computational resources to the image more efficiently [14]. Using symmetric AdaBoost can help produce linear combinations, which means that stability of positive and negative errors can be achieved using the AdaBoost algorithm.

2.3.2 Convolutional Neural Network Architecture

Convolutional neural networks (CNNs) have revolutionized face detection by automatically learning and extracting hierarchical features from raw images [15]. Unlike traditional approaches that rely on manually designed features and cascaded classifiers such as Viola-Jones algorithms, CNNs use multiple layers of convolutional filters to capture the spatial hierarchy in the data. This enables CNNs to recognize low-level features such as edges and textures in the initial layers, as well as more complex patterns such as facial structure and expression in deeper layers [15]. CNN methods typically involve training deep networks on large datasets of labeled images, enabling the network to learn complex patterns associated with faces. Recent advances, such as the development of architectures such as YOLO (You Only Look Once) and its variants, have further improved the speed and accuracy of CNN-based face detection, making it suitable for real-time applications.

CNNs are neural networks specialized for processing grid-like data such as images and videos [16]. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers as shown in Figure 2.4. Convolutional layers apply filters to detect local patterns while pooling layers down to sample the data to reduce its size and dimensionality while retaining key information. Activation functions introduce nonlinearity to the network, enabling it to learn more complex patterns. Common activation functions are ReLU and sigmoid functions. Fully connected layers perform classification tasks based on the extracted features. CNNs are generally well-suited for tasks such as classification of image, object detection, and image segmentation.

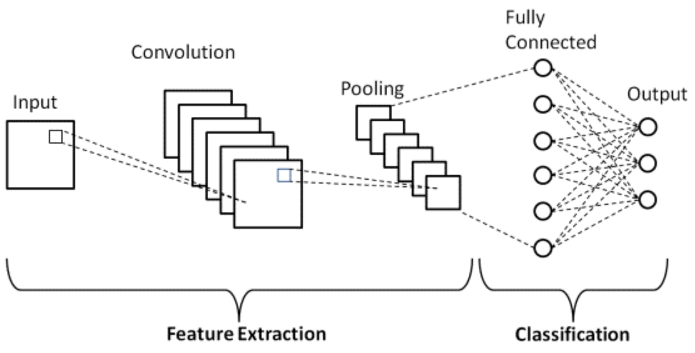


Figure 2.4: CNN architecture [16]

Despite the many advantages of CNNs, they also face some challenges in implementation. Training and running CNNs are computationally expensive and require powerful hardware resources to ensure adequate performance. CNNs are prone to overfitting, especially when working with small datasets. In addition, due to the complex internal representation of CNNs, it is difficult to understand how they make decisions, resulting in poor feature interpretability. Overall, convolutional neural networks are an excellent tool for solving problems such as image classification because they can learn and extract relevant features from images [15]. Despite the challenges, CNNs need to be studied and analyzed on various platforms to better understand and apply them to improve human life.

2.3.3 Skin Colour Technique

For skin colour detection, the process is implemented using colour space transformation. A binary image is obtained and the pixel region around the largest connected component is considered as the search region. A robust face detection that employs hybrid skin colour under different illuminations is proposed [17]. Using the International Commission on Illumination colour space, skin-like pixels and skin contours are detected using a finite threshold and the facial region is identified based on that threshold. The application of skin colour in face detection is straightforward as it identifies the largest connected component, which includes various features such as eyes, nose, and mouth. However, a significant limitation is that false detections may occur when the algorithm encounters a skin colour background or when the user wears skin colour clothes.

To improve the accuracy of unidentified photos, three colour spaces which are RGB, YcbCr, and HSI are combined. This combination made it possible to develop a new skin colour-based detection algorithm, which improved accuracy. Figure 2.5 shows how the combination conducted the skin colour-based face detection algorithm [18]. There are several disadvantages to using skin colour as a feature for face detection. The facial colour representation obtained by the camera can be affected by factors such as ambient light and object motion. Different types of cameras can also produce significantly different colour values. In addition, colour cues affect the algorithm's sensitivity to changes in lighting colour, such as RGB to lighting intensity.

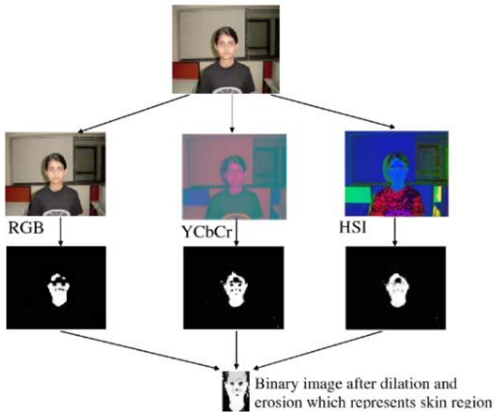

RGB
YCbCr
HSI
Binary image after dilation and erosion which represents skin region

Figure 2.5: Skin Colour Detection [18]

2.4 Drowsiness Signs

Detecting signs of drowsiness is a key factor in improving safety, especially in situations such as driving or operating heavy machinery. Monitoring eye activity has emerged as one of the most effective methods for detecting drowsiness. This method involves analyzing various eye parameters that indicate a person’s level of alertness. Based on the human face, many human behaviors can be measured as they contain rich information. This project focuses on detecting human behaviors related to signs of drowsiness based on the eyes and mouth as they provide key information for drowsy drivers and non-intrusive systems.

2.4.1 Eye Activities

The human eye is one of the main components of the human body and contains rich information that can be used to distinguish the degree of sleepiness. A study was conducted to determine the parameters suitable for detecting sleepiness and found that eye characteristics are one of the best indicators for detecting sleepiness [19]. Based on these eye characteristics, three eye activity parameters can provide important insights into the vigilance state of a person. The key parameters include blink rate, duration of eye closure (ECD), and percentage of eyelid closure (PERCLOS).

2.4.1.1 Eye Blink

Blinking is the action of opening and closing the eyelids at the same time and is one of the obvious signs of sleepiness. If a person is drowsy, he will blink more frequently than someone who is not sleepy [19]. Figure 2.6 shows the difference between the pupil colour and the white of the eye is the basis for detecting blinks [20]. The pupil is not a suitable parameter for detecting blinks due to its small size and easy occlusion. In addition, it may be occluded by light reflected from the glass. In terms of size, the iris is more suitable as a parameter for detecting blinks. Blink detection methods based on the iris area are easier to build even if they are occluded. The iris area comes from the aspect ratio of the iris bounding box. Blinks are observed according to the calculated ratio, and when the area is reduced, a blink of the eye is detected.

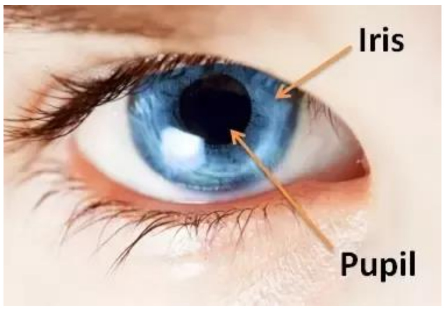


Figure 2.6: Eye [20]

2.4.1.2 Eye Closure Duration (ECD)

Eye closure duration, commonly referred to as ECD, is another parameter that can be used to indicate drowsiness [21]. ECD can be defined as the amount of time the eyes remain closed. Typically, if a person is drowsy, their eyes will remain closed longer than usual. Eye closure duration measures the time interval during each blink cycle that the eyes remain closed. Prolonged eye closure is a clear sign of drowsiness as it reflects difficulty keeping the eyes open. Systems used to detect drowsiness will often set an ECD threshold to trigger an alarm to alert the individual or operator. ECD is determined by applying equation 2.1.

$$ECD = Total_{EyeClose} \times Duration_{OneFrame} \qquad (2.1)$$

2.4.1.3 Percentage of Eyelid Closure (PERCLOS)

Blinking frequency and duration are reliable indicators of drowsiness. When people are drowsy, blinking frequency may decrease, and the duration of each blink increases. This is because drowsy people tend to have slower reaction times and reduced eye muscle control, resulting in longer blinks. Therefore, monitoring blinking patterns can provide real-time data on the onset of drowsiness. PERCLOS, or percentage of eyelid closure over the pupil over time, is the percentage of time that the eyes are closed for a specific period [22]. It is widely considered to be one of the most accurate indicators of drowsiness. High PERCLOS values are strongly associated with a higher risk of falling asleep, making it a key parameter in drowsiness detection systems. By continuously monitoring PERCLOS, these systems can issue timely warnings, helping to prevent accidents caused by drowsiness. Equation 2.2 shows how the PERCLOS is determined.

$$PERCLOS = \frac{Eye\ Close\ Time}{Total\ Time} \qquad (2.2)$$

2.4.2 Mouth Activities

In addition to eye activity, monitoring mouth activity can also provide valuable information about a person's drowsiness. Certain behaviors, such as yawning, are strongly associated with the onset of drowsiness and fatigue [21]. By referencing the mouth area, only one feature can be used to indicate a sign of sleepiness, which is yawning activity. Generally, there is a way to measure mouth opening to indicate yawning, mainly by tracking lip movement and quantifying the width of the mouth.

2.4.2.1 Yawning

Yawning is an involuntary mouth-opening movement that usually occurs when a person feels tired or sleepy. Moreover, yawning can be classified as an early sign of sleepiness before a person enters a full-blown sleepy mode. Yawning is one of the prominent parameters that point out signs of sleepiness that a person exhibits [21]. Yawning is a physiological response that usually indicates tiredness and decreased alertness. The frequency and duration of yawning can serve as a reliable indicator of sleepiness. An increased number of yawns is usually associated with the need for rest and decreased concentration and alertness. When the detection identifies frequent yawning, it can signal that the individual is becoming drowsy and may need to take a break or engage in activities that increase alertness. This approach is particularly useful in environments such as driving, where maintaining high alertness is crucial for safety.

2.4.3 Combination of Few Parameters Drowsiness Signs Detection

While individual parameters such as eye activity and mouth activity can provide valuable indicators of sleepiness, combining multiple parameters can improve the accuracy and reliability of drowsiness detection systems. By integrating data from a variety of sources, these systems can provide a more comprehensive assessment of an individual's alertness [23].

One effective approach is to simultaneously monitor blink rate, eye closure duration (ECD), percentage of eyelid closure (PERCLOS), and yawn frequency [23]. Each of these parameters provides unique information about the state of alertness. For example, increased eye closure duration and high PERCLOS values indicate prolonged eye closure, while frequent yawning indicates high levels of fatigue. By combining these indicators, the system can cross-validate the presence of sleepiness more strongly than relying on a single parameter.

Advanced drowsiness detection systems use machine learning algorithms to analyze the combined data in real-time. These algorithms can be trained on datasets containing a variety of sleepiness indicators, allowing them to identify complex patterns associated with sleepiness episodes. When multiple parameters reach their respective thresholds simultaneously, the system can issue a more reliable alarm, prompting the individual to take necessary actions to prevent accidents or errors. This multi-parameter approach greatly improves the effectiveness of drowsiness detection, ensuring timely and accurate responses in critical situations.

2.5 Field Programmable Gate Arrays

A field programmable gate array (FPGA) is an integrated circuit that combines reconfigurability with high performance. Unlike traditional processors such as application-specific integrated circuits (ASICs), FPGAs can be reprogrammed, allowing users to customize them for specific needs and applications. This flexibility makes FPGAs well-suited for a wide range of applications, enabling researchers to reuse them in different projects [24].

FPGAs are semiconductor devices built around a matrix of configurable logic blocks (CLBs) interconnected by programmable connections. FPGAs consist of multiple logic blocks that act as building blocks and are configured to perform specific logic functions as required by the algorithm [24]. Common components in FPGAs include lookup tables (LUTs), flip-flops, multiplexers, block RAMs (BRAMs), and digital signal processing (DSP) units. LUTs are essentially tables that produce outputs based on given inputs. Flip-flops (FFs) maintain the state of the chip, storing single bits of information. Multiplexers select one input from multiple inputs. BRAMs are used as memory within FPGAs to store large amounts of data, while DSP units handle complex mathematical calculations.

FPGAs are programmed using a hardware description language (HDL) such as Verilog or VHDL [24]. These languages enable designers to specify the desired functionality of a digital circuit, detailing how logic blocks should be interconnected and how data should flow through the system. The inherent parallelism in FPGA processing gives them superior performance in certain applications, providing speed and efficiency advantages over traditional processors such as CPUs and GPUs. However, programming and optimizing FPGAs can be difficult without a full understanding of their architecture, resource utilization, and timing constraints, all of which must be carefully configured for optimal performance.

2.5.1 High Level Synthesis

High-level synthesis (HLS), also known as C synthesis, is an automated process that converts abstract behavioral specifications of digital systems into register transfer level (RTL) structures that implement the specified behavior. HLS enables programmers to write algorithms in high-level programming languages such as C, C++, and Python. Through the HLS process, these high-level algorithms are converted into hardware description language (HDL) code that can then be used in FPGAs. This automation is critical because designing complex algorithms directly in HDL is a very complex task. HLS simplifies the implementation of complex models on FPGAs by automatically synthesizing high-level language descriptions into low-level HDL code.

Despite the many benefits of HLS, using HLS effectively requires considerable knowledge of the hardware architecture to be implemented. Designers must carefully manage data flow, memory usage, and performance constraints to ensure that the synthesized hardware performance is comparable to traditional computing platforms. Many FPGA vendors provide integrated HLS tools in their development environments. For example, Xilinx provides Vivado and Vitis as HLS tools for its FPGA boards.

In summary, high-level synthesis helps implement high-level language algorithms onto hardware platforms such as FPGAs that require low-level HDL coding. By automating the synthesis from high-level languages to low-level languages, HLS enables designers to implement complex algorithms on FPGAs more easily and efficiently.

2.5.2 PYNQ

PYNQ is an open-source project from AMD that aims to simplify the use of the Adaptive Compute Platform, as shown in Figure 2.7. By leveraging the Python language, Jupyter notebooks, and a broad ecosystem of Python libraries, PYNQ enables designers to harness the power of programmable logic and microprocessors to develop more advanced and innovative electronic systems [25]. PYNQ facilitates the creation of high-performance applications by enabling parallel hardware execution, high frame rate video processing, hardware-accelerated algorithms, real-time signal processing, high bandwidth I/O, and low latency control.

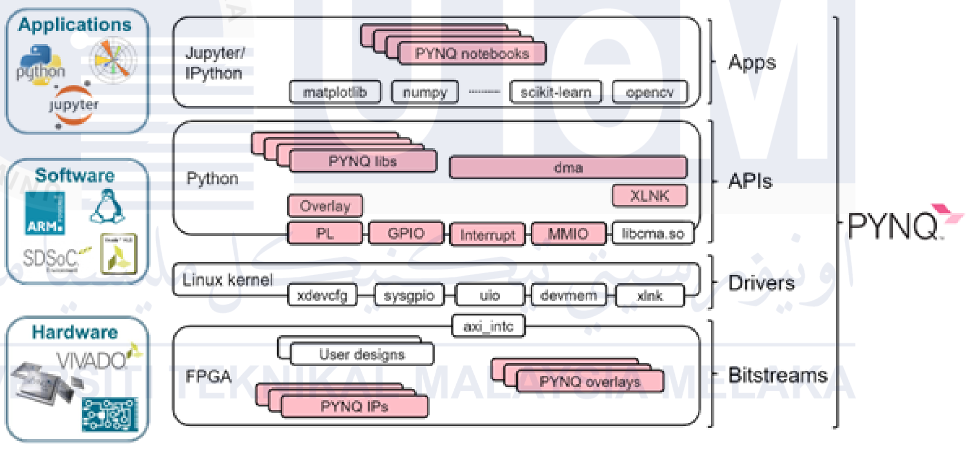


Figure 2.7: PYNQ Framework [25]

PYNQ is intended for a wide range of designers and developers, including software developers who want to exploit the capabilities of the Adaptive Compute Platform without resorting to traditional ASIC design tools, system architects seeking a user-friendly software interface and framework to rapidly prototype and develop Zynq, Alveo, and AWS-F1 designs, and hardware designers who want to make their designs accessible to the widest audience possible [25]. PYNQ is compatible with a wide range of AMD devices and boards, including Zynq 7000, Zynq UltraScale, Kria, Zynq RFSoC, Alveo Accelerator Board, and AWS-F1.

Jupyter Notebook is a browser-based interactive computing environment that allows users to create documents containing live code, interactive widgets, charts, explanatory text, equations, images, and videos. PYNQ-enabled boards can be programmed directly in Jupyter Notebook using Python. Developers can take advantage of hardware libraries and overlays on programmable logic that improve software performance on Zynq or Alveo boards and allow for customization of hardware platforms and interfaces [25]. By integrating these technologies, PYNQ makes it easier for developers to fully exploit the potential of adaptive computing platforms using a familiar and flexible programming environment.

2.6 Literature Review

After comparing all the researcher journals that had been using the architecture of drowsiness detection in their project, there were six related projects. This part would make a comparison of the algorithm and platform they were using, which is known as the drowsiness detection algorithm and FPGA platform in their research.

2.6.1 System-on-Chip Based Driver Drowsiness Detection and Warning System

In the research of this article, the project aims to develop a driver drowsiness detection system that combines high accuracy and low response time, using a cost-effective method suitable for implementation on a single processor system [26]. The initial implementation of the project used PERCLOS and CNN methods, achieving over 98% accuracy using the MobileNetV2 network. However, this setup was too slow, taking over 2 seconds to detect drowsiness on a single processor. The final system uses a combination of facial landmarks, haar cascade classifiers, and eye aspect ratio (EAR) methods to achieve a balance between speed and accuracy.

The optimized system detects driver drowsiness with 92% accuracy and 0.8 seconds of latency. The system consumes as little as 2W of power and can run all day using a power bank of over 10,000 mAh [26]. The main drawback is the performance degradation at night, which can be mitigated by using a night vision camera. The audio module of the PYNQ-Z2 board is used to implement the sound warning system. The system outperformed methods using SVM, random forest, and naive Bayes, with an accuracy only 2% lower than the neural network-based application. In conclusion, the developed driver fatigue detection system has high accuracy and efficiency, making it a viable option for practical applications. The integration of different detection methods ensures strong performance even on low-cost hardware, although nighttime performance still needs to be improved.

2.6.2 Heterogeneous FPGA-based System for Real-Time Drowsiness Detection

The project proposes an efficient hardware architecture to achieve real-time drowsiness detection by monitoring the driver's blinking behavior using the PERCLOS (Percentage of Eye Closure) metric [27]. The key features of the project include real-time detection, processing 250 VGA frames per second at low power consumption (1.6W) on a Xilinx Zynq XC7Z020 FPGA SoC. In terms of efficiency and performance, the proposed system is 33.3 times faster and occupies 2.6 times less area than the state-of-the-art system, enabling efficient integration into modern vehicles. In terms of hardware-software co-design, the project adopts a hybrid hardware-software approach to balance the computational load. Time-consuming tasks such as face detection and eye state monitoring are offloaded to dedicated hardware accelerators designed using high-level synthesis (HLS). In terms of pre-processing to improve accuracy, the system includes pre-processing steps such as

RGB to grayscale conversion and histogram equalization to improve detection accuracy under different lighting conditions.

As for face and eye detection, the architecture uses the Viola-Jones face detection algorithm and a novel eye state analysis method based on the standard deviation of the saturation channel in the HSV colour space [27]. The system calculates PERCLOS parameters by analyzing the eye closure rate over time to help determine the driver's drowsiness. In terms of robustness and adaptability, the design can adapt to different environmental conditions, such as lighting changes, by setting appropriate eye state detection thresholds during the calibration phase. Overall, this project demonstrates significant progress in the field of Advanced Driver Assistance Systems (ADAS) by providing a robust, efficient, and real-time solution to detect driver drowsiness, thereby improving road safety.

2.6.3 Design of ADAS Fatigue Control System using Pynq z1 and Jetson Xavier NX

This project aims to develop an Advanced Driver Assistance System (ADAS) that uses computer vision techniques to detect driver fatigue and drowsiness [28]. It utilizes two main platforms, Pynq Z1 and Jetson Xavier NX, using their capabilities for efficient processing and detection. The system consists of several steps, namely training a classifier for facial feature detection using Haar techniques, acquiring and processing images, detecting faces and using facial landmark algorithms, and finally analyzing the state of the driver's eyes to determine fatigue. Eye aspect ratio (EAR) is used to detect whether the eyes are open or closed, providing a basis for determining the driver's alertness. Various video resolutions were tested to evaluate the performance of the system on different platforms. Jetson Xavier NX showed superior performance and faster processing time compared to Pynq Z1 and CPU-based implementations [28]. For example, for a video resolution of 1024x768 pixels, Jetson Xavier NX is 18.75 times faster than Pynq Z1 and 2.62 times faster than the CPU implementation. This project successfully designed and implemented a robust and low-cost driver fatigue detection system on the Jetson Xavier NX and Pynq Z1 platforms. The system was proven to be effective in real-time detection with significant speed advantages and design flexibility.

2.6.4 Instruction Set Extension of a RiscV Based SoC for Driver Drowsiness Detection

This paper presents a driver drowsiness detection (DDD) system implemented using a modified RiscV processor on an FPGA [29]. The system uses a trained convolutional neural network (CNN) to classify four driver expressions, which are distracted, natural, sleeping, and yawning, achieving 81.07% accuracy on validation data. The RiscV processor is enhanced with three custom instructions (custom store, conv2d(2×2), and MAC) to increase computation speed. The latency of the modified processor is improved by 1.7 times compared to the base processor. Automotive companies have invested heavily in systems that detect drowsiness and alert drivers to prevent accidents. Neural networks, especially CNNs, are known for their high accuracy in classification tasks, making them well-suited for drowsiness detection. Implementing these networks on FPGAs offers a viable solution due to the adaptability and efficiency of this hardware.

Due to limited memory on low-cost FPGAs, the CNN model is designed with a small number of weights and biases [29]. Dynamic memory allocation is employed to efficiently manage intermediate variables, and custom instructions are used to improve performance. These optimizations ensure that the system can run on resource-constrained hardware without significantly reducing accuracy. This paper shows that extending the instruction set of the RiscV processor with custom instructions can significantly improve the performance of a CNN-based driver drowsiness detection system implemented on an FPGA. The approach operates within the memory constraints of low-cost FPGAs while balancing the need for accuracy and computational efficiency.

2.6.5 Real-Time Drowsiness Alert System from EEG Signal Based on FPGA

The project presents a comprehensive approach to detecting driver drowsiness using EEG signals processed in real-time on an FPGA platform [30]. The system uses EEG signals to monitor brain activity, capturing data indicating the driver’s alertness level. FPGAs, selected for their high-speed processing capabilities and flexibility, process these signals using complex algorithms designed to accurately detect drowsiness. The primary goal is to alert the driver in a timely manner, thereby preventing accidents caused by drowsiness. The implementation involves capturing EEG signals through electrodes placed on the driver’s scalp, which are then amplified and digitized for processing by the FPGA. The algorithm developed for this purpose classifies the EEG data into various states of alertness. This classification is achieved using machine learning techniques that are trained to recognize patterns in the EEG data that correspond to different levels of drowsiness.

The real-time nature of the system ensures that any signs of drowsiness are detected immediately, and an alert is issued to the driver. The project results demonstrate that the FPGA-based system is capable of high-precision real-time processing, making it a viable solution for integration into vehicles to improve safety [30]. Using FPGAs allows for a balance of speed and power efficiency, which is critical for in-vehicle applications. The study concluded that the system not only meets the performance requirements for real-time drowsiness detection but also provides a scalable and cost-effective solution for automotive safety systems.

2.6.6 A Fast FPGA Hardware Accelerator for Remote Heart Rate Detection Based on RGB Vision

This project focuses on developing a hardware accelerator to estimate heart rate from video recorded by an RGB camera using a FPGA [31]. The technology used is remote photoplethysmography (rPPG), which detects physiological signals by analyzing subtle colour changes on the skin surface. This FPGA-based implementation aims to significantly increase computational speed compared to traditional software methods, making it suitable for real-time applications such as heart failure early warning for athletes and driver drowsiness detection. The core of the system involves capturing images using an RGB camera, processing the data to extract the blood volume pulse, and then determining the heart rate. The process begins by selecting a region of interest (ROI) on the subject's face where the underlying physiological signals are strongest. The data is then preprocessed to remove noise, center, and whiten to facilitate independent component analysis (ICA). ICA helps separate the blood volume pulse from other signals by maximizing their statistical independence.

The transformed signal is then analyzed in the frequency domain to identify peaks corresponding to the heart rate. The FPGA implementation of the algorithm offers several advantages, including reduced computation time and improved accuracy [31]. The study showed that the system can achieve heart rate detection accuracy of -0.76 ±

5.09 bpm and -0.70 ± 8.71 bpm in short recording times of 16 seconds and 8 seconds, respectively. This performance exceeds previous methods and effectively combines speed and accuracy. The ability of FPGAs to process data makes it an excellent choice for applications that require immediate physiological monitoring and response.

2.6.7 Comparison of Related Literature Review

Table 2.1 compares 6 related studies, understands the differences with previous research results, and compares FPGA-based drowsiness detection.

Table 2.1: Literature Comparison

Board	Latency	Power	Method	Pros and Cons
FPGA (Xilinx PYNQ-Z2) [26], 2022	0.8ms in 640x480	2W	- Haar Cascade - EAR	Advantages Outperformed methods using SVM, random forest, and naive Bayes Disadvantages Performance degradation at night
FPGA (Xilinx ZYNQ XC7Z020) [27], 2022	250 fps in 640x480	1.6W	- Haar Cascade - PERCLOS	Advantages Low latency Disadvantages No Mouth Activity
FPGA (Xilinx PYNQ-Z1) [28], 2022	12s in 720x576	-	- Haar Cascade - EAR	Disadvantages High latency
Jetson Xavier NX [28], 2022	0.2s in 320x240			Advantages Low latency Disadvantages Eye Activity Only
FPGA (Xilinx Nexys 4 DDR) [29], 2022	231ms in 320x240	-	- CNN	Advantages 1.7 frame rate improvement Disadvantages High memory usage
FPGA (Xilinx) [30], 2021	-	1.116W	- EEG Signal	Advantages Low Power Consumption Disadvantages Intrusive method
FPGA (Intel Altera DE-10 Standard) [31], 2024	16s in 640x480	-	- Heart Rate Detect	Advantages RGB Vision Disadvantages High latency

2.6.8 Research Gap

The review “A High Performance and Robust FPGA Implementation of a Driver State Monitoring Application” by Christakos, P. et al. points out areas we can focus on to push the state of the art in drowsiness detection [32]. As mentioned in the paper, optimizing the drowsiness detection computation process is critical. This involves further extending and optimizing the rules to improve the robustness, and dynamically choosing the appropriate shape alignment ratio. In addition, access optimization requires more research on other data access methods and further improvements in hardware acceleration techniques. Proper management of scheduling and allocation issues, as detailed in the paper, can significantly improve the performance of FPGA implementations. Given the current research gaps in this area, it is worth further exploration.

2.7 Summary

The literature review comprehensively analyzes the existing research on drowsiness detection using FPGAs. It explores the basic concepts in depth, surveys related research, and identifies gaps in current research. This background study places the research in the broader field of hardware acceleration, highlighting the rationale for choosing FPGAs. This chapter lays the foundation for the objectives of the paper, which include FPGA-based drowsiness detection design, optimization exploration, and comprehensive performance analysis.

Among the reviewed techniques, behavior-based measurements stand out as the most effective method due to their user-friendliness and cost-effectiveness. Driver drowsiness is mainly indicated by eye, mouth, and head behaviors. Advanced drowsiness signs such as nodding and turning your head indicate that the driver may be extremely drowsy or has entered a sleep mode. Therefore, eye and mouth behaviors are selected as early and obvious signs of drowsiness in this project.

Eye feature indicators include blinking, eye closure time (ECD), and percentage of eyelid closure (PERCLOS). Yawning is an indicator of mouth features. According to previous studies, these four indicators are most suitable for detecting sleepiness. A convolutional neural network feature-based method is adopted to analyze these indicators and use the parallel computing power of FPGA to accelerate the algorithm.

CHAPTER 3

METHODOLOGY

3.1 Introduction

To achieve the research objectives, the methodology used in this study includes the comprehensive development and implementation of a convolutional neural network based on drowsiness detection. The methodology emphasizes the fusion of machine learning and hardware acceleration and aims to optimize the deployment of trained models on FPGAs and leverage the capabilities of the PYNQ-Z1 board for drowsiness detection. The subsequent chapters will step through the processes involved in data preparation, model training, high-level synthesis (HLS), FPGA implementation, fatigue detection, verification, and performance evaluation. The methodology will describe the implementation of the project. The first step is to select the appropriate dataset. The next step is to train two convolutional neural network models with different weight quantization. Next, after validating the two models, they are exported into model types that can be recognized by HLS. After that, HLS is used to convert the two models into HDL in preparation for the implementation of the models on the FPGA. After generating the bitstream of the models, the bit files of the two models are transferred to the FPGA, and the drowsiness detection algorithm is added for test execution verification. A thorough analysis will be performed to gain a deep understanding of the FPGA.

3.2 Project Planning

The flowchart of this project is shown in Figure 3.1. The process starts with conducting background research and a comprehensive literature review. After this, a dataset is selected. Two models with different weights are trained on Google Colab using the WIDER FACE dataset. These models are then tested using the same dataset. The model configuration is adjusted until the mean average precision (mAP) for both training and testing exceeds 0.3. This threshold is chosen because the Yolov3-Tiny model also achieves a mAP of about 0.3 when quantized to 416×416 8-bit integers. Specifically, for the Yolov3-Tiny model in Xilinx Vitis, the mAP drops from 0.362 to 0.296 after quantization [33]. After achieving the desired mAP, the model is exported to the ONNX format to be compatible with the FINN library. The FINN library runs in a Docker container, which is installed on an Ubuntu system on the host. The advantage of using a Docker container is that it provides a self-contained package capable of running all necessary applications. The software can be easily installed on the host by running script commands.

After that, both the models are HLS converted using the FINN library. The HLS conversion and configuration steps optimize the hardware implementation of the models. Next, simulation and synthesis are performed to test the functionality of the models before the actual deployment. After the synthesis process runs without any critical errors, bitstream generation is performed. The models are then deployed on the PYNQ-Z1 board by transferring the bit file, hardware configuration files, and drivers to the board and verifying them. After that, the drowsiness detection algorithm is added and verified using the yawning detection dataset, and the process continues if the latency remains at or below 200 milliseconds. If not, adjustments are made in the previous steps and retested. All the processes are logged and analyzed to get a report on the implementation and accuracy. These steps provide an in-depth understanding of the drowsiness detection implementation on the PYNQ-Z1 FPGA board.

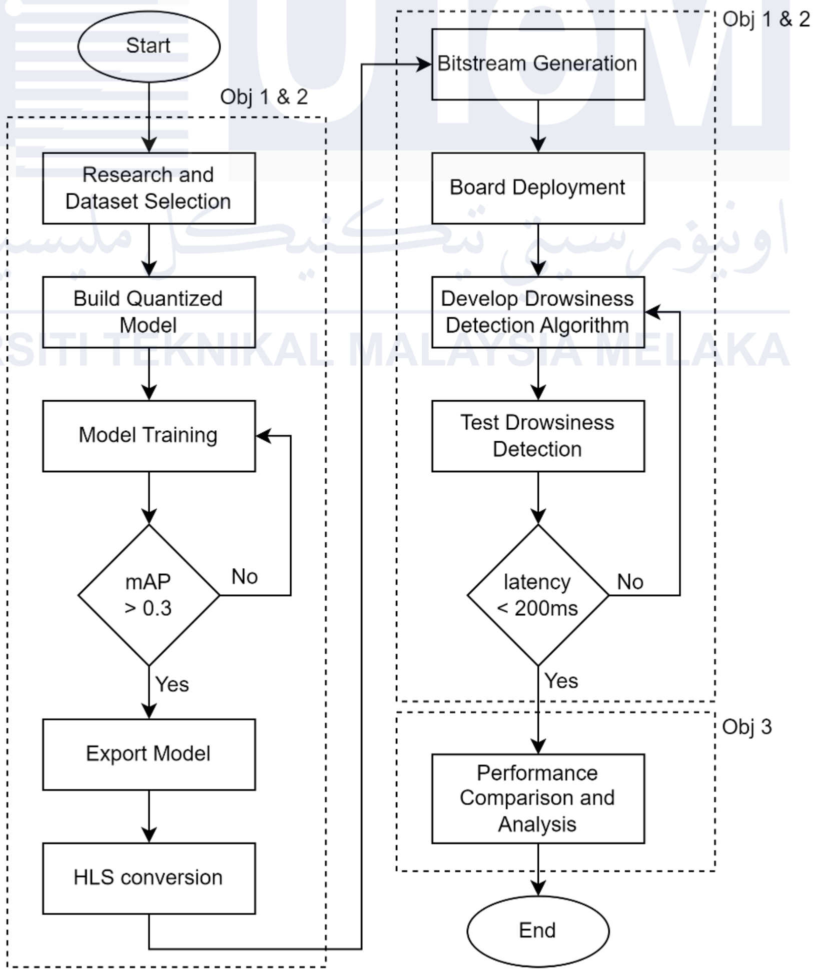


Figure 3.1: Flow Chart

3.2.1 Dataset Selection

The WIDER FACE dataset is a good dataset for performing image recognition and classification tasks using convolutional neural networks [34]. The reason why the WIDER FACE dataset is suitable for the task is that it contains 60 event categories covering a wide range of real-world scenarios, which makes this dataset relevant for real-world applications, such as face detection of car drivers. The data quality of the WIDER FACE dataset is the key to its strength as it exhibits a high degree of variability in scale, pose, occlusion, expression, makeup, and lighting. The data quality that is closest to the real-world environment ensures that the model can adapt to images captured in real time. WIDER FACE is a large-scale face detection dataset that is often used as a benchmark. If a model is successfully implemented on WIDER FACE, it shows the capabilities of the model. Figure 3.2 shows the data in the WIDER FACE dataset and its labels.

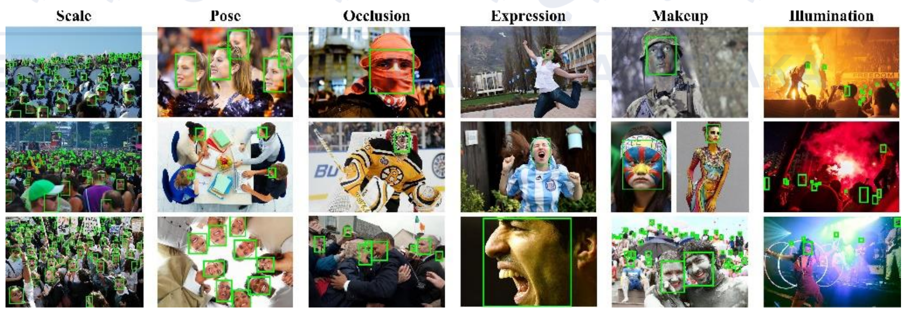


Figure 3.2: WIDER DACE Dataset [34]

This project requires testing several scenarios to investigate the accuracy and speed of drowsiness detection implemented in FPGA. Six different video files from YawDD are used to investigate the accuracy and performance of drowsiness detection in FPGA. YawDD is a video dataset recorded by an onboard camera of drivers in real cars with various facial features, such as male and female, with and without glasses or sunglasses, different ethnicities, and in situations of talking, singing, silence, and yawning [35]. It is primarily used to develop and test algorithms and models for yawn detection but can also be used to recognize and track faces and mouths. The videos are captured under natural and varying lighting conditions as shown in Figure 3.3. The camera is mounted on the driver's dashboard. The set of videos provides different scenarios, each containing silent driving, driving while talking, and driving while yawning.

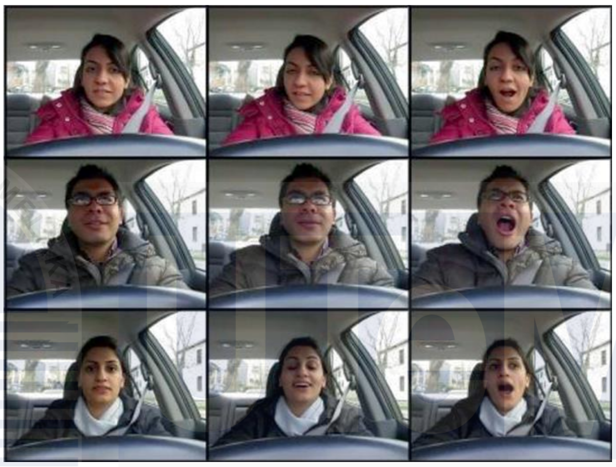


Figure 3.3: Yawning Detection Dataset (YawDD) [35]

3.2.2 FINN

FINN is a Python package for FPGA machine learning inference [36]. The library uses a high-level synthesis language (HLS) to create firmware implementations of machine learning algorithms. It can be used to convert traditional open-source machine learning package models to HLS and can be configured for desired situations based on user needs. Figure 3.4 shows the hardware generation of the FINN compiler.

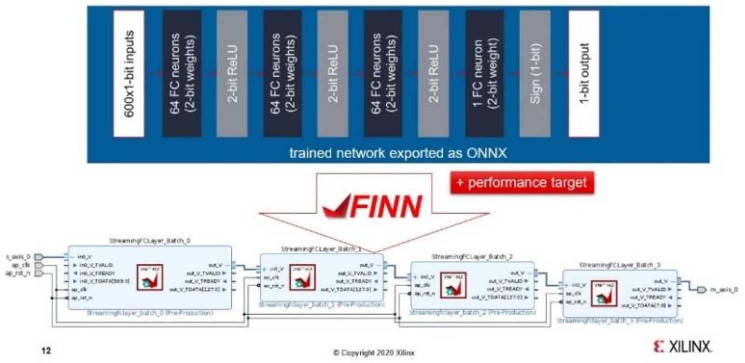


Figure 3.4: FINN Compiler [36]

With the help of open-source tools such as PyTorch and Brevitas, machine learning models can be quickly and efficiently converted to high-level synthesis (HLS) code, which can then be translated and executed on FPGAs. Subsequently, HLS projects can be used to generate IP that can be integrated into complex designs or used to develop cores for co-processing CPUs. Users are free to define many parameters of the algorithm to best meet their needs. The FINN package can quickly prototype machine learning algorithms in FPGAs, greatly reducing the time required to obtain results [36]. It also provides users with guidance on how to design the best machine learning algorithm for their application while balancing latency, resource consumption, and performance requirements. FPGAs can be specifically programmed to perform a certain task, in this case, evaluating a neural network given a set of inputs, and can therefore be highly optimized for that task through tricks such as pipelining and parallel evaluation. However, this means that dynamic remapping at runtime is effectively impossible.

FPGAs also typically have a relatively low power cost compared to CPUs and GPUs. This enables FINN to build HLS code from compressed neural networks, achieving latency predictions in the microsecond range. The FINN tool saves the time investment required to convert neural networks into hardware design languages or even HLS code, allowing for rapid prototyping. In summary, FINN is a bridge between advanced machine learning model development and efficient FPGA implementation. By automating the translation process and handling FPGA-specific optimizations, FINN simplifies the deployment of machine learning models on hardware, providing a valuable tool for developers seeking to take advantage of FPGA acceleration.

3.2.3 PYNQ-Z1 Board Implementation

With the new open-source framework PYNQ, embedded programmers can harness the power of the Xilinx PYNQ-Z1 development board to create programmable logic circuits. Python is used to program the PYNQ-Z1 and is used to test and develop the code. The process of importing and programming programmable logic circuits is very similar to that of software libraries, which are imported as hardware libraries and programmed through their respective APIs. The PYNQ development board uses the Zynq system-on-chip, which combines multiple functions in a single chip but still can use multiple chips on the board to perform the same desired function.

An ARM processor is also included in the Zynq SoC. This makes it possible to implement hardware acceleration of CPU, DSP, and other components on the same chip or board. The flexibility of the PYNQ board is also an advantage since it can reprogram the SoC as needed. In summary, the PYNQ-Z1 FPGA board was chosen because of its flexibility, reprogramming ability, and lower challenges compared to other FPGAs. The PYNQ image and its built-in Python interface with Jupyter Notebook in the Zynq SoC provide a huge advantage for implementing the drowsiness detection for this project on board. The physical board is shown in Figure 3.5.

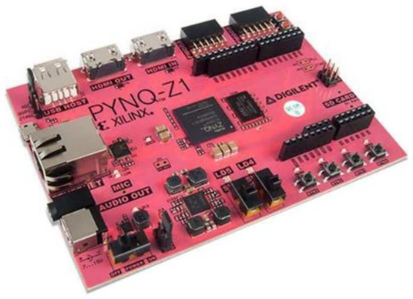


Figure 3.5: PYNQ-Z1 FPGA [25]

3.3 Model Preparation

Model preparation for our project involves four key steps to ensure efficient, accurate detection optimized for FPGA deployment. In terms of model selection, we chose YOLOv3-tiny for face detection because of its smaller architecture, fewer convolutional layers, and parameters. This choice strikes a balance between computational efficiency and accuracy, making it suitable for limited computing resources.

For model quantization, the model was quantized using the Brevitas library to apply 2-bit and 6-bit weights to each convolutional layer. This step reduces the size and computational requirements of the model, further improving efficiency. Both versions of the quantized model are trained to learn the features required for accurate face detection. Finally, the models are validated to ensure that they achieve the required mean average precision (mAP). After validation, the models are exported to ONNX format, enabling the FINN library to convert them to high-level synthesis (HLS) to generate FPGA bitstreams.

3.3.1 Model Selection

Yolov3-Tiny is a simplified version of the Yolov3 object detection CNN model designed for real-time applications with limited computational resources as shown in Table 3.1. It uses a smaller architecture with fewer convolutional layers and parameters, making it faster and more efficient while still maintaining reasonable accuracy. Yolov3-Tiny is particularly well suited for applications such as face detection that require fast and accurate responses.

The max pooling layer, or max pooling, is a down sampling operation that reduces the dimensionality of each feature map while retaining the most important information. It does this by sliding a window over the input feature map and selecting the maximum value within the window. This process reduces the spatial size of the feature map, which reduces the number of parameters and computations in the network and helps control overfitting. The ReLU (Rectified Linear Unit) layer introduces nonlinearity to the CNN. It applies the ReLU activation function to each element in the feature map, setting all negative values to zero and leaving positive values unchanged. This activation function helps the network learn complex patterns and relationships by introducing nonlinearity to the model. ReLU is computationally efficient and helps alleviate the vanishing gradient problem during training.

Table 3.1: Yolov3-Tiny Architecture

Layer Type	Filters	Size	Input	Output	Activation
Conv	16	3 x 3 / 1	416 x 416 x 3	416 x 416 x 16	ReLU
MaxPool		2 x 2 / 2	416 x 416 x 16	208 x 208 x 16	-
Conv	32	3 x 3 / 1	208 x 208 x 16	208 x 208 x 32	ReLU
MaxPool		2 x 2 / 2	208 x 208 x 32	104 x 104 x 32	-
Conv	64	3 x 3 / 1	104 x 104 x 32	104 x 104 x 64	ReLU
MaxPool		2 x 2 / 2	104 x 104 x 64	52 x 52 x 64	-
Conv	128	3 x 3 / 1	52 x 52 x 64	52 x 52 x 128	ReLU
MaxPool		2 x 2 / 2	52 x 52 x 128	26 x 26 x 128	-
Conv	256	3 x 3 / 1	26 x 26 x 128	26 x 26 x 256	ReLU
MaxPool		2 x 2 / 2	26 x 26 x 256	13 x 13 x 256	-
Conv	512	3 x 3 / 1	13 x 13 x 256	13 x 13 x 512	ReLU
MaxPool		2 x 2 / 2	13 x 13 x 512	13 x 13 x 512	-
Conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x 1024	ReLU
Conv	256	1 x 1 / 1	13 x 13 x 1024	13 x 13 x 256	ReLU
Conv	512	3 x 3 / 1	13 x 13 x 256	13 x 13 x 512	ReLU
SimpleConv	255	3 x 3 / 1	13 x 13 x 512	13 x 13 x 255	Sigmoid

3.3.2 Model Quantization

During the model quantization step, the Convolutional Neural Network (CNN) model was modified several times to optimize its efficient execution on FPGA hardware. The key changes include reducing the number of cores and using integer quantized forms of convolutional layers and activation functions. The number of filters in each convolutional layer was reduced by five times. This significant reduction helps reduce the complexity and computational load of the model, making it more suitable for resource-constrained environments such as FPGAs. The model uses *QuantConv* and *QuantReLU* layers, which are integer quantized versions of traditional convolutional and activation functions as shown in Table 3.2. These quantized layers replace floating-point operations with integer operations, which are more efficient and faster to compute on FPGA hardware.

The CNN model accepts an input image of size *416x416x3* and outputs a *13x13x18* result. This output represents a *13x13* grid of the image, where each grid cell contains the center *x* and *y* coordinates, width and height, category information, and the confidence score of the detected object. These six outputs are calculated for three different anchor boxes, enabling the model to detect objects with different aspect ratios. After the results are calculated, the resulting matrix is used to visualize the bounding boxes on the image. Non-maximum suppression (NMS) is then applied to eliminate overlapping bounding boxes, retaining only the most confident ones to provide clear and accurate detection output.

To strike a balance between efficiency and accuracy, the first and last convolutional layers are preferably used with 8-bit integer values. This adjustment helps maintain better accuracy where precision is most critical. The Brevitas library in PyTorch is used to implement these quantized layers and activation functions, providing the necessary tools to convert floating point operations into efficient integer operations suitable for FPGA deployment. This quantization process ensures that the model remains fast and efficient without significantly reducing accuracy.

Table 3.2: Quantized Yolov3-Tiny Architecture

Layer Type	Weights	Filters	Size	Input	Output	Activation
QuantConv	8 bits	8	3 x 3 / 1	416 x 416 x 3	416 x 416 x 8	QuantReLU
MaxPool			2 x 2 / 2	416 x 416 x 8	208 x 208 x 8	-
QuantConv	2/6 bits	8	3 x 3 / 1	208 x 208 x 8	208 x 208 x 8	QuantReLU
MaxPool			2 x 2 / 2	208 x 208 x 8	104 x 104 x 8	-
QuantConv	2/6 bits	16	3 x 3 / 1	104 x 104 x 8	104 x 104 x 16	QuantReLU
MaxPool			2 x 2 / 2	104 x 104 x 16	52 x 52 x 16	-
QuantConv	2/6 bits	32	3 x 3 / 1	52 x 52 x 16	52 x 52 x 32	QuantReLU
MaxPool			2 x 2 / 2	52 x 52 x 32	26 x 26 x 32	-
QuantConv	2/6 bits	56	3 x 3 / 1	26 x 26 x 32	26 x 26 x 56	QuantReLU
MaxPool			2 x 2 / 2	26 x 26 x 56	13 x 13 x 56	-
QuantConv	2/6 bits	104	3 x 3 / 1	13 x 13 x 56	13 x 13 x 104	QuantReLU
MaxPool			2 x 2 / 2	13 x 13 x 104	13 x 13 x 104	-
QuantConv	2/6 bits	208	3 x 3 / 1	13 x 13 x 104	13 x 13 x 208	QuantReLU
QuantConv	2/6 bits	56	3 x 3 / 1	13 x 13 x 208	13 x 13 x 56	QuantReLU
QuantConv	2/6 bits	104	3 x 3 / 1	13 x 13 x 56	13 x 13 x 104	QuantReLU
QuantSimpleConv	8 bits	18	3 x 3 / 1	13 x 13 x 104	13 x 13 x 18	QuantHardTanh

3.3.3 Model Training

The model training process is a critical step in preparing the CNN model for accurate face detection. The training uses a dataset of 3.6k images from the WIDERFACE dataset, which is well-known for collecting images of faces in various scenarios. The dataset is split into 90% for training and 10% for validation. The model is trained for 120 epochs. Each epoch represents a complete pass over the entire training dataset. The batch size used is 128, which means 128 images are processed before updating the model parameters. This batch size helps balance computational efficiency and training stability.

During training, the model learns to detect faces by adjusting its parameters to minimize the difference between the predicted output and the actual face locations and classes in the training images. The training process involves forward propagation (computing the model's predictions) and backward propagation (updating the model's parameters based on the prediction errors). After each epoch, the model's performance is evaluated on the validation set. This helps monitor the model's generalization ability and prevents overfitting by ensuring that the model performs well on unseen data.

The model is trained using quantized values. Quantization involves representing the weights and activations with lower bit widths, which significantly reduces the computational and memory requirements. This step is crucial for ensuring that the model runs efficiently on FPGA hardware without sacrificing too much accuracy. Once training is complete, the model is saved as a *.pt* file (PyTorch model file). This file contains the learned parameters of the model and can be used later for inference. This training process ensures that the model is fully prepared to accurately detect faces in a variety of images using the robust features learned from the WIDERFACE dataset.

3.3.4 Model Validation

The model validation process is critical to ensure that the trained model performs well on unseen data and is optimized for deployment. Once the validation is completed, the trained model is exported in the ONNX (Open Neural Network Exchange) format. ONNX is an open-source format designed for representing machine learning models, providing an intermediate representation that facilitates interoperability between different frameworks. Exporting the model to ONNX is a key step as it allows the model to be further processed and optimized by the FINN framework. FINN is a framework developed by Xilinx for accelerating quantized neural networks on FPGAs.

During training, the Sigmoid function is used as the activation function for the last layer. The Sigmoid function is beneficial for learning because it smoothly maps the input values to a range between 0 and 1, which is ideal for probability predictions in classification tasks. For deployment, particularly for lowering latency and improving compatibility with the FINN framework, the Sigmoid function is replaced with a rescaled HardTanh function. HardTanh is a piecewise linear approximation of the Tanh function, which is computationally less expensive and thus reduces latency. The relationship between Tanh and Sigmoid function can be defined as equation 3.1.

$$\sigma(x) = \frac{1+\tanh\left(\frac{x}{2}\right)}{2} \qquad (3.1)$$

3.4 Board Implementation

Each step of the board level implementation of this project is attached in this section. First, the quantized model is converted to HLS using the FINN framework. Next, the bitstream is generated to port the detection accelerator on the FPGA. At this stage, the output folder will contain two subfolders called the bit file folder and the driver folder for porting the bitstream accelerator on the FPGA. The output folder will also contain the reports generated by FINN and Vivado. The bit file is deployed on the FPGA and compiled with the drowsiness detection algorithm. Blinks and yawns are calculated using the eye aspect ratio and mouth aspect ratio.

3.4.1 High-Level Synthesis Conversion

The FINN library is utilized to convert and compile a quantized convolutional neural network model into a hardware description language (HDL) representation suitable for FPGA (Field-Programmable Gate Array) deployment. The process of High-Level Synthesis (HLS) conversion for the quantized model involves several key steps to transform the trained neural network model into an FPGA-compatible format using the FINN framework. Initially, the model undergoes a series of transformations to prepare it for synthesis. These transformations include inferring shapes and data types, folding constants, and assigning unique and readable names to tensors and nodes in the computational graph. Specifically, the model is transformed using functions such as `*InferShapes*`, `*FoldConstants*`, `*GiveUniqueNodeNames*`, `*GiveReadableTensorNames*`, `*InferDataTypes*`, and `*RemoveStaticGraphInputs*`.

Next, preprocessing steps are integrated into the model. Using the `*ToTensor*` function from the `*finn.util.pytorch*` module, the preprocessing step is designed to normalize input images by dividing *uint8* inputs by 255. This preprocessing model is exported in ONNX format and merged with the core quantized model using the `*MergeONNXModels*` transformation. Additionally, an input quantization annotation is added, specifying that the input data type should be *UINT8* for compatibility with the models. The prepared model is saved and re-validated through repeated transformations to ensure all shapes, constants, and data types are correctly inferred and optimized. Finally, the model is saved in the ONNX format suitable for further processing.

3.4.2 Simulation, Synthesis and Bitstream Generation

Simulation and synthesis are performed using the command build from the FINN library. Estimated resource utilization reports are generated for analysis. The configuration of FINN is slightly different from the synthesis part, which uses the Vivado accelerator as the backend, while the synthesis process uses Vivado as the backend.

The next phase involves setting up the configuration for dataflow build using the FINN framework. The `*DataflowBuildConfig*` is defined with various parameters, including the output directory, the folding configuration file specific to the Pynq-Z1 board, and synthesis clock period settings. The build configuration specifies the target FPGA board and the desired output types, such as estimation reports, bit files, PYNQ drivers, and a deployment package.

The `*build_dataflow_cfg*` function is called with the model file and the configuration, initiating the conversion process. This step compiles the neural network into a hardware-friendly representation, generating the necessary files and reports for deploying the model on FPGA hardware, thus enabling efficient and high-speed detection using the model on the PYNQ-Z1 board.

3.4.3 Deployment and Validation

Download the image of the board and load it into the SD card so that the PYNQ-Z1 board can work. After that, the driver can be used for drowsiness detection inference for analysis. The accuracy of yawning and blinking on the board is being compared with that of the host computer.

3.4.4 Eye State Determination

For the blink detection method, Eye Aspect Ratio (EAR) algorithm will be used. EAR is defined as the ratio of the height and width of the eye [37]. First, extraction of the eye region from a set of facial landmarks. Based on Figure 3.6, there are six coordinates for each eye. These six coordinates will be used for calculation of EAR value. The calculation is done for both left and right eye.

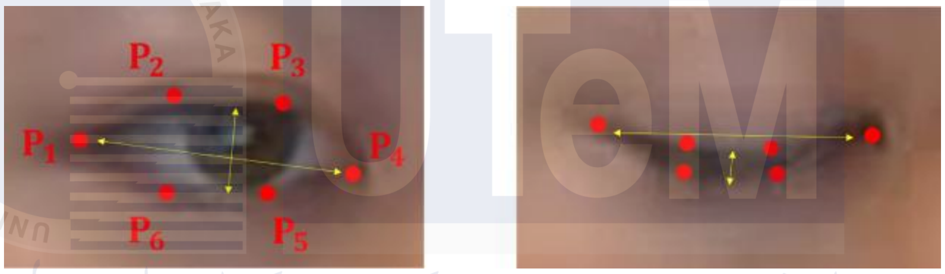


Figure 3.6: Eye Aspect Ratio (EAR) [37]

The equation 3.2 for EAR can be derived as the following:

$$EAR = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2 \times \|P_1 - P_4\|} \quad (3.2)$$

Where P1, P2, P3, P4, P5 and P6 are the facial landmark coordinates that have been obtained. Next, the system calculates the average of two EAR together with assumption that a person blinks both eyes at the same time. EAR value will be compared with the threshold value taken as 0.2. If the EAR value is below the threshold, the eye will be considered closed [37]. When an eye is closed, the two types, which are eye closure and eye blink will be differentiated. When the duration of eye closed is more than 0.2 seconds, it will be considered as eye closure or else as eye blink.

3.4.5 Mouth State Determination

For the yawn detection method, Mouth Aspect Ratio (MAR) algorithm will be used. MAR is defined as the ratio of the height and width of the mouth. First, extraction of the mouth region from a set of facial landmarks. To calculate the MAR, only those coordinates will be used at which are at the outer mouth [38]. There are 12 coordinates as shown in Figure 3.7.

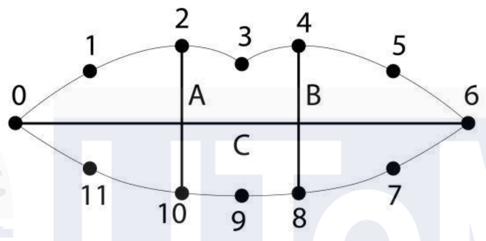


Figure 3.7: Mouth Aspect Ratio (MAR) [38]

The equation 3.3 for MAR can be derived as the following:

$$MAR = \frac{\|P_2 - P_{10}\| + \|P_4 - P_8\| + \|P_0 - P_6\|}{3} \quad (3.3)$$

Where P1 to P12 are the facial landmark coordinates that this study obtained before. MAR value will be compared with the threshold value taken as 20. The value of threshold value was established by trial and error, with several values of threshold value being tested to ensure that the algorithm accurately classifies an instance of yawning and closed mouth. It shows that if the MAR value is bigger than threshold value, the mouth will be considered as yawning [38].

3.5 Performance Analysis

Performance analysis of a drowsiness detection system on an FPGA focuses on several key metrics to assess its efficiency and effectiveness. Measuring inference time provides insight into the speed of the system, specifically its latency in milliseconds per frame. Analyzing the resource utilization of the FPGA helps determine how efficiently the hardware resources, including logic elements, memory, and DSP blocks, are being used. Evaluating power consumption is critical to understanding the energy efficiency of the FPGA implementation. Evaluating the accuracy of detecting blinks and yawns ensures the reliability of the drowsiness detection system. This metric is critical to verifying the usefulness of the system in real-world scenarios. By examining these metrics, the performance analysis aims to gain a detailed understanding of the capabilities and advantages of the FPGA-based drowsiness detection system over traditional computing platforms.

3.5.1 Inference Time

Measure and discuss the inference time of the deployed model. In the context of FPGA deployment, the inference time is a critical metric reflecting the speed at which the model processes input data and produces classification results. The FPGA’s parallel processing capabilities are harnessed to optimize inference time, and measurements are taken to quantify the reduction achieved compared to a purely software-based implementation. This metric is critical to understanding the real-time capabilities of an FPGA-based system

3.5.2 Resource Utilization

Assess the FPGA resource utilization for the implemented system. Resource Utilization is the number of resources used by an FPGA for the design, in my project, it is the HLS quantized convolutional model. The aspects that are often considered are Lookup table, Digital Signal Processing (DSP), Flip Flops (FF), Block RAM and I/O block. They are affected by the architecture of the model and the designation of the model and the HLS conversion technique. This evaluation is critical to optimizing the system and ensuring that it fits within the constraints of the target FPGA device.

3.5.3 Power Consumption

Analyze the power consumption of the deployed system. Power Consumption is the power needed for the FPGA to run the model. It is usually affected by the function and the frequency of the FPGA running. The design complexity and operating condition can also change the power consumption of FPGA.

3.5.4 Blink and Yawn Count

Analyze the system's performance in detecting signs of drowsiness and assess the system's effectiveness in monitoring eye and mouth activity. Accurate detection of blinks and yawns are important indicators of an individual's drowsiness. By tracking the frequency and duration of blinks and yawns, a drowsiness detection system can measure alertness or fatigue levels in real time. Blink frequency and duration are important indicators because prolonged blinks or increased intervals between blinks are often associated with drowsiness. Similarly, yawn detection can provide valuable insights into an individual's physiological state, with frequent or prolonged yawning indicating a higher likelihood of drowsiness. Accurate counting and analysis of blinks and yawns helps improve the overall effectiveness and reliability of the drowsiness detection system, enabling timely intervention to prevent potential incidents or errors caused by reduced alertness.

3.5.5 Comparison with Traditional Platform

Compare the performance with traditional platforms such as CPU. To visualize the advantages and limitations of FPGA comparing to the CPU, the comparison between them needs to be made and analyzed for gaining insight on both platforms implementing the same model performing the same classification task. Comparing the performance of an FPGA-based system to traditional platforms such as CPUs can highlight advantages and potential trade-offs. This comparison provides a comprehensive view of the advantages in terms of speed, power efficiency, and resource utilization.

3.5.6 Trade-offs and Optimization Strategies

Discussing trade-offs and optimization strategies provides insight into the design choices made to balance performance, accuracy, and resource constraints. This analysis helps identify potential improvements and best practices for future implementations.

CHAPTER 4

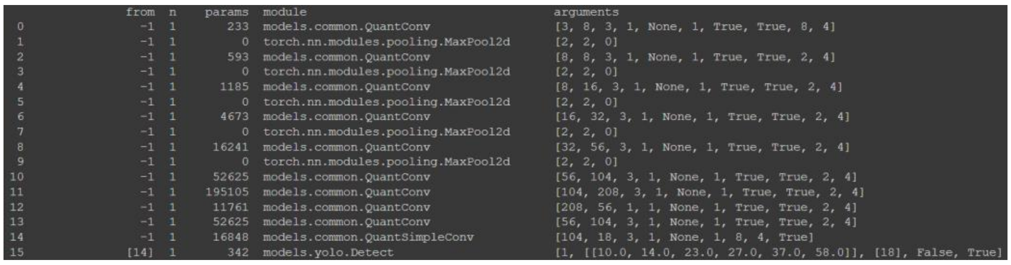
RESULTS AND DISCUSSION

4.1 Introduction

This chapter will record and measure all the results and output data of this project as data analysis. This section will discuss the latency, power consumption, and resource utilization of drowsiness detection. The performance of the two models in PYNQ-Z1 FPGA will also be presented in the form of results.

4.2 Two-bit Quantization Model

This section studies the performance of the 2-bit quantized model implemented on the FPGA platform shown in Figure 4.1. The analysis focuses on resource utilization, power consumption, and adherence to clock constraints and frequency. Through a comprehensive evaluation, we aim to gain insight into the feasibility and effectiveness of deploying low-precision models on FPGA platforms, thereby facilitating the development of efficient and scalable machine learning solutions.

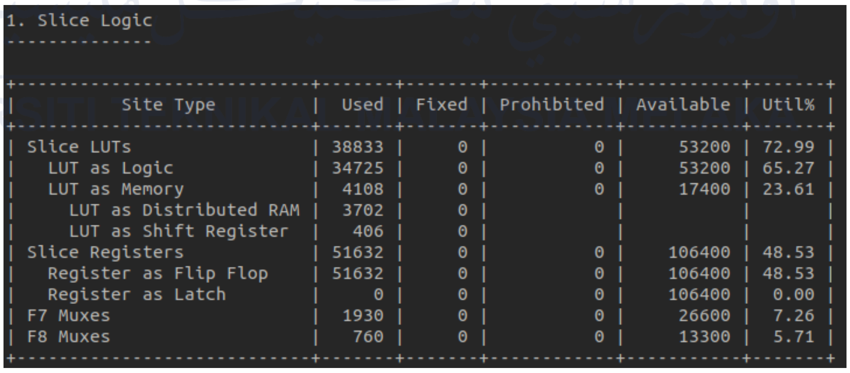


```
                 from  n    params  module                                  arguments
  0                -1  1       233  models.common.QuantConv                 [3, 8, 3, 1, None, 1, True, True, 8, 4]
  1                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  2                -1  1       593  models.common.QuantConv                 [8, 8, 3, 1, None, 1, True, True, 2, 4]
  3                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  4                -1  1      1185  models.common.QuantConv                 [8, 16, 3, 1, None, 1, True, True, 2, 4]
  5                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  6                -1  1      4673  models.common.QuantConv                 [16, 32, 3, 1, None, 1, True, True, 2, 4]
  7                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  8                -1  1     16241  models.common.QuantConv                 [32, 56, 3, 1, None, 1, True, True, 2, 4]
  9                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
 10                -1  1     52625  models.common.QuantConv                 [56, 104, 3, 1, None, 1, True, True, 2, 4]
 11                -1  1    195105  models.common.QuantConv                 [104, 208, 3, 1, None, 1, True, True, 2, 4]
 12                -1  1     11761  models.common.QuantConv                 [208, 56, 1, 1, None, 1, True, True, 2, 4]
 13                -1  1     52625  models.common.QuantConv                 [56, 104, 3, 1, None, 1, True, True, 2, 4]
 14                -1  1     16848  models.common.QuantSimpleConv           [104, 18, 3, 1, None, 1, 8, 4, True]
 15              [14]  1       342  models.yolo.Detect                      [1, [[10.0, 14.0, 23.0, 27.0, 37.0, 58.0]], [18], False, True]
```

Figure 4.1: 2-bit Quantized Model

4.2.1 Resource Utilization

This section presents the implementation report of the resource utilization of the Vivado implementation of the 2-bit quantized YOLOv3-Tiny model on the PYNQ-Z1 FPGA board. The resource utilization is shown in Figures 4.2, 4.3, and 4.4. Of the 53,200 available Slice LUTs, 38,833 are used, which is 72.99% utilization as shown in Figure 4.2.

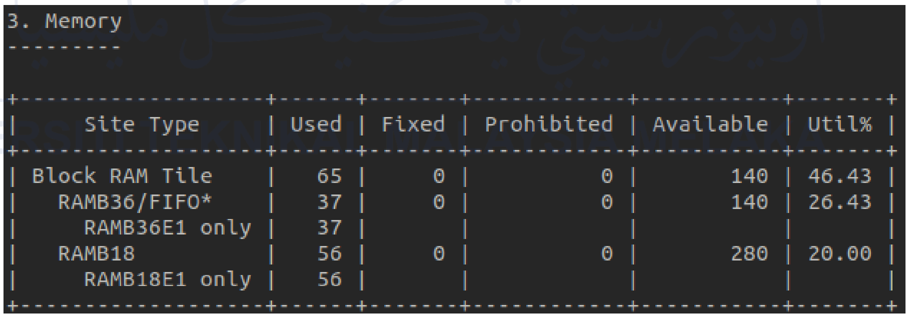


```
1. Slice Logic
--------------

+----------------------------+-------+-------+------------+-----------+-------+
|          Site Type         |  Used | Fixed | Prohibited | Available | Util% |
+----------------------------+-------+-------+------------+-----------+-------+
| Slice LUTs                 | 38833 |     0 |          0 |     53200 | 72.99 |
|   LUT as Logic             | 34725 |     0 |          0 |     53200 | 65.27 |
|   LUT as Memory            |  4108 |     0 |          0 |     17400 | 23.61 |
|     LUT as Distributed RAM |  3702 |     0 |            |           |       |
|     LUT as Shift Register  |   406 |     0 |            |           |       |
| Slice Registers            | 51632 |     0 |          0 |    106400 | 48.53 |
|   Register as Flip Flop    | 51632 |     0 |          0 |    106400 | 48.53 |
|   Register as Latch        |     0 |     0 |          0 |    106400 |  0.00 |
| F7 Muxes                   |  1930 |     0 |          0 |     26600 |  7.26 |
| F8 Muxes                   |   760 |     0 |          0 |     13300 |  5.71 |
+----------------------------+-------+-------+------------+-----------+-------+
```

Figure 4.2: 2-bit Model Slice Logic

These LUTs are used to implement combinatorial logic, of which 65.27% are configured for general logic use and 23.61% are allocated for memory use. Some Slice LUTs are dedicated to specific functions, such as distributed RAM and shift registers, of which 3,702 LUTs are used as distributed RAM and 406 LUTs are configured as shift registers. The utilization of the slice registers used to store intermediate values and results is 48.53%. All 51,632 registers are used, all configured as flip-flops, indicating that they are mainly used for sequential logic. The F7 and F8 multiplexers, which are critical for signal routing within the FPGA, are utilized at 7.26% and 5.71% respectively, with 1,930 of the 26,600 F7 multiplexers being used and 760 of the 13,300 F8 multiplexers being used.

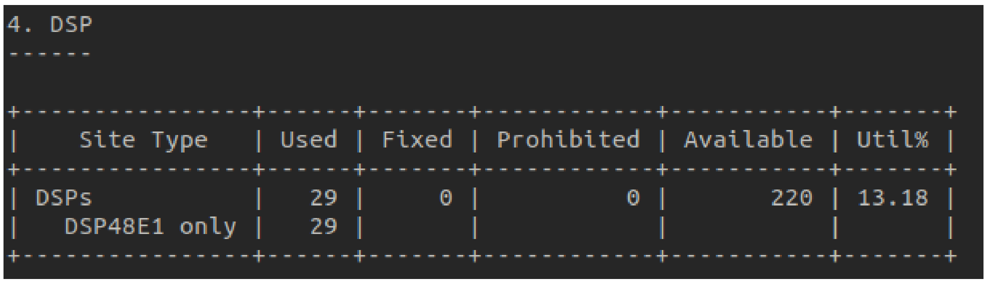
Figure 4.3 details the memory utilization, where the Block RAM Tile is used as the basic memory component, with 65 of the 140 instances being used, for a utilization of 46.43%. Specific memory instances such as RAMB36/FIFO and RAMB36E1 each have 37 instances used, with both types having a utilization of 26.43%. These are critical for applications that require specialized memory structures such as FIFO implementations. The RAMB18 and RAMB18E1 blocks, known for their capacity and versatility, are each used at 56 of the 280 instances available, for a utilization of 20.00%.



```
3. Memory
---------

+-------------------+------+-------+------------+-----------+-------+
|     Site Type     | Used | Fixed | Prohibited | Available | Util% |
+-------------------+------+-------+------------+-----------+-------+
| Block RAM Tile    |   65 |     0 |          0 |       140 | 46.43 |
|   RAMB36/FIFO*    |   37 |     0 |          0 |       140 | 26.43 |
|     RAMB36E1 only |   37 |       |            |           |       |
|   RAMB18          |   56 |     0 |          0 |       280 | 20.00 |
|     RAMB18E1 only |   56 |       |            |           |       |
+-------------------+------+-------+------------+-----------+-------+
```

Figure 4.3: 2-bit Model Memory

The DSP resource utilization shown in Figure 4.4 shows that 29 instances are in use out of the available 220, a utilization of 13.18%. DSP blocks are critical for accelerating complex mathematical computations and signal processing tasks in FPGA designs. A comprehensive analysis of Slice Logic, DSP, and memory utilization demonstrates a holistic approach to resource management. A considerable portion of the DSP resources utilization indicates effective coordination of computational tasks with specialized hardware functions.



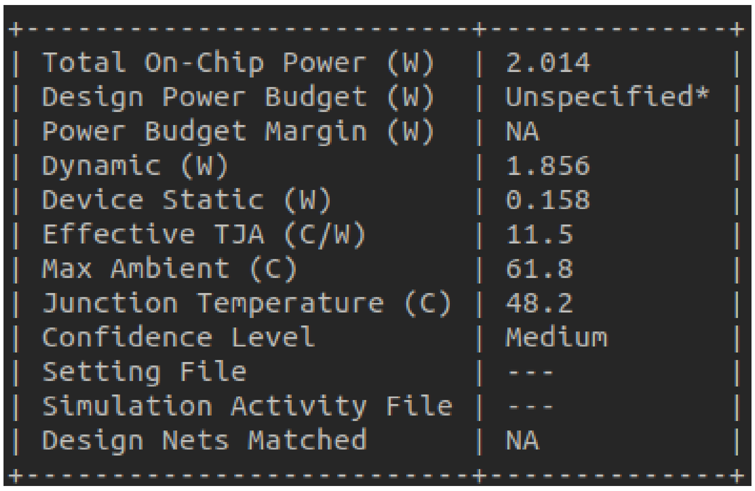
```
4. DSP
------

+----------------+------+-------+------------+-----------+-------+
|    Site Type   | Used | Fixed | Prohibited | Available | Util% |
+----------------+------+-------+------------+-----------+-------+
| DSPs           |   29 |     0 |          0 |       220 | 13.18 |
|   DSP48E1 only |   29 |       |            |           |       |
+----------------+------+-------+------------+-----------+-------+
```

Figure 4.4: 2-bit Model DSP

The balance between Slice Logic and memory utilization indicates that resource allocation for computational and data storage requirements is coordinated. A comprehensive analysis of resource utilization highlights the efficiency and effectiveness of the 2-bit quantized YOLOv3-Tiny model implementation on the PYNQ-Z1 board.

4.2.2 Power

Figure 4.5 shows the power report, which details the total power consumption of the chip in watts. The total on-chip power is approximately 2.014W, indicating the overall power consumption of the chip. The dynamic power consumption (indicating the power used when the chip is active or running) is approximately 1.856W. The device's static power (indicating the power consumed when the chip is idle) is approximately 0.158W.



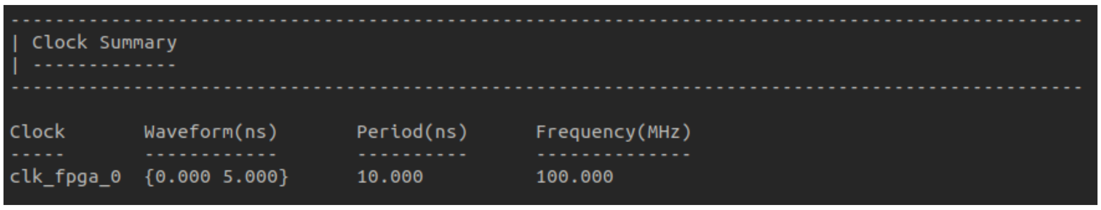
```
+--------------------------+--------------+
| Total On-Chip Power (W)  | 2.014        |
| Design Power Budget (W)  | Unspecified* |
| Power Budget Margin (W)  | NA           |
| Dynamic (W)              | 1.856        |
| Device Static (W)        | 0.158        |
| Effective TJA (C/W)      | 11.5         |
| Max Ambient (C)          | 61.8         |
| Junction Temperature (C) | 48.2         |
| Confidence Level         | Medium       |
| Setting File             | ---          |
| Simulation Activity File | ---          |
| Design Nets Matched      | NA           |
+--------------------------+--------------+
```

Figure 4.5: 2-bit Model Power

In addition, the report also includes the effective TJA (C/W), which stands for the junction-to-ambient thermal resistance. This metric measures how effectively the chip transfers heat from the junction (the hottest part of the chip) to the surrounding environment. Lower TJA values indicate better cooling efficiency. The maximum ambient temperature represents the maximum ambient temperature (in degrees Celsius) at which the chip can operate effectively. The junction temperature refers to the temperature of the chip junction, which is usually the hottest point. This detailed power report provides a comprehensive overview of the power consumption and thermal management of the chip, which is critical to evaluating the efficiency and reliability of the 2-bit quantized model implementation on the PYNQ-Z1 board.

4.2.3 Clock Constraint and Frequency

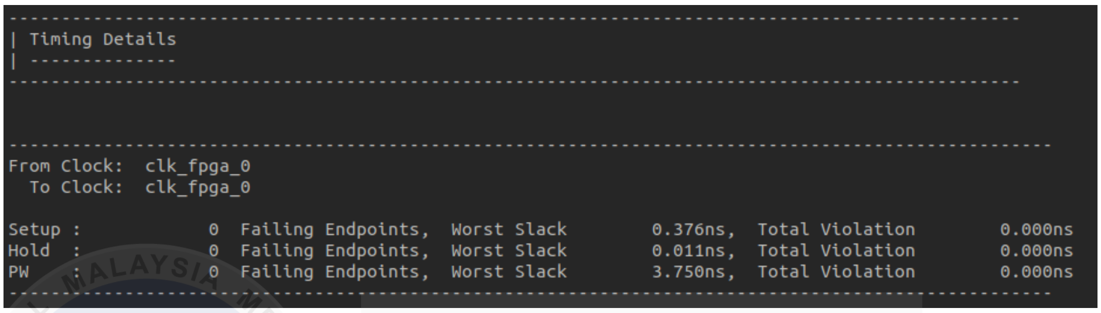
As shown in Figures 4.6 and 4.7, a clock frequency of 100 MHz is used and no clock setup, hold, or pulse width requirements are violated, highlighting the robust design characteristics. In Vivado, the “worst slack” in the clock report refers to the timing margin of the critical path with the smallest margin to meet the specified timing constraints. These constraints define the desired performance goals of the model design, including maximum clock frequency, setup time, and hold time requirements.



```
-----------------------------------------------------------------------------------------------
| Clock Summary
| -------------
-----------------------------------------------------------------------------------------------

Clock       Waveform(ns)       Period(ns)      Frequency(MHz)
-----       ------------       ----------      --------------
clk_fpga_0  {0.000 5.000}      10.000          100.000
```

Figure 4.6: 2-bit Model Clock Summary

Timing margin represents the amount of time a signal can be delayed without violating the specified timing constraints. Positive margin values indicate that the design meets timing requirements, while negative margin values indicate timing violations. The “worst slack” is the smallest (most negative) margin value among all critical paths in the design. A negative worst margin means that the design fails to meet timing on the critical path. This failure can be caused by various factors, such as routing congestion, inefficient logic element placement, or insufficient clock-to-q delays for sequential elements on the critical path.



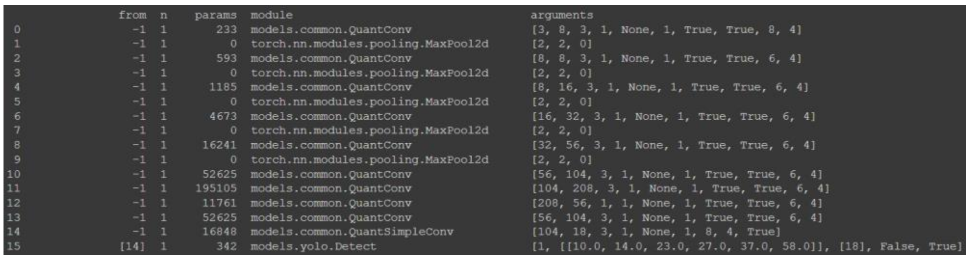
```
------------------------------------------------------------------------------------------------
| Timing Details
| -------------
------------------------------------------------------------------------------------------------


---------------------------------------------------------------------------------------------------
From Clock:  clk_fpga_0
  To Clock:  clk_fpga_0

Setup :            0  Failing Endpoints,  Worst Slack        0.376ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.011ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        3.750ns,  Total Violation        0.000ns
---------------------------------------------------------------------------------------------------
```

Figure 4.7: 2-bit Model Timing Details

This clock summary and timing analysis highlights the efficiency and reliability of the 2-bit quantized model implementation on the PYNQ-Z1 board, ensuring that the design meets the required performance criteria without violating timing.

4.3 Six-bit Quantization Model

In this section, we study the 6-bit quantized models as shown in Figure 4.8. Our analysis focuses on resource utilization, power consumption, and adherence to clock constraints and frequency. Through a comprehensive evaluation, we aim to gain insights into the feasibility and effectiveness of deploying low-precision models on FPGA platforms. This detailed investigation will highlight the potential benefits and challenges of implementing 6-bit quantized models, helping to advance the optimization of machine learning deployments on FPGA hardware.

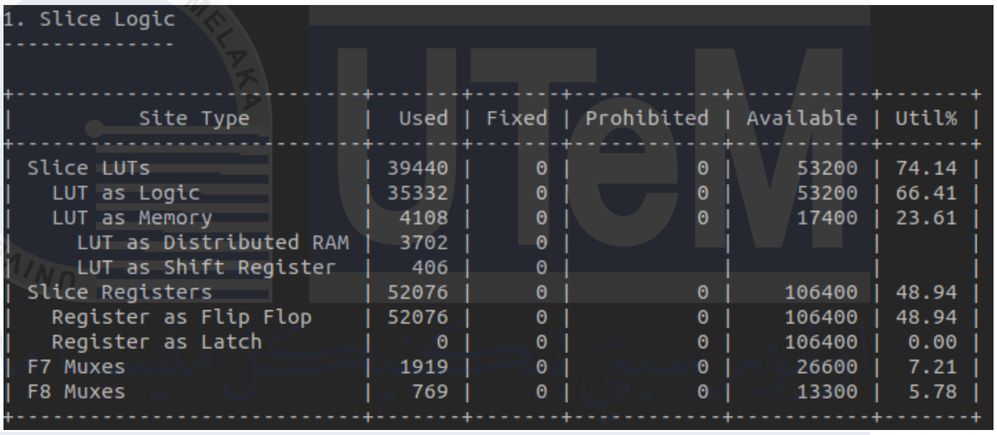


```
                 from  n    params  module                                  arguments
  0                -1  1       233  models.common.QuantConv                 [3, 8, 3, 1, None, 1, True, True, 8, 4]
  1                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  2                -1  1       593  models.common.QuantConv                 [8, 8, 3, 1, None, 1, True, True, 6, 4]
  3                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  4                -1  1      1185  models.common.QuantConv                 [8, 16, 3, 1, None, 1, True, True, 6, 4]
  5                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  6                -1  1      4673  models.common.QuantConv                 [16, 32, 3, 1, None, 1, True, True, 6, 4]
  7                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
  8                -1  1     16241  models.common.QuantConv                 [32, 56, 3, 1, None, 1, True, True, 6, 4]
  9                -1  1         0  torch.nn.modules.pooling.MaxPool2d      [2, 2, 0]
 10                -1  1     52625  models.common.QuantConv                 [56, 104, 3, 1, None, 1, True, True, 6, 4]
 11                -1  1    195105  models.common.QuantConv                 [104, 208, 3, 1, None, 1, True, True, 6, 4]
 12                -1  1     11761  models.common.QuantConv                 [208, 56, 1, 1, None, 1, True, True, 6, 4]
 13                -1  1     52625  models.common.QuantConv                 [56, 104, 3, 1, None, 1, True, True, 6, 4]
 14                -1  1     16848  models.common.QuantSimpleConv           [104, 18, 3, 1, None, 1, 8, 4, True]
 15              [14]  1       342  models.yolo.Detect                      [1, [[10.0, 14.0, 23.0, 27.0, 37.0, 58.0]], [18], False, True]
```

Figure 4.8: 6-bit Quantized Model

4.3.1 Resource Utilization

Figures 4.9, 4.10, and 4.11 show the resource utilization report, detailing the performance of the 6-bit quantized model on the FPGA platform. The slice lookup tables (LUTs) that implement arbitrary Boolean logic functions have a utilization of 74.14%, indicating that a large portion of the design logic relies on these LUTs. Specifically, 66.41% of the LUTs are used for general logic purposes, while 406 LUTs are used as small memories or shift registers, accounting for 23.61% of the total utilization.



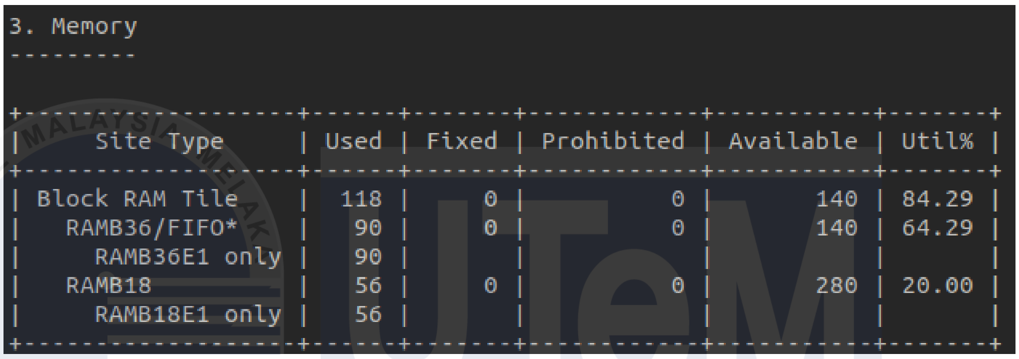
```
1. Slice Logic
--------------

+----------------------------+-------+-------+------------+-----------+-------+
|         Site Type          |  Used | Fixed | Prohibited | Available | Util% |
+----------------------------+-------+-------+------------+-----------+-------+
| Slice LUTs                 | 39440 |     0 |          0 |     53200 | 74.14 |
|   LUT as Logic             | 35332 |     0 |          0 |     53200 | 66.41 |
|   LUT as Memory            |  4108 |     0 |          0 |     17400 | 23.61 |
|     LUT as Distributed RAM |  3702 |     0 |            |           |       |
|     LUT as Shift Register  |   406 |     0 |            |           |       |
| Slice Registers            | 52076 |     0 |          0 |    106400 | 48.94 |
|   Register as Flip Flop    | 52076 |     0 |          0 |    106400 | 48.94 |
|   Register as Latch        |     0 |     0 |          0 |    106400 |  0.00 |
| F7 Muxes                   |  1919 |     0 |          0 |     26600 |  7.21 |
| F8 Muxes                   |   769 |     0 |          0 |     13300 |  5.78 |
+----------------------------+-------+-------+------------+-----------+-------+
```

Figure 4.9: 6-bit Model Slice Logic

The utilization of slice registers, which are used to store data or state information in sequential logic, is 48.94%, and all used registers are configured as flip-flops, matching the utilization of slice registers. This indicates that the design primarily uses flip-flops as memory elements. The higher utilization percentage compared to the 2-bit quantized model indicates that the 6-bit quantized model requires more resources.

Block RAMs, which implement larger memory arrays that are necessary to store large amounts of data or coefficients, have a utilization of 84.29%, indicating that almost all available block RAM blocks are used. This marks a significant improvement in memory utilization compared to the 2-bit quantized model. Quantization reduces the precision of the neural network weights, such as converting

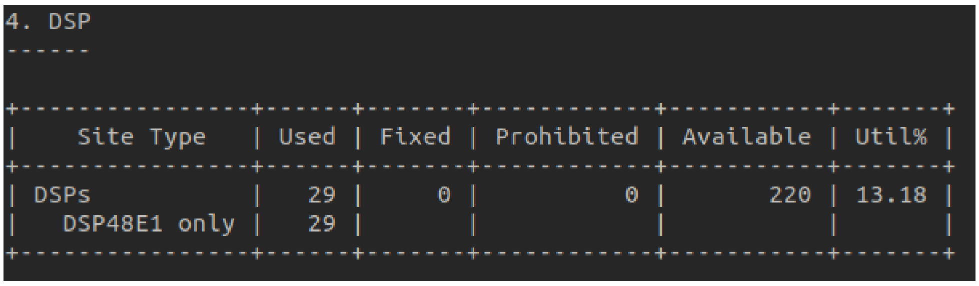
32-bit floating point numbers to 6-bit integers, thereby reducing the memory and compute requirements of the model. The report shows that this low memory utilization is attributed to the effective use of quantization techniques, which shrinks the model's memory footprint, allowing it to fit on the FPGA with sufficient resources available. This is particularly beneficial for embedded systems with limited memory and computer resources.



```
3. Memory
---------

+-------------------+------+-------+------------+-----------+-------+
|     Site Type     | Used | Fixed | Prohibited | Available | Util% |
+-------------------+------+-------+------------+-----------+-------+
| Block RAM Tile    |  118 |     0 |          0 |       140 | 84.29 |
|   RAMB36/FIFO*    |   90 |     0 |          0 |       140 | 64.29 |
|     RAMB36E1 only |   90 |       |            |           |       |
|   RAMB18          |   56 |     0 |          0 |       280 | 20.00 |
|     RAMB18E1 only |   56 |       |            |           |       |
+-------------------+------+-------+------------+-----------+-------+
```

Figure 4.10: 6-bit Model Memory

The digital signal processor (DSP), which is critical for performing arithmetic functions such as multiply-accumulate operations in convolutional layers, has a utilization of 13.18%, indicating that a considerable portion of the DSP is used in the design.



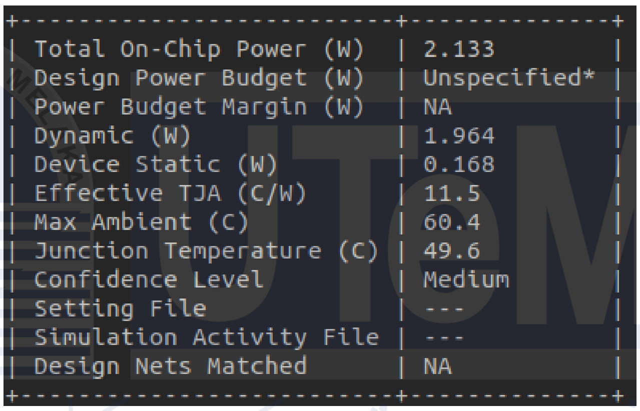
```
4. DSP
------

+----------------+------+-------+------------+-----------+-------+
|    Site Type   | Used | Fixed | Prohibited | Available | Util% |
+----------------+------+-------+------------+-----------+-------+
| DSPs           |   29 |     0 |          0 |       220 | 13.18 |
|   DSP48E1 only |   29 |       |            |           |       |
+----------------+------+-------+------------+-----------+-------+
```

Figure 4.11: 6-bit Model DSP

Overall, the detailed resource utilization highlights the efficiency and increased requirements of the 6-bit quantized model, proving its feasibility and effectiveness in deployment on FPGA platforms while maintaining efficient resource management.

4.3.2 Power

Figure 4.12 details the power report, illustrating the power consumption metrics for a 6-bit quantized model implemented on an FPGA platform. The total on-chip power is approximately 2.133 Watts, representing the overall power consumption of the chip. Dynamic power represents the power consumed when the chip is active, which is approximately 1.964 Watts. Device static power represents the power consumed when the chip is idle, which is approximately 0.168 Watts.



```
+--------------------------+--------------+
| Total On-Chip Power (W)  | 2.133        |
| Design Power Budget (W)  | Unspecified* |
| Power Budget Margin (W)  | NA           |
| Dynamic (W)              | 1.964        |
| Device Static (W)        | 0.168        |
| Effective TJA (C/W)      | 11.5         |
| Max Ambient (C)          | 60.4         |
| Junction Temperature (C) | 49.6         |
| Confidence Level         | Medium       |
| Setting File             | ---          |
| Simulation Activity File | ---          |
| Design Nets Matched      | NA           |
+--------------------------+--------------+
```

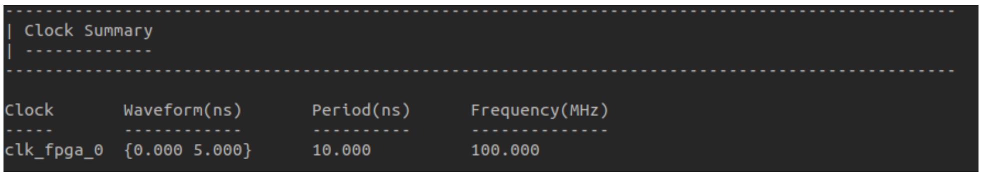
Figure 4.12: 6-bit Model Power

Effective TJA (C/W) or junction-to-ambient thermal resistance measures how effectively the chip transfers heat from the junction (the hottest part of the chip) to the surrounding environment. Lower TJA values indicate better cooling efficiency. Maximum ambient temperature represents the maximum ambient temperature (in degrees Celsius) at which the chip can operate effectively. Junction temperature refers to the temperature of the chip junction, which is typically the hottest point.

The total on-chip power consumption of the 6-bit quantized model is higher compared to the 2-bit quantized model, indicating that the increased accuracy of the 6-bit model results in increased power consumption. This comprehensive power analysis highlights the trade-off between model accuracy and power efficiency, which is critical for optimizing FPGA deployments for machine learning tasks.

4.3.3 Clock Constraint and Frequency

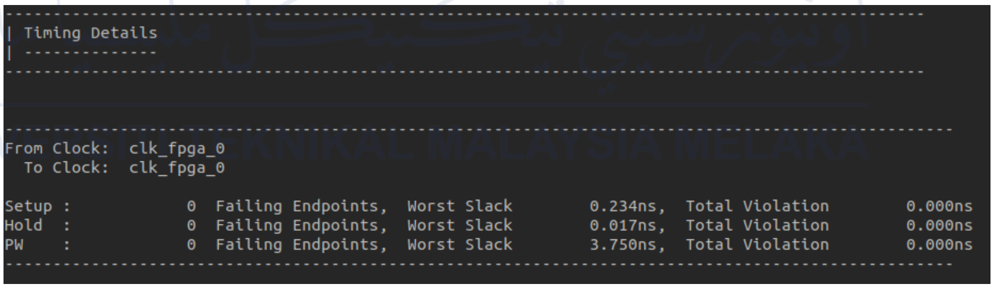
As stated in the Clock Summary report for the 6-bit quantized model, the clock frequency used is 100 MHz as shown in Figure 4.13.



```
------------------------------------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------------------------------------

Clock       Waveform(ns)       Period(ns)      Frequency(MHz)
-----       ------------       ----------      --------------
clk_fpga_0  {0.000 5.000}      10.000          100.000
```

Figure 4.13: 6-bit Model Clock Summary

Figure 4.14 shows that no clock violations were found after the implementation process. This indicates that the design meets all setup, hold, and pulse width requirements, ensuring reliable performance. The absence of clock violations highlights the robustness of the 6-bit quantized model implementation, maintaining adherence to critical timing constraints and enabling efficient operation on FPGA platforms.



```
------------------------------------------------------------------------------------------
| Timing Details
| --------------
------------------------------------------------------------------------------------------


------------------------------------------------------------------------------------------
From Clock:  clk_fpga_0
  To Clock:  clk_fpga_0

Setup :            0  Failing Endpoints,  Worst Slack        0.234ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.017ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        3.750ns,  Total Violation        0.000ns
------------------------------------------------------------------------------------------
```

Figure 4.14: 6-bit Model Timing Details

4.4 PYNQ-Z1 board

The final deployed model is validated on FPGA and its performance is summarized. The performance metrics obtained by FPGA implementation of 2-bit and 6-bit quantized models are compared. This step aims to analyze the performance of quantized model on FPGA. This ensures that the drowsiness detection based on FPGA implementation is meaningful.

4.4.1 Validation

The validation process is done which is shown in Figure 4.15 and to make sure the practical implementation of the drowsiness detection with 2-bit quantized model and the 6-bit quantized model works on the PYNQ-Z1 FPGA board. The inference process is done using the driver which is the communication protocol, the bit file and configuration file.

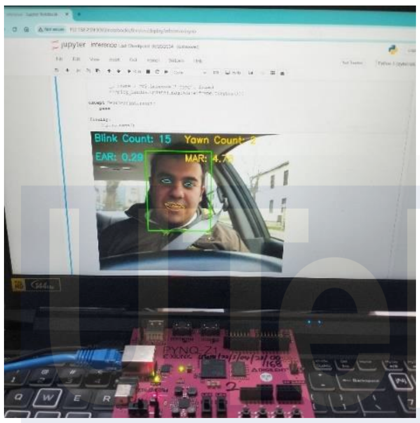


Figure 4.15: PYNQ-Z1 Implementation

4.4.2 Performance

The performance of the drowsiness detection with blink and yawn accuracy with latency is tested in PYNQ-Z1 FPGA as shown in Table 4.1.

Table 4.1: Performance on PYNQ-Z1

Model	Blink Accuracy	Yawn Accuracy	Total Accuracy	Latency
Haar Cascade	-	-	-	910.25ms/frame
Yolov3 Tiny	-	-	-	3530.14ms/frame
2-bit Quantized Yolov3 Tiny	72%	76%	74%	191.72ms/frame
6-bit Quantized Yolov3 Tiny	77%	90%	83.5	224.35ms/frame

4.5 Central Processing Unit

The yawning and blinking counts in drowsiness detection are also inferred on PC using CPU workspace for comparative analysis.

4.5.1 Performance

The performance of the blink and yawn accuracy with latency is tested in CPU as shown in Table 4.2.

Table 4.2: Performance on CPU

Model	Blink Accuracy	Yawn Accuracy	Total Accuracy	Latency
Haar Cascade	78%	85%	81.5%	22.25ms/frame
Yolov3 Tiny	85%	92%	88.5%	49.10ms/frame

4.5.2 Resource Utilization

The resource utilization graph of the personal computer can be seen in Figure 4.16. The CPU usage of the drowsiness detection fluctuated from 0 to 49%, the memory usage for the models is at 40% average.

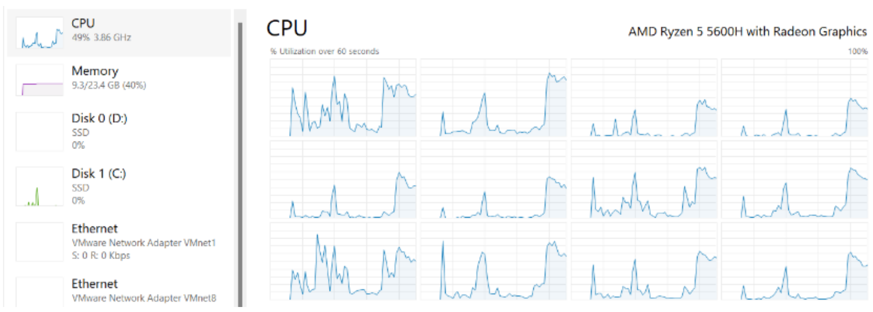

CPU
49% 3.86 GHz
Memory
9.3/23.4 GB (40%)
Disk 0 (D:)
SSD
0%
Disk 1 (C:)
SSD
0%
Ethernet
VMware Network Adapter VMnet1
S: 0 R: 0 Kbps
Ethernet
VMware Network Adapter VMnet8
CPU
AMD Ryzen 5 5600H with Radeon Graphics
% Utilization over 60 seconds
100%

Figure 4.16: Resource Utilization

4.6 Comparison Table

Table 4.3 shows not only the comparison of drowsiness detection on different platforms with different models in terms of latency, power, and resource utilization, but also the comparison with other research results. All the data that had been measured and recorded had been compared.

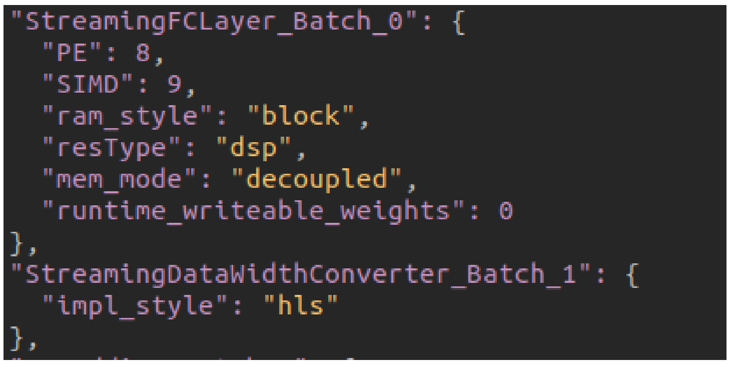
Table 4.3: Result Comparison

	Device	**Model**	**Power**	**Resource Utilization**	**Latency**
Proposed in this project	CPU (Ryzen 5600H)	Haar Cascade	65W	-	22.25ms/frame
	CPU (Ryzen 5600H)	Yolov3 Tiny	65W	-	49.10ms/frame
	PYNQ-Z1 (PS)	Haar Cascade	2.4W	-	791.62ms/frame
	PYNQ-Z1 (PS)	Yolov3 Tiny	2.4W	-	8344.53ms/frame
	PYNQ-Z1 (PS + PL)	2-bit Quantized Yolov3 Tiny	2.014W	72.99% LUT 48.53% Registers 13.18% DSP 46.43% Memory	180.92ms/frame
	PYNQ-Z1 (PS + PL)	6-bit Quantized Yolov3 Tiny	2.133W	74.14% LUT 48.94% Registers 13.18% DSP 84.29% Memory	199.12ms/frame
S. K. Mousavikia *et al.* [29], 2022	Xilinx Nexys 4 DDR	CNN	-	9.57% LUT 1.77% Registers 4.17% DSP 99.26% Memory	231ms/frame

4.7 Discussion

The PYNQ-Z1 FPGA achieved a significant 45.24% reduction in memory usage after converting from a 6-bit model to a 2-bit model. In addition, the quantized model on the PYNQ-Z1 reduced power consumption by 95.52% compared to CPU and improved latency by 46.12 times after converting from a pure PS (processing system) to a PS+PL (programmable logic) combined approach. As mentioned in [29], the FPGA uses a 320x240 (QVGA) resolution with low FPGA resource utilization, using only 9.57% of the look-up tables (LUTs), 1.77% of the registers, and 4.17% of the digital signal processing (DSP) blocks. This lower utilization allows for more flexibility in adding additional functionality or logic. However, this work also results in a memory utilization of almost 99.26% and a longer processing time of 231 milliseconds per frame, which may not be suitable for real-time applications. The proposed work, while requiring more FPGA resources, provides faster processing time and more available memory, which is beneficial for real-time applications.

The PYNQ-Z1 FPGA uses advanced parallel techniques such as pipelining to improve computational efficiency and performance. This hardware design strategy divides computation into multiple stages, allowing each stage to run simultaneously. By overlapping the execution of different stages, pipelining enhances concurrency and enables efficient parallel processing of multiple data elements in each stage, thereby optimizing overall computational performance. These parallelization techniques are automatically implemented by the FINN library, significantly improving the performance of the FPGA in terms of resource utilization, power consumption, and thermal management. However, due to the limitations of onboard RAM resources, the training process cannot be performed on the board itself. Figure 4.17 shows the degree of parallelism is determined by the number of processing elements (PEs) and single instruction multiple data (SIMD) units used per layer, which can be adjusted by setting the folding parameters. Increasing the number of PEs and SIMDs can increase parallelism and speed, but this also consumes more logic resources, so a balance needs to be struck between performance and resource utilization.



```
"StreamingFCLayer_Batch_0": {
  "PE": 8,
  "SIMD": 9,
  "ram_style": "block",
  "resType": "dsp",
  "mem_mode": "decoupled",
  "runtime_writeable_weights": 0
},
"StreamingDataWidthConverter_Batch_1": {
  "impl_style": "hls"
},
```

Figure 4.17: Parallelism

4.8 Environmental and Sustainability

Sustainable development is essential to ensure a balance between environmental protection and social progress, highlighting the importance of engineering solutions that minimize ecological impact while benefiting communities.

4.8.1 Needs and Importance for Sustainable Development

In today's rapidly evolving technology landscape, especially in the field of automation, developing advanced computing for AI applications is essential. To meet the growing demand for AI applications, advanced computing solutions such as Field Programmable Gate Arrays (FPGAs) have become essential. FPGAs play a key role in promoting innovation and building resilient infrastructure aligning with SDG 9 as shown in Figure 4.18 [39]. Their scalability and flexibility are particularly beneficial for developing a neural network-based drowsiness detection system. By leveraging the adaptability of FPGAs, the project ensures that the technology can evolve with the rapid development of neural networks. This adaptability not only supports continued innovation, but also strengthens the infrastructure required for complex AI applications. Improving energy efficiency is a key aspect of the project, contributing to more sustainable consumption and production patterns.

9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
12 RESPONSIBLE CONSUMPTION AND PRODUCTION

Figure 4.18: Sustainable Development Goals [39]

FPGAs are known for their high energy efficiency, making them an ideal solution for reducing the environmental impact of technology. By optimizing resource utilization and improving energy efficiency, the project promotes sustainability in technological development. This not only meets the goals of SDG 12, but also helps improve human lifestyles by minimizing the ecological footprint of advanced computing. By promoting innovation, strengthening infrastructure and improving energy efficiency, the project contributes to the achievement of SDG 9 and SDG 12, ensuring that technological progress is both sustainable and impactful.

4.8.2 Impact of the Engineering Solution on Society

Applying drowsiness detection on FPGA can promote the development of driver fatigue detection crisis to improve vigilance. Autonomous driving systems are becoming a trend. This project provides insights into real-time image recognition systems to improve the safety of advanced driver assistance systems (ADAS). When using reconfigurable FPGA-based convolutional neural networks for drowsiness detection, resource waste can be avoided.

4.9 Summary

Through comparative analysis of latency, power consumption, and resource consumption, the results of my project “Implementation and Performance Analysis of Drowsiness Detection using Hardware Acceleration on PYNQ-Z1 FPGA” show that high-level computing has both advantages and disadvantages for drowsiness detection. The advantage is the optimization of latency, power consumption and resource utilization. The disadvantage is that the training process cannot be completed on the PYNQ-Z1 board due to resource limitations. The results highlight the potential of FPGAs in meeting high-performance computing needs and provide a path for the development of drowsiness detection.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Introduction

This chapter will summarize the project results in terms of the project goals and future work that can improve the system. This chapter will present the conclusions of the project in Section 5.2 and future work on the project in Section 5.3.

5.2 Conclusion

This thesis focuses on the development of an optimized quantized convolutional neural network-based model for drowsiness detection on the PYNQ-Z1 FPGA. All goals of the project were achieved, successfully implementing drowsiness detection on the FPGA, using optimization techniques on the model, and analyzing the drowsiness detection performance of the quantized model in FPGA.

The first goal was to design and implement drowsiness detection on the FPGA. This goal was achieved by completing on-board validation and documenting the results. The second goal was to explore drowsiness detection optimization techniques in the model, which was also achieved by quantizing the model. Parallelization techniques were also used when converting the model to an HLS model, where pipelining, inlining, and partitioning arrays were performed to exploit the parallelization capabilities of the FPGA. The third goal was to perform a drowsiness detection performance analysis on the results to gain insight into how traditional computing platforms compare to FPGAs. This goal was achieved when we were able to infer the advantages and disadvantages of FPGAs compared to traditional computing platforms based on the power consumption, latency, and implementation reports generated.

In summary, the successful achievement of these goals highlights the importance of FPGAs for drowsiness detection applications. The research results presented in this paper not only advance the current understanding of hardware-accelerated neural networks but also lay a solid foundation for future research and development in the pursuit of optimized, high-performance computing solutions in the field of artificial intelligence.

5.3 Future works

To further improve the limitations found in this project, several advanced techniques and methods can be considered. By quantizing the activation layers in the convolutional neural network model, we can potentially improve performance and efficiency. Integrating and optimizing more complex drowsiness detection methods (such as head pose analysis) can push the limits of FPGA capabilities. It is critical to investigate further optimization techniques for specific FPGA architectures, as different FPGAs have unique capabilities and constraints that can be exploited to improve performance and resource utilization.

In addition, fine-tuning the configuration of each layer in the convolutional neural network may help improve latency, power consumption, and resource utilization. Investigating other optimization techniques is also a promising direction to improve the performance of FPGA-based neural networks. In addition to CNNs, exploring other neural network models or drowsiness detection methods can fully understand the potential of FPGAs in various applications. Real-time inference using cameras on FPGAs presents significant challenges, but also provides opportunities to explore the ability of FPGAs to exploit real-time data. Ensuring cross-platform compatibility is critical to adapt to various FPGA architectures and expand the scope of implementation.

Evaluating power consumption and efficiency through power-aware design techniques and dynamic reconfiguration can produce more energy-efficient models. Integrating the FPGA implementation with an edge computing platform and thoroughly benchmarking it against alternative FPGA-based solutions will provide valuable insights into the strengths and weaknesses of the proposed approach compared to existing solutions. In summary, these future research directions aim to push the boundaries of FPGA-accelerated drowsiness detection and advance the field of hardware-accelerated deep learning and edge computing.

REFERENCES

[1] W. Ameer *et al.*, "Identifying Factors Associated with Sleep Quality among Manufacturing Workers Riding to Work in Klang Valley," 2021.

[2] K. S. Gill, V. Anand, R. Chauhan, S. Thapliyal, and R. Gupta, "A Convolutional Neural Network-Based Method for Real- Time Eye State Identification in Driver Drowsiness Detection," in *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, IEEE, Dec. 2023, pp. 1–5. doi: 10.1109/SMARTGENCON60755.2023.10442238.

[3] J. Ye and W. Zhang, "A Scalable ARM+FPGA-Based CNN Accelerator with Limited Hardware Resources," in *2023 42nd Chinese Control Conference (CCC)*, IEEE, Jul. 2023, pp. 2498–2503. doi: 10.23919/CCC58697.2023.10241078.

[4] V. Kartsch, S. Benatti, M. Guermandi, F. Montagna, and L. Benini, "Ultra Low-Power Drowsiness Detection System with BioWolf," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, IEEE, Mar. 2019, pp. 1187–1190. doi: 10.1109/NER.2019.8717070.

[5] V. Kalisetti, V. S. C. Vasarla, S. B. Kolli, R. Varaparla, V. Enireddy, and M. Mohammed, “Analysis of Driver Drowsiness Detection Methods,” in *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*, IEEE, Mar. 2023, pp. 1481–1485. doi: 10.1109/ICEARS56392.2023.10084986.

[6] S. Yaacob, N. A. Izzati Affandi, P. Krishnan, A. Rasyadan, M. Yaakop, and F. Mohamed, “Drowsiness detection using EEG and ECG signals,” in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, IEEE, Sep. 2020, pp. 1–5. doi: 10.1109/IICAIET49801.2020.9257867.

[7] Y. Ma *et al.*, “Driving Drowsiness Detection with EEG Using a Modified Hierarchical Extreme Learning Machine Algorithm with Particle Swarm Optimization: A Pilot Study,” *Electronics (Basel)*, vol. 9, no. 5, p. 775, May 2020, doi: 10.3390/electronics9050775.

[8] “Lane Departure Warning Overview Benefits.” [Online]. Available: www.intelli-vision.com

[9] H. Oishi, H. Kawanaka, and K. Oguri, “Effectiveness of Data Screening for Driver Drowsiness Estimation Using Drive Recorder,” in *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, IEEE, Oct. 2021, pp. 766–769. doi: 10.1109/GCCE53005.2021.9622071.

[10] Z. Li, L. Chen, L. Nie, and S. X. Yang, “A Novel Learning Model of Driver Fatigue Features Representation for Steering Wheel Angle,” *IEEE Trans Veh*

Technol, vol. 71, no. 1, pp. 269–281, Jan. 2022, doi: 10.1109/TVT.2021.3130152.

[11] E. N. Pratama and W. F. Al Maki, “Drowsiness Detection System for Masked Face Based on Deep Neural Network and Haar Cascade,” in *2022 1st International Conference on Software Engineering and Information Technology (ICoSEIT)*, IEEE, Nov. 2022, pp. 233–237. doi: 10.1109/ICoSEIT55604.2022.10029948.

[12] Sumanto, B. Wijonarko, M. Qommarudin, A. Sudibyo, P. Widodo, and A. M. Lukman, “Viola-Jones Algorithm for Face Detection using Wider Face Dataset,” in *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, IEEE, Sep. 2022, pp. 1–4. doi: 10.1109/CITSM56380.2022.9935830.

[13] I. Arrieta-Arellano, F. López-Orozco, J. I. Hernández-Hernández, and J.-G. Ruiz-Ruiz, “HCI based on eye movements for unlocking mobile devices,” *Avances en Interacción Humano-Computadora*, vol. 6, no. 1, pp. 6–10, Nov. 2021, doi: 10.47756/aihc.y6i1.78.

[14] C. Zhang, G. Liu, X. Zhu, and H. Cai, “Face Detection Algorithm Based on Improved AdaBoost and New Haar Features,” in *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, IEEE, Oct. 2019, pp. 1–5. doi: 10.1109/CISP-BMEI48845.2019.8965841.

[15] R. Xie, Q. Zhang, E. Yang, and Q. Zhu, “A Method of Small Face Detection Based on CNN,” in *2019 4th International Conference on Computational*

Intelligence and Applications (ICCIA), IEEE, Jun. 2019, pp. 78–82. doi: 10.1109/ICCIA.2019.00022.

[16] J. Boone, C. Goodin, L. Dabbiru, C. Hudson, L. Cagle, and D. Carruth, "Training Artificial Intelligence Algorithms with Automatically Labelled UAV Data from Physics-Based Simulation Software," *Applied Sciences*, vol. 13, no. 1, p. 131, Dec. 2022, doi: 10.3390/app13010131.

[17] Md. A. A. Akash, M. A. H. Akhand, and N. Siddique, "Robust Face Detection Using Hybrid Skin Color Matching under Different Illuminations," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, IEEE, Feb. 2019, pp. 1–6. doi: 10.1109/ECACE.2019.8679481.

[18] M. Diba, A. R. Sokhango, M. Sabouri, A. Haji Poor, and M. S. Student, "Iranian Conference on Fuzzy Systems 'Human face detection by fuzzy filter and Pattern matching in HSI and YCbCr color space' Human face detection by fuzzy filter and Pattern matching in HSI and YCbCr color space", doi: 10.13140/2.1.4688.2883.

[19] T. T. Phuong, L. T. Hien, D. N. Toan, and N. D. Vinh, "An Eye Blink detection technique in video surveillance based on Eye Aspect Ratio," in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, IEEE, Feb. 2022, pp. 534–538. doi: 10.23919/ICACT53585.2022.9728891.

[20] Daniel Georgiev, "What is the difference between the Iris and the pupil in the eye?," IrisTech. Accessed: Apr. 19, 2024. [Online]. Available:

https://iristech.co/what-is-the-difference-between-the-iris-and-the-pupil-in-the-eye/

[21] W. Tipprasert, T. Charoenpong, C. Chianrabutra, and C. Sukjamsri, "A Method of Driver's Eyes Closure and Yawning Detection for Drowsiness Analysis by Infrared Camera," in *2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, IEEE, Jan. 2019, pp. 61–64. doi: 10.1109/ICA-SYMP.2019.8646001.

[22] M. A. Zulkarnanie, K. S. Shanmugam, N. Badruddin, and M. N. M. Saad, "Enhancements to PERCLOS Algorithm for Determining Eye Closures," in *2022 International Conference on Future Trends in Smart Communities (ICFTSC)*, IEEE, Dec. 2022, pp. 76–81. doi: 10.1109/ICFTSC57269.2022.10039811.

[23] A. Pondit, A. Dey, and A. Das, "Real-time Driver Monitoring System Based on Visual Cues," in *2020 6th International Conference on Interactive Digital Media (ICIDM)*, IEEE, Dec. 2020, pp. 1–6. doi: 10.1109/ICIDM51048.2020.9339604.

[24] Advanced Micro Devices Inc., "Field Programmable Gate Array (FPGA): What is an FPGA?" Accessed: Feb. 24, 2024. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

[25] Advanced Micro Devices Inc., "What is PYNQ?" Accessed: Mar. 11, 2024. [Online]. Available: https://www.pynq.io/#

[26] B. Yazici, A. Ozdemir, and T. Ayhan, “System-on-Chip Based Driver Drowsiness Detection and Warning System,” in *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*, IEEE, Sep. 2022, pp. 1–5. doi: 10.1109/ASYU56188.2022.9925481.

[27] A. Migali, F. Spagnolo, and P. Corsonello, “Heterogeneous FPGA-based System for Real-Time Drowsiness Detection,” in *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, IEEE, Jun. 2022, pp. 169–172. doi: 10.1109/PRIME55000.2022.9816816.

[28] Y. Kortli, S. Gabsi, L. F. C. Lew Yan Voon, and M. Jridi, “Design of ADAS Fatigue Control System using Pynq z1 and Jetson Xavier NX,” in *2022 IEEE 9th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, IEEE, May 2022, pp. 65–68. doi: 10.1109/SETIT54465.2022.9875823.

[29] S. K. Mousavikia, E. Gholizadehazari, M. Mousazadeh, and S. B. O. Yalcin, “Instruction Set Extension of a RiscV Based SoC for Driver Drowsiness Detection,” *IEEE Access*, vol. 10, pp. 58151–58162, 2022, doi: 10.1109/ACCESS.2022.3177743.

[30] N. Tabassum and N. Tabassum, “Real-Time Drowsiness Alert System from EEG Signal Based on FPGA,” in *2021 3rd International Conference on Electrical & Electronic Engineering (ICEEE)*, IEEE, Dec. 2021, pp. 129–132. doi: 10.1109/ICEEE54059.2021.9718788.

[31] J.-Y. Hsu, T.-Y. Jiang, and P. C.-P. Chao, “A Fast FPGA Hardware Accelerator for Remote Heart Rate Detection Based on RGB Vision,” *IEEE Trans Biomed*

Circuits Syst, vol. 18, no. 3, pp. 592–607, Jun. 2024, doi: 10.1109/TBCAS.2024.3354505.

[32] P. Christakos, N. Petrellis, P. Mousouliotis, G. Keramidas, C. P. Antonopoulos, and N. Voros, "A High Performance and Robust FPGA Implementation of a Driver State Monitoring Application," *Sensors*, vol. 23, no. 14, p. 6344, Jul. 2023, doi: 10.3390/s23146344.

[33] P. Dipl and I. M. Wess, "FPGA optimized dynamic post-training Quantization of Tiny-YoloV3," 2021.

[34] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A Face Detection Benchmark." Accessed: Feb. 12, 2024. [Online]. Available: http://shuoyang1213.me/WIDERFACE/

[35] Shabnam Abtahi, Mona Omidyeganeh, Shervin Shirmohammadi, and Behnoosh Hariri, "YawDD: Yawning Detection Dataset." Accessed: Feb. 17, 2024. [Online]. Available: https://dx.doi.org/10.21227/e1qm-hb90

[36] AMD Xilinx, "Getting Started - FINN documentation." Accessed: Apr. 12, 2024. [Online]. Available: https://finn.readthedocs.io/en/latest/getting_started.html

[37] A. Kuwahara, K. Nishikawa, R. Hirakawa, H. Kawano, and Y. Nakatoh, "Eye fatigue estimation using blink detection based on Eye Aspect Ratio Mapping(EARM)," *Cognitive Robotics*, vol. 2, pp. 50–59, 2022, doi: 10.1016/j.cogr.2022.01.003.

[38] P. Awasekar, M. Ravi, S. Doke, and Z. Shaikh, “Driver Fatigue Detection and Alert System using Non-Intrusive Eye and Yawn Detection,” *Int J Comput Appl*, vol. 180, no. 44, pp. 1–5, May 2018, doi: 10.5120/ijca2018917140.

[39] Gradesens, “Sustainable Development Goals.” Accessed: May 24, 2024. [Online]. Available: https://gradesens.com/sdg/