UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Implementation of FPGA Based Smart Collision Avoidance Alert System Algorithm

This report is submitted in accordance with the requirement of the Universiti Teknikal Malaysia Melaka (UTeM) for the Bachelor of Computer Engineering Technology (Computer System) with Honours

By

LIM SIAU LI

B071310529

FACULTY OF ENGINEERING TECHNOLOGY

2016

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

BORANG PENGESAHAN STATUS LAPORAN PROJEK SARJANA MUDA

TAJUK: **IMPLEMENTATION OF FPGA BASED SMART COLLISION AVOIDANCE ALERT SYSTEM ALGORITHM**

SESI PENGAJIAN: **2016/17 Semester 1**

Saya **LIM SIAU LI**

mengaku membenarkan Laporan PSM ini disimpan di Perpustakaan Universiti Teknikal Malaysia Melaka (UTeM) dengan syarat-syarat kegunaan seperti berikut:

1. Laporan PSM adalah hak milik Universiti Teknikal Malaysia Melaka dan penulis.
2. Perpustakaan Universiti Teknikal Malaysia Melaka dibenarkan membuat salinan untuk tujuan pengajian sahaja dengan izin penulis.
3. Perpustakaan dibenarkan membuat salinan laporan PSM ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (✓)

☐ SULIT (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia sebagaimana yang termaktub dalam AKTA RAHSIA RASMI 1972)

☐ TERHAD (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☐ TIDAK TERHAD

Disahkan oleh:

_____ _____

Alamat Tetap:

Cop Rasmi:

_____

_____

_____

Tarikh: _____ Tarikh: _____

** Jika Laporan PSM ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan PSM ini perlu dikelaskan sebagai

DECLARATION

I hereby, declared this report entitled “Implementation of FPGA Based Smart Collision Avoidance Alert System Algorithm” is the results of my own research except as cited in references.

Signature	:	…….
Author’s Name	:	LIM SIAU LI
Date	:	9 DECEMBER 2016

APPROVAL

This report is submitted to the Faculty of Engineering Technology of UTeM as a partial fulfillment of the requirements for the degree of Bachelor Degree of Engineering Technology (Computer Systems) (Hons.). The member of the supervisory is as follow:

.....

EN. AIMAN ZAKWAN BIN JIDIN

ABSTRAK

Pada masa kini, kemalangan jalan raya menyumbang sejumlah besar kematian dan kecederaan. brek secara tiba-tiba dan memandu terlalu rapat dengan kenderaan depan adalah salah satu faktor yang menyebabkan kemalangan. kesilapan manusia adalah faktor utama yang menyebabkan kemalangan jalan raya. Banyak kes kemalangan jalan raya berlaku kerana ketaksedaran manusia. Oleh itu, kajian ini akan mencipta satu algoritma yang mampu untuk memberi amaran kepada pemandu dengan meggunakan sensor untuk mengesan jarak antara sensor tersebut dengan kenderaan atau objek depan untuk mengurangkan kejadian kemalangan jalan raya. Algoritma ini mengandungi 2 mod. Dalam mod memandu, ia mengesan kadar perubahan jarak antara sensor dengan kenderaan depan dan juga mengesan jarak selamat antara dua kenderaan. Dalam mod letak kereta, sensor juga akan mengesan jarak selamat antara sensor dengan kenderaan depan. Ini membantu untuk mencegah atau mengurangkan kerosakan yang disebabkan oleh perlanggaran. Komponen yang telah digunakan dalam projek ini adalah FPGA, buzzer dan led. Sensor ultrasonik telah digunakan untuk mengesan jarak. Bahasa pengaturcaraanasa yang telah digunakan dalam projek ini adalah Verilog dan software yang telah digunakan untuk projek ini adalah Altera Quartus II. Projek ini telah berjaya dilaksanakan dan objektif projek ini telah dicapai. Perbandingan antara sistem pencegah pelanggaran yang berdasarkan FPGA dan sistem pencegah pelanggaran yang berdasarkan Arduino telah menunjukkan bahawa sistem pencegah pelanggaran yang berdasarkan FPGA mempunyai prestasi yang lebih baik dan ia lebih dipercayai apabila berbanding dengan sistem pencegah pelanggaran yang berdasarkan Arduino.

ABSTRACT

Nowadays, road accidents accounts for a large number of deaths and injuries. Sudden braking and driving too closely are one of the reasons causing accidents. Human error is the main factor that causing road accident. Many cases of road accident occur because of human's unawareness. Therefore, this study was going to design an algorithm that able to alert driver by detecting the range from the detector to the vehicle or object ahead to reduce the occurrence of road accident. This project was focused only on the algorithm implementation and tested by using ultrasonic sensor. This algorithm contained 2 modes. In driving mode, it would detect the rate of change of range between the detector and the vehicle ahead and also detect the safe distance between two vehicles. In parking mode, the detector would detect the safe distance between the detector and the preceding vehicle as well. This helped the vehicle to prevent or to take reduced damage from collisions. The hardware components that have been used in this project were FPGA, a buzzer and a led. An ultrasonic sensor was used for distance detection. The language that has been used to configure FPGA was Verilog and the software that has been used for the configuration of FPGA was Altera Quartus II. It has been proved that the project could be successfully implemented and the objective of this project has been achieved. The comparison between FPGA based collision avoidance alert system and Arduino based collision avoidance alert system has shown that FPGA based collision avoidance alert system has better performance and it was more reliable to be used compared to Arduino based collision avoidance alert system.

DEDICATION

Special dedicated to my beloved parent, siblings and friends who give me encouragement and support to help me in completing my final year project successfully. My supervisor, En Aiman Zakwan Bin Jidin also gave me a lot of guidance throughout the project implementation. Thank you.

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my project supervisor, En. Aiman Zakwan bin Jidin for his continuous guidance throughout the project and help me in completing my degree final year project. I would like to thank him for his contribution to my project by sharing me with his experience on how to handle the project and how to do research on topics that related to my project. He has shared me with his knowledge and helped me throughout the process of developing the project. He provided me an opportunity to explore to more technological knowledge by using technology device in my project. He has also provided me suggestion when I faced difficulties in doing the project. Besides that, he has helped me in dealing with critical situation and problem solving. Without his guidance and encouragement, this project might not be able to be completed on time. Thank you so much for his contribution.

I would also like to thanks to all my friends who has supported me throughout the process of implementing my final year project. Thank you for their encouragement and support through all the ups and downs during the process of completing this project. Besides that, I would like to thanks to my family for supporting me all the way. Last but not least, I appreciated all the help and thanked you so much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURE

LIST OF ABBREVIATIONS, SYMBOLS AND NOMENCLATURE

ASIC	-	Application Specific Integrated Circuit
ASSP	-	Application-Specific Standard Parts
DC	-	Direct Current
DSP	-	Digital Signal Processing
ECM	-	Engine Control Module
FPGA	-	Field-Programmable Gate Array
GPS	-	Global Positioning System
GSM	-	Global System for Mobile Communication
HDL	-	Hardware Description Language
IF	-	Intermediate Frequency
LCD	-	Liquid Crystal Display
MPGA	-	Mask Programmed Gate Arrays
NRE	-	Non-Recurring Engineering
PLD	-	Programmable Logic Devices
UMC	-	United Microelectronics Corporation
VHDL	-	VHSIC Hardware Description Language
VLSI	-	Very large Scale Integration
V2V	-	Vehicle to Vehicle

CHAPTER 1
INTRODUCTION

1.0 Introduction

Transportation nowadays has become more and more convenient and safe; however, there are still a lot of people dying because of road accident. One of the factors that caused road accidents to occur are sudden braking and driving too closely with vehicle ahead. Majority of road accidents are caused by carelessness of driver, especially when they drive at midnight, the strong desire of reaching destination in a short time always cause them leave unnoticed to the surrounding vehicle. Therefore, a collision avoidance alert system is required to alert driver to avoid any crashes or minimize the impact of collision. A range detector could be used for this. A range detector is a detector that will detect the range between two vehicles on the road. It can be used to detect the safe distance between two vehicles. Therefore, this study was going to design an algorithm that was able to alert driver by detecting the range from the detector to the vehicle ahead. This algorithm contained 2 modes. In driving mode, it would detect the rate of change of range between the detector and the preceding vehicle and also detect the safe distance between two vehicles. In parking mode, the detector detects the safe distance between the detector and the preceding vehicle as well. Once the vehicle was not in the safe condition, the system would activate a signal to alert the driver to reduce the probability of collision.

1.1 Project Background

According to statistics produced by Bukit Aman Traffic Unit (2014), there were about 65,883 accident cases on Malaysian roads involving car drivers and motorcyclists which were at least 5.4 percent higher than the 62519 cases recorded in 2013. From this phenomena, it can be seen that road accidents account a large number of deaths and injuries, the number of road accident increasing year by year. According to Malaysia Institute of Road Safety Research (Miros) director general Professor Dr Wong Shaw Voon, there are several factors that cause road accidents increasing year by year which includes transportation, road constraint, driver behavior, attitude and human error, distracted driving and illegal racing. Among all of these factors, the most causative factor is human factor. Any road accidents that occurred because of human's behavior, human's unawareness of road condition, human's reaction speed and how human make a decision are included in human factor. After these years, it has been found that human factor always the main causes of vehicle collision. Since 1985, it has been found that there were 93% of vehicle collisions were caused by human factor based on British and American crash data. From the article written by Olivia Olarte (2011), Bob Joop Goos, chairman of the International Organization for Road Accident Prevention pointed that road accident is mainly caused by human factor where there were 90% of road accidents were caused by human factor. Jose Miguel, chairman of the Portuguese Society for Road Accidents Prevention, claimed that the quality of road transport system or a break system of a car and how the driver applies the car break system corresponding to the environmental demand is the condition of occurring road accident. Therefore, it is very important to make people to be conscious that the behavior of driver in driving is the main factor that causing accidents. In order to reduce this problem, the traffic safety program should be focusing on people by telling them the consequences of road accidents and the way that all of us can do to prevent road accidents. Besides that, an alert system can be used to alert driver when driver is in dangerous. Due to the behavior of the driver, they rather choose to ignore any risk that may cause accident. With an alert

system installed in a car, it can be used to alert driver so that he or she notice that they are not in the safe area.

It was reported (Free Malaysia Today, 2014) that in 3rd December 2014, a road accident that involved 13 vehicles had occurred in Kuala Pilah. It was an accident occurred because of driver unaware of road condition. Even though the driver had applied on the brakes but it still caused the road accident to occur. The distance between the vehicle and the object was too close to each other, therefore the driver unable to brake in time. 13 vehicles and 39 people were involved in that accident. Since the accident involved a tanker carrying palm oil, the leaking palm oil caused the road to become slippery.

Figure 1.1: Sudden braking causes 13 vehicles and 39 people involved in an accident

According to the statistic shown above, the main factor that causing road accident since 1985 remain unchanged, the danger on the road always cannot be apart from human's behavior. Other than organizing safety programmed, there are still ways of reducing probability of road accidents occur. One of the ways to attract driver's attention during driving is to build a smart collision avoidance alert system in the car to

alert driver. In this project, the implementation of FPGA based smart collision avoidance alert system algorithm has been done to alert driver when they were not aware of road condition while driving.

A field-programmable gate array (FPGA) is an integrated circuit. It was designed to be configured by using hardware description language (HDL). FPGA contains an array of programmable logic block which the logic blocks is used to be configured to perform simple logic gates and combinational functions. There are differences between FPGA and microcontroller. Microcontrollers are mini computers that built in an integrated circuit and perform specific task while FPGAs built up from logic blocks and can be reprogrammed and rewired electrically. FPGAs can run concurrently while microcontroller is always sequential, thus FPGA is faster than microcontroller.

There are two hardware description languages that can be used to configure FPGA which are Verilog and VHDL. Two of these hardware description languages are difference in both their concept and syntax. VHDL is more on ADA programing language while Verilog is more C programming language.

1.2 Problem Statement

Nowadays, car continues to become safer and more convenience, however, they are still a lot of traffic accidents occur. Traffic accidents occur for several reasons. Most of the traffic accidents are caused by driver's careless, especially when vehicle ahead brake suddenly and driver driving too close to the vehicle in front, this cause the driver unable to brake in time and accident occurred. Many accidents occur due to the driver's failure to recognize danger. Many people unable to estimate the safe distance between own vehicle and the vehicle ahead on the road so that when the vehicle in front make sudden braking, the driver still hit the vehicle ahead although they had applied the brakes. Drivers with stronger desire to arrive at their destination as soon as possible are more likely to take risk. Sometimes, when driver driving too long for the journey will

cause them cannot pay well attention on driving and leave unnoticed when the vehicle ahead change their speed. Therefore, an alert system is required to alert the driver with warning when the system determines that there is possibility of collision and allow the driver to keep a safe distance with the vehicle in front.

1.3 Objectives

1. To study on how to detect the rate of change of range between vehicles
2. To develop an algorithm to detect and alert occurrence of slowing or stalling vehicle ahead on FPGA.
3. To analyze the functionality and reliability of the alert system in the aspect of distance detection.

1.4 Work Scope

The aim of this project was to design an algorithm to detect the range between the detector and the vehicle or object ahead on the road by using FPGA to prevent or minimize the risk of road accidents. An ultrasonic sensor was used for testing the functionality of the system. The ultrasonic sensor was connected with an FPGA which was generally configured by using hardware description language. This project was focusing only on the algorithm implementation and testing by using ultrasonic sensor. The test was done in the following situation:

In driving mode:

- If vehicle ahead makes sudden braking, the system should alert the driver.
- If two vehicles are driving too close to each other and their gap is less than the safe distance, the system should alert the driver.

In parking mode:

- If the vehicle is not in the safe distance with vehicle ahead during parking, the system should alert the driver as well. Due to the speed of vehicle in driving and in parking are different, the safe distance in driving mode and in parking mode are also different.

To make an analysis on the overall performance of the alert system based on FPGA, a comparison with the existing system based on Arduino would be done.

1.5 Conclusion

This chapter mainly brief about introduction of this project. Nowadays road accidents account a large number of deaths and injuries. The main reason for a road accident to occur is never been apart from human behavior. Therefore, a FPGA based collision avoidance system is required to alert driver if there are possibilities of collision. The component in building the system included FPGA and the language used to configure FPGA could be either Verilog or VHDL. This chapter also discussed about the objectives and work scope of this project where the main objective of building this project was to detect and alert occurrence of slowing or stalling vehicle ahead on FPGA. This project would focus only on the algorithm implementation and testing by using ultrasonic sensor.

CHAPTER 2
LITERATURE REVIEW

2.0 Introduction

This chapter will discuss the history of the detector and the application of field programmable gate array (FPGA). The comparison between FPGA and other microcontroller as well as the advantages and disadvantages of FPGA will also be discussed in this chapter. Besides that, this chapter will also discuss about two hardware description languages which are Verilog and VHDL, these two languages have their own applications and advantages as well as disadvantages.

2.1 A brief history of detector

Detectors or sensors have been around for a long time in different forms. According to the article written by Ken Smyers (2013), the first electric thermostat came to market in 1883. The inventor of this first thermostat was Warren S. Johnson, a professor at State Normal College in Whitewater. This thermostat has been considered as the first modern, manmade sensor. Based on a research conducted by A.Rogalski (2012), infrared sensors have been discovered in 1940. This sensor has been extensively developed since 1940's.

Other than thermostat and infrared sensor, Tuteja et al. (2014) stated from their article that the first motion sensor was invented by Samuel Bagno in the mid-1940. According to the article, the motion sensor was known as ultrasonic alarm where it sent ultrasonic waves throughout a room. From the ultrasonic wave that spread throughout the room, when the wave was disrupted by something, a return echo triggered the alarm.

Throughout the inventions of these sensors, people start to realize the importance and application of a sensor. The inventions of sensors and their applications had created a commercial demand for people. In 1970s, the principle of Bagno's ultrasonic technology continued to be used, the motion sensor turned into alarm system by using the same principle. The system transmitted an ultrasonic signal and detected changes in the response. If there was a changed occurred in the response, the detector notified the alarm system's control panel. But in 1970s, technologies were not so advanced, false alarms were common, a little sound like clock chiming could change the ultrasonic wave's echo.

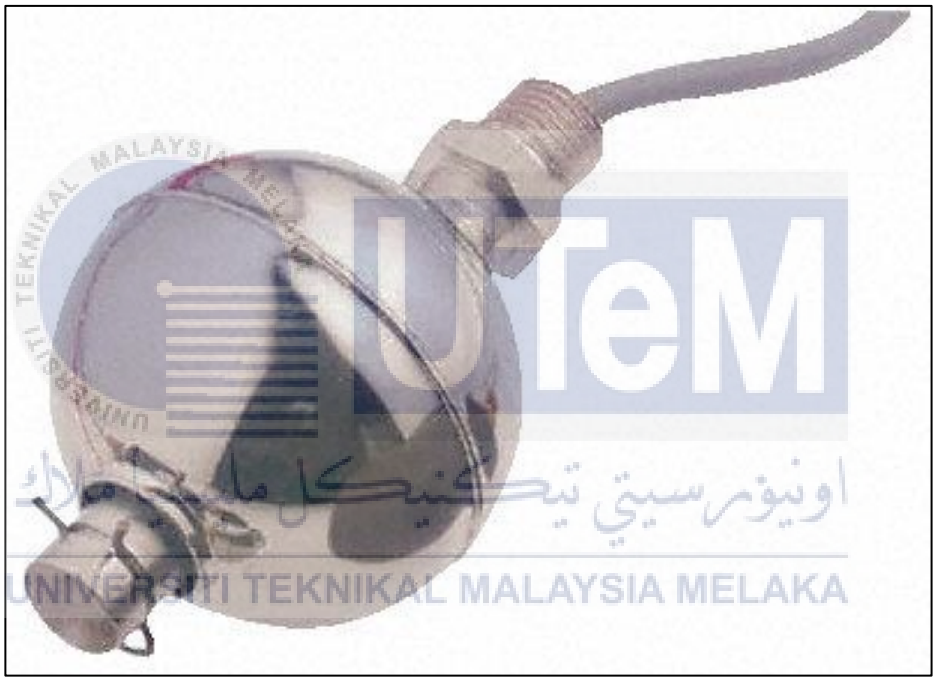


Figure 2.1: The first manmade motion sensor

According to same article, in 1980s, infrared motion sensor began to replace radar sensor. These sensors became more and more widely in used. Initially, the prices of these devices were costly but with these devices became more and more widely in used, the prices became lower and could be afforded by most people. Bagno's device made use of ultrasonic frequencies as well as Doppler Effect.

2.2 Sensor in a vehicle

With sensor has been widely used in vehicle, vehicle continues to become safer and convenient. There are several parts of vehicle that required sensor. In a real world system, a vehicle needs to be well communicated with an outside world. Many conditions have to be considered during driving and there may have some unexpected situations to be occurred. A sensor can help to improve the performance of a vehicle and communicate accurately between vehicle operator and outside world. It also helps to guarantee the safety of drivers and passengers. Therefore, a vehicle needs a reliable, accurate and effective sensor. According to John Vetelino and Aravind Reghu (2010), sensor plays a very important role in a vehicle, the sensor functions may range from a simple sensing of water temperature, oil pressure, and fuel level to the control of the engine and transmission to optimize economy and performance while reducing the potentially dangerous emission effluents. It can be concluded that a vehicle will be unable to function without these sensor. Figure 2.2 shows the areas that sensors are in used.

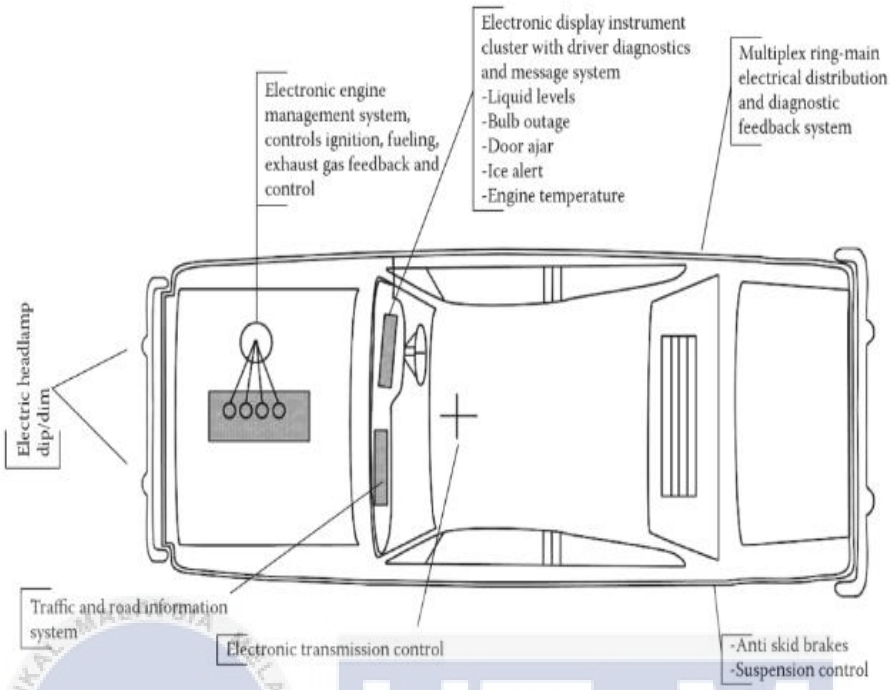


Figure 2.2: Areas where sensors can be used in vehicle. (John Vetelino and Aravind Reghu, 2010)

According to the same book, there are some sensors correspond to the engine and transmission, it can also be known as power train. These sensors are very important to the vehicle performance and relate to engine timing, manifold vacuum pressure and mass airflow, transmission control valve position, transmission input and output speed, exhaust gas oxygen level, and throttle and accelerator position. These sensors must meet the requirements such as accuracy and operating temperature range. In order to meet technical specification, these sensors must also meet space and weight requirements. Besides, these sensors also have to be of minimal cost, reliable, accurate and have high performance. These sensors are very important to a vehicle in order to allow the vehicle run smoothly so that they must be maintained in a good condition.

Gas sensor also used in a vehicle to control the combustion mixtures in car engines. The function of the sensor in a vehicle is to reduce the atmospheric pollution while increasing fuel economy. The combustion control gas sensor was introduced in the

early of 1970s when the Environment Protection Agency (EPA) of the United States legislated Clean Air Act. This Act required automobile manufacturers to minimize exhaust gases by about 90%. Before introducing the combustion control gas sensor, a few method have been tried to reduce exhaust gases including optimizing the fuel supply and the ignition system, this method was failed. After that, they tried to convert the polluting gases into inert species, but this method increased automobile production cost and the fuel economy decreased. Therefore, this method also cannot be used. Failure of previous method and high cost of automobile production caused the industry to look towards catalytic converters. This converter solves the oxidation of CO and CH_x and reduces NO_x to convert all polluting material into harmless byproducts. However, this there-way catalytic converter is effective only if the engine is fed with near-stoichiometric air/fuel (A/F) mixtures. Due to the effectiveness of this method, this A/F mixture requirement created a market for an A/F sensor.

Sensor technology has become more and more advance, there have also been advances in speed detection technology. Kumar et al. (2014) stated that speed detection of a vehicle can be done by using laser guns. Recently, laser gun has been used by police to enforce speed limits. Speed of light from a laser gun is much faster compared to sound sensor.

2.3 Development of anti-collision system for vehicles

In recent years, the number of road accidents keep increasing resulted in many people died. Some of them were because of environment error but most of them caused by human error. Therefore, many ideas and research have been proposed to reduce the occurrence of road accidents especially accidents caused by human error. Shival Dubey and Abdul Wahid Ansari (2013) stated that many researches have been conducted on the anti-collision system device based on different components and provide different ways to avoid collision including used wireless network, vehicle to vehicle (V2V) communication, global positioning system (GPS) and radar implementation. Based on

the paper conducted by Ajit Kumar et al. (2014), occurrence of a road accident can be reduced by building an anti-collision system in a vehicle. According to the paper, the system detects the speed and distance of vehicle by using BLINDER laser detector. The information of vehicles can be shared by using laser beam detection. This vehicle detection system is used to provide alert message and to decrease the speed of vehicles. When two vehicles are on the road, the system detects the distance between the vehicle and vehicle ahead. If distance of the two vehicle too close to each other and crosses the safe distance, the system will decelerate the speed automatically or apply emergency brake.

As stated in the same paper, this system can be divided into three parts. First part is source of laser light beam to produce laser beam. Second part is BLINDER laser detector which will detect the speed and distance of vehicle according to the reflection beam of laser light. Third part is alert and control system based on the detection of the speed and distance that obtained by using BLINDER laser detector. The output of the system is produced based on the comparison between the input and stored value.

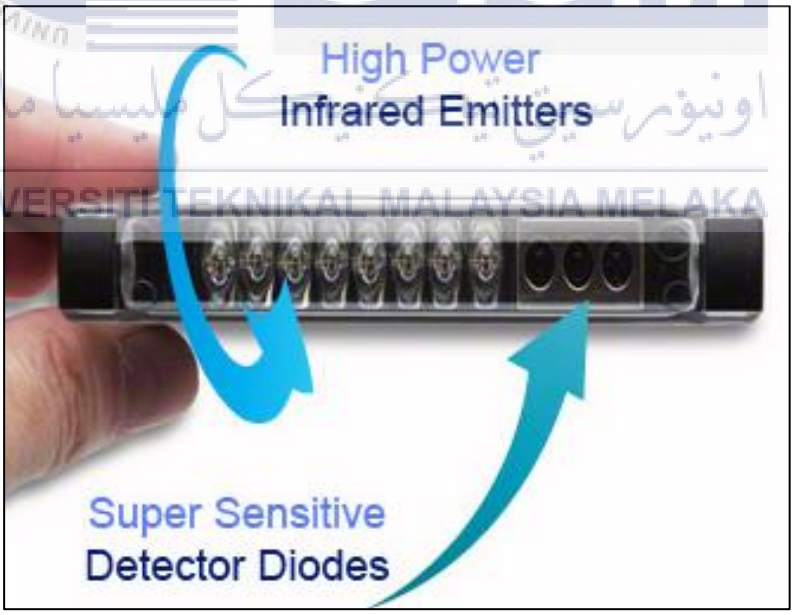


Figure 2.3: BLINDER laser detector that used to detect the speed and distance of vehicle.

The BLINDER laser detector works better under full daylight or at night. This is because the BLINDER laser detector detects the infrared light of laser beam. The infrared light is also a part of sunlight. At daytime, the brighter the day, the more infrared light is scattered. At night, there is only infrared light is presence, makes the BLINDER laser detector detects easier. The effectiveness of the laser detector is depends on the color of the object, the brighter the object, the easier the detection. According to the paper, there are also disadvantages of using this BLINDER laser detector. It will only detect object with color. For object that are transparent such as glass, the BLINDER laser detector will not be able to detect because this transparent object do not have reflection.

When the distance between two vehicles crosses the safe distance, the system will decrease the speed of current vehicle automatically. The speed control of this system is done by a DC motor where the DC driver called L293D is used. This DC motor can control two DC in the same time. The deceleration of speed can be done by generate a back EMF to change the direction of rotation. Power supply of DC motors is depends on distance between two vehicles.

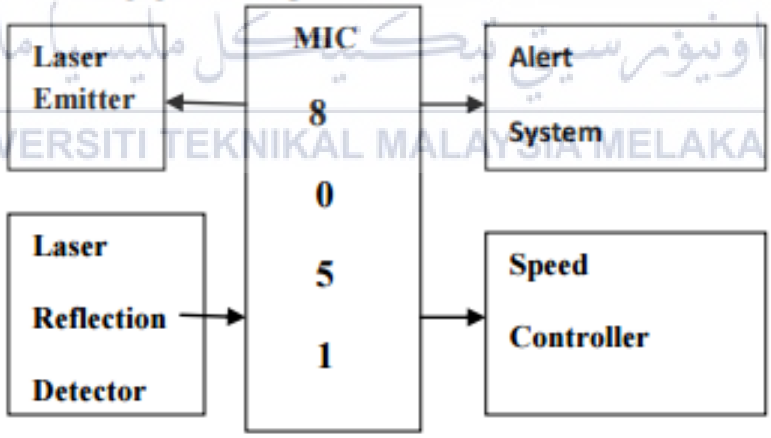


Figure 2.4: Block diagram of the vehicle anti-collision system

Triveni Shinde and Prof.B.V. Pawar (2013) had proposed a car Anti-Collision and Intercommunication System using Communication Protocol to help reducing the occurrence of road accident. This system is run by using a microcontroller to control the brake system. The system detects the object or vehicle in front by using ultrasonic

sensor. If there is object or vehicle detected, the ultrasonic sensor continues to detect the distance between current vehicle and vehicle ahead. If two of these vehicle are in the safe distance, the vehicle continue to run with constant speed but when the system detected that two of these vehicles are not in the safe distance, the system will trigger the microcontroller to start applying brake until the distance is within the safe range. As long as two of these vehicles not in the safe range, the process will be continuous until the vehicle comes to a stop. The microcontroller used in this system is ARM7 which is used to control the whole system includes the DC motor. ARM7 will give instruction or in other words to increase or decrease the speed of DC motor via Pulse width modulation based on the detection result obtained by ultrasonic sensor.

As stated in the same paper, this system not only able to know the distance between two vehicle, this system also able to communicate with the vehicles that are close by. The communications between two vehicles are enabled by using zigbee. If the vehicle ahead brake drastically or decrease their speed suddenly, the vehicle at the back can be noticed so that the driver can apply on his or her brake to avoid any collision. Besides that, this system also contains GPS/GSM to allow driver to communicate about the road condition, traffic condition as well as weather condition.

The components that used in this anti-collision and intercommunication system includes buzzer, DC motor, Liquid Crystal Display, ARM7 Microcontroller, Ultrasonic sensor, ZigBee, Global system for mobile communication (GSM) and Global Positioning System (GPS). The buzzer is used to alert the driver when any emergency occurred while the Liquid Crystal Display (LCD) is used to display the output of the application. It can also be used to check the speed of car, location of car as well as the question asked by other car. ZigBee that used to enable the communications of two vehicles is a specification of suit of high level communication protocol that based on IEEE802 standard. The main usage of ZigBee is to transmit data over a longer distance.

Besides that, there is also vehicle anti-collision system using electromagnet and ultrasonic sensor. According to the paper proposed by Shival Dubey and Abdul Wahid Ansari (2013), vehicle anti-collision system using electromagnet and ultrasonic sensor works in two stages. First, the range detector which is ultrasonic sensor will continuously detects the distance between two vehicles moving and sends it to the

Engine Control Module (ECM). Second, ECM received the input and use these input to decide whether to activate the sensor strip for Electromagnetic induction. Shival Dubey and Abdul Wahid Ansari declared that their system is an automatic vehicle anti-collision device that can be used to reduce the possibility of vehicular head to head or head to back collision. It used ultrasonic sensor to detect the range between vehicles and generate an electromagnetic field to repel vehicles. This device not only provides alert system to driver but also automatically activate the safety switches before emergency situation occurred.

As stated in the paper, the microcontroller that used in this system is ATMEGA 16 that will receives response signals from ultrasonic sensor. This signal will be sent to ECM and used it to trigger solenoids to create electromagnetic field. The distance between two vehicles is continuously read by the ultrasonic sensor and the dashboard of the vehicle will show the output. If the distance goes on reducing until crosses the safe distance, circuits starts working to create electromagnetic field. Figure 2.5 shows the block diagram of this anti-collision system.

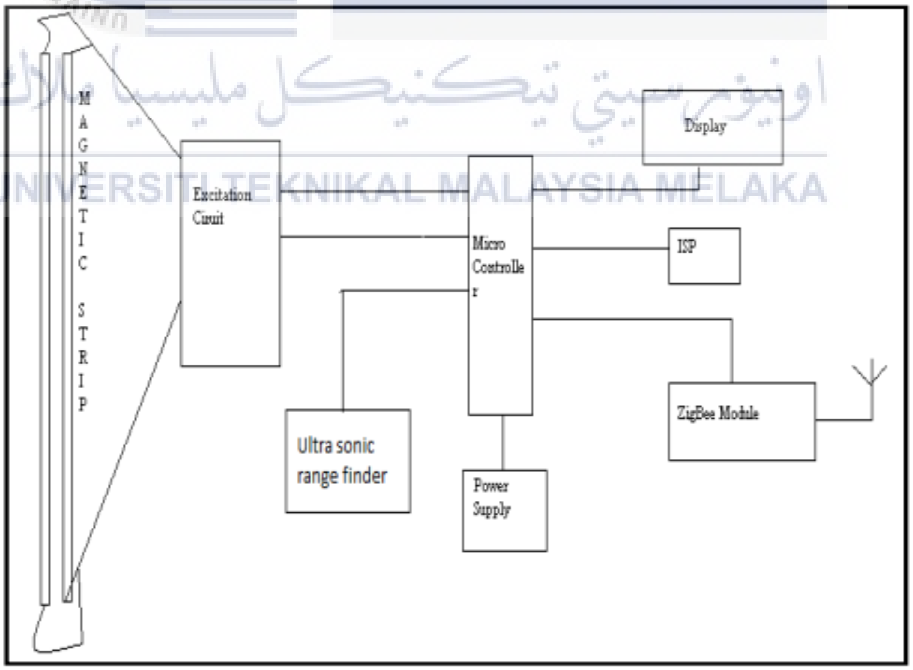


Figure 2.5: Block diagram of Anti-collision system

The microcontroller which is ATmega 16 is configured by using C programming language. It has been programmed to test the hardware. WinAVR[2] is used to program the microcontrollers C language. It is a compiler for a high level language that helps to minimize the production time. After compilation, C program was converted into machine language which can only be understood by a microcontroller. Machine language (hex) file of the compiled program was burned into the program memory which attached to a PC's peripheral.

2.4 Field Programmable Gate Arrays

2.4.1 Overview of Field Programmable Gate Arrays

Clive Maxfield (2011) stated that Field programmable gate arrays (FPGAs) are digital integrated circuit that are designed to perform variety of tasks such as combinational functions, or simply logic gates like OR, AND, NOR and XOR gates. It can be configured or reprogrammed by customers or designer engineers after manufacturing. FPGAs contain an array of programmable logic blocks along with configurable interconnects between these blocks. Not every FPGA can be programmed for unlimited times. It depends on how these FPGA are implemented. Some FPGAs may be able to be programed for many times but some FPGAs may only able to be programmed for a single time only. One-time programmable refer to the device that can be programmed for only one time.

There are various type of digital integrated circuit on the market such as Programmable Logic Devices (PLD), Application Specific Integrated Circuit (ASIC), and FPGAs. Similar to FPGAs, PLDs also consider devices that can be configured by engineers in the field to perform different tasks. The internal architecture of PLDs are predetermined by the manufacturer. However, these devices contain only limited number of logic gate, and the functionality of these

devices are limited and simpler. Compared to FPGAs that contain large number of logic gates, PLDs may not be able to perform complex task like FPGA.

On the other hand, ASIC and Application-Specific Standard Parts (ASSP) contains a large number of logic gate, it can contain hundreds of millions of logic gates and can be used to create tremendous and complex functions. Similar to other digital integrated circuit, ASIC and ASSP also built to implement specific task by user. However, ASIC is only built and designed to order for use by a specific company. An ASSP is designed and built to multiple customers. There is a disadvantage of using ASIC. The devices that is designed and built based on ASIC cannot be modified without creating a new version of the device. This is because the final design of the device is "frozen in silicon".

Although ASIC contain a large number of logic gate, complexity and provide very good performance, the process of designing and building one is very time-consuming and costly. In comparison with FPGA, the cost of an FPGA design is much reasonable than that of an ASIC. Therefore, FPGA stands in the middle between PLDs and ASICs due to the functionality of FPGA is better to be used than PLDs and cost of designing and building is much lower than ASIC.

Table 2.1: Comparison between PLD, FPGA and ASIC

Criterion	PLD	FPGA	ASIC
Performance	Medium	High	Very high
Development Cost	Low	Medium	Very high
Design change Cost	Medium	High	Very high
Time to market	Short	Medium	Long

Vaughn Betz, Jonathan Rose, and Alexander Marquardt (2012) declared that FPGA has two key advantages: First, lower non-recurring engineering (NRE) cost, Second, faster time-to-market. In order to implement a circuit with other circuit implementation such as Standard Cells, one is required to send the completed design to a silicon foundry to manufacture a chip according to the

design. The NRE fees to manufacture the first chip normally is around $100 000 and $250 000 where this fee includes the cost of making lithography masks and of running a new design through the fabrication plant. The non-recurring engineering (NRE) of FPGA is lower because the design that is implemented in a FPGA can be easily by programming the FPGA based on the desired functionality, there are no NRE cost need to be charged. Time-to-market is another benefit of FPGAs. The process of completed a chip usually takes 6-8 weeks. If there are problems found in the completed chip, the chip has to be thrown away and another 6-8 weeks is needed to fabricate another new chip. On the other hand, the process of program a FPGA takes only a few second. If there are bugs found in the chip, it can be corrected by reprogramming the FPGA. This takes only a few minutes. Fabrication of FPGA in a chip is a fast process, thus, it is faster time-to-market.

Compared to other circuit implementation such as Mask Programmed Gate Arrays (MPGA), a circuit that implemented in an FPGA is ten times larger and three times slower than the same circuit implemented in MPGA in a same process. Due to the large size of FPGA circuitry, the FPGA implementation is expensive than MPGA and the limited speed of FPGA cause FPGA unable to make of their use in very high-speed designs.

Same to what has been stated by Clive Maxfield, Vaughn Betz, Jonathan Rose, and Alexander Marquardt also stated that FPGA is built up with a large amount of programmable logic blocks. Each logic block carry out a small amount of digital logic and programmable routing which connects the logic block inputs and outputs to form larger circuits. The global routing architecture of a FPGA specifies the width of several types of wiring channels within the chip.

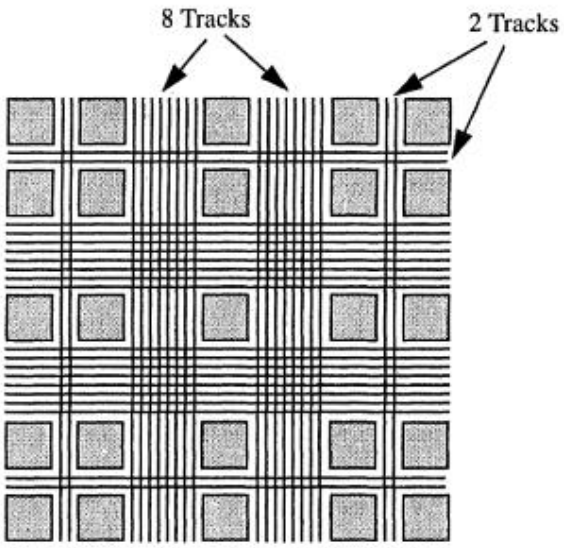


Figure 2.6: Global routing architecture

From Figure 2.6, it can be noticed that the channels near the center of FPGA is wider than other channel. All routing resources in FPGA are prefabricated, thus, manufacturer is the one set the width of all the routing channels. Therefore, it is required to find the distribution of routing resources, or known as tracks, to the different types of channels that permits their efficient utilization by the largest class of circuits. In case there are too few tracks in certain area of the chip, many circuit will be not able to route, but if there are too many tracks, it could be wasted.

With FPGA using cluster-based logic blocks, interconnections in FPGA can be made faster. The use cluster-based logic blocks in a FPGA allow many connections can be made using the local interconnect within a cluster. Cluster is refers to grouping of logic blocks. Figure 2.7 shows the cluster of look-up tables (LUTs) and flip flop along with local routing to interconnect the LUTs within a cluster.

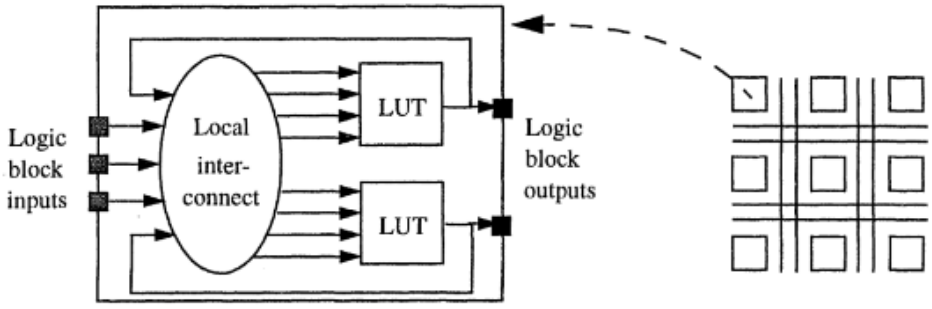


Figure 2.7: Logic cluster that containing two LUTs

These cluster-based logic blocks can increase the speed of FPGA due to the interconnections in FPGA.

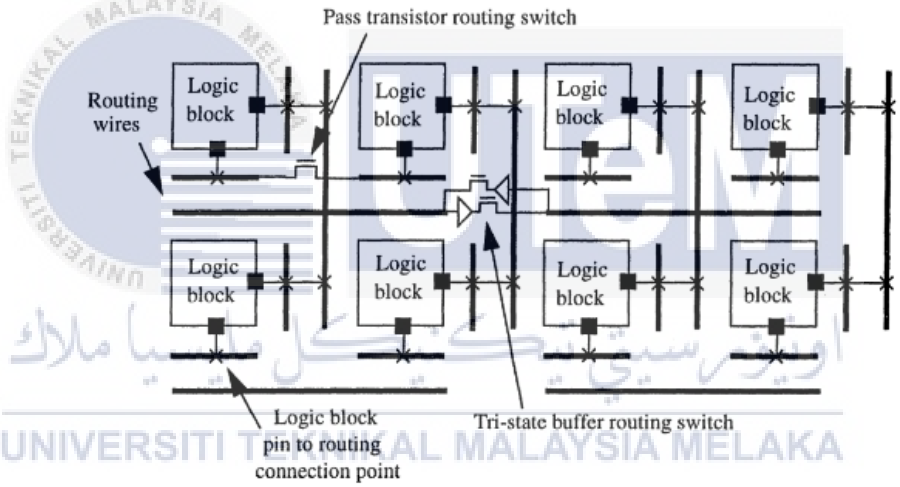


Figure 2.8: Detailed routing architecture

Figure 2.8 shows the example of detailed routing architecture which describes how logic block inputs and outputs can be connected.

2.4.2 Application of FPGA

According to Clive Maxfield (2011), FPGA was first introduced in 1980. Initially, FPGA was used for glue logic, medium-complexity state machines, and

data processing task. At the early 1990s, due to the increasing of size and complexity of FPGA, it start to be used for processing large blocks of data and pushing that data around, their market at that time was mostly on telecommunications and networking arenas. As the functionality and performance of FPGA continue to increase, the market of FPGA continues to expand. At the end of 1990s, the market of FPGA includes automotive and industrial applications.

Normally FPGA is used to build ASIC designs and to provide a hardware platform to verify the physical implementation of new algorithms. Since the cost of development a FPGA is cheaper and it is faster time-to-market, many vendors are finding their ways to make it as final product and so that it can compete directly with ASIC. High performance of FPGA that contains a large number up to millions of gates are currently available. It is used for embedded microprocessor cores, high speed input or output devices and etc. Nowadays, FPGA can be used to implement a lot of devices, including software-defined radio, communications devices such as mobile phone; radar, image, and other digital signal processing applications; and system on chip (SoC) components that contain both hardware and software elements.

Major market segments for FPGA

Table 2.2: Major market segments for FPGA

Application	Explanation
ASIC and custom silicon	FPGAs are increasingly being used to implement designs that previously were realized by using only ASICs and custom silicon.
Digital Signal Processing	Today's FPGA can contain embedded multipliers, dedicated arithmetic routing, and large amounts of on-chip RAM, all of which facilitate DSP operations. When coupled with the massive parallelism provided by FPGAs, this results in outperforming the fastest DSP chips by a factor of 500 or more.

Embedded microcontrollers	Low-cost microcontrollers, which contain on-chip program and instruction memories, timers and I/O peripherals wrapped around a processor core, and used in small control functions. With falling of FPGA prices, however, and increased capability to implement a soft processor core combined with a selection of custom I/O functions, FPGAs are becoming increasingly attractive for embedded control applications.
Physical layer communications	FGPAs have long been used for the glue logic that interfaces between physical layer communication chips and high-level networking protocol layers. Now high–end FPGAs can contain multiple high speed transceivers, which means that communications and networking functions can be consolidated into a single device.
Reconfigurable computing (RC)	FPGAs have created this new market segment. This refers to exploiting the inherent parallelism provided by FPGAs to 'hardware accelerates' software algorithms. Various companies are currently building huge FPGA-based reconfigurable computing engines for tasks ranging from hardware simulation to cryptography analysis to discovering new drugs.

2.4.3 Comparison between FPGA and microcontroller

Similar to most microcontrollers, FPGA do contains memory. In most FPGA, the logic blocks of FPGA contain memory element such as flip-flops or more complete blocks of memory, but FPGA is different with microcontroller, it is not a family of microcontroller. With the development of FPGA, a microcontroller can be replaced by a FPGA, therefore, there has always been hot discussion between differences of FPGA and microcontroller. The performance of FPGA has been increased in these years.

Based on what has been stated by Aflab Sarwar (2012), differences of FPGA and microcontroller can be discussed in several criteria. In terms of structure, microcontrollers are mini computers that built in an integrated circuit and perform specific task while FPGAs built up from logic blocks and can be reprogrammed and rewired electrically. This makes FPGA more flexible to be used. In terms of power, FPGAs consume more power than microcontroller, thus, microcontroller is more power efficient than FPGAs. In terms of speed, FPGAs can run concurrently while microcontroller is always sequential. In this case, the concurrently processing made the system better and more suitable to be used which it can transmit and receive signal and process the signal at the same time. Therefore, FPGAs are more suitable for real-time applications such as executing digital signal processing (DSP) algorithms.

In terms of flexibility, FPGAs are more flexible since it allow user to add or subtract the functionality as required. The functionality of microcontrollers is fixed during manufacture, thus the functionality of a microcontroller cannot be changed as required. In terms of development time, FPGAs take longer time than microcontroller. The peripherals of microcontrollers are readily available and have been pre-tested by vendor. User does not have to worry about their functionality. In terms military application, FPGAs are more likely to be used in military application because FPGA is hard-wired, the memory areas of FPGA not easy be attacked or destroyed by alpha rays, the functionality of FPGAs are not easy be corrupted. Besides that, life time of FPGA based development is longer. This makes it can be adopted for advanced chip. In comparison to microcontroller, the life times of microcontrollers are shorter, it change too frequent and a lots of re-work required need to do to keep pace with changing technology. In terms of costs, the costs of microcontrollers are much lower than FPGA.

According to the same article, there are also some vendors such as Altera, Xilinx, and Atmel that used microcontroller and FPGAs simultaneously. They provide configurable logic along-with processor core as well. But it is very difficult to work on both together due to microcontroller are very different with

FPGA. Although FPGA use Hardware Description Language such as Verilog that may look similar to C in syntax, but it is very different and confuse in use. There are many pros and cons of microcontroller and FPGA. For the field that demands a lot of DSP work and heavy DLD works are involved, FPGA is more suggested, otherwise, microcontroller can be used.

In terms of transient failure, Samarjit Chakraborty and Jorg Eberspacher (2012) said that FPGAs are sensitive in transient failures. As stated above, FPGA can run concurrently, due to no temporal logic correlation between two processing cycles, the transient failure of FPGA is negligible. Once FPGA detected that program code is affected, the failure will be remain there until the next system reboot which also means that the configuration of FPGA must be tested against systematic failures and transient-permanent.

The FPGA processing result can be tested by integrate the test pattern of beginning and ending of every image. This pattern is processed with the same algorithm like the current image. The correct result of the test pattern will be store in the microcontroller memory and it is used to compare with the current result.

In microcontroller, the transient failure of microcontroller is not negligible. In comparison with FPGA, what makes microcontroller different in term of transient failure is that the post processing task of microcontroller contains tasks with temporal correlation since microcontroller is different with FPGA, it cannot run concurrently. Permanent failures in microcontroller are also fatal. Both must be detected solidly. A concept with comparator and two different algorithm chains prevents false action. There are two assumptions required for this comparator concept: First, based on the same FPGA preprocessing it is possible to develop two sufficient diverse algorithms for the complete post processing. Second, two diverse algorithms deliver not the same result on defective hardware.

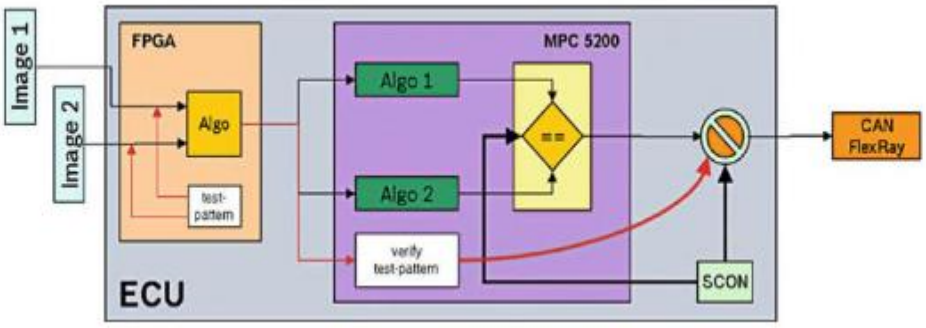


Figure 2.9: Safety concept with test pattern and comparator

According to Subhas C. Mukhopadhyay and Joe-Air Jiang (2013), FPGA has been widely used in wireless system. The main benefit of FPGA which are "reprogrammability" and faster time-to-market make it able to compete with other components or devices. FPGA have become a dominant technology in the first-stage intermediate frequency (IF). It has been applied to variety of wideband wireless application. However, there are still discussions on the differences of using FPGA and microcontroller in wireless system. Table below shows the comparison between FPGA and microcontroller applied in wireless system.

Table 2.3: Comparison between FPGA and microcontroller applied in wireless system.

FPGA	Microcontroller
Permits design upgrades with no hardware replacement.	Hardware is fixed and can't be upgraded.
Provides a good platform now a days on ASIC design.	ASIC is not possible.
Support a wide range of Digital Signal processing (DSP) operations.	Not efficient for implementing a complicated DSP.
Power consumption and chip size are still considerable.	Some processors now a day can be found with a very small size and low power.

2.4.4 Comparison between Xilinx and Altera

There are many vendors of FPGA, among all of these vendors, the most two famous vendors are Altera and Xilinx. Jeff Johnson (2011) stated that Xilinx has been the leader of FPGA in the market for many years. Xilinx was the inventor of FPGA. It was founded in 1984 and introduced its first product in 1985. Xilinx cooperates with leading semiconductor manufacturers, for example: IBM Microelectronics, United Microelectronics Corporation (UMC) and Seiko. On the other hand, Altera was founded in 1983. The first commercial product introduced by Altera was reprogrammable logic device (PLD) in 1984.

In recent years, Xilinx has covered the high-end FPGA family with Virtex series and covered the low-end FPGA family with Spartan series while Altera offers Stratix series at the high-end FPGA family and Cyclone series at the low-end FPGA family. All of these series in Xilinx as well as in Altera are direct components.

There are several key factors that can be used to distinguish FPGA from Xilinx and Altera:

- Processor system

  Both of Xilinx and Altera are using an embedded processor, a dual-core Cortex-A9 with NEON extensions. It is five to ten times faster than a soft-core.

- Memory

  Altera has L1 2x32K per core, L2 512K shared, 64K RAM while Xilinx has L1 2x32K per core, L2 512K shared, 256K RAM.

- Hardware Peripherals

  The hardware peripherals of both FPGAs are quite similar. The main difference is that Xilinx have ADC on chip (XADC) on their high end families for system management but Altera does not have. This makes Xilinx more useful than Altera. In order to transfer all the data in and out of these peripherals, both of Xilinx and Altera FPGA contain 8 channel

DMA engines. The peripherals on both of it are wired to pin through a big multiplexer.

- FPGA fabric

  Altera have two ports (one fast, one slow) to transfer data to FPGA and one port to transfer data from FPGA. There are also memory ports in Altera FPGA. In larger device, there are 1 to 3 more hard memory controllers connected directly to the FPGA fabric. In Xilinx, there are only two ports to transfer data in and out from FPGA and 4x64 bits ports from FPGA to memory.

2.5 Hardware Description Language

There are two hardware description languages that can be used to configure FPGA. These two languages include Verilog and VHDL. Based on James E. Stine (2015), Verilog was introduced by Phil Moorby in 1984 at the Gateway Design Automation conference. It has become an IEEE standard in 1995 as IEEE standard 1364-1995[IEEE95]. The overall goal of the Verilog language is the framework and methodology for modeling and stimulation. There are two important aspects of Verilog: First, levels of system specification which will describe the behavior of digital system and how it provides the mechanism that makes it work. Second the system specification formalism which allows designers to utilize abstractions to represent their Very large Scale Integration (VLSI) or digital system.

Digital system such as VLSI are highly complex, it may contain millions of elements. By configuring these highly complex digital systems, Verilog language provides a wide range of levels of abstraction. At the most detailed level, Verilog provide access to computer aided design tools to contribute in the design process. With VLSI implementation that using place and route programs, Verilog provides better ways for modeling these circuits. It also allows engineers to increase the speed and decrease the area of the VLSI chip by optimizing the logical circuits and VLSI layouts. Thus,

Verilog can be known as the more efficient and useful tool for making design of VLSI and digital systems to an engineer.

For a VLSI designer, Verilog language is easier to use and there are many Verilog compilers publicly as well as commercially available.

VHDL is another HDL language that used to describe digital and mixed signal systems. The application of VHDL includes field-programmable gate arrays and integrated circuits as well as parallel programming language. VHDL was introduced at the behest of the U.S Department of Defense. VHDL borrows both concepts and syntax from the Ada programming language. This is because the Department of Defense requiring as much of the syntax as possible to be based on Ada. In order to avoid re-inventing concepts that had already been thoroughly tested in the development of Ada, VHDL borrows both concepts and syntax from the Ada programming language.

Normally VHDL is used to write text models that describe a logic circuit. The logic designed model is commonly processed by a synthesis program. VHDL is more on Ada, it is strongly and not case sensitive, but there are also features in VHDL that are not found in Ada such as an extended set of Boolean operators including NAND and NOR. In Ada, most programming language only ascending indexing is available, but in VHDL, its programming language include ascending or descending direction where these both conventions are used in hardware. VHDL can be used as general-purpose language for text processing due to VHDL has file input and output capabilities. These files are usually used by a simulation test bench for stimulus or verification data.

VHDL is a dataflow language, it allows the description of a concurrent system. Another benefit of VHDL is that once the VHDL project is created, it can be used in many other projects. Besides that, a VHDL project is portable. Being created for one element base, a computing device project can be ported on another element base, for example VLSI with various technologies.

Both Verilog and VHDL have their own advantages and disadvantages. VHDL is more on ADA programing language in both concept and syntax while Verilog is more C programming language. VHDL uses strong typing which does not allow the intermixing of variables with different classes. Verilog is a weakly typed language which opposite with VHDL. In terms of case sensitivity, Verilog is case sensitive and not easy to

recognize a variable if the case used is not consistent while VHDL is not case sensitive. VHDL allow user to change the case as long as the character in the name, and the order, stay the same. Verilog is simple than VHDL, therefore, it is easier to learn Verilog compared to VHDL because the way to write in Verilog is more like C programming which are more common to most programmer. VDHL is a little bit more difficult to learn and program.

2.6 Ultrasonic sensor

Based on Rockwell Automation (2016), ultrasonic sensor is used to detect the distance of object ahead. The concept same with bats use echolocation to identify objects in their surroundings. With the echolocation, bats can estimate the distance of object around and it helps bats to hunt their food. Same with dolphins, they also use echolocation to estimate distance. Ultrasonic sensors emit a sound wave that reflects off of objects entering the wave field. The sound wave is then reflected and received by ultrasonic sensor. By emitting and receiving the sound wave, the distance between current object and object ahead can be determined. Detection of the sound wave will produce an output signal for use by an actuator, controller, or computer. The output signal can be analog or digital.

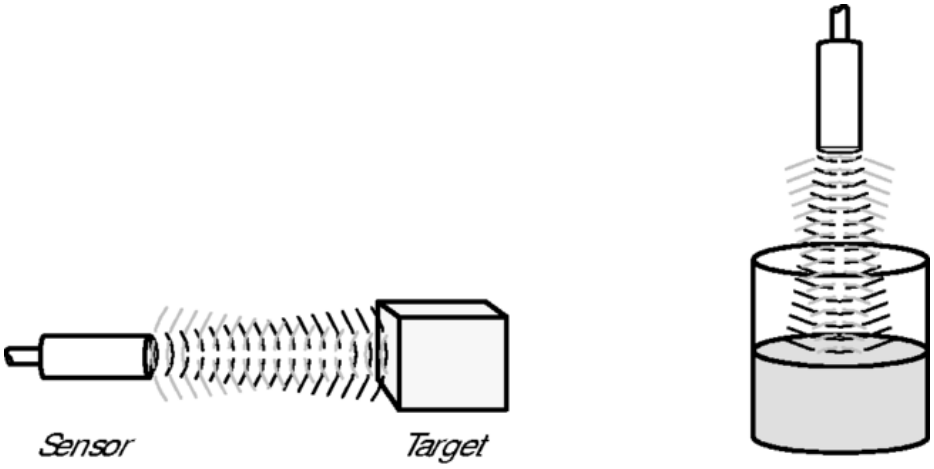


Figure 2.10: Sound wave can reflect both solid and liquid target

Ultrasonic sensing technology is according to the principle of velocity. Sound has a constant velocity which is 343.2m/s. The time for emitting and receiving sound wave is directly proportional to the distance to the object. Therefore, ultrasonic sensor is usually used for distance measurement applications such as distance control or liquid level control.

Ultrasonic sensor can detect most of the objects have sufficient reflectivity includes metal or nonmetal, clear or opaque, liquid, solid, or granular. There are also materials like sound absorbing materials that do not have ideal reflectivity such as foam, soft rubber, cloth, and flour. These lower the performance of ultrasonic sensor, because the poor reflectivity of these objects makes the ultrasonic sensor unable to estimate the actual distance to these objects.

The four basic components of an ultrasonic sensor :

- Transducer/receiver
- Comparator
- Detector circuit
- Solid-state output

Figure 2.11: Components of ultrasonic sensor

Transducer can be also known as receiver which it emits sound wave from the ultrasonic sensor to the object and receives the reflection of the sound wave from the object. The comparator and detector circuit calculates the distance to the object based on the reflected echo. The calculation of distance can be done by comparing the time

frames between the emitting and receiving sound waves to the speed of sound. After the calculation of distance by comparator and detector circuit, the solid-state output produces an electrical signal to be interpreted by an interface device. There are two electrical signals that will be produced which are signal from digital sensor and signal from analog sensor. Signal from digital sensors shows the presence or absence of an object in the sensing field while signal from analog sensors shows the distance to an object in the sensing field.

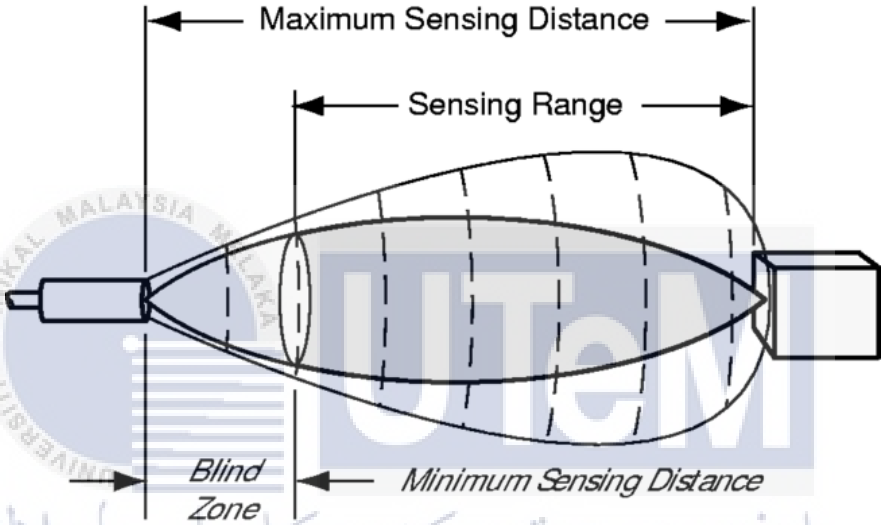


Figure 2.12: Distance of ultrasonic sensing

Figure 2.12 shows the distance of ultrasonic sensing. The sensing area of an ultrasonic sensor is the area between the minimum and maximum sensing limits. From the figure, it can be seen that there is a blind zone. This blind zone indicates the unusable area. The sensor is unable to receive the reflection accurately if the ultrasonic beam leaves the sensor, reach the target object and reflects before the sensor has completed its transmission. The minimum sensing distance is the minimum distance an object can be from the sensor. In this area, it will not have reflecting echoes that will be ignored by the sensor. Maximum sensing distance indicates the maximum distance that the sensor is capable to see the target object and material. The easier an object is to detect, the longer the maximum sensing distance can be.

There are several environmental situations that need to be considered:

- Ambient noise

  The noise suppression circuitry in ultrasonic sensor allows them to function reliably in noisy environments.

- Air pressure

  Measurement accuracy of the ultrasonic sensor may be affected under normal atmospheric pressure. It is not encouraged to use ultrasonic sensor in high or low air pressure environment because the transducer or sensor face may be damaged under extreme pressure.

- Air temperature

  Increasing in temperature will slowing down the speed of sound, thus, the sensing distance will be increased.

- Air turbulence

  Air turbulence affects the reflection of sound wave, it may cause refraction of sound wave and weaken or divert the sound wave to the extent that it is not receive at all.

2.6.1 Advantages of an ultrasonic sensor (Rockwell Automation, 2016)

There are several advantages of ultrasonic sensor:

- The detection of an ultrasonic sensor does not depend on surface color or optical reflectivity of the object.
- Ultrasonic sensors with digital outputs have high sensing accuracy. It can detect an object ahead directly by ignoring the background of the object.
- The reflection of analog ultrasonic sensors is linear with distance. The visual indication of target distance can be obtained by interfacing the sensor to an LED display. Thus, ultrasonic sensor is usually used for level controlling and linear motion controlling application.

2.6.2 Disadvantages of an ultrasonic sensor

There are also several disadvantages of using ultrasonic sensor:

- The surface of the target object to be detected must be squarely or perpendicularly to receive sufficient sound echo. Also, different sensor type requires different minimum target area.
- The ultrasonic sensors are still likely response to false noise such as “hissing” sound produced by air hoses and relief valves.
- There is minimum sensing distance in ultrasonic sensor.
- Ultrasonic response will be affected by environment changes.
- It is not easy to detect sound absorbing materials such as clothes, foam and etc.

2.7 Conclusion

This chapter mainly discuss about literature review on history of detector, existing project of anti-collision system of a vehicle and component used in this project. A detector has been widely used for a very long time, the first detector was invented in 1883. It has been used in most of the technology device including range detection in anti-collision application. Many anti-collision system based on different component have been introduced to avoid car accidents. These components include radar, infrared sensor, ultrasonic sensor and etc. This chapter also discuss about the components that are going to be implemented in this project such as FPGA and ultrasonic sensor. FPGAs are digital integrated circuits that are designed to perform variety of tasks. Compared to ASIC, cost of FPGA is much lower than ASIC. Changing of design in FPGA is much easier and time-to-market of FPGA is faster. In comparison with microcontroller, the main benefit of FPGA is that it can run concurrently while microcontroller is always run in sequential. Ultrasonic sensor is another component that will be used in this project, it is used to detect distance between current object and object ahead by calculate the time

of emitting and receiving sound wave correspond to the speed of sound. Verilog and VHDL are the languages that can be used to configure FPGA. The main difference of Verilog and VHDL is that the concept and syntax of Verilog is more on C programming while the VHDL is more on Ada programming.

CHAPTER 3
RESEARCH METHODOLOGY

3.0 Introduction

This chapter will mainly discuss about the research methodology of this project. It covers all the detail explanations of methodology that being used in this project. A project methodology is very important to ensure a project can be completed smoothly by follow the correct sequences. The project methodology normally consists of a few stages or phases to ensure the project objective can be achieved. Besides that, this chapter also discuss about the project overview by including the block diagram and related calculation of this project. Implementation flowchart and project flowchart will also be included in this chapter so that there is a clear picture on how did the project be implemented and how did the project run. Furthermore, this chapter will also cover the project schedule. The project schedule would list out the activities that would be done, together with the duration for each of the activity. This was necessary to ensure the project could be done in time.

3.1 Flow Chart of Project Methodology

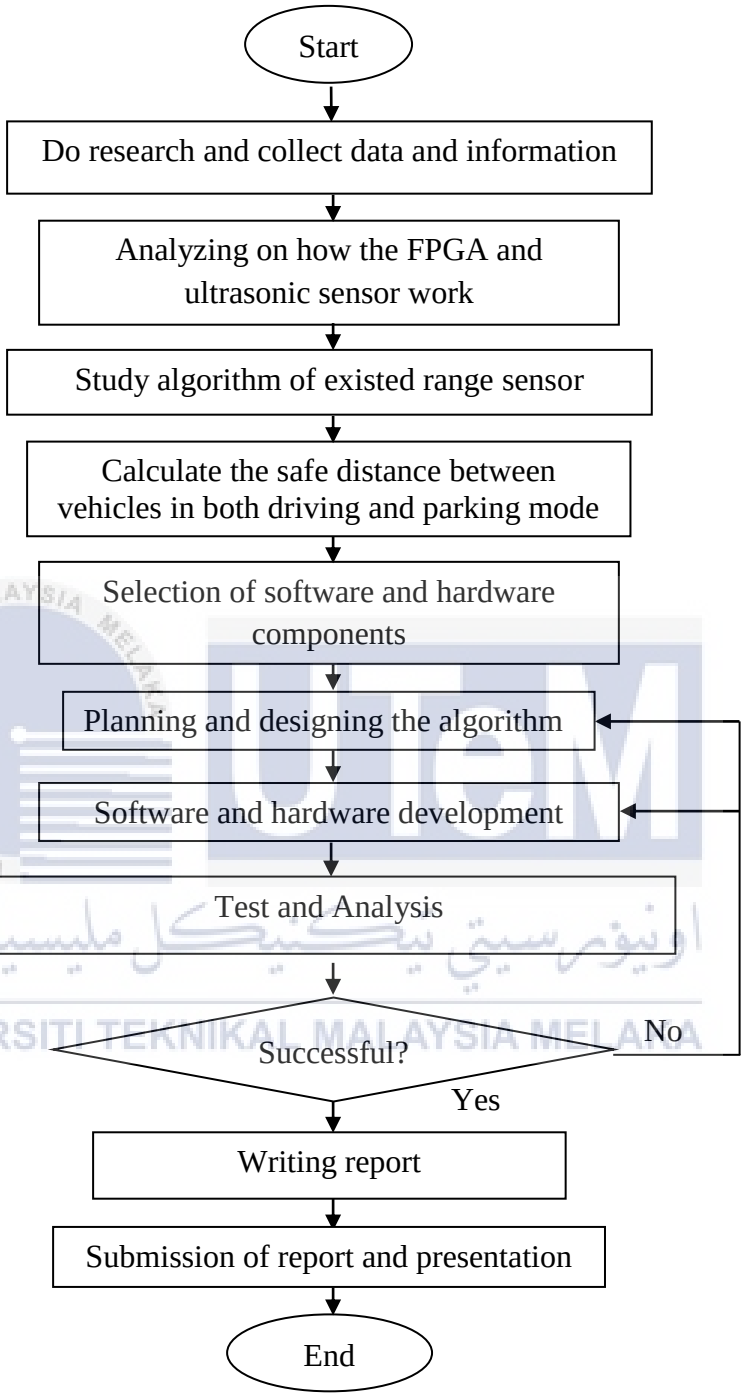


Figure 3.1: Flow chart of project methodology

Figure 3.1 shows the flow chart of implementing this project. The details of the project methodology would be discussed in the following session.

3.2 Project Methodology

This session will discuss about project methodology. The project methodology was developed so that a project could be completed in a good manner. A project methodology is to provide an efficient method and sequences of processes for developer to ensure the project can be conducted in a smooth way. It is a process that describes work scopes and details of each stage from the beginning of the project until the project submitted. There were several stages included in project methodology. The process of development of smart collision avoidance alert system is depicted in the Figure 3.2.

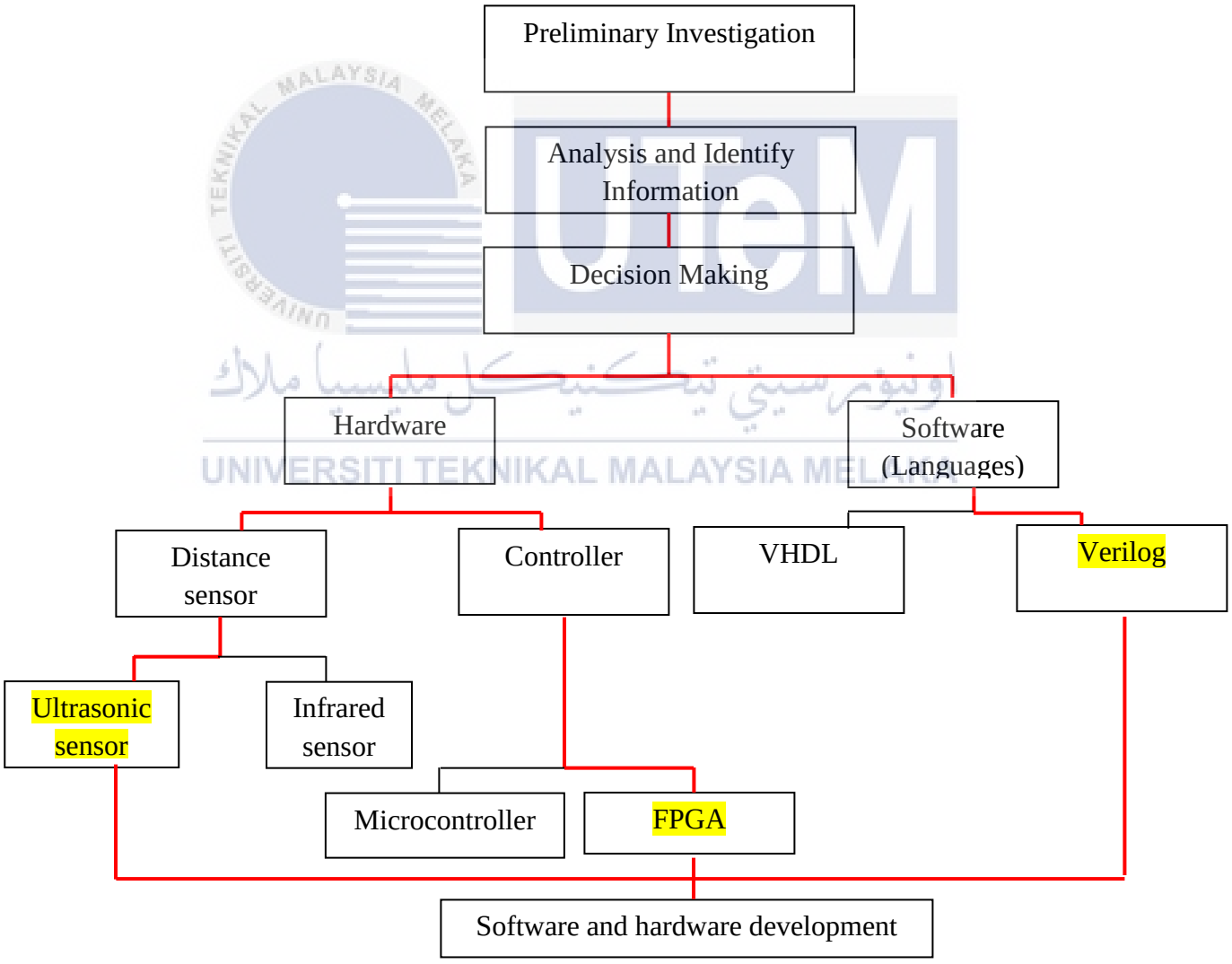


Figure 3.2: Smart Collision Avoidance System Chart

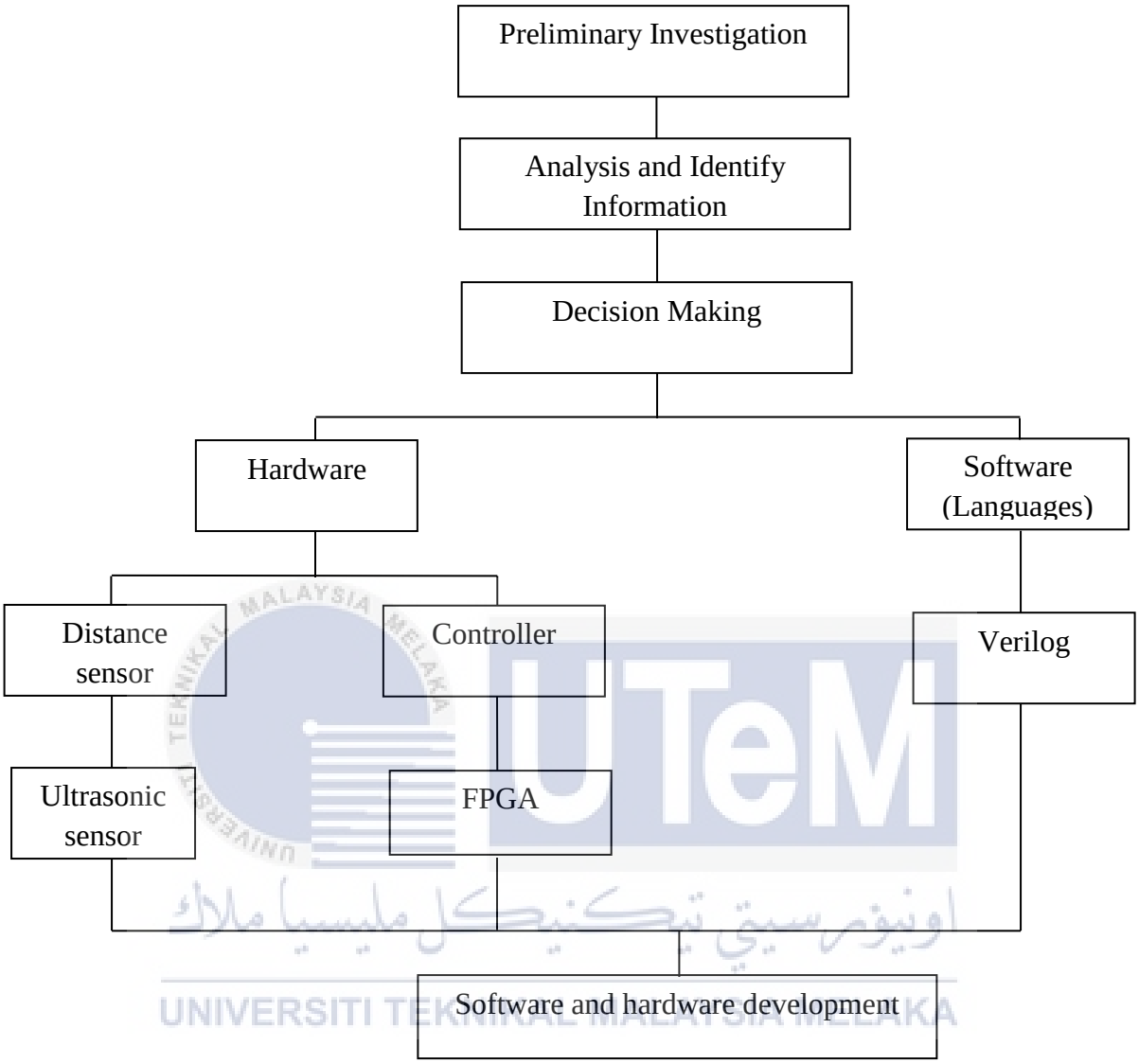


Figure 3.3: Summary Chart

Figure 3.3 shows the summary chart of the development of smart collision avoidance alert system. From Figure 3.3, it can be seen that there were several stages involved in the development of the project included preliminary investigation, analysis and identify information, decision making and last but not least, software and hardware development. In the stage of decision making, several components have been considered based on their functions and specifications. Hardware components that have been chosen for this project were ultrasonic sensor and FPGA. For software, there were two programming languages could be used for the configuration of FPGA, the language that has been

chosen for FPGA configuration was Verilog. The details and explanations of each stage including the selection of components and languages will be discussed in the following session.

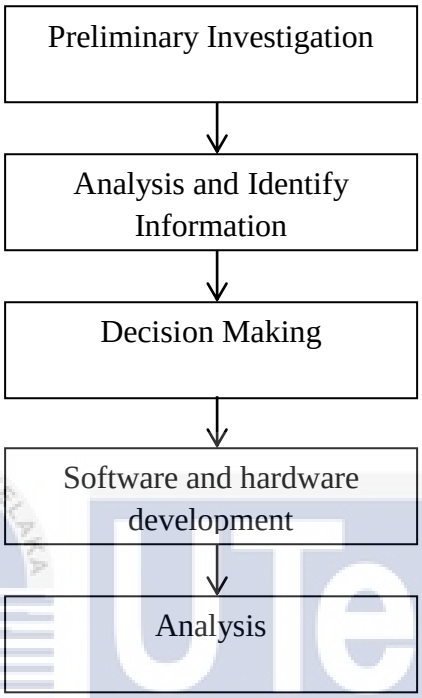


Figure 3.4: Stages of project methodology

3.2.1 Stage I: Preliminary Investigation

Preliminary investigation was the first step of developing the project. Before starting any project, it is very important to list out the objective of the project. The objective of the project must be related to the problem that stated in problem statement. At the beginning of this project, the objectives of this project have been listed out which the main objective of this project was to develop an algorithm to detect and alert occurrence of slowing or stalling vehicle ahead on FPGA to reduce the probability of occurrence of road accident. Researches regarding the specifications of FPGA and other components that were going to

be used in this project have been done. This was important to ensure the direction of developing this project was correct and reduced the mistake that might occur along the developing process. A preliminary proposal has been prepared in order to allow the developing process could be run smoothly.

There were a few aspects included in the proposal:

- Project briefing
- Problem statement
- Objective of the project
- Work scope
- Project methodology
- Expected result
- Project planning

3.2.2 Stage II: Analysis and Identify Information

After defining the objective of the project and have a clear concept and direction of doing the project, the second step was collecting information and data that might be helpful in developing the project. These information and data could be collected from journal, reference book, newspaper, article or any reliable website. The collected information and data were analyzed and identified. The information and data that have been gathered for this project including history of detector, sensors that have been used in vehicle, previous study on the development of anti-collision systems for vehicle and components and languages that were available for this project. Comparison between available components and languages were also included in this stage. From these information and data collected, the functions and specifications of each component could be seen clearly and it gave a clearer concept on the developing of the project.

3.2.3 Stage III: Decision Making

Decision making is an essential step to decide which components to be used in developing a project. This project contained two parts: Hardware and software. There were two components that have to be chosen in the hardware part which were Controller and Distance Sensor.

Hardware: Controller

The main component of this project was the controller. The controller played an important role in this project, it controlled all activities of the system and decided how the system run. In order to choose the correct controller to allow this system to work in the best way, all the specifications have to be considered. The ideal controller to be used in this project needed to be fast, flexible and has good performance. Controllers that could be used in this project were FPGA, Arduino and microcontroller such as ARM7. Field programmable gate arrays (FPGAs) are digital integrated circuit that are designed to perform variety of tasks such as combinational functions, or simply logic gates like OR, AND, NOR and XOR gates. It can be configured or reprogrammed by customers or designer engineers after manufacturing. FPGAs contain an array of programmable logic blocks along with configurable interconnects between these blocks, it can do several process at the same time.

Arduino is an open source physical computing platform based on a simple input board and development. It is cheaper compared to FPGA and other programs language. An arduino usually use in sensors and actuators. The disadvantages of Arduino are: it does not have security; it cannot do several processes at the same time.

For microcontroller, unlike FPGA, the functionality of microcontrollers is fixed during manufacture, thus the functionality of a microcontroller cannot be changed as required. By comparing microcontroller to FPGA, FPGAs were more

likely to be used in military application because FPGA is hard-wired, the memory areas of FPGA not easy be attacked or destroyed by alpha rays, the functionality of FPGAs are not easy be corrupted. Besides that, life time of FPGA based development is longer.

After comparing three of the controllers, FPGA was more preferable in this project because its ability to do several processes at the same time makes it has fast response compared to Arduino and microcontroller. It can be reprogrammed by designer or developer, this makes it more flexible to be used by user. Besides that, life time of FPGA based development is longer compared to microcontroller based development.

Hardware: Range Sensor

Range sensor was required in this project to detect the range between current vehicle and vehicle or object ahead. The data or output produced by the range sensor would be used for analyzing and interpreting by controller. There are many types of sensor in the market such as infrared sensor, ultrasonic sensor and etc.

Ultrasonic sensor is a type of sensor that is designed to measure the distance between current object and non-contact distance object. Ultrasonic sensors emit a sound wave that reflects off of objects entering the wave field. The sound wave is then reflected and received by ultrasonic sensor. By emitting and receiving the sound wave, the distance between current object and object ahead can be determined. The advantage of ultrasonic sensor is that it does not depend on surface color or optical reflectivity of object. Ultrasonic sensors with digital outputs have high sensing accuracy. It can detect an object ahead directly by ignoring the background of the object. Furthermore, ultrasonic sensor is relatively inexpensive to be used.

Infrared sensor is different with ultrasonic sensor, the reflection of infrared sensor depend on the surface of the object, different surface, different

colors and different shades provide different reading to the sensor even if the range is the same. Ultrasonic sensor emits sound waves to detect range but infrared sensor emits infrared light and therefore infrared sensor is not the best out there, it cannot work accurately if there is direct or indirect sunlight.

After went through the advantages and disadvantages of both ultrasonic sensor and infrared sensor, ultrasonic sensor was selected for this project due to its accuracy compared to infrared sensor. Infrared sensor is cheaper than ultrasonic sensor, but the accuracy of infrared sensor is lower than ultrasonic sensor, it required a narrow beam width for the reflection and cannot function well under sunlight. Ultrasonic sensor can provide more accurate reading than infrared sensor because its reflection does not depends on the surface of the object and it can operates in both dark and bright environment.

Software: Programming languages

Since the controller that to be used in this project was FPGA, there were two programming languages could be used to configure it. These two languages were Verilog and VHDL. Both of these languages are considered as digital design languages. There are differences between two of these language. In the terms of concept and syntax, VHDL is more like ADA, it is strongly typed and is not case sensitive. Verilog is more on C programming language which is more familiar and easy to understand. Compared to VHDL that is more on ADA, Verilog is easier to be learned and to be used. The structure of VHDL is more complex, it does not allow user to intermix variables with different classes, thus it a strongly typed language. Verilog is opposite with VHDL, it is weakly typed and it does not contain as much rules as VHDL. It is more concise with efficient notation. VHDL is deterministic and more verbose than Verilog.

From the comparison between VHDL and Verilog, Verilog has been chosen for configuration of FPGA because Verilog is easier to be used and it does not as complex as VHDL. This is more convenient and easy to developer to

allow the developer to complete the project in time and to reduce the mistake that may occur along the implementation of project.

3.2.4 Stage IV: Software and Hardware Development

After the selection of hardware components and programming language to be used, development of the project could be started. Altera Quartus II was the software that has been used for the configuration of FPGA by using Verilog. After the program has been written, it was compiled and stimulated in the software Altera Quartus II. The program was executed in the Altera Quartus II. After made sure that there was no error detected in the program, the hardware part has been developed and the program has been embedded into the FPGA. The system was tested and modified repeatedly until it achieved the objectives of this project.

3.2.5 Stage V: Analysis

In order to make an analysis on the overall performance of the alert system based on FPGA, a comparison with the existing system based on Arduino has been done. The comparison was done by comparing the speed of the system to respond to the detected distance between FPGA based system and Arduino based system. The execution time of Arduino based system has been obtained from existing data while the speed of FPGA based system has been determined by calculating the number of clock cycle required for the system to process the detection.

3.3 Project Overview

The main objective of implementing this project was to design an algorithm to detect the range between the detector and the vehicle or object ahead on the road by using FPGA to prevent or minimize the risk of road accidents. As mentioned earlier, this algorithm contained two modes.

In driving mode:

- This algorithm allowed the detector to detect the rate of change of range between current vehicle and vehicle or object ahead. If the range between current vehicle and vehicle or object ahead reduced drastically, it was possible that the vehicle ahead brake suddenly, the algorithm would activate the buzzer and LED to alert the driver.
- This algorithm would calculate the safe distance between two vehicles and alert the driver through buzzer and LED if two of the vehicles crossed the safe distance.

In parking mode:

- The algorithm also allowed the detector to detect the distance between the detector and the vehicle or object ahead during parking. Buzzer and LED would be activated to alert driver if the vehicle was about to crash with the vehicle or object ahead.

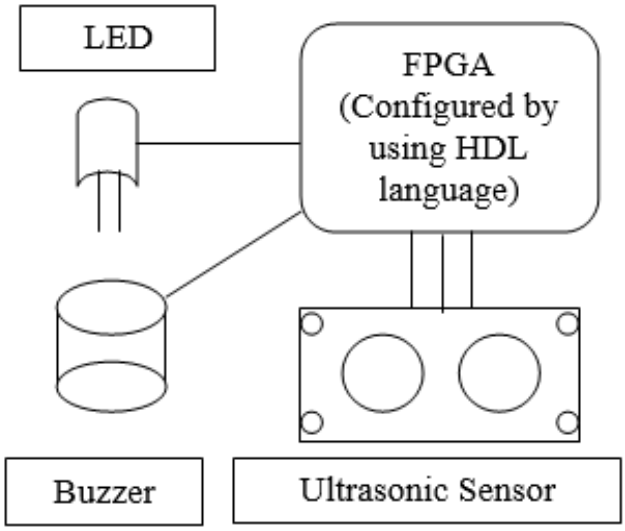


Figure 3.5: Block diagram of FPGA Based Smart Collision Avoidance Alert System

From Figure 3.5, it can be seen that the Ultrasonic Sensor was connected with FGPA, LED and a buzzer. An ultrasonic sensor was used in this project only for testing the functionality of this system. It was used to detect the range of current vehicle and vehicle or object ahead by calculating the elapsed time of emitting and receiving sound waves. The elapsed time that obtained by ultrasonic sensor would be transmitted to FPGA for processing. The FPGA acted as a controller in this project. It controlled the activities of this system and sent out signals to activate buzzer and LED. There were two HDL languages that could be used to configure FPGA which were Verilog and VHDL. The HDL language that has been used in configuring FPGA was Verilog because it was easier to be used compared to VHDL. FPGA controlled the activities of the system based on the elapsed time obtained and safe distance between vehicles.

In FPGA, if the driver turned on driving mode and if elapsed time of emitting and receiving sound waves reduced drastically, FPGA would send out a signal to activate buzzer and LED to alert driver. If FPGA detected that the distance between two vehicles crossed the safe distance that has been calculated, FPGA sent signals to activate the buzzer and LED as well. If there was no vehicle or object found in front of the vehicle on the road, ultrasonic sensor continued to send out the sound wave and no alert

signal would be activated. When driver was going to park a car, he or she could turned on parking mode. The safe distance between vehicles for parking mode and driving mode were different. When a car is running, it is necessary to estimate the time and distance for a car to decelerate its speed until the car is stop, but when a car is about to parking, its speed is slow, no time is required for the car to slow down, therefore, the safe distance between vehicles for parking mode was shorter compared to driving mode.

3.3.1 Formula for calculating range between sensor and vehicle ahead

Ultrasonic sensor was used to emit and receive sound waves. The elapsed time obtained from the emitting and receiving sound waves could be used to calculate the range between current vehicle and vehicle or object ahead. Since the ultrasonic technology is based on the principle of velocity, the velocity of sound is required in calculating the range between vehicles. Sound has a constant velocity of 343.2m/s. The time for emitting and receiving sound wave is directly proportional to the distance to the object.

The formula for calculating the range between the sensor and the vehicle ahead using ultrasonic sensor are:

Distance that sound travels = Speed of sound in air * Time that sound travels

Distance to the vehicle = 0.5 * Distance that sound travels

Speed of sound in air = constant = 343.2m/s

Since the time acquired from ultrasonic sensor is the total time of emitting and receiving sound waves, which indicates that the time is twice of the distance, therefore, when calculate distance, the value that obtained from multiplication of speed of sound in air and time that sound travels has to be multiply with 0.5.

3.4 System Operation Flow

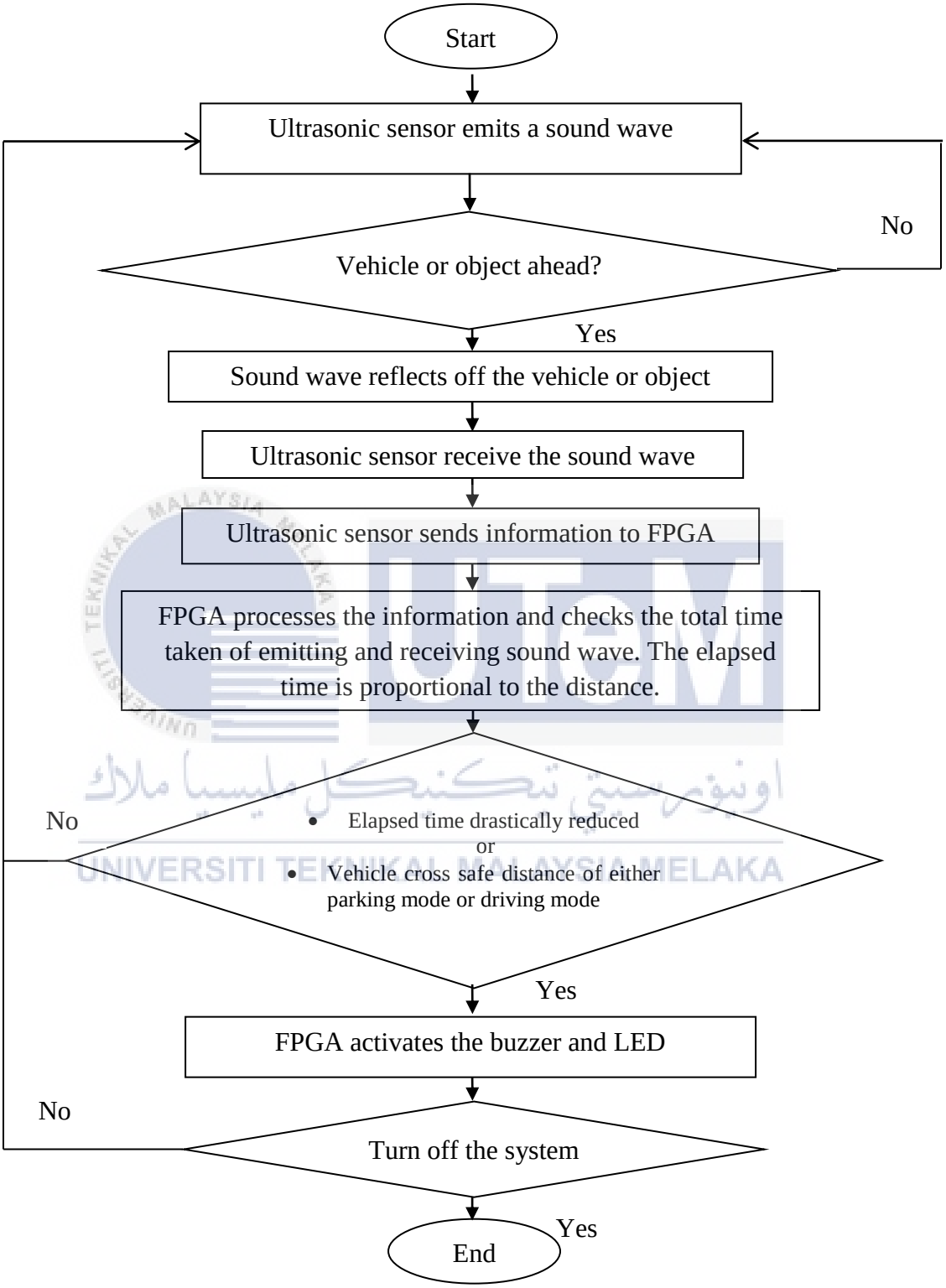


Figure 3.6: Flowchart of FPGA based smart collision avoidance alert system

3.5 List of components

Table 3.1: List of components

No	Name of components	Units	Function	Specifications
1	FPGA	1	Analyze output of ultrasonic sensor, control activities of the system and send signals to activate buzzer and LED.	• Faster response • Can be reprogrammed • Flexible • Longer life time
2	Ultrasonic sensor	1	Detect range between current vehicles and vehicle or object ahead.	• Accurate • Does not depend on surface of object
3	Buzzer	1	Alert driver	• Produce sound
4	LED	1	Alert driver	• Emit light

Table 3.1 shows the components that have been used in this project.

3.6 Project Planning

Project planning is also known as project management that states how to complete a project within a certain timeframe. It was very important to ensure a project could be completed in time and allow the developing process to be systematic and smooth. The project planning provided a step by step approach to complete the project. It related to the use of schedules such as Gantt charts. In project planning, every task that has to be done have been listed out. If there were resources for each task, the resources were identified. One of the most important parts in project planning was the estimation of time. The time to complete every task has to be estimate so that the duration for the whole project to be completed could be known. Estimation of time also could help developer to manage their time in a well manner. After deciding the components to be used in the project, developer has to estimate the cost that would be used for purchasing components to avoid over budget. Besides that, the tasks that were dependent on other task have to be determined so that the developer could determine which task has to be done first. Gantt chart was useful in a project planning. It listed out all the tasks that needed to be done in a well arrangement together with the duration for each task.

Table 3.2: Gantt Chart of project planning

PROJECT PLANNING																																							
	2016																										2016												
Project Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
PSM 1																																						
Project Proposal Preparation	**BP Briefing**	x	x	x				**Mid Term Break**							**Study Week**	**Final Semester Examination**			**Semester Break**										**Mid Term Break**							**Study Week**	**Final Semester Examination**		
Research on HW		x	x	x	x																																	
Research on SW			x	x	x																																	
Refining system architecture					x	x	x																															
PSM1 Report Writing						x	x		x	x	x	x	x	x																								
PSM1 Presentation preparation													x	x																								
Purchasing Materials									x	x	x																											
Early basic coding test											x	x	x	x																								
PSM 2																																						
Implementation of coding																					x	x	x	x														
System Development																					x	x	x	x														
System Integration																								x	x	x												
Test & Analysis																										x	x		x	x								
Report: early chapters corrections																							x	x	x	x	x											
Report writing for thesis																									x	x	x		x	x	x	x	x	x				
Presentation preparation																																	x	x				

3.7 Conclusion

This chapter mainly discuss about the methodology of this project. A project methodology is an efficient method and sequences of processes for developer to ensure the project can be conducted in a smooth way. Project methodology is required to ensure the project can be completed in the estimated time. There were four stages included in the project methodology. These four stages were preliminary investigation, analysis and identify information, decision making and hardware and software development. In preliminary investigation, a proposal which contained the preliminary concept and idea for the project has been prepared. The main part of analysis and identify information was do research on all components and elements that was related to this project. Decision making was the stage where the desired components were chosen based on their specifications. FPGA, ultrasonic sensor and the programming language Verilog were the components and program language that have been chosen and have been used in this project. After development of hardware and software, an analysis by comparing this system with Arduino based collision avoidance system has been done. Flowchart was an important element in project methodology. Flowchart gave a clear picture to people of how did the project be implemented and how did the project run. In order to have a good time management in developing a project, Gantt chart has been used for listed out all the activities that would be done in developing this project. Besides that, the duration for each task to be completed also stated in the Gantt chart.

CHAPTER 4
RESULT AND DISCUSSION

4.0 Introduction

This chapter will mainly discuss about the simulation result and hardware implementation result of this project based on the methodology that has been stated in chapter 3. Besides, this chapter will also discuss about the hardware and software operation of this project. The comparison between performance of FPGA and Arduino in the aspect of speed will also be discussed in this chapter.

4.1 Hardware Implementation

As what have been stated in methodology, the main controller and components used in this project were FPGA, ultrasonic sensor, a led and a buzzer. The ultrasonic sensor was used to detect the range between current vehicle and vehicle or object ahead in order to alert the driver when there is possibility of collision. FPGA acted as the controller to process the signal received from ultrasonic sensor and responsible to make decision on whether to activate the alert system. The led and buzzer were parts of the alert system, they would be activated when high signal is received.

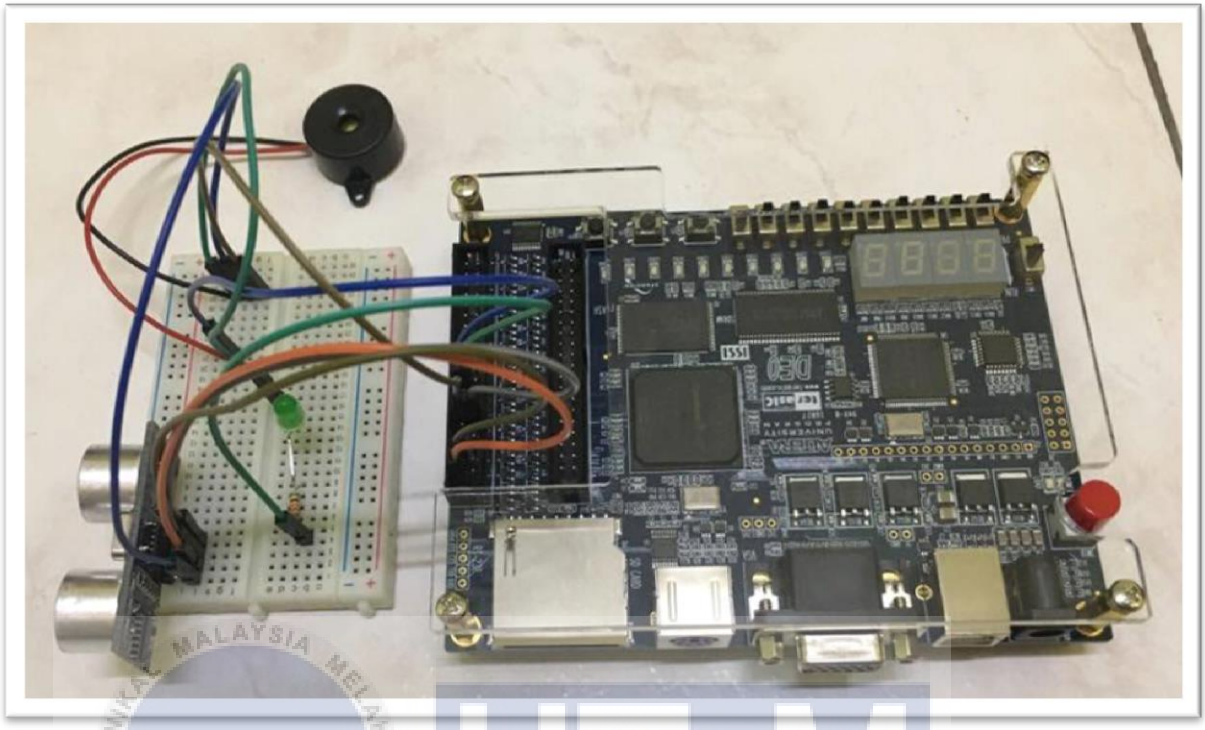


Figure 4.1: Connection of FPGA and the components on breadboard

Figure 4.1 shows the connection between FPGA and the components on breadboard. The Altera DE0 Board is equipped with Altera Cyclone III 3C16 FPGA device; it consists of two 40-pin expansion header. According to Terasic Technologies (2012), each header connects directly to 36 pins of the Cyclone III FPGA. Each header also provide two VCC with +5V and +3.3V respectively and two GND pins. There are 4 pins among these 36 I/O pins connected to the PLL clock input and output pins of FPGA.

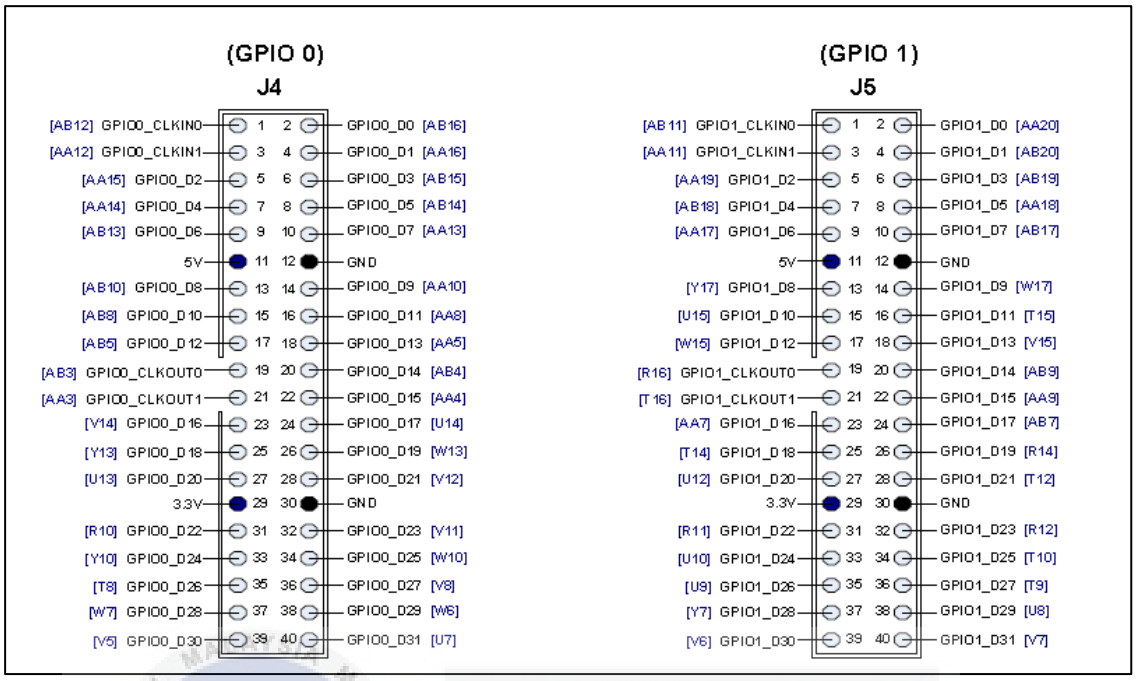


Figure 4.2: I/O distribution of the expansion headers of DE0

Figure 4.2 shows the I/O pins of DE0 board. Only a few pins of GPIO 1 were used in order to build the connection between this board and the components on breadboard as shown in Figure 4.1. Pins that involved for the connection were PIN_H2, PIN_AB20, PIN_AA20, PIN_J6, PIN_V15 and PIN_AB9. DE0 board includes a built-in 50MHz clock input where the pin assignment for the clock input was PIN_G21. Table 4.1 shows the pin assignment for each input and output of this project.

Table 4.1: Pin Assignment and Pin Usage

DE0 board pin value	Pin Usage
PIN_H2	This pin was connected to a push button for reset purpose.
PIN_AB20	This pin was connected to trigger pin of ultrasonic sensor.
PIN_AA20	This pin was connected to echo pin of ultrasonic sensor to receive echo.
PIN_J6	This pin was connected to a switch to allow change of speed mode.
PIN_V15	This pin was connected to a LED for alert purpose.
PIN_AB9	This pin was connected to a buzzer for alert purpose.
PIN_G21	This pin was connected to a 50MHz clock input.

4.2 Algorithm Implementation in FPGA

This project was mainly focus on the algorithm implementation, thus, the algorithm of this project played an important role in making this project successfully worked. There were two finite state machines included in this project which were sensor controller finite state machine and comparator finite state machine. Finite state machine is used to design sequential logic circuit. In finite state machine, one state is available only at a time and the state that available at the time is called current state. It can change from one state to another only when some condition is true or is initiated by triggering event. The changing of the state is called transition. Finite state machine was used in this project because only one state of operation was required at a time. The processes of sending and receiving the signal from ultrasonic sensor, interpreting and processing the signal could be complex, but with implementing of state machine, the processes could be simpler. It made the code more efficient.

4.2.1 Sensor Controller Finite State Machine

Sensor controller finite state machine was used to control the operation of the ultrasonic sensor. This state machine was designed based on the operation of ultrasonic sensor. According to Cytron Technologies (2013), in order to allow an ultrasonic sensor to start a measurement, trigger pin of ultrasonic sensor must receive a high pulse for at least 10us to allow the transmission of eight cycle of ultrasonic burst at 40kHz. When there was ultrasonic detection at the receiver, echo pin was set to high and delay for a period which proportional to the distance. Thus, the distance obtained can be measured by the width of the echo pin. 60ms of measurement cycle is required for an ultrasonic sensor in order to avoid trigger signal to the echo signal.

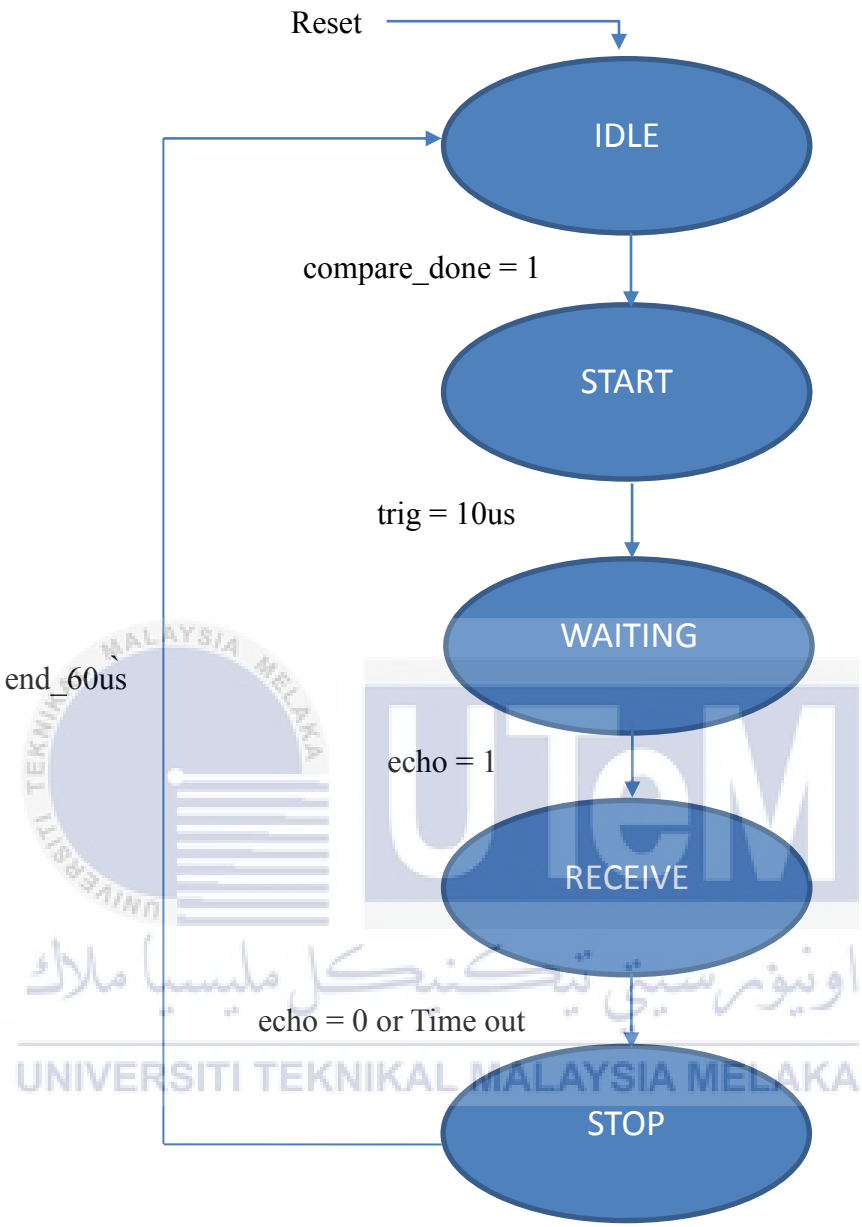


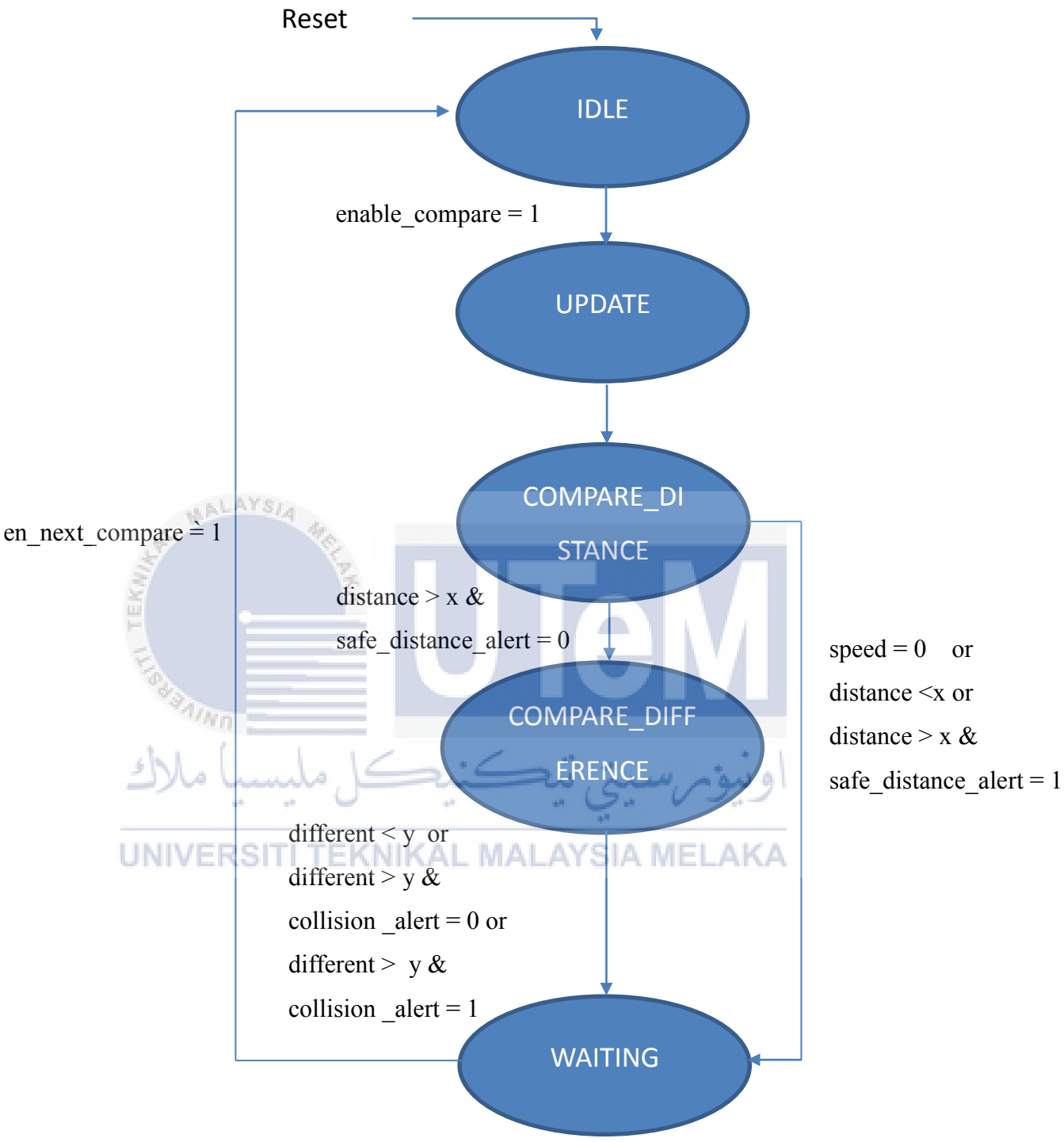
Figure 4.3: Sensor controller finite state machine

According to Figure 4.3, there were five states in the sensor controller state machine which were **idle, start, waiting, receive** and **stop**. There was no operation in the **idle** state, when the comparator done the comparison, the state machine moved to the next state which was **start** state. At **start** state, trigger was set to 1 and the state machine started to count until 10us as the trigger pin of

ultrasonic sensor must receive a high pulse for at least 10us, and moved to **waiting** state. At **waiting** state, trigger was set to 0. When high echo was detected, the state machine moved to the **receive** state. **receive** state waited for the low echo, when low echo was detected, the state machine moved to the **stop** state and distance was obtained based on the delay of the high echo. When there was no detection, the state machine also moved to the **stop** state and waiting for 60ms for next measurement cycle. After 60ms, the sensor controller state machine sent a signal to comparator state machine to enable next comparison.

4.2.2 Comparator Finite State Machine

The sensor controller finite state machine sent the detected distance to comparator finite state machine for comparison. The comparator finite state machine responded to the distance that received from the sensor controller finite state machine and checked whether the detected distance was less than the safe distance. Safe distance means the safe following distance from current vehicle to vehicle ahead. If the detected distance was less than the safe distance, indicating that the current vehicle was following too close to the vehicle ahead and it might cause collision to occur, the alert system would be activated. The comparator also compared current following distance and previous following distance to check the rate of change of distance between current following distance and previous following distance. If the distance between current vehicle and vehicle ahead reduce drastically, it was possible that the vehicle ahead had made a sudden braking and the comparator finite state machine would activate the alert system.



x = Safe distance

y = Difference between current following distance and previous following distance of vehicle

Figure 4.4: Comparator state machine

Figure 4.4 shows the states of comparator finite state machine, there were also five states in the comparator finite state machine which were **idle, update, compare_distance, compare_different** and **waiting**. At **idle** state, comparator state machine enabled the sensor controller state machine to send out signal for detection, it moved to **update** state when distance between current vehicle and vehicle or object ahead was detected. At **update** state, previous distance and current distance was updated before it moved to the **compare_distance** state. Current distance was compared to the safe distance in this state. If current distance less than safe distance, means that current vehicle was following too close to the vehicle ahead, the LED would be activated. The state machine moved to the **waiting** state. If current distance bigger than safe distance but the LED was activated, the LED would be deactivated. In driving mode, if current distance bigger than safe distance and the LED was deactivated, the state machine moved to the **compare_difference** state to check if vehicle ahead made sudden braking, if vehicle ahead break suddenly, collision_alert would be activated. If vehicle ahead drove smoothly without sudden braking but the collision_alert was activated, the collision_alert would be deactivated. If there was no sudden braking and the collision_alert was deactivated, the state machine would move to the **waiting** state. In parking mode, since the speed of the vehicle was very slow, detection of sudden braking was not required, the comparator state machine moved directly to the **waiting** state after **compare_distance** state. The comparator state machine returned back to **idle** state when next comparison was enabled.

4.2.3 Main Module of Collision Avoidance Alert System

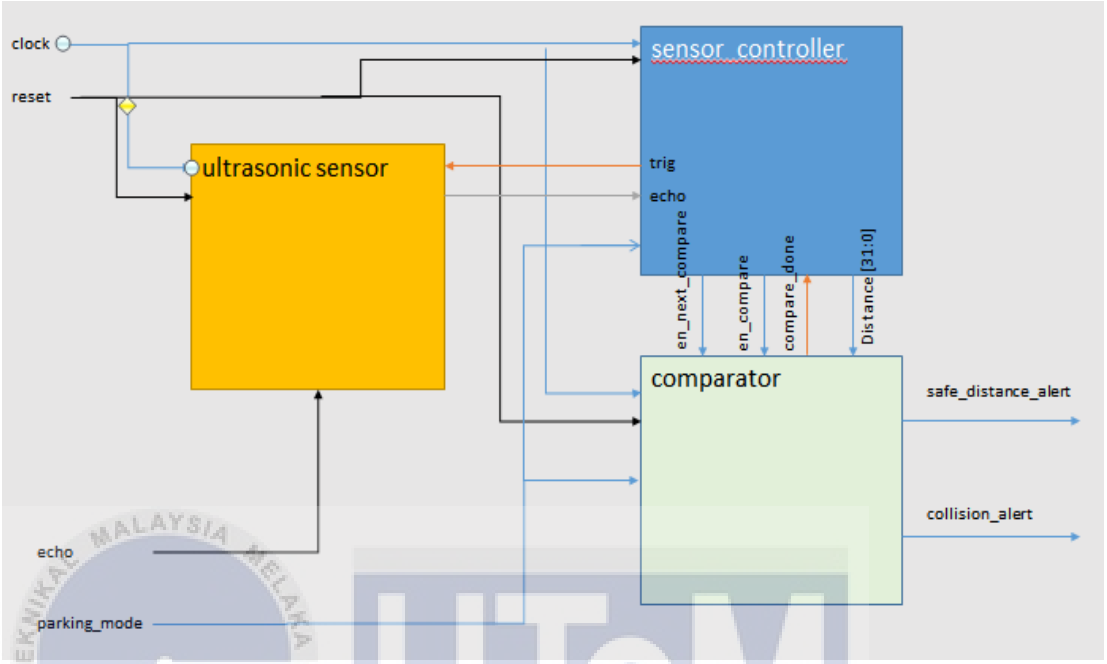


Figure 4.5: Top plane of collision avoidance alert system

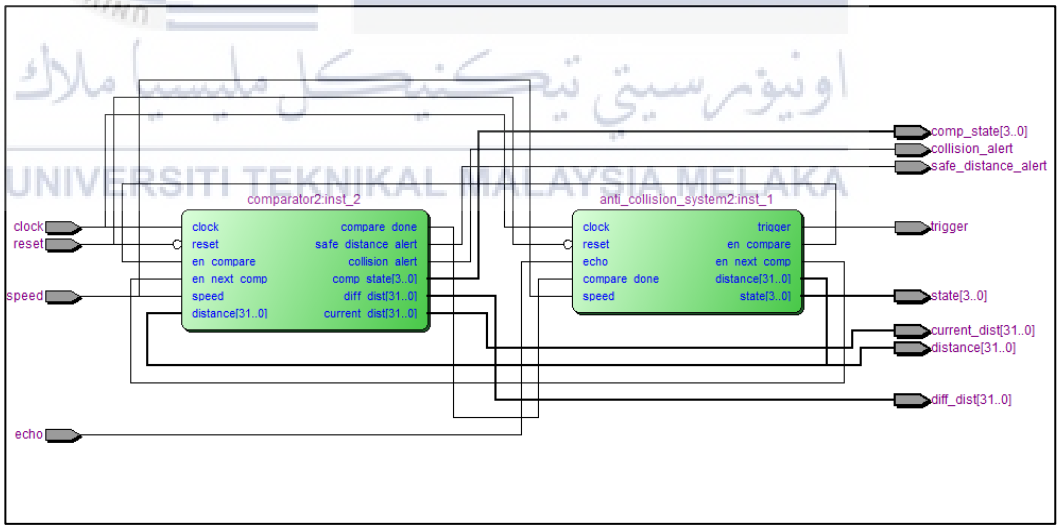


Figure 4.6: RTL viewer of main module on Quartus II

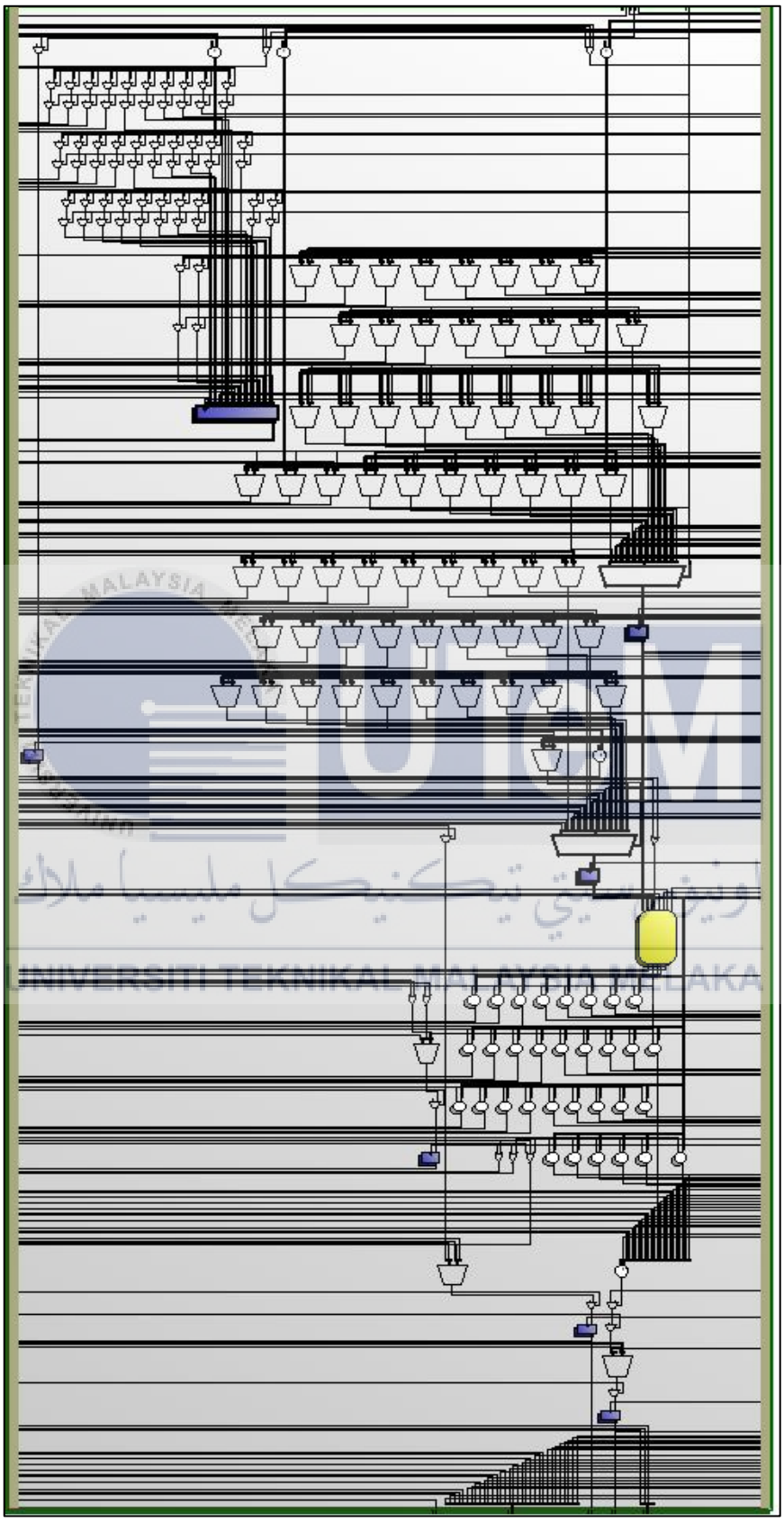


Figure 4.7: RTL viewer of sensor controller module on Quartus II

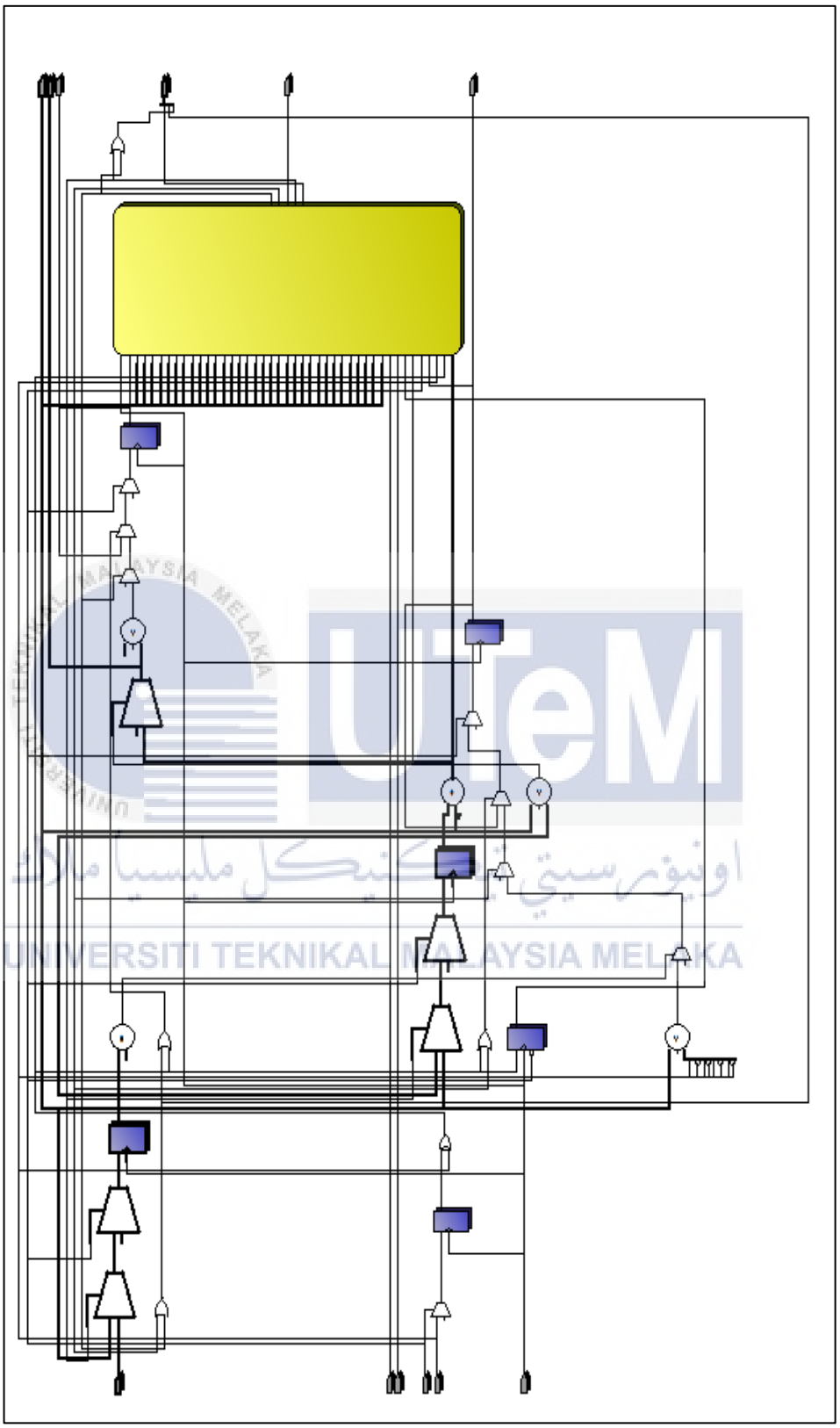


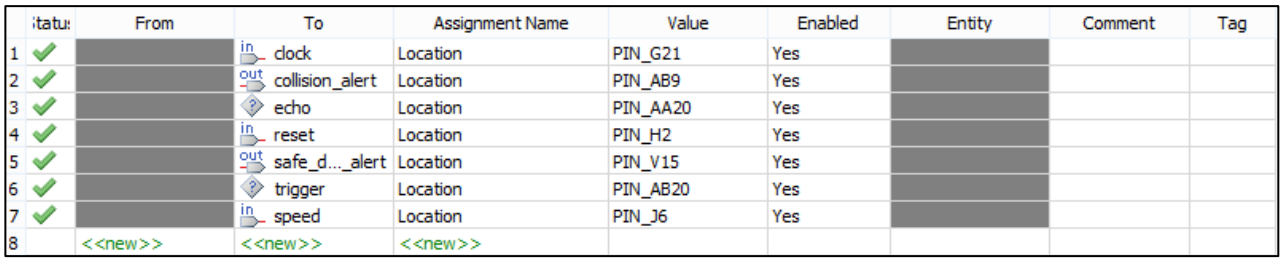
Figure 4.8: RTL viewer of comparator module on Quartus II

Both sensor controller finite state machine and comparator finite state machine were developed in different module. In order to allow the communication of these two modules, a main module was developed to combine each of them. Figure 4.5 shows the combination of sensor controller finite state machine and comparator finite state machine with ultrasonic sensor. The modules were combined by using module instantiation where both of the finite state machines were instantiated within the main module. Port connection for each state machine were done by ordered connection, thus, the order of the ports of each state machine must match with the order of the instantiated module.

Figure 4.6 shows RTL viewer of main module after the combination of sensor controller finite state machine and comparator finite state machine. From Figure 4.6, it can be seen that each of the finite state machines were interacting with each other by exchanging signal simultaneously on multiple ports. The ultrasonic sensor sent a signal to sensor controller finite state machine when echo was received, the sensor controller finite state machine then sent a signal to comparator finite state machine to allow the comparison. Figure 4.7 and Figure 4.8 show the RTL viewer of sensor controller module and comparator module respectively. From the RTL viewers, it can be seen that both of the module was made up of an integrated circuit that contained large amount of gates to allow their operation.

4.3 Interaction between Software and Hardware

In order to allow the interaction between Altera Quartus II and FPGA, ports of main module needed to be configured and connected to the pins of FPGA. Pin assignment was required to assign ports of main module in Altera Quartus II to pins on DE0 board. The pin assignment was made in the Assignment Editor environment.



	Status	From	To	Assignment Name	Value	Enabled	Entity	Comment	Tag
1	✓		clock	Location	PIN_G21	Yes			
2	✓		collision_alert	Location	PIN_AB9	Yes			
3	✓		echo	Location	PIN_AA20	Yes			
4	✓		reset	Location	PIN_H2	Yes			
5	✓		safe_d..._alert	Location	PIN_V15	Yes			
6	✓		trigger	Location	PIN_AB20	Yes			
7	✓		speed	Location	PIN_J6	Yes			
8		<<new>>	<<new>>	<<new>>					

Figure 4.9: Assignment Editor

Figure 4.9 shows the Assignment Editor which located in the Altera Quartus II. There were two ways of pin assignment. It can be either modify in the pin planer of Altera Quartus II or insert name of each ports into the pin assignment editor as shown in Figure 4.9. Each port was assigned to their corresponding pin that connected to hardware by adding the pin value to the value column based on the pin value given in the I/O distribution table of DE0 board. A 50MHz of clock signals that included in the DE0 board was used and pin assignment for that clock input was PIN_G21.

4.4 Simulation

After the development of state machines on Altera Quartus II, the design and requirement was tested by using simulation before it was implementing on hardware. It is better to test a design by using simulation first instead of implementing it directly on hardware. Problem can be found easily on simulation because the interaction between variables can be seen. Each data that stored in a variable can also be seen while the design is running. Besides, the input could be generated from user to check if the output results match with the input. This helps to check if the result is correct. When there was error occurred, the causes could be figured out easier by using simulation because it showed out every variable that could not be seen in hardware implementation. There were two available simulation tools, which were university program VWF and ModelSim. University program VWF was used at the beginning of simulation. By using this tool, the input could be generated directly on the VWF program but the time allowed

for the simulation was limited, the total time that available for the simulation was only 100us which was not enough for testing the design, thus ModelSim was used instead in testing the design.

ModelSim is a HDL simulation environment that available for simulation, verification, and debugging of hardware description languages such as Verilog and VHDL. Unlike university program VWF, input of ModelSim could not be generated inside the simulation window; a test bench module was required to generate inputs for the simulation. Test bench was used to simulate the design without implementation of the design on hardware. It was written in Verilog language as well. The module to be simulated was instantiated in the test bench module as shown in Figure 4.10 and the inputs was generated by using delay of unit.

```
always begin
   #1 clock = ~clock;
end

main2 u_main(
clock,
reset,
speed,
trigger,
echo,
safe_distance_alert,
collision_alert,
state,
comp_state,
diff_dist,
current_dist,
distance);
```


Figure 4.10: Instantiation of module in test bench

```
`timescale 10 ns / 1 ns
module main_tb2();

reg clock, reset, echo, speed;
wire trigger, safe_distance_alert, collision_alert;
wire [3:0]  state, comp_state;
wire [31:0] distance;
wire [31:0] diff_dist;
wire [31:0] current_dist;

initial begin
    clock = 1;
    reset = 1;
    echo  = 0;
    speed = 1;

    #10      reset = 0;
    #20      reset = 1;

    #2000    echo  = 1;  //1000 clk cycle,start from 20300ns
    #210000  echo  = 0;  //105000 clk cycle,from 2120300ns

    #5960000 echo  = 1;  //2980000 clk cycle, after 2980000 clk cyle, echo = 1, start from 61720300ns
    #120000  echo  = 0;  //60000 clk cycle,start from 62920300ns
```

Figure 4.11: Script of test bench in Verilog language

Figure 4.11 shows the script of test bench used in the simulation. Timescale was required in every test bench because it was used to determine time for each unit. As shown in the script, 10ns was used for one unit, which means that delay of 10 units was equal to delay of 100ns. In test bench module, no input and output port was required, all the inputs needed to be initialized before the simulation got started. Each of the input was generated by using the symbol delay # followed by unit of time as shown in Figure 4.11. Since the period of one clock cycle was 20ns, thus, 1000 clock cycle was equal to 2000 unit. No limitation of time was required for running the simulation in ModelSim, the end time could be adjusted based on the design. Changes of any variable in each of the state machine could be viewed easily by right clicking the variable and add wave to the simulation window as shown in Figure 4.12. Therefore, every signal that was in the design could be inspected during running of the simulation on the simulation window.

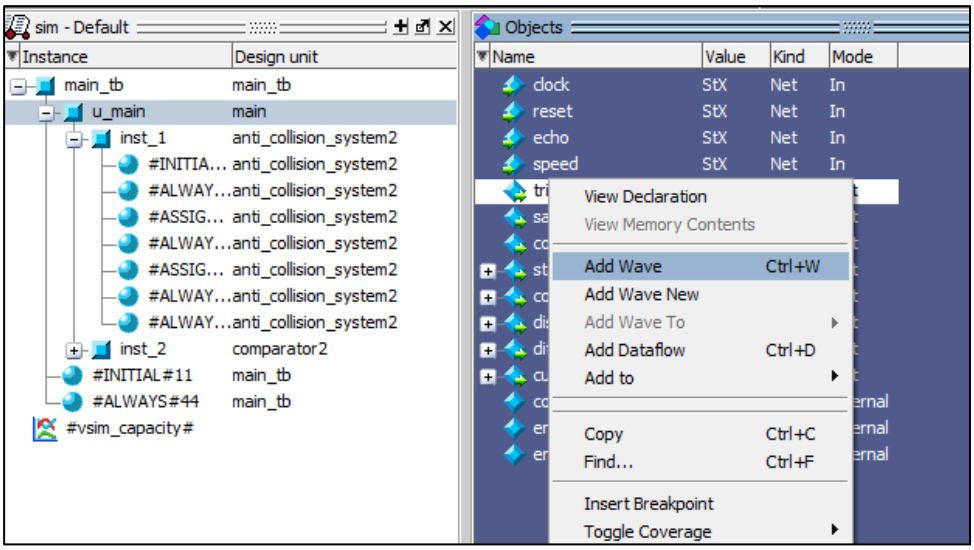


Figure 4.12: Object window of module

4.4.1 Simulation Result

The safe distance for simulation and real implementation was set to different value respectively. This was because the value for real implementation on hardware was very big and it would be very time consuming in order to run the simulation whole. Furthermore, the objective of running the simulation was to check the functionality and performance of the design. The value can be changed during real implementation. In simulation, the safe distance was set to 300 clock cycle when it was in driving mode, which means that when the input showed that the current distance was less than 300 clock cycle, implied that current vehicle was following too near to the vehicle ahead and there might cause a collision to occur, thus the safe_distance_alert should be activated. The maximum difference value of previous following distance and current following distance allowed was set to 100 clock cycle, implied that when the difference between previous following distance and current following distance exceeded 100 clock cycles, the vehicle ahead has made a sudden braking and collision_alert should be activated.

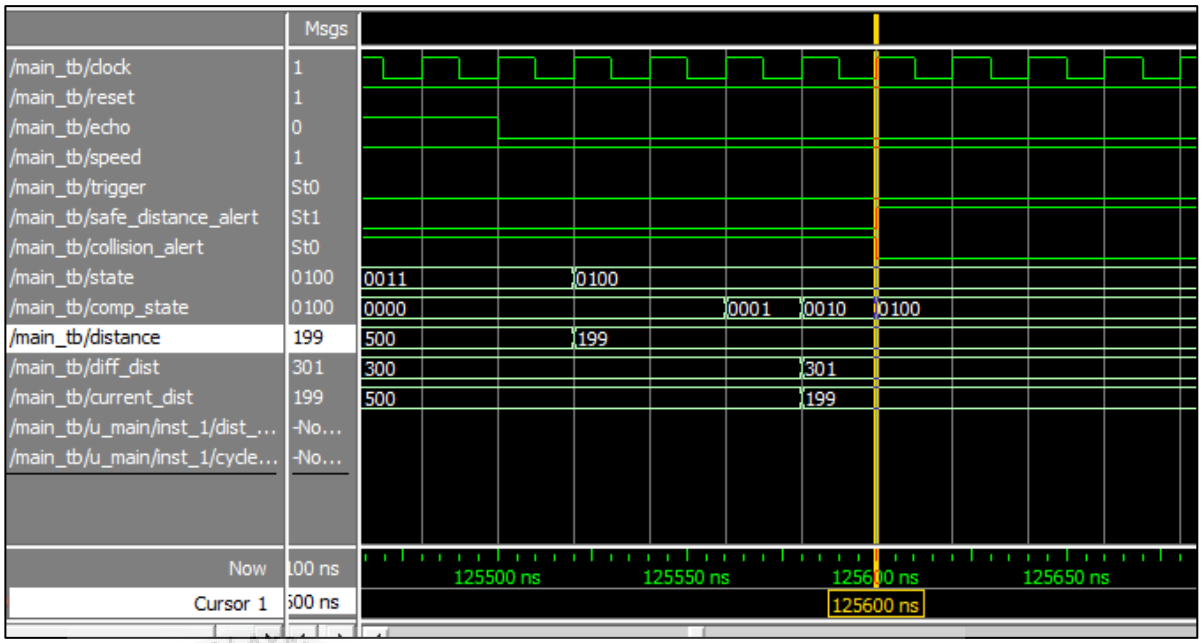


Figure 4.13: Waveform of simulation result when current distance less than safe distance

Figure 4.13 shows waveform of the simulation result. From Figure 4.13, the sensor controller state machine moved to the **waiting** (0100) state and enabled the comparison in comparator state machine. The distance obtained from sensor controller finite state machine was passed to the comparator state machine and comparison was started. It can be seen that current distance was updated to 199 clock cycles after the **update** (0001) state of comparator state machine. After the **compare_distance** (0010) state, the safe_distance_alert was activated because current distance was 199 clock cycles which was less than the safe distance 300 clock cycles. The result was similar with the expected result. The safe_distance_alert would continue to be activated as long as the vehicle was not in the safe distance with vehicle ahead.

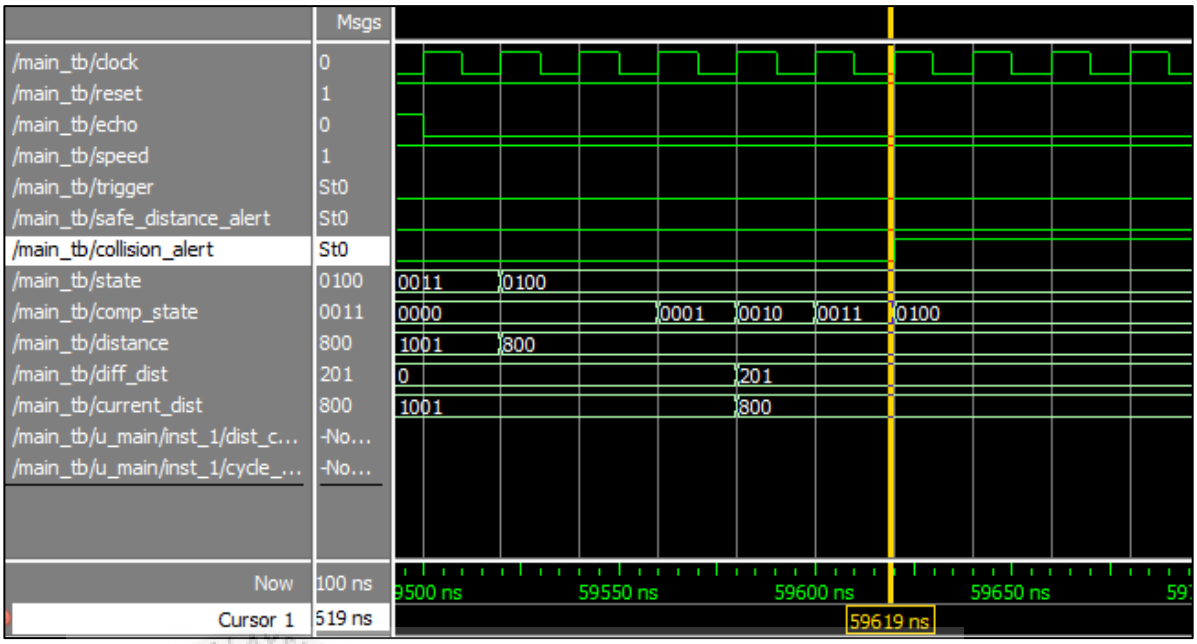


Figure 4.14: Waveform of simulation result in when difference distance more than maximum difference distance allowed

Figure 4.14 shows the result when difference between previous following distance and current following distance was bigger than 100 clock cycles, which means that vehicle ahead had made a sudden braking. From Figure 4.14, it can be seen that in **compare_distance** (0010) state of comp_state, current distance was 800 clock cycles which was bigger than the safe distance 300 clock cycles, the safe_distance_alert remained in deactivated, thus the comparator moved to the next state which was the **compare_different** (0011) state. The collision_alert had been activated in this state because the difference distance was 201 clock cycles which was bigger than 100 clock cycles, the maximum difference distance allowed.

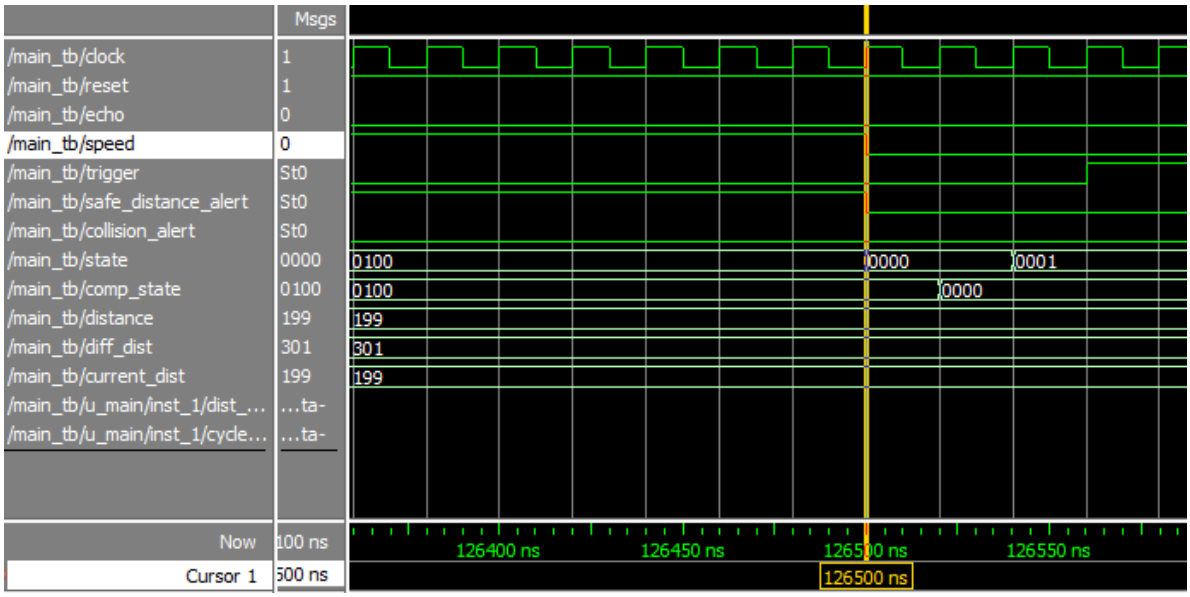


Figure 4.15: Waveform of simulation result during switching of mode

Figure 4.15 shows the waveform simulation result when user switched from driving mode to parking mode. It can be seen that both finite state machine has went back to **idle** (0000) state. Switching of the mode was indicated by the speed, speed was high during driving mode and it turned to low when user switched from driving mode to parking mode. When user switched from one mode to another, the system would be reset to prevent occurrence of error.

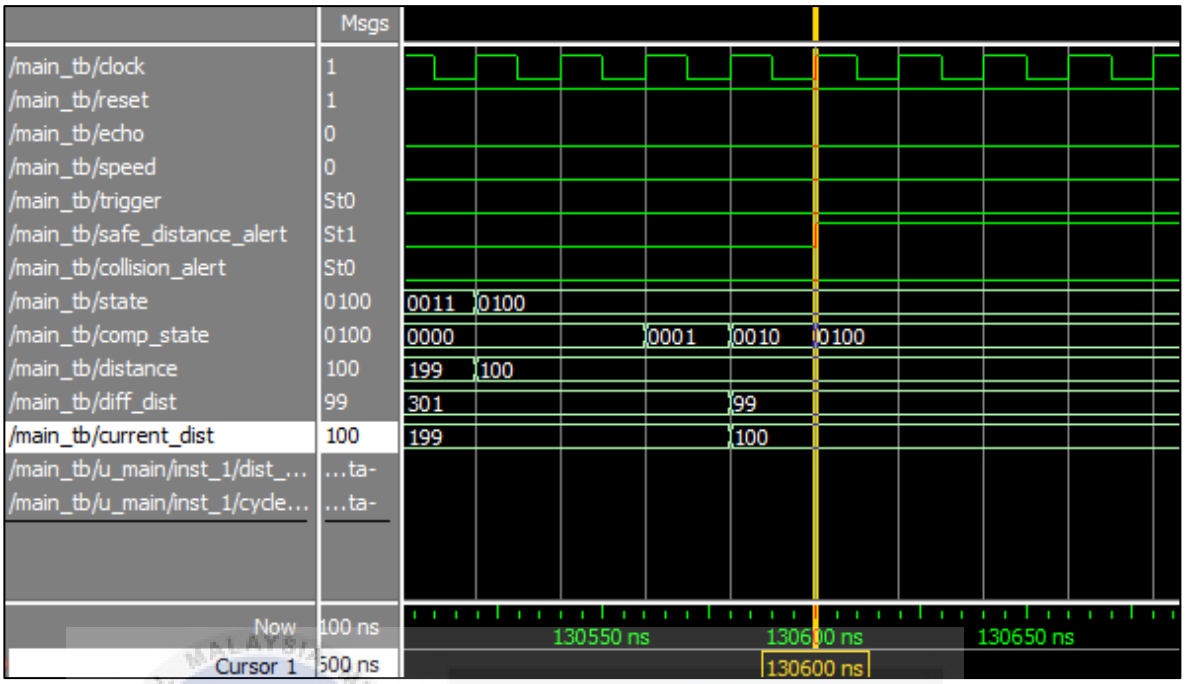


Figure 4.16: Waveform of simulation result during parking mode

The safe distance for driving mode and parking mode was different since the speed of vehicle in driving and in parking are different. In simulation, the safe distance were set to 300 clock cycles and 100 clock cycles for driving mode and parking mode respectively. According to Figure 4.16, the safe_distance_alert was activated when current distance equaled to 100 clock cycles, which was similar with the expected result.

4.5 Hardware Validation

After the simulation has been successful, the design was implemented on hardware. A debugging tool called Signal Tap II included in the Quartus II software has been used to display signals in real time in the FPGA design. The setting must be changed according to the requirement of the project before the SignalTap analyzer can work. Once the hardware has been setup and connected to the Quartus II, the project on Quartus II with SignalTap II instantiated loaded onto the DE0 board and the analysis

was run. The hardware started to run and the changes of signal on each variable have been displayed on the SignalTap II window.

In real time hardware implementation, the safe distance was set to 10cm and 5cm for driving mode and parking mode respectively. The maximum difference between previous following distance and current following distance allowed was set to 5cm.

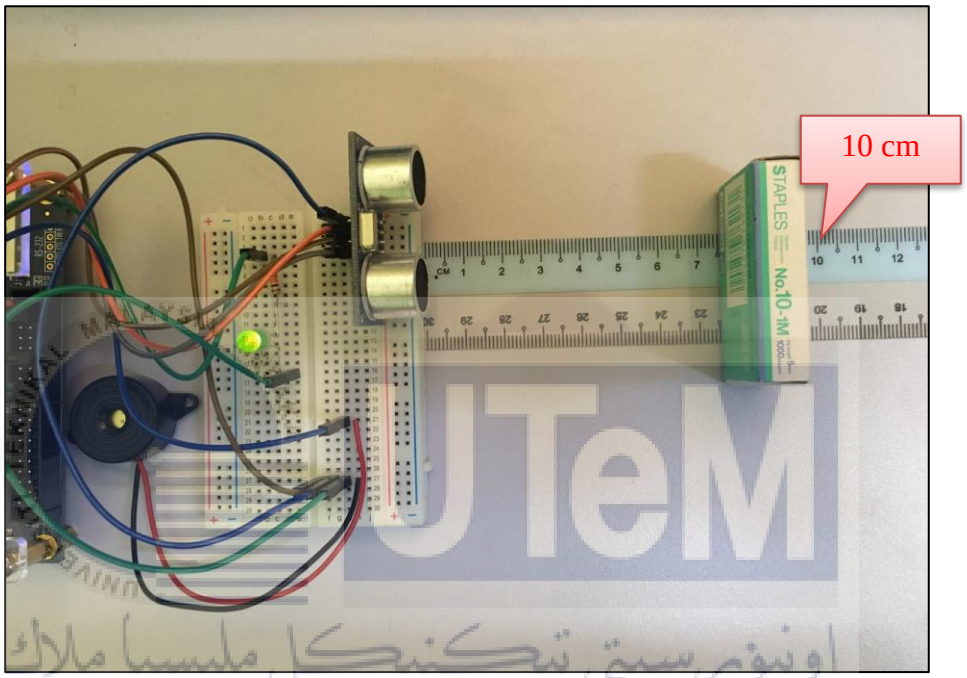


Figure 4.17: Object detected during driving mode

The safe distance was set to 10 cm for driving mode in real time implementation on hardware. According to Figure 4.17, it can be seen the LED has been lighted up as expected when there was object detected less than the safe distance.

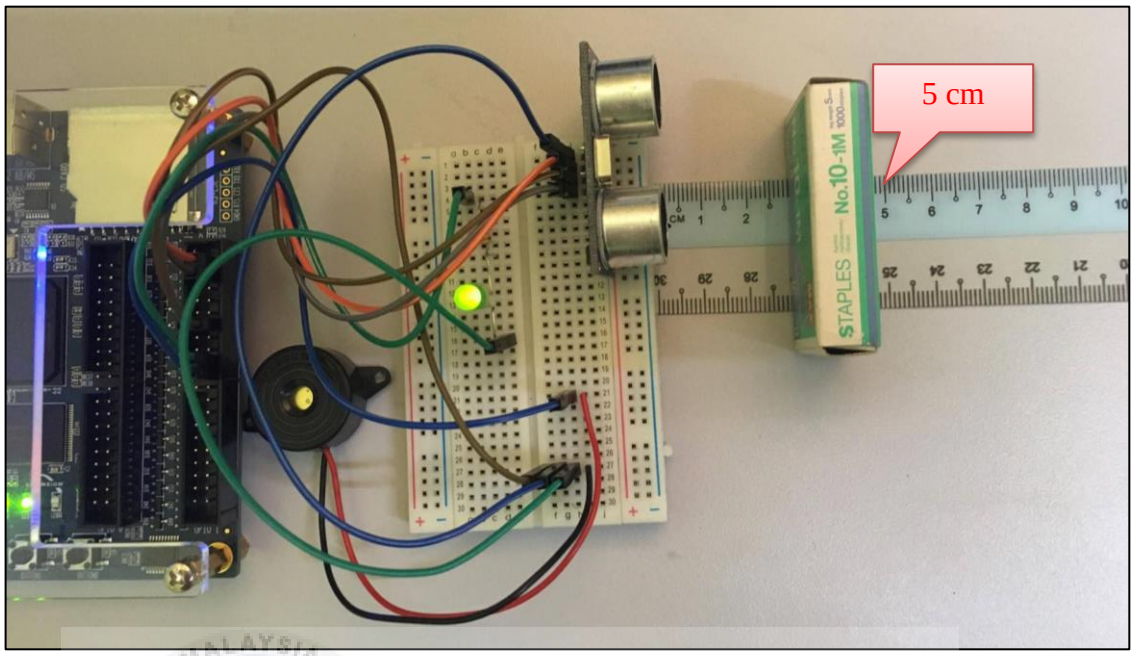


Figure 4.18: Object detected during parking mode

The safe distance was set to 5 cm for parking mode in real time implementation on hardware. According to Figure 4.18, it can be seen the LED has been lighted up as expected when there was object detected less than the safe distance.

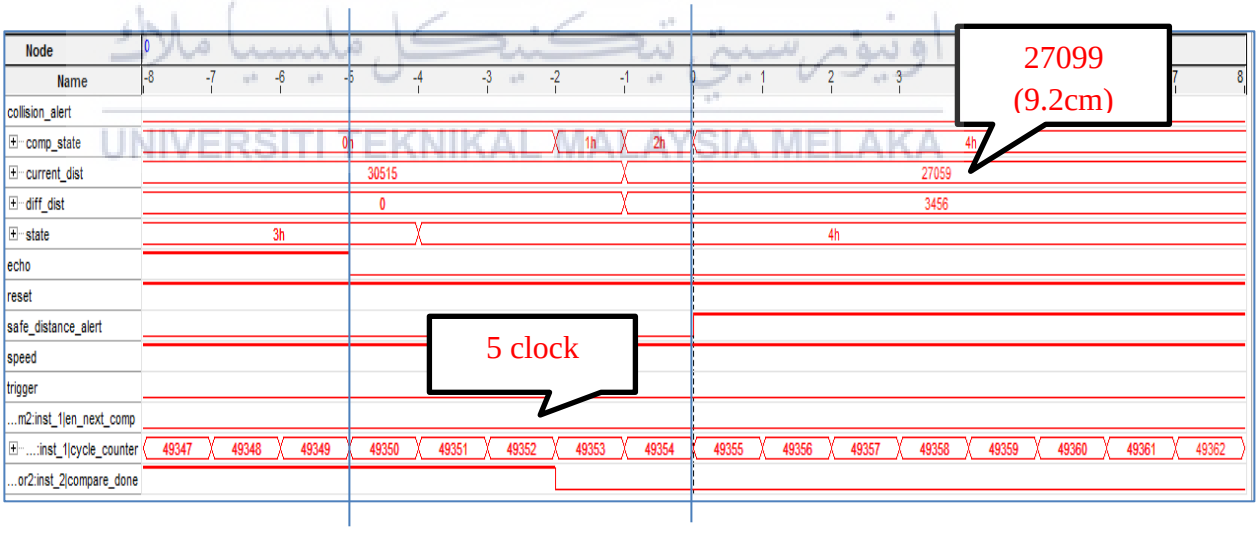


Figure 4.19: Logic Analyzer display when object detected less than safe distance in driving mode

From Figure 4.19, the safe_distance_alert has been activated as expected when the object detected was less than the safe distance (10cm) as shown in Figure 4.15.

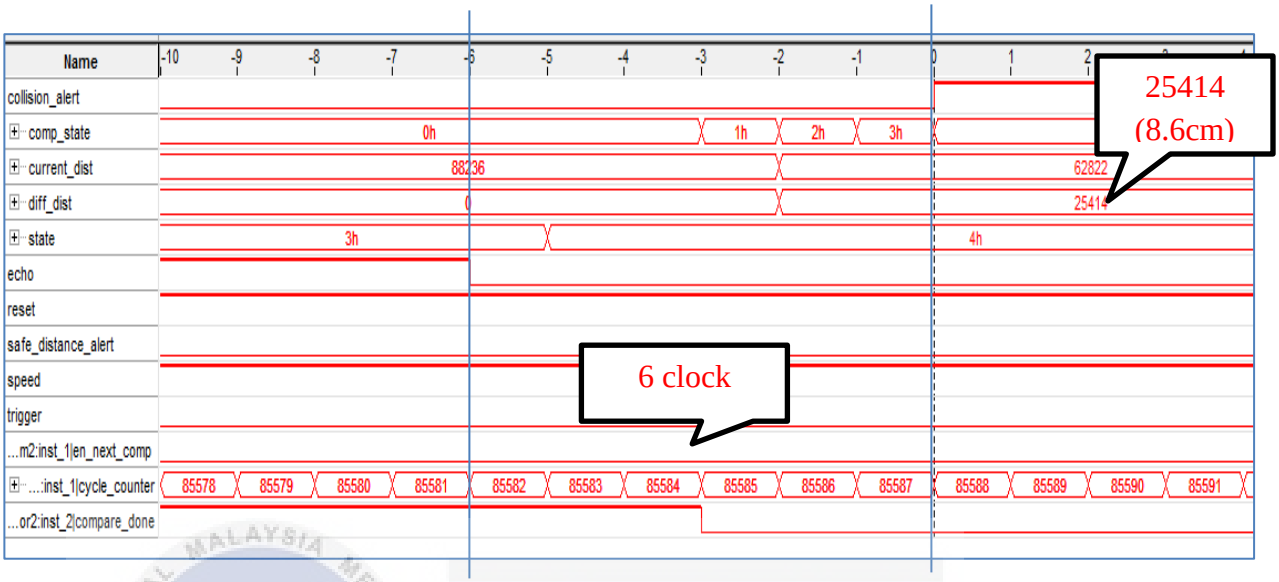


Figure 4.20: Logic Analyzer display when difference distance bigger than the maximum difference allowed

Figure 4.20 shows the logic analyzer display when the difference between previous distance of ultrasonic sensor and object and current distance of ultrasonic sensor and object was bigger than the maximum difference allowed (5cm). Based on Figure 4.20, the collision_alert has been activated as expected, proved that the buzzer has been activated.

4.5.1 Analysis on Overall Performance

To make an analysis on the overall performance of the alert system based on FPGA, a comparison with the existing system based on Arduino has been done. Since both of the system were using ultrasonic sensor for detection of distance, their comparison was done by only compared the time taken for each system to respond to the echo received. From Figure 4.19 and Figure 4.20, it can

be seen that the time taken for the FPGA reacted to the echo received was only approximately from five clock cycles to six clock cycles.

Time taken for one clock cycle = 20ns.

Five clock cycles = 5 * 20ns = 100ns

Six clock cycles = 6 * 20ns = 120ns

From the calculation shown, the maximum time taken for the FPGA responded to the echo received and generated output for the system was only 120 ns.

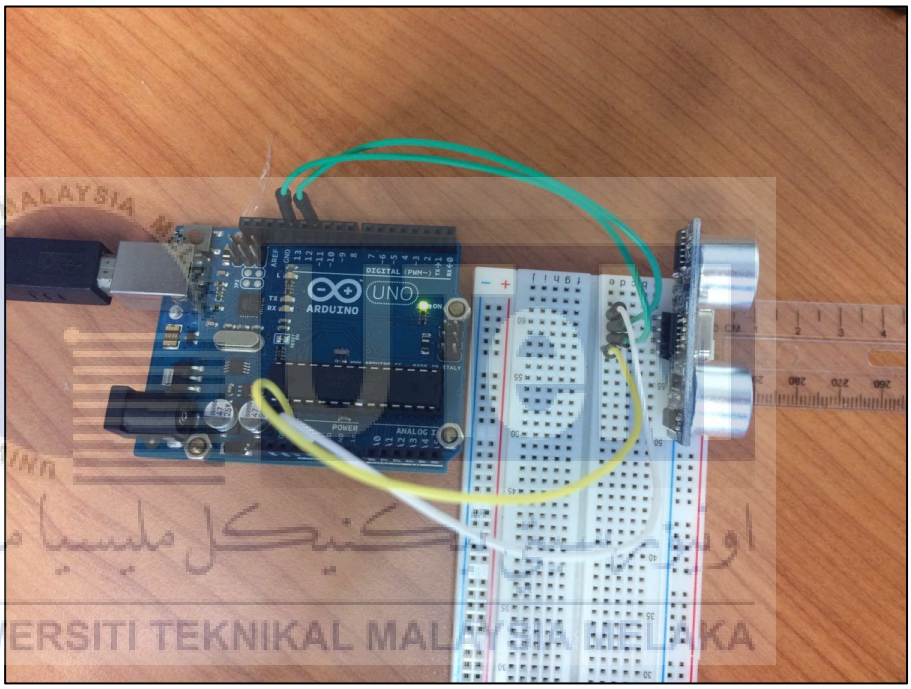


Figure 4.21: Arduino based collision avoidance alert system

Figure 4.21 shows the Arduino based collision avoidance alert system which had been setup for comparison purpose. From Figure 4.21, it can be seen that the ultrasonic sensor also acted as the distance detector in Arduino based collision avoidance alert system.

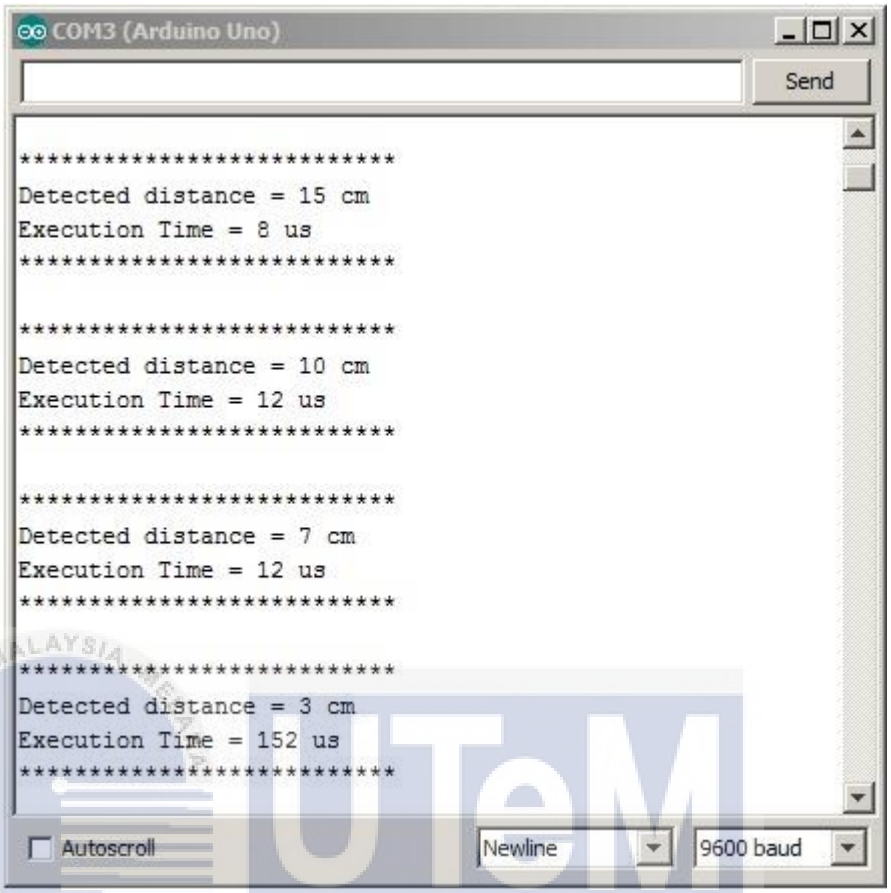


Figure 4.22: Execution time of Arduino based collision avoidance alert system when object detected less than safe distance

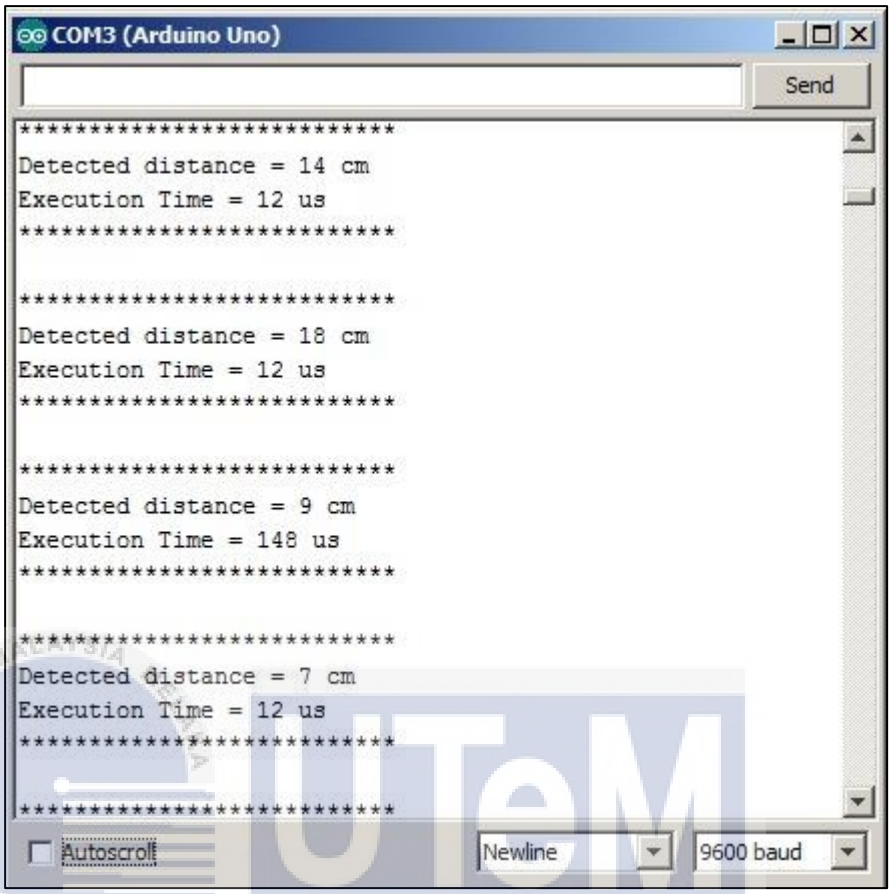


Figure 4.23: Execution time of Arduino based collision avoidance alert system when difference distance bigger than the maximum difference allowed

Figure 4.22 and 4.23 shows the detected distance and execution time of Arduino based collision avoidance alert system. The safe distance of the system was set to 5cm. According to the data shown in Figure 4.22, the maximum time taken for the system to respond to the echo received was 12 us when the system did not need to activate the alert. When there was detection, the time taken for the system to react to the echo received was 152 us since the system needs to activate the alert.

From Figure 4.23, when the distance between the object and the ultrasonic sensor decreased from 18cm to 9cm, the execution time was 148 us which was much longer compared to FPGA based collision avoidance system. This was because FPGA could run the detection and activate the alert simultaneously, while Arduino need to wait for the alert triggering to finish first, then only can go back to the next

detection. Run simultaneously means that the FPGA could transmit and receive signal and process the received signal at the same time. Thus in FPGA, it could activate the alert and at the same time, the new detection would be continued. In Arduino, the system needed to wait until the alert stopped, only it would continue with the new detection. This made a collision avoidance alert system more effective and provided a better performance in order to prevent a collision to occur.

Besides that, the maximum frequency of the design shown from timing analysis was 72MHz, but the frequency used for the project was only 50MHz, which indicates that when we maximize the clock frequency until 72MHz, the execution time could be shorter and the system could be more effective.

4.6 Limitation

There were some of limitations by using ultrasonic sensor for distance detection. The ultrasonic sensor could only detect the object or vehicle ahead when the object or vehicle ahead was as perpendicular as possible to the sensor. There was difficulty for the sensor to identify vehicle that traveling side by side.

Besides, one of the limitations of an ultrasonic sensor was its flexibility in detection, the ultrasonic sensor only able to detect obstacle ahead without accurately identify the size and the exact location of the obstacle. Other than that, result of the detection might be not consistent because changes in environment such as temperature, pressure and air turbulence might affect the ultrasonic response. But due to limited cost provided for this project, an ultrasonic sensor was used in this project for distance detection.

This system did not have auto braking system. When the driver unable to response to the alert that have been activated, there was high possibility for the vehicle to crash with the vehicle ahead since the vehicle did not have auto braking system.

4.7 Conclusion

This chapter mainly discussed about the result of both simulation and hardware implementation. From the result shown in both simulation and hardware implementation, it has been proved that the system could be successfully worked since the result shown was similar with the expected result. This chapter also described about the operation of this system. This system was developed from two state machines which were sensor controller finite state machine that used to control the operation of the ultrasonic sensor and comparator finite state machine that used to allow the comparison of the detected distance and react to the detected distance. There were also some limitations found in this system. Ultrasonic sensor might not be the most suitable sensor to be used in this system but due to limited of budget, the ultrasonic sensor was used in this system.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Introduction

This chapter will discuss about the conclusion of this project according to the scope and objective that had been stated in introduction. This chapter will compare the achievement of the result with the objectives that have been introduced in the introduction. Besides that, this chapter will also discuss about the improvement that can be made in this project regarding to the limitations that have been pointed out in chapter 4. The commercial potential of this project will also be discussed in this chapter.

5.2 Conclusion

In order to reduce the occurrence of accident on the road and the impact during a collision, a collision avoidance alert system is required to help to alert the driver when there is possibility of collision. This project was about the implementation of FPGA based collision avoidance alert system algorithm that has two modes, which were driving mode and parking mode. In driving mode, the system would detect the range between current vehicle and vehicle ahead, when the range between them less than the safe distance that has been set, the system would alert the driver. When the rate of change of distance between two vehicles change drastically indicates that the driver ahead had made a sudden braking, the system would alert the driver as well. In parking mode, the system would also detects the distance between current vehicle and vehicle or object ahead to check if current vehicle is too close with vehicle or object ahead but the

safe distance in parking mode was different with the safe distance in driving since the speed of vehicle in different mode are different. User could switch the mode of the system accordingly. Parking mode could also be activated when the vehicle is driving in low speed such as in traffic jam to prevent the driver drive too close with vehicle ahead.

The first objective of this project was to develop an algorithm to detect and alert occurrence of slowing or stalling vehicle ahead on FPGA. This objective has been successfully achieved as the result shown in Chapter 4 has proved that the designed algorithm could be successfully implemented on the FPGA based collision avoid alert system. An ultrasonic sensor was used to detect distance. As shown in the result in chapter 4, the LED has been lighted up when there was object detected near to the ultrasonic sensor and the distance between the object and ultrasonic sensor was less than the safe distance. The buzzer has been activated when there was object moved in high speed near to the ultrasonic sensor.

The second objective was to study on how to detect the rate of change of distance between vehicles. The distance between vehicles was detected by using ultrasonic sensor, in order to obtain the rate of change of distance, previous distance that obtained from ultrasonic sensor was saved in a variable in the algorithm, current distance that obtained from ultrasonic sensor was used to compare with the previous distance, if current distance detected less than the previous distance, which means that the vehicle ahead had applied brake on his vehicle, the different between previous distance and current distance has been calculated in the algorithm. If the difference between previous distance and current distance was big, means that the rate of change of distance between vehicles was high implied that the vehicle ahead made a sudden braking, the buzzer would be activated to alert the driver.

The third objective of this project was to analyze the functionality and reliability of the alert system in the aspect of distance detection. In order to achieve this objective, a comparison between FPGA based collision avoidance alert system and Arduino based collision avoidance alert system has been done. From the result shown in Chapter 4, it has been proved that FPGA based collision avoidance alert system has better

performance and it was much more reliable to be used than Arduino based collision avoidance alert system since the FPGA could run concurrently during the execution and the processing speed was much higher than Arduino, thus FPGA abled to provide prompt response to the detected distance. If there was possibility of collision, the system would activate the alert component in a shorter time compared to Arduino. This was very important to a collision avoidance alert system since a collision on the road can be happened in seconds, if the collision avoidance alerts system unable to response to the dangerous situation in short period, a collision might be occurred.

5.3 Future Improvement

It has been stated in Chapter 4 that there were some limitations discovered in this project. Ultrasonic sensor that has been used as the distance detector in this project could only be able to detect object or vehicle ahead when the object or vehicle ahead was perpendicular to the sensor, thus, there might be some blind spot that were unable to be detected. A 24GHz radar sensor could be used in order to improve or to overcome this problem. Michael Klotz and Hermann Rohling (2010) stated that 24GHz radar sensor is very good in distance detection especially at detecting objects that reflect electromagnetic radiation such as metal object. It has longer range, higher update rates and higher precision compared to ultrasonic sensor. It allows the blind spot detection of object or vehicle. The speed of radar sensor in the aspect of detection is higher than ultrasonic sensor since the radar sensor is works with electromagnetic waves. Same as ultrasonic sensor, the electromagnetic wave hits the object and returns the wave at known speed but the speed is much higher than the sound wave emitted by ultrasonic sensor.

The result of detection by ultrasonic sensor was not consistent; it might be affected by the surrounding temperature and pressure. A radar sensor provides more consistent and accurate reading because it is much less affected by temperature.

Auto braking system could be installed on the collision avoidance system in order to stop the vehicle automatically when the driver fails to response to the activated alert component. It is better to alert the driver before a collision could happen but if the driver unable to response to the alert that has been activated, an auto braking system is required to avoid collision.

5.4 Commercialization Potential

The unique of this project is the controller. FPGA was used in this project in order to process and response to the detected distance in a high speed. It provided more reliable and better performance than other controller used in collision avoidance alert system. Nowadays, the number of road accidents increasing year by year and caused many people died. Most of the road accident was caused by human's unawareness. Therefore, a collision avoidance alert system with high performance is very important in order to reduce the occurrence of road accident. The collision avoidance alert system could be implemented in real time on every vehicle so that the driver could be aware of the road condition all the time.

REFERENCE

Deepak Tuteja1, Dhruv Jain2, Hemant Singla3 et al. (2014) Detailed Survey on Motion Sensing. *Journal of Basic and Applied Engineering Research* 1(8):27-31.

Ajit Kumar, Ankit Jaiswal, Neha Jaiswal et al. (2014) Vehicles Anti-collision System. International *Journal of Computer Applications (0975 – 8887)* 99(19): 7-9.

Triveni Shinde and Prof. B.V.Pawar (2013) Car Anti-Collision and Intercommunication System using Communication Protocol (A Prototype Model). *International Journal of Engineering Sciences & Research Technology [2087-2093]* 2(8): 25-32.

Michal Kelemen, Ivan Virgala, Tatiana Kelemenova et al. (2015) Distance Measurement via Using of Ultrasonic Sensor. *Journal of Automation and Control* 3(3):71-74.

Nishad Vivek Kumbhojkar and Chaitanya Avadhutchintan Kuber (2014) Ultrasonic Automatic Braking System For Forward Collision Avoidance With Accelerator Pedal Disengagement Mechanism. *International Journal and Magazine of Engineering Technology, Managemnt and Research* 4(3):12-18.

Hua Huo, Jun Qiang Liu, and Yong Jie Wang (2015) Flood Diversion Algorithm for Anticollision in RFID System. *International Journal of Distributed Sensor Networks* 2015(2015).

Chetan Sharma (2011) Designing Of Four Port Controlled Switch Using Verilog. *Journal of Global Research in Computer Science*. 1(3).

Michael Klotz and Hermann Rohling (2010) 24 GHz radar sensors for automotive applications. *Journal of Telecommunications and Information Technology (JTIT).*

Shival Dubey and Abdul Wahid Ansari (2013) Design and Development of Vehicle anti-collision System using Electromagnet and Ultrasonic Sensors. *International Journal on Theoretical and Applied Research in Mechanical Engineering (IJTARME)*, Volume-2, Issue-1.

Vaughn Betz, Jonathan Rose and Alexander Marquardt (2012) Architecture and CAD for Deep-Submicron FPGAS. Berlin: Springer Science & Business Media.

Clive Maxfield (2011) FPGAs: Instant Access. Newnes.

Samarjit Chakraborty and Jörg Eberspächer (2012) Advances in Real-Time Systems. Berlin: Springer Science & Business Media.

Subhas C. Mukhopadhyay and Joe-Air Jiang (2013) Wireless Sensor Networks and Ecological Monitoring. Berlin: Springer Science & Business Media.

John Vetelino, Aravind Reghu (2010) Introduction to Sensors. United States of America : CRC Press.

Terasic Technologies (2012) DE0 User Manual. Available at: file:///C:/Users/User/Downloads/DE0_User_manual_2012.pdf

Cytron Technologies (2013) Product User's Manual – HC-SR04 Ultrasonic Sensor, Available at: https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit.

Aflab Sarwar (2012) FPGA vs Microcontrollers. Available at: http://electrodesigns.net/fpgas-microcontrollers/

Ken Smyers (2013) StatTalk: Warren S. Johnson Patented the Thermostat in 1883. Thanks ASME Milwaukee for the History & Heritage. Available at: http://controltrends.org/building-automation-and-integration/06/stattalk-warren-s-johnson-patented-the-thermostat-in-1883-thanks-asme-milwaukee-for-the-history-heritage/#more-9415

A. Rogalski* (2012) History of infrared detectors. Available at: http://antonirogalski.com/wp-content/uploads/2012/12/History-of-infrared-detectors.pdf

XILINX (2016) Field Programmable Gate Array (FPGA). Available at: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm

Jeff Johnson (2011) List and comparison of FPGA companies. Available at: http://www.fpgadeveloper.com/2011/07/list-and-comparison-of-fpga-companies.html

Verilog Dot Com (2012) Verilog Resources. Available at: http://www.verilog.com/

Adela Megan Willy (2015) 5 common causes of road accidents in Malaysia. Available at: http://www.motorme.my/5-common-causes-of-road-accidents-in-malaysia

Free Malaysia Today. 2014. Sudden braking causes 13-vehicle pile-up. 3 December 2014.

International News. 2011. Human Error Accounts For 90% of Road Accidents.

APPENDICES

A. Coding of sensor controller finite state machine

```
module sensor_controller(clock, reset, echo,
compare_done, speed, trigger, distance, en_compare,
en_next_comp, state);

input  clock, reset, echo, compare_done, speed;

output trigger, en_compare, en_next_comp;
output [31:0] distance;
output [3:0]  state;

reg  trigger, en_compare, en_next_comp, old_speed;
reg  [31:0] distance;
reg  [31:0] trig_counter  = 32'd0;
reg  [31:0] dist_counter  = 32'd0;
reg  [31:0] cycle_counter = 32'd0;
reg  [3:0]  current_state;
reg  [3:0]  next_state;
wire [3:0]  state;
wire speed_change;

parameter idle    = 3'b000,
                start   = 3'b001,
                waiting = 3'b010,
                receive = 3'b011,
                stop    = 3'b100;

//reset
always @(posedge clock)
begin
     if(reset == 1'b1)
     begin
          current_state <= idle;
     end
```

```
     else if (speed_change == 1'b1)
     begin
          current_state <= idle;
     end

     else
     begin
          current_state <= next_state;
     end
end

//state always equal to current_state
assign state = current_state;

//to allow detection of change of speed
always @ (posedge clock)
begin
     if(reset == 1'b1)
     begin
          old_speed <= 1'b0;
     end

     else
     begin
          old_speed <= speed;
     end
end

//to detect the change of speed
assign speed_change = (speed != old_speed)?
1'b1 :1'b0;


//next state
always @(current_state or compare_done or trig_counter
or cycle_counter or echo or en_next_comp or distance
or dist_counter)
begin
     next_state <= current_state;
```

```
      case (current_state)
            idle:
                  begin
                        if (compare_done == 1'b1)
                        begin
                              next_state <= start;
                        end
                  end

            start:
                  begin
                        if(trig_counter == 32'd500)
//trigger for 10us
                        begin
                              next_state <= waiting;
                        end
                  end

            waiting:
                  begin
                        if (echo == 1'b1)
                        begin
                              next_state <= receive;
                        end
                  end

            receive:
                  begin
                        if (echo == 1'b0)
                        begin
                              next_state <= stop;
                        end

//assume max distance to detect echo is 30cm
                        else if(dist_counter == 88235)
                                                begin
                              next_state <= stop;
                        end
                  end
```

```
          stop:
               begin
                    distance <= dist_counter;



                    //1 cycle measurement = 60ms =
3000000
                    if(cycle_counter == 32'd2999999)
                                             begin
                         next_state <= idle;
                    end
               end

          default: next_state <= idle;
     endcase
end

//Action in each state
always @(posedge clock)
begin

     if(current_state != idle)
     begin
          cycle_counter <= cycle_counter + 1;
     end

     if(reset == 1'b1)
     begin
          cycle_counter <= 32'd0;
          trig_counter  <= 32'd0;
          dist_counter  <= 32'd0;
          trigger          <= 1'b0;
          en_compare    <= 1'b0;
          en_next_comp <= 1'b0;
     end

     else
     begin
          case (current_state)
```

```
                idle:
                begin
                     cycle_counter <= 32'd0;
                     trig_counter  <= 32'd0;
                     dist_counter  <= 32'd0;
                     trigger          <= 1'b0;
                     en_compare    <= 1'b0;
                     en_next_comp <= 1'b0;
                end

                start:
                begin
                     trigger      <= 1'b1;


//count trigger time
                     trig_counter <= trig_counter + 1;
                                          end

                waiting:
                begin
                     trigger <= 1'b0;
                end

                receive:
                begin                                          //if
echo ==1, count distance
                     dist_counter = dist_counter +
1'd1;                                          end

                stop:
                begin
                     if(distance != 32'd0)

//to enable the comparison of distance
                          en_compare   <= 1'b1;

                     //1 cycle measurement = 60ms =
3000000
```

```
                    if(cycle_counter == 32'd2999999)
                                                begin

//to enable next comparison
                            en_next_comp <= 1'b1;


//to disable the comparison of distance
                            en_compare   <= 1'b0;

end
                end
           endcase
     end
end
endmodule
```

B. Coding of comparator finite state machine

```
module comparator2
(clock,
reset,
en_compare,
en_next_comp,
speed,
distance,
compare_done,
safe_distance_alert,
collision_alert,
comp_state,
diff_dist,
current_dist);

input clock, reset, en_compare, en_next_comp, speed;
input  [31:0] distance;
output [31:0] diff_dist;
output [31:0] current_dist;
output compare_done, safe_distance_alert,
collision_alert;
output [3:0]  comp_state;

reg   compare_done, safe_distance_alert,
collision_alert, hold_on;
reg   [31:0] safe_distance;
```

```
reg   [31:0] prev_dist;
reg   [31:0] current_dist = 32'd0;
reg   [3:0]  current_state;
reg   [3:0]  next_state;
reg   en_collision_alert;
reg   en_safe_distance_alert;
reg   old_speed;

wire [31:0] diff_dist;
wire [3:0]  comp_state;
wire speed_change;

parameter                       idle                    =
3'b000,
                update                      = 3'b001,
                compare_distance = 3'b010,
                compare_diff         = 3'b011,
                waiting                     = 3'b100;

//reset
always @(posedge clock)
begin
     if(reset == 1'b1)
          begin
               current_state <= idle;
          end

     else if (speed_change == 1'b1)
          begin
               //delay for one clock
                hold_on       <= 1'b1;
                current_state <= next_state;
          end

     else if (hold_on == 1'b1)
          begin
               current_state <= idle;
               hold_on       <= 1'b0;
          end

     else
          begin
               current_state <= next_state;
          end
end
```

```
assign comp_state = current_state;

//update prev and current distance
always @(posedge clock)
begin
     if(reset == 1'b1)
     begin
          prev_dist    <= 32'd0;
          current_dist <= 32'd0;
     end

     else
     begin
          if(current_state == update)
          begin
               //update previous distance
               prev_dist    <= current_dist;
               //update current distance
               current_dist <= distance;
          end
     end
end

assign diff_dist = (prev_dist >= current_dist)?
prev_dist - current_dist : 32'd0;

//safe distance alert
always @(posedge clock)
begin
     if(reset == 1'b1)
     begin
          safe_distance_alert  <= 1'b0;
     end

     else
     begin
          if (current_state == compare_distance ||
speed_change == 1'b1)
          begin
               safe_distance_alert <=
en_safe_distance_alert;
          end
     end
end

//collision alert
```

```
always @(posedge clock)
begin
     if(reset == 1'b1)
     begin
          collision_alert <= 1'b0;
     end

     else
     begin
          if (current_state == compare_diff ||
current_state == compare_distance || speed_change ==
1'b1)

          begin
               collision_alert <= en_collision_alert;
          end
     end
end

//To allow detection of change of speed
always @ (posedge clock)
begin
     if(reset == 1'b1)
          begin
               old_speed <= 1'b0;
          end
     else
          begin
               old_speed <= speed;
          end
end
assign speed_change = (speed != old_speed)?
1'b1 :1'b0;

//next state and action
always @(current_state or speed_change or speed or
en_compare or en_next_comp or distance
or current_dist or safe_distance_alert or diff_dist or
collision_alert or prev_dist)

begin
     compare_done           = 1'b0;
     en_safe_distance_alert = 1'b0;
     en_collision_alert     = 1'b0;
     safe_distance = (speed == 1'b0)? 32'd14706 :
32'd29412; //5cm , 10cm
```

```
     //if there is switching of mode, reset the system
            if (speed_change == 1'b1)
     begin
          en_safe_distance_alert = 1'b0;
          en_collision_alert     = 1'b0;
     end

     next_state <= current_state;

     case (current_state)
          idle:
               begin
                    compare_done <= 1'b1;
                    if (en_compare == 1'b1)
                    begin
                         next_state <= update;
                    end
               end

          update:
               begin
                    next_state <= compare_distance;
               end

          compare_distance:
               begin
                         //if no distance detected,
deactivate the alert system
                    if (current_dist == 32'd0)
                    begin
                         en_collision_alert     <=
1'b0;
                         en_safe_distance_alert <=
1'b0;
                         next_state             <=
idle;
                    end

                    //safe distance = 10cm/5cm
                    else if (current_dist <=
safe_distance)
                    begin
                         if(collision_alert <= 1'b1)
                         begin
                              en_collision_alert <=
```

```
1'b0;
                         end
                         en_safe_distance_alert <=
1'b1;
                         next_state             <=
waiting;
                    end

                    //when it is safe but the alert
still activated, turn it off
                    else if (current_dist >
safe_distance && safe_distance_alert == 1'b1)
                    begin
                         en_safe_distance_alert <=
1'b0;
                    end

                    else if (current_dist >
safe_distance && safe_distance_alert == 1'b0)
                    begin
                      //in parking mode, speed = 0
                         if (speed == 1'b0)
                         begin
                              next_state <= waiting;
                         end

                         else
                         begin
                              next_state  <=
compare_diff;
                         end
                    end
               end

          compare_diff:
               begin
                    //diff_dist = difference between
current and previous distance, difference = 5cm
                    if (diff_dist >= 32'd14706)
                    begin
                         en_collision_alert <= 1'b1;
                         //to get the next distance
                         next_state         <=
waiting;
                    end
```

```
                    //when it is safe but the alert
still activated, turn it off
                    else if (diff_dist < 32'd14706 &&
collision_alert == 1'b1 )
                    begin
                         en_collision_alert <= 1'b0;
                         next_state         <=
waiting;
                    end

                    else if (diff_dist < 32'd14706 &&
collision_alert == 1'b0)
                    begin
                         next_state <= waiting;
                    end
               end

          waiting:
               begin
                    if(en_next_comp == 1'b1)
                    begin
                         next_state <= idle;
                    end
               end

          default: next_state <= idle;
     endcase
end

endmodule
```


C. Coding of main module

```
module main2
(clock,
reset,
speed,
trigger,
echo,
safe_distance_alert,
collision_alert,
state,
comp_state,
diff_dist,
current_dist,
distance);
```

```
input clock, reset, echo, speed;

output trigger, safe_distance_alert, collision_alert;
output [3:0]  state, comp_state;
output [31:0] distance;
output [31:0] diff_dist;
output [31:0] current_dist;

wire echo, speed, compare_done, en_compare,
en_next_comp;
wire [31:0] distance;


sensor_controller inst_1 (
clock,
~reset,
echo,
compare_done,
speed,
trigger,
distance,
en_compare,
en_next_comp,
state);

comparator2 inst_2 (
clock,
~reset,
en_compare,
en_next_comp,
speed,
distance,
compare_done,
safe_distance_alert,
collision_alert,
comp_state,
diff_dist,
current_dist);

endmodule
```