

CAR TRACTION CONTROL SYSTEM

By

ADIBAH BINTI MOHD ISMAIL

FINAL PROJECT REPORT

**Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)**

**Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan**

© Copyright 2007

by

Adibah Binti Mohd Ismail, 2007

CERTIFICATION OF APPROVAL


CAR TRACTION CONTROL SYSTEM

by

Adibah Binti Mohd Ismail

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:



Dr. Nor Hisham Hamid
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2007

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Adibah Binti Mohd Ismail

ABSTRACT

This project explores the potential of implementing fuzzy logic algorithm for traction control system using VHDL. Previously, the project on car traction control was done by simulation using fuzzy logic approach. The Fuzzy Logic Toolbox in MATLAB software is used to create simulation for fuzzy logic system. The challenge of the project is to design the control system using hardware description language for future implementation on hardware using FPGA. Fuzzy logic controller provides optimum control according to the conditions specify. It is useful when the driving condition is uncontrolled. The core programming language which will be used as the hardware description language is VHSIC Hardware Description Language (VHDL). VHDL is used in FPGA – based implementation. The methodology includes designing the fuzzy logic controller, development of the algorithm and codes programming. After that, the following phase includes testing and troubleshooting. Lastly, carry out the documentation. In conclusion, it is possible to develop the algorithm for fuzzy - based car traction control system using VHDL. The implementation of the control system using VHDL is viable for future implementation onto FPGA. Thus the performance of the car traction control would be enhanced

ACKNOWLEDGEMENTS

"In the name of Allah, Most Gracious, Most Merciful"

The project would not have been possible without the help of a number of people. I would like to express my utmost gratitude to all of them.

My foremost gratitude goes to my Final Year Project supervisor, Dr. Nor Hisham Hamid for his guidance and endless supports in accomplishing this project. The experience of being supervised by him is priceless. He has continuously encouraged and challenged me throughout this project. His critics and suggestions were valuable in determining the final outcome of the project.

I would also like to express my gratitude to all UTP technicians for their guidance and assistant while completing my project.

Last but not least, I would like to thanks to my beloved family for their love and support which is the source of my strength and motivation. Not forgetting to all of my friends who have directly or indirectly given their assistance in various aspects throughout the development of the project, I thank them all.

TABLE OF CONTENTS

LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF ABBREVIATIONS	x
CHAPTER 1 INTRODUCTION	1
1.1 Background of Study	1
1.2 Problem Statement	1
1.3 Objectives	4
1.4 Scope of Study	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Car Traction Control System	5
2.1.1 Traction	5
2.1.2 Traction Control System	5
2.2 Fuzzy Logic	6
2.2.1 Fuzzy Logic	6
2.2.2 Fuzzy Logic Controller	10
2.2.3 Fuzzy Logic on Hardware	10
2.3 Discussion	11
CHAPTER 3: METHODOLOGY / PROJECT WORK	16
3.1 The Overall Fuzzy Logic Controller Simulation	16
3.1.1 The FIS Editor	16
3.1.2 The Membership Function Editor	17
3.1.3 The Rule Editor	17
3.2 Architecture, Design Specification & Familiarize with VHDL Coding	19
3.3 Development VHDL Coding for Individual Sub - Block	20
3.4 Integrating All Sub - Blocks	20
3.5 Simulation & Design Verification	21
CHAPTER 4: RESULTS & DISCUSSION	22
4.1 Simulation Using MATLAB	22
4.2 The Development of VHDL Coding	29

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS.....	32
5.1 Conclusion.....	32
5.2 Recommendations for Future Works.....	33
REFERENCES.....	34
APPENDICES.....	36
APPENDIX A VHDL SOURCE CODES.....	37
APPENDIX B TEST BENCH SOURCE CODES.....	42

LIST OF TABLES

Table 1: Membership functions for input parameters (Load).....	12
Table 2: Membership functions for input parameters (Corner).....	12
Table 3: Membership functions for input parameters (Speed).....	12
Table 4: Membership functions for output parameters (Front Spring Rate).....	13
Table 5: Membership functions for output parameters (Rear Spring Rate).....	13
Table 6: If – Then Rules.....	14

LIST OF FIGURES

Figure 1:	Car Traction Control.....	6
Figure 2:	Classical Logic versus Fuzzy Logic.....	7
Figure 3:	A Typical Membership Function.....	8
Figure 4:	Fuzzy Logic Phases.....	8
Figure 5:	Crisp Sets versus Fuzzy Sets.....	9
Figure 6:	MATLAB FIS Editor Main Screen.....	17
Figure 7:	MATLAB Rule Editor Main Screen.....	18
Figure 8:	The Block Diagram for a Fuzzy Logic Controller.....	19
Figure 9:	The Block Diagram for Fuzzification.....	20
Figure 10:	Aldec Active HDL 5.1 Main Screen.....	21
Figure 11:	Membership Function for Corner Input.....	22
Figure 12:	Membership Function for Load Input.....	23
Figure 13:	Membership Function for Speed Input.....	23
Figure 14:	Membership Function for Front Spring Rate Output.....	23
Figure 15:	Membership Function for Rear Spring Rate Output.....	24
Figure 16:	Rule Viewer When Corner = -4, Load = 1.49 and Speed = 0.....	25
Figure 17:	Rule Viewer When Corner = 3.5, Load = 1.49 and Speed = 0.....	27
Figure 18:	Rule Viewer When Corner = 0, Load = 1.49 and Speed = 0.....	28
Figure 19:	The Simulation Results for Subtractor.....	29
Figure 20:	The Simulation Results for Multiplication.....	30
Figure 21:	The Simulation Results for Comparator.....	30
Figure 22:	The Output Waveform of Verification for the Integrated Sub – Block.....	31

LIST OF ABBREVIATIONS

ASIC : Application Specific Integrated Circuit

EDA : Electronic Design Automation

FIS : Fuzzy Inference System

FPGA : Field Programmable Gate Array

GUI : Graphical User Interface

MF : Membership Functions

VHDL : VHSIC Hardware Description Language

VHSIC: Very High Speed Integrated Circuit

CHAPTER 1

INTRODUCTION

1.1 Background of Study

The advancement in control system and technology has lead to an improved safety feature in automotive industry. Currently, many modern vehicles are equipped with the traction control system. This feature is very important in providing a better and enhanced safety and performance of vehicles while on road.

Basically, the word of ‘traction’ in Oxford Advanced Learner’s Dictionary means the capability of a wheel to grip the road without slipping [1]. Traction control system will prevent the loss of grip on the wheels [2]. Traction control system is beneficial to both road and race cars [3]. It enables a car to accelerate and also allows cars to maintain the maximum traction in order to reduce wheel slip when the car is under acceleration. As mentioned earlier, most modern car manufacturers have offered the traction control system and the control system is part of the main component in the suspension system of cars [4]. The suspension system enables the vehicle to move on rough surface with minimum of bumpy ride and also reduces tendency of loosing traction [4].

These control systems will ensure a smooth driving and also provide an improved safety measures to the driver and also the passengers in the vehicle.

1.2 Problem Statement

Control system is typically design based on either conventional control method or fuzzy logic method [5]. A conventional control method or design initially involves understanding of the physical or the objectives of the system and its requirements. Then, the control system designer has to develop the model of the system based on the understanding from the previous step. The following step is to determine a

simplified version of the control system using the linear – control theory. This next step involves the development of an algorithm for the simplified control system. The final step is the simulation of the control system design. If the results or performance from the observation of the simulation does not achieve the requirement, the system has to be modified.

In fuzzy logic design, the designer has to initially understand and determine the system behavior through knowledge and experience. This is followed by developing the control algorithm using the fuzzy rules. Fuzzy rules explain the relationship between inputs and outputs. The final step is to simulate and debug the control system design [5].

From both types of designing approaches, fuzzy logic approach offers a more simple method. The fuzzy rule – based feature is mainly focusing in application instead of programming. Fuzzy logic is flexible and tolerant of imprecise data [6]. Fuzzy logic controller can deal with complicated and imprecise nonlinear processes [7]. Following are the reasons why fuzzy control system is favorable among engineers [8]:

- 1) Rule – based inference is powerful and easy
- 2) Fuzzy control offered efficient technology at low cost for automation in industry
- 3) Fuzzy control was suitable for practicing engineers and people without mathematical background
- 4) Fuzzy control could easily deal with nonlinear control problems with language by introducing human – knowledge into controllers
- 5) Fuzzy technology was able to solve problems which had not been solved

Fuzzy logic plays the major role in the traction control system. Since the traction control system needs to determine when the wheels are losing the grip, fuzzy logic

will help the traction control system in decision making in order to take action so that the vehicle will maintain at maximum traction.

The implementation of fuzzy logic controller on standard hardware such as general – purpose microprocessors is very favorable. Hardware implementation of the fuzzy logic controller can be divided into dedicated hardware and standard hardware. In dedicated hardware, fuzzy chips are implemented on an Application Specific Integrated Circuit (ASIC). The implementation of ASIC offers high speed. Unfortunately, once chips are created, no changes can be done. Therefore, the implementation of fuzzy logic – based car traction control system on Field Programmable Gate Array (FPGA) is considered. The consideration is due to its flexibility. FPGA is a semiconductor device that consists of programmable logic components and programmable interconnects. FPGA can be reconfigured by user and reprogram for many different designs. By using an appropriate electronic design automation (EDA) tool, FPGA can produce the desired system design for quick prototyping. FPGA can execute desired algorithm instead of a sequence of instructions on predefined hardware resources. Designs for FPGA are defined in a hardware language such as VHSIC hardware description language (VHDL) and are verified by simulation. Many designs can be implemented due to reusability of the FPGA. Hence, FPGA is suitable for fast implementation and quick hardware verification.

In the electronic system, the hardware description languages that are most widely used are VHDL and Verilog. However, VHDL promotes portable descriptions. This is because VHDL is independent of design methodology. It describes the behavior of an electronic system. Such description allows simulator to simulate the behavior of the system without having to actually construct the system [9]. Therefore, the design cycle could be reduced. Thus this project would utilize VHDL as the hardware descriptive language.

In this project, the main concern is implementing traction control system using VHDL for future implementation on FPGA. The traction control system will be designed using the fuzzy logic approach. In other words, the traction control is an implementation of a fuzzy logic controller.

1.3 Objectives

The main objective of this project is to develop the fuzzy logic based control system using VHDL for future implementation on FPGA.

1.4 Scope of Study

This project only involves with the electronic part of the traction control system. The scope of study will not deal with the mechanical part. By leaving the mechanical portion out, the outcome of the project will not be affected. The scope of the study is divided into the following sections:

1) Understanding on how fuzzy logic based traction control system work

The understanding is done by using the existing design (previous work) [4]. Then, based on the findings from the existing design, the simulation of the fuzzy logic is done on MATLAB.

2) Development of the fuzzy logic basic traction controller using VHDL for hardware implementation

Upon developing the controller using VHDL, the main blocks for the structure of the control system need to be determined. The construction of VHDL coding is done using Aldec Active HDL 5.1.

CHAPTER 2

LITERATURE REVIEW

This project is related to the traction control system, fuzzy logic and FPGA. Thus, this section of the report provides the related literature reviews which have been thoroughly reviewed. This section will increase the understanding and knowledge related to the project.

2.1 Car Traction Control System

2.1.1 Traction

Traction is referring to the car's ability to maintain adhesive friction between the vehicle and the surface [3].

2.1.2 Traction Control System

The system is consisted of springs and the shock absorbers. The purpose of the springs is to sustain the weight and the load of the vehicle. The shock absorbers help to reduce and control spring actions. The shock absorber will quickly remove the unwanted bouncing when the vehicle travels over holes or bumpy roads. The shock absorbers will only allow the necessary spring movement. Thus, the safety and performance of the driving will be enhanced. Figure 1 shows the electronically controlled air suspension that consists of the following components [4]:

1) Sensors

The sensors will receive the appropriate data and channel to the controller. Basically these sensors are used in measuring acceleration and force.

II) Electronic Control Unit

The control unit will receive the data from the sensors. It will process them to decide the appropriate actions to be taken according to the data received. In this project the control unit will control the damper setting depending on the road conditions.

II) Dampers

The functions of the dampers are for controlling the springs. The dampers can be medium, soft or super – soft. These dampers are controlled by solenoids valves.

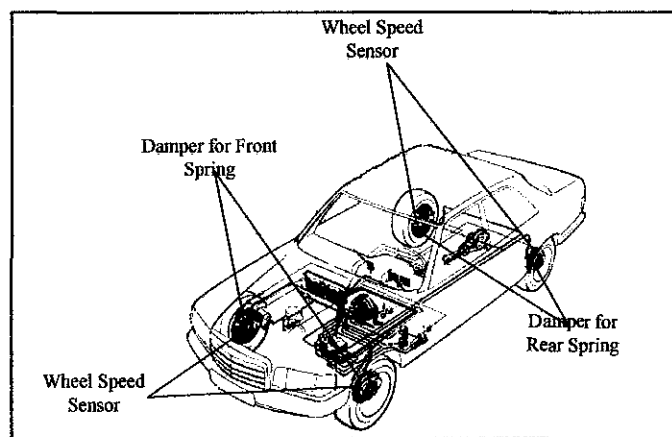


Figure 1: Car Traction Control [10]

Traction control system deals with the loss of friction during acceleration. The control system is also deals with the suspension system which reduces the possibility of losing the traction between tires and road surface during cornering [4].

2.2 Fuzzy Logic

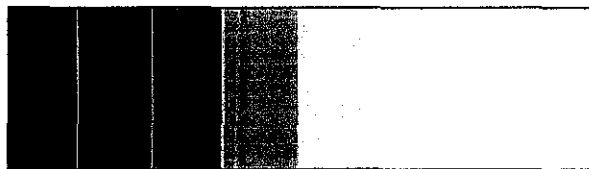
2.2.1 Fuzzy Logic

The concept of fuzzy logic was first introduced in 1960's by Professor Lofti Zadeh from the University of California at Berkley. Fuzzy logic is a methodology for describing operational laws of a system in linguistic terms instead of mathematical equations [11]. Fuzzy logic begins with the theory of a fuzzy set. A fuzzy set is a set with blurry boundaries. Fuzzy set expresses vague concepts [12]. Fuzzy logic allows

the programmer to deal with natural, “linguistic sets” of states, such as very hot, warm, cool, cold, etc [5]. Fuzzy logic is able to estimate human decision – making using natural – language terms instead of quantitative terms [5].It is helpful in formulating an extremely difficult algorithm solutions.



(a) Classical Logic: Elements are either



(b) Fuzzy Logic: Allowing elements to exist in between

Figure 2: Classical Logic versus Fuzzy Logic [13]

Fuzzy theory is a mathematical theory that describes ambiguities using quantitative description [14]. Hence, fuzzy logic caters for elements that are existing in between true or false unlike the classical logic. These elements are partially true and false as shown in Figure 2.

The limits of a fuzzy set are governed by degree of membership in a fuzzy set. Membership function is a curve that classifies how input is mapped to a degree of membership. The degree of membership of an element to a fuzzy set is a representation of the degree of the participation of the element to the set. Fuzzy set allows each element of X to belong to the set with a membership degree characterized by values that vary between 0 and 1 [14]. A typical membership function for a fuzzy set is displayed in Figure 3.

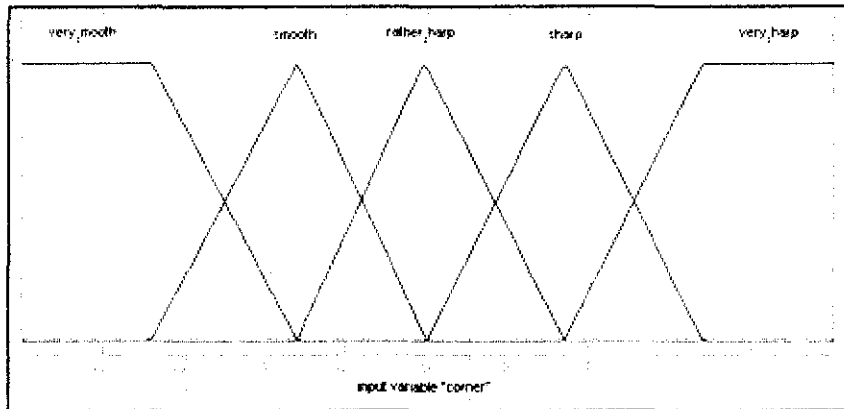


Figure 3: A Typical Membership Function

Figure 3 shows that the input for Corner can be classified to Very Smooth, Smooth, Rather Sharp, Sharp and Very Sharp. The x – axis represent the input whereas the y – axis represents the degree of the membership function.

Figure 4 show the application of fuzzy logic involves three steps: fuzzification, rule evaluation, and defuzzification [4].

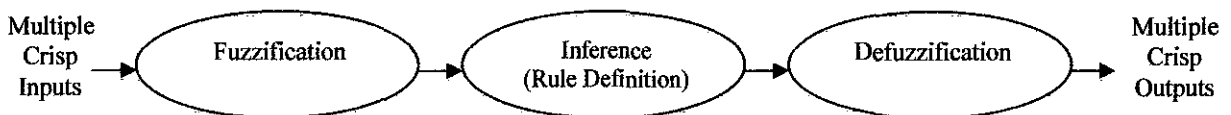


Figure 4: Fuzzy Logic Phases [11]

1) Fuzzification

In this phase, the crisp inputs are converted into fuzzy membership functions. Crisp inputs are non - fuzzy inputs. Figures 5 show the differences between crisp sets and fuzzy membership sets.

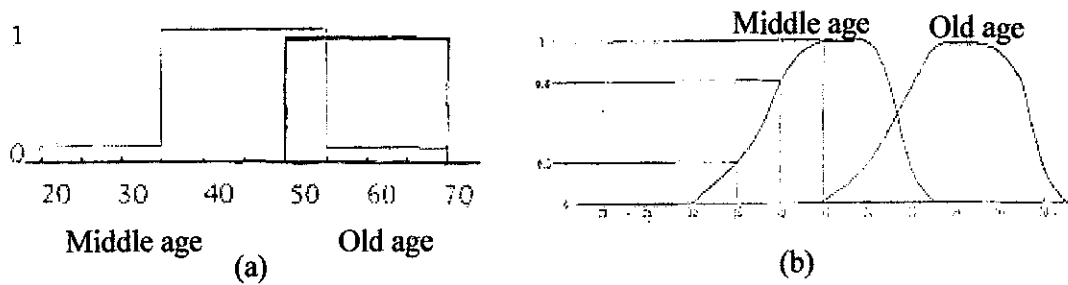


Figure 5: Crisp Sets versus Fuzzy Sets; (a) Crisp sets of middle age and old age (b) Fuzzy sets of middle age and old age (membership function) [8]

The development of the membership functions for the inputs is important. These membership functions are defined by both a range of values and a degree of membership. The variable must be clearly mapped in order to observe which membership functions it belongs to and the relative degree to which it is a member. This will determine the “weighted” membership in the membership function [13]. A variable is allowed to have a weighted membership in several membership functions simultaneously. Fuzzy membership functions consist of a range of values and can overlap.

II) Inference Rule Design

The function of the inference is to relate the outputs action of the controller to the observed inputs. This is done after both inputs and outputs have been defined by the membership functions.

Example:

If corner is **VERY SMOOTH**, then set front spring rate is **VERY STIFF**

If corner is **SMOOTH**, then set front spring rate is **STIFF**

If corner is **RATHER SHARP**, then set front spring rate is **ORDINARY**

If corner is **SHARP**, then set front spring rate is **SOFT**

If corner is **VERY SHARP**, then set front spring rate is **VERY SOFT**

These If – Then rules can relate multiple input and output variables. Fuzzy logic controller is capable of defining any relationship that can be described in linguistic

variables. This is due to the rules that are based on word descriptions instead of mathematical expressions. The rules which composed of these variables are also weighted since the variables have weighted memberships. Depending on the input variable, different rules have different impacts on the controller.

There are two types of inference systems that can be implemented for any fuzzy logic controller: Mamdani - type and Sugeno – type. Mamdani – type of inference is most commonly used for a fuzzy logic controller [12]. This type of inference expects the membership functions to be fuzzy sets. Thus each of the variable outputs needs to be defuzzified. Sugeno – type does not deal with membership functions. Hence, this type of inference does not require defuzzification. For this project, the author implements Mamdani – type because the outputs of the fuzzy logic controller are in the form of fuzzy sets.

III) Defuzzification

This phase involves producing of crisp output. The fuzzy logic controller must produce a meaningful output to the system involves. The meaningful output is converted from the internal fuzzy output. The meaningful output is the output that can be used by the controlled system.

2.2.2. Fuzzy Logic Controller

Fuzzy logic controllers have been widely used in both consumer products and industrial process controls. Fuzzy logic controllers are suitable for complicated and imprecise processes where mathematical model does not exist. Fuzzy logic controller can approximate human mind's behavior which is able to work well under imprecise conditions [11]

2.2.3 Fuzzy Logic Controller on Hardware

Fuzzy logic controller can be divided into:

- General purposed fuzzy processors with specialized fuzzy computations
- Dedicated fuzzy hardware for specific applications

The general purposed fuzzy processors offers a fast implementation and flexible application but a lower performance. The dedicated fuzzy hardware requires a long development time but provides an optimum performance. Recently, the implementation on FPGA has been encouraging since FPGA provides a compromise between special purpose ASIC hardware and general purposed processors. The implementation of fuzzy logic controller on FPGA is attractive due to the greatly reduced development time. [15]

2.3 Discussion

From the reading done in the existing design [4], the author had able to identify 3 inputs and two outputs for the car traction control system. Following are the lists of the inputs and outputs along with their membership functions for the fuzzy logic based car traction control system:

- **First Input - Sharpness of the Corners**
Membership Functions (MF):
 - a) VERY SMOOTH
 - b) SMOOTH
 - c) RATHER SHARP
 - d) SHARP
 - e) VERY SHARP

- **Second Input – Vehicle Load**
MF:
 - a) NOT HEAVY
 - b) HEAVY

- **Third Input – Vehicle Speed**
MF:
 - a) VERY SLOW
 - b) SLOW
 - c) RATHER FAST
 - d) FAST

e) VERY FAST

- Output – Spring Rate of Front and Rear Wheel

MF:

- a) VERY STIFF
- b) STIFF
- c) ORDINARY
- d) SOFT
- e) VERY SOFT

Each of these membership functions is representing certain range of input and outputs values. The representation of the values can be observed in Table 1 to Table 5.

Table 1: Membership functions for input parameters (Load)

MF	MF Values	Equivalent weight of passengers (kg)
NOT HEAVY	0 – 0.333	0 – 233.1
HEAVY	0.167 – 0.5	116.9 - 350

Table 2: Membership functions for input parameters (Corner)

MF	MF Values	Equivalent steering angle (°)
VERY SMOOTH	0 – 0.333	0 – 119.88
SMOOTH	0.167 – 0.5	60.12 – 180
RATHER SHARP	0.333 – 0.667	119.88 – 240.12
SHARP	0.5 – 0.833	180 – 299.88
VERY SHARP	0.667 – 1.0	240.12 - 360

Table 3: Membership functions for input parameters (Speed)

MF	MF Values	Equivalent velocity (km / h)
VERY SLOW	0 – 0.333	0 – 59.94
SLOW	0.167 – 0.5	30.06 – 90

RATHER FAST	0.333 – 0.667	59.94 – 120.06
FAST	0.5 – 0.833	90 – 149.94
VERY FAST	0.667 – 1.0	120.06 – 180

Table 4: Membership functions for output parameters (Front Spring Rate)

MF	MF Values	Equivalent percentage (%)
VERY SOFT	0 – 0.333	0 – 33.3
SOFT	0.167 – 0.5	16.7 – 50.0
ORDINARY	0.333 – 0.667	33.3 – 66.7
STIFF	0.5 – 0.833	50.0 – 83.3
VERY STIFF	0.667 – 1.0	66.7 - 100

Table 5: Membership functions for output parameters (Rear Spring Rate)

MF	MF Values	Equivalent percentage (%)
VERY SOFT	0 – 0.333	0 – 33.3
SOFT	0.167 – 0.5	16.7 – 50.0
ORDINARY	0.333 – 0.667	33.3 – 66.7
STIFF	0.5 – 0.833	50.0 – 83.3
VERY STIFF	0.667 – 1.0	66.7 - 100

These inputs and outputs of the control system are governed by a set of if – then rules. For this project, Table 6 shows the rules which enable the control system to get stabilized according to the given inputs.

Table 6: If – Then Rules

If			Then	
Corner	Load	Speed	Front Spring Rate	Rear Spring Rate
VERY SMOOTH	NOT HEAVY	VERY FAST	VERY STIFF	VERY SOFT
VERY SMOOTH	HEAVY	RATHER FAST	VERY STIFF	VERY SOFT
VERY SMOOTH	HEAVY	FAST	VERY STIFF	VERY SOFT
VERY SMOOTH	HEAVY	VERY FAST	VERY STIFF	VERY SOFT
SMOOTH	NOT HEAVY	VERY FAST	STIFF	SOFT
SMOOTH	HEAVY	RATHER FAST	STIFF	SOFT
SMOOTH	HEAVY	FAST	STIFF	SOFT
SMOOTH	HEAVY	VERY FAST	STIFF	SOFT
RATHER SHARP	NOT HEAVY	VERY FAST	ORDINARY	ORDINARY
RATHER SHARP	HEAVY	RATHER FAST	ORDINARY	ORDINARY
RATHER SHARP	HEAVY	FAST	ORDINARY	ORDINARY
RATHER SHARP	HEAVY	VERY FAST	ORDINARY	ORDINARY
SHARP	NOT HEAVY	VERY FAST	SOFT	STIFF
SHARP	HEAVY	RATHER FAST	SOFT	STIFF
SHARP	HEAVY	FAST	SOFT	STIFF
SHARP	HEAVY	VERY FAST	SOFT	STIFF
VERY SHARP	NOT HEAVY	VERY FAST	VERY SOFT	VERY STIFF
VERY	HEAVY	RATHER	VERY SOFT	VERY STIFF

SHARP		FAST		
VERY SHARP	HEAVY	FAST	VERY SOFT	VERY STIFF
VERY SHARP	HEAVY	VERY FAST	VERY SOFT	VERY STIFF

Based on these findings, the author will be able to proceed to simulation of the control system in MATLAB. Eventually, the author could continue to develop an algorithm for the car traction control system.

CHAPTER 3

METHODOLOGY / PROJECT

This chapter describes the methodology involved in order to achieve the objective. The methodology begins with the simulation of the whole fuzzy logic controller in MATLAB. This is then followed by the development of the algorithms.

3.1 The Overall Fuzzy Logic Controller Simulation

Before developing the algorithms for the control system, the simulation of the fuzzy logic controller design in MATLAB is recommended. The simulation is to verify that the control design is practical to be implemented. It is unreasonable and wasting of time to develop algorithms of an uncertain design.

The author utilized the MATLAB's Fuzzy Logic Toolbox to create and edit fuzzy logic controller. The author conveniently made use of the graphical user interface (GUI) tools that are user – friendly in simulating the fuzzy logic based control design. The toolbox provides the Fuzzy Inference System (FIS) Editor, the Membership Function Editor, the Rule Editor and the Rule Viewer [12].

Following are the editors in the Fuzzy Logic Toolbox used by the author to perform specific functions for the simulation.

3.1.1 The FIS Editor

The FIS Editor deals with the high – level issues of a system (e.g. numbers of inputs and outputs, the names for the inputs and outputs). Figure 6 show the main screen for the FIS Editor. From this figure, the author defined the numbers of inputs and output, the type of inference used and the type of the defuzzification used. There were three inputs which were highlighted in yellow and two outputs which were highlighted in blue.

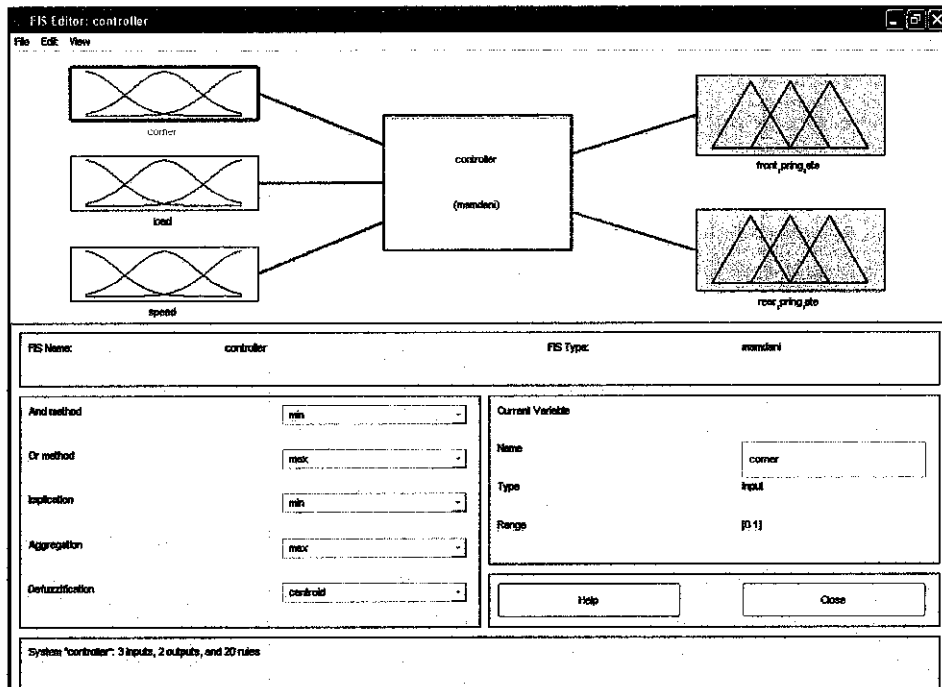


Figure 6: MATLAB FIS Editor Main Screen

3.1.2 The Membership Function Editor

In this editor, the author defined the shapes of all membership function related to each variable and the number of the membership functions required for each input or output.

3.1.3 The Rule Editor

The author edited the list of if – then rules that describe the characteristic of the system in this editor. Figure 7 shows that the author had included the rules that as shown in Table 6 from Chapter 2.

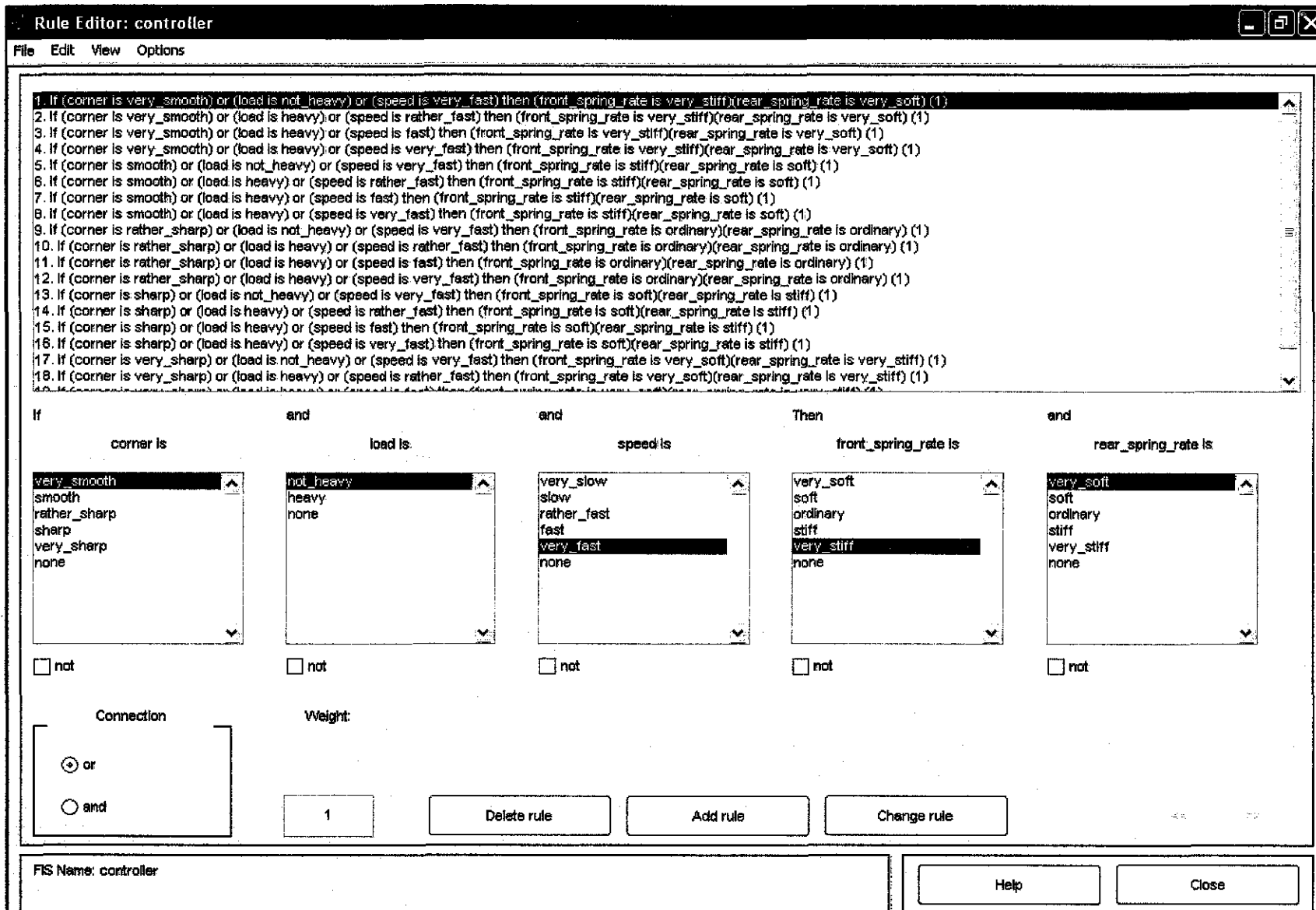


Figure 7: MATLAB Rule Editor Main Screen

3.2 Architecture, Design Specification & Familiarize with VHDL Coding

Figure 8 shows the overall structure of the fuzzy logic controller that the author had identified [15].

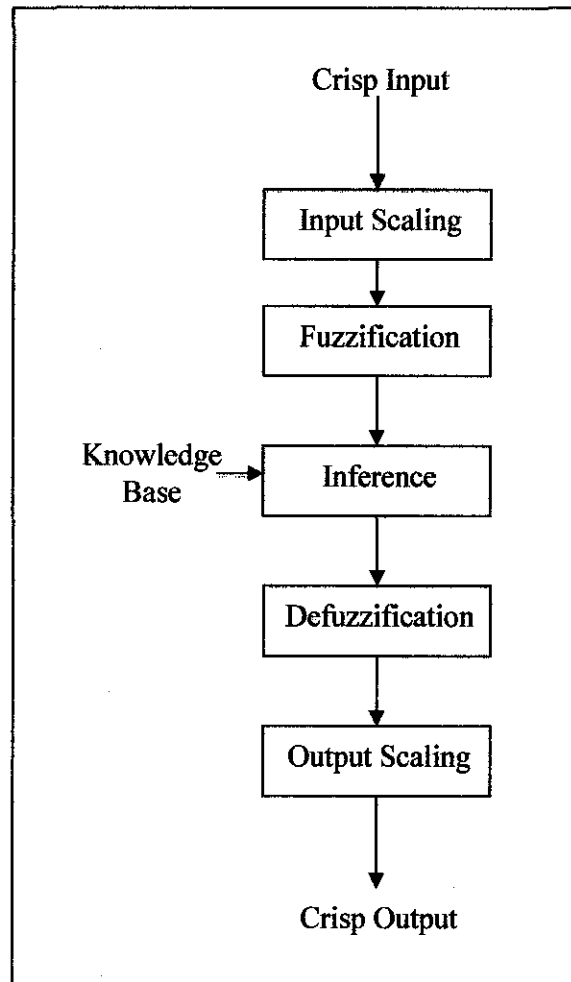


Figure 8: The Block Diagram for a Fuzzy Logic Controller [15]

Each of these main blocks consists of multiple sub – blocks [15]. Then, the author decided to begin developing the algorithms with fuzzification.

3.3 Development VHDL Coding for Individual Sub – Block

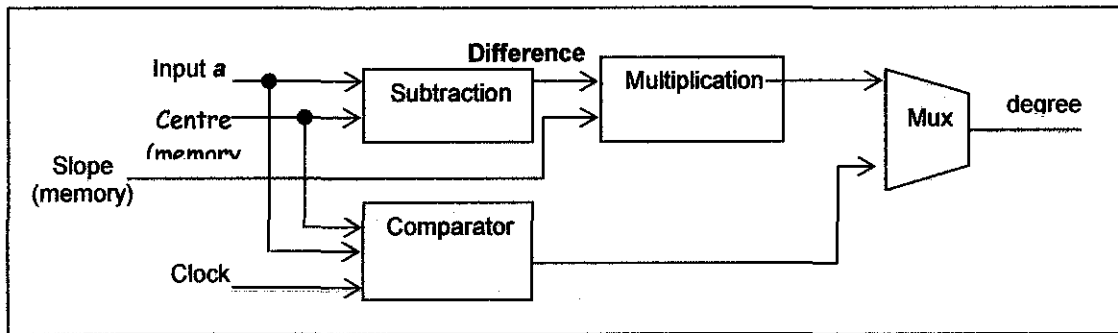


Figure 9: The Block Diagram for Fuzzification [15]

The author's main focus was the fuzzification block. The author started by writing the coding for the individual sub – blocks. In this project, the concerned sub – blocks were subtraction, multiplication and comparator as shown in Figure 9. VHDL codes were entered using the Aldec Active HDL 5.1. Each of the VHDL coding for each sub – block were stored in separate source files. Different source files were then saved in a common design or workspace. Figure 10 shows that the individual source codes were kept in the common workspace. This would allow author to recall these codes easily. These codes were then compiled to check for syntax errors.

3.4 Integrating All Sub – Blocks

The completed individual sub – blocks were then combined to become one main block. The written code for the main block was also compiled for syntax errors.

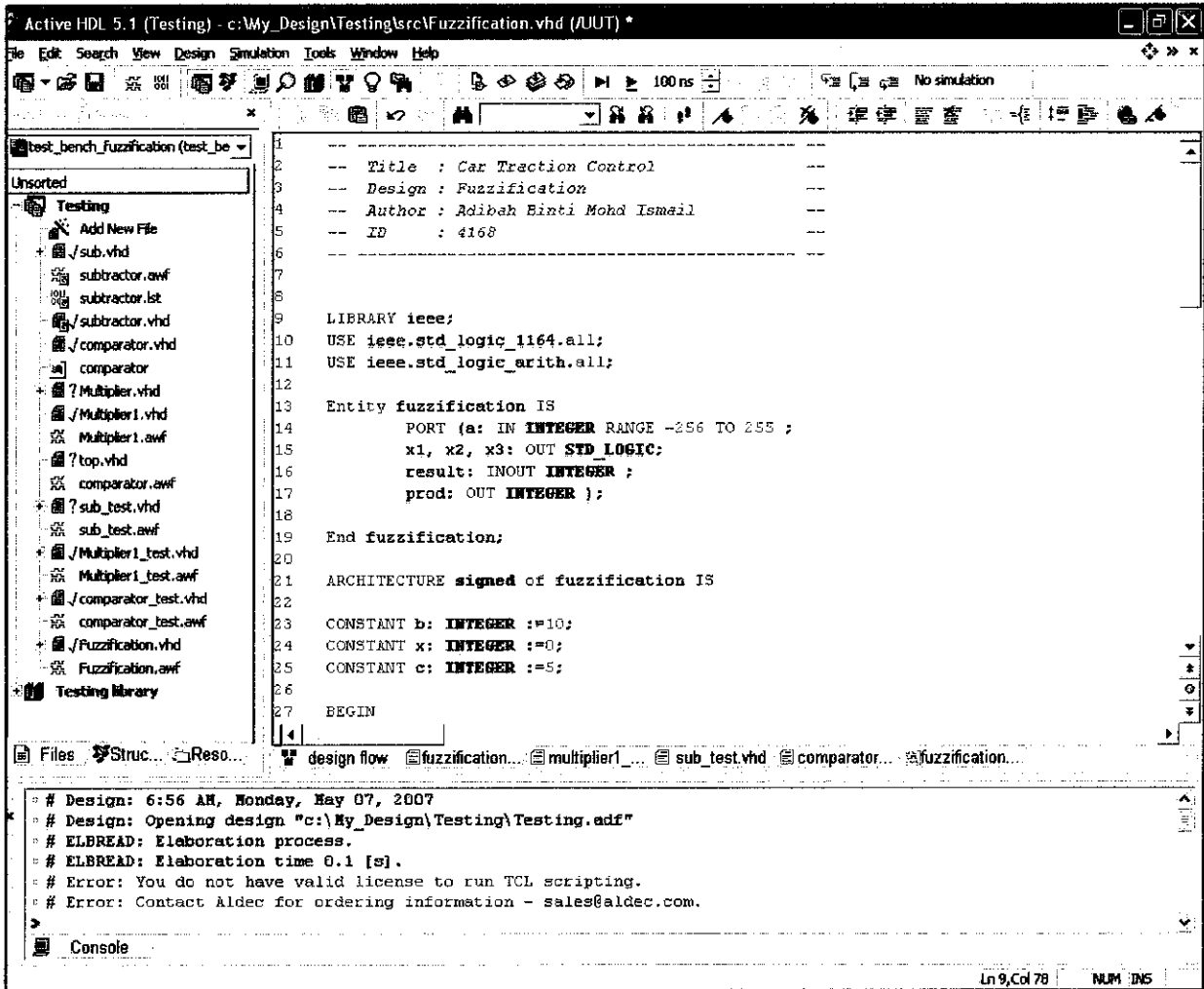


Figure 10: Aldec Active HDL 5.1 Main Screen

3.5 Simulation & Design Verification

The author used Aldec Active HDL 5.1 for simulation and design verification. A test bench is used for testing VHDL designs in VHDL environment [16]. The design is checked by applying signals or stimuli and monitoring the outcomes. The author observed the response through output waveforms.

CHAPTER 4

RESULTS & DISCUSSION

This chapter will discuss results from two main parts. The first part would be the results from MATLAB. Then, this chapter will cover the outcome from the construction of the VHDL coding. The main idea of this control system is to get the system back to neutral or stable during cornering. All inputs and outputs were governed by Table 6.

4.1 Simulation Using MATLAB

The simulation of the fuzzy logic controller had been done using MATLAB. After defining each of the inputs together with its membership function, the following graphs were obtained. Figure 11 to Figure 15 show the membership functions for inputs and outputs correspondingly to Table 1 to Table 5.

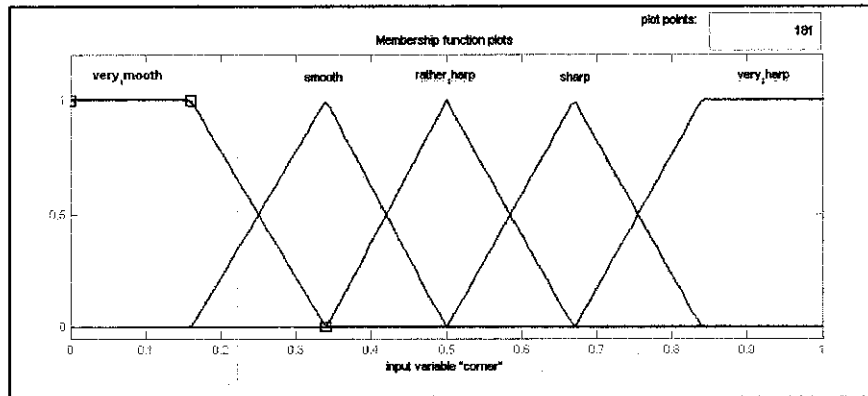


Figure 11: Membership Function for Corner Input

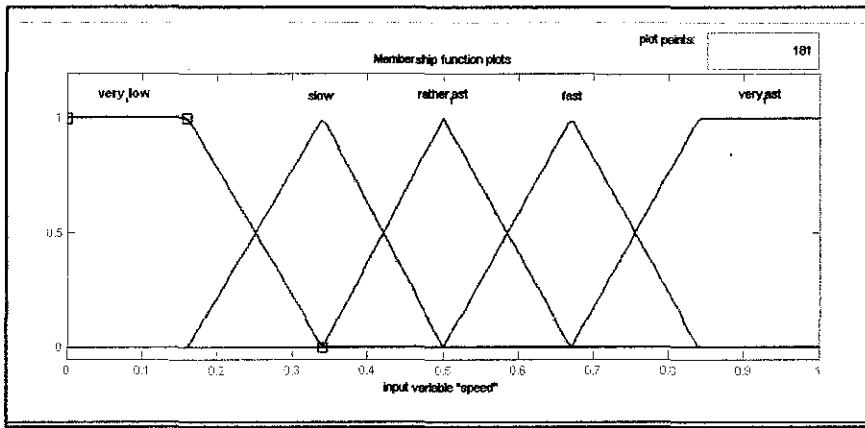


Figure 12: Membership Function for Load Input

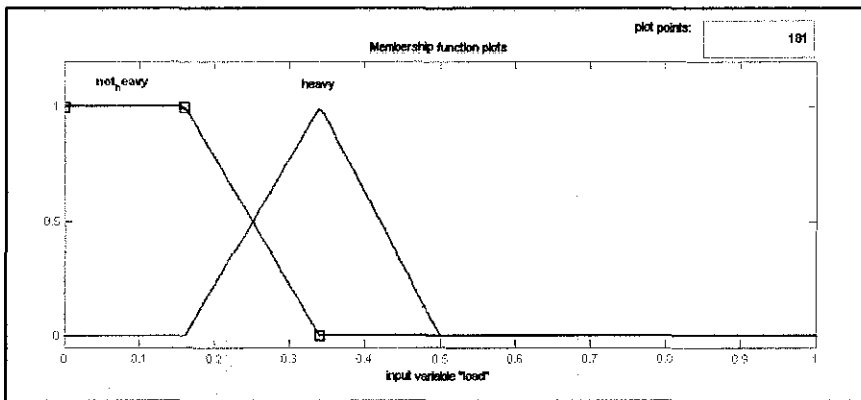


Figure 13: Membership Function for Speed Input

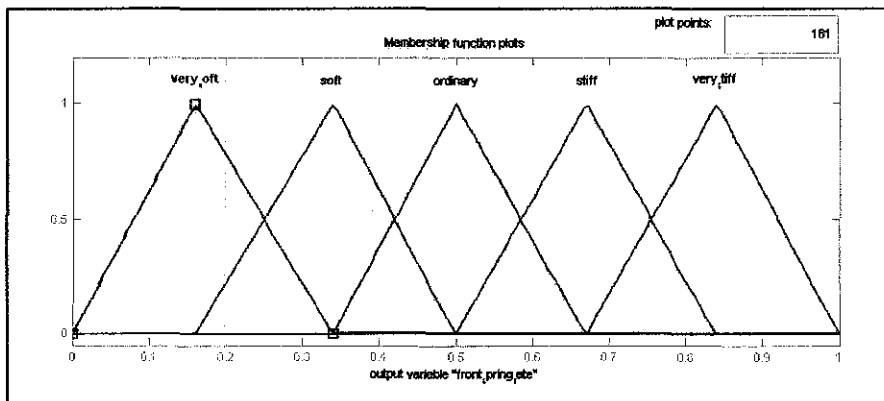


Figure 14: Membership Function for Front Spring Rate Output

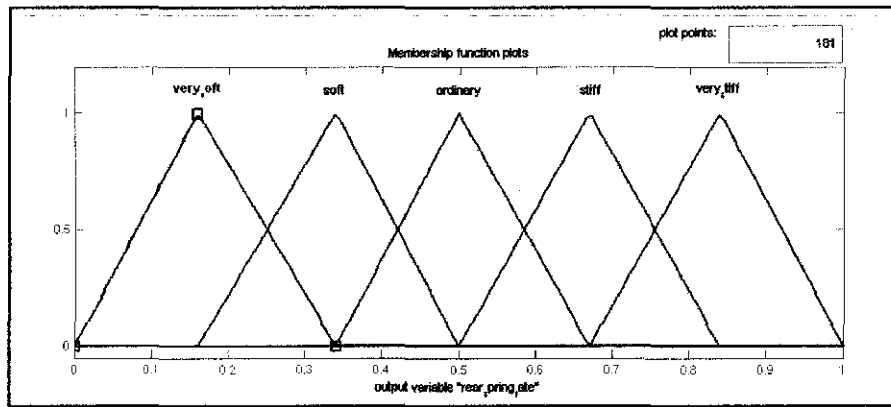


Figure 15: Membership Function for Rear Spring Rate Output

The if – then rules from Table 6 were then applied to the MATLAB by defining these rules using the Rule Editor as shown in Figure 7. Then the outcome was observed from the Rule Viewer which displayed the overall of the fuzzy inference process. The results were based on the defined inference rules. The Rule Viewer allows users to observe how each membership function is affected accordingly to each inference rule when the inputs are applied.

Several set of inputs were given to verify the fuzzy logic controller. Figure 16 shows the results for the first set of inputs. Firstly the corner input was maintained at the very smooth condition (corner = -4), then the corner was made heavy (load= 1.49) and the speed was given rather fast (speed = 0). Thus, according the rules in Table 6, the rule no. 6 was in active. This could be observed from all three inputs were highlighted in yellow in the 6th row. From Figure 16, the front spring rate indicates stiff and the rear spring rate indicates soft.

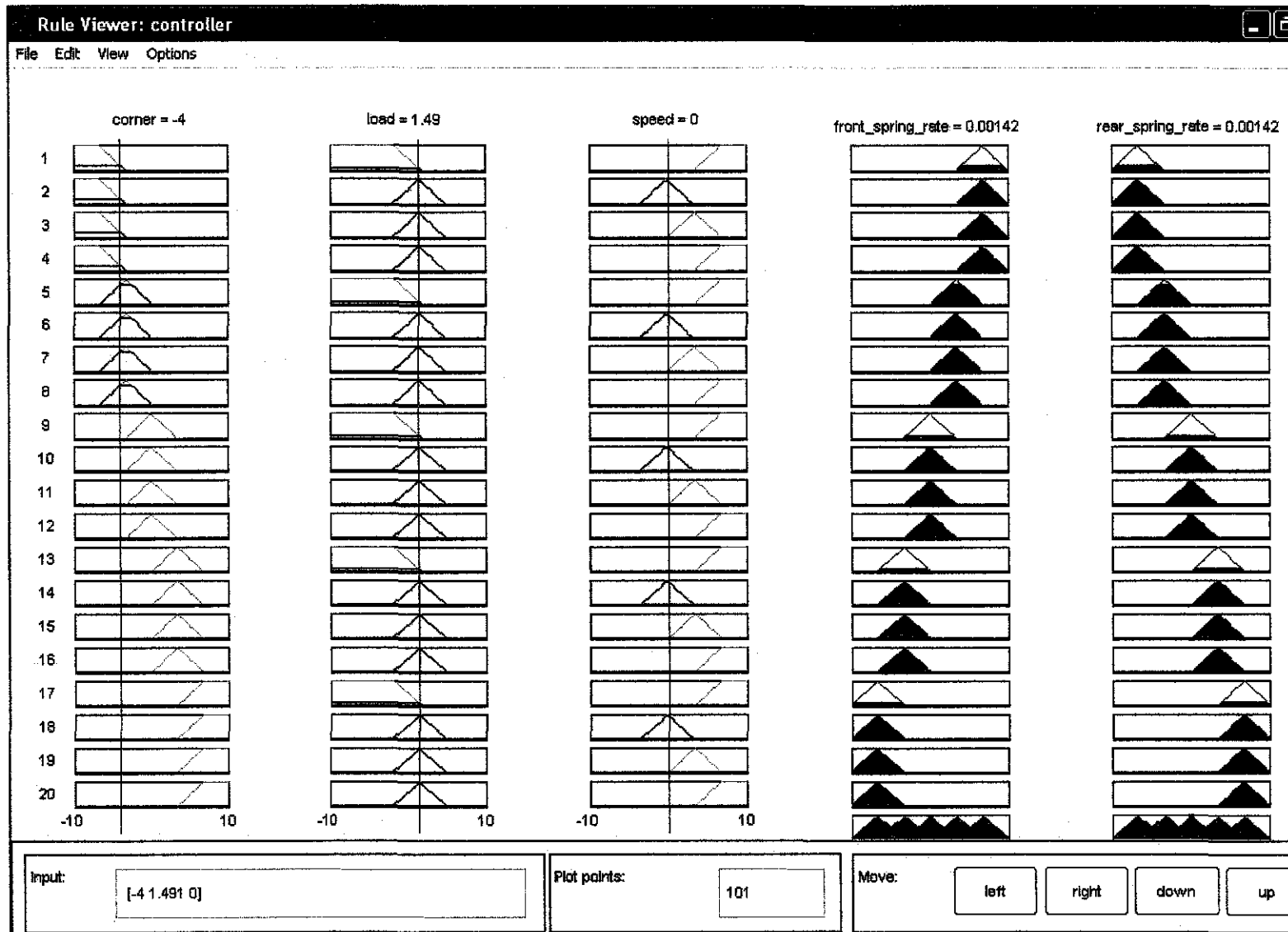


Figure 16: Rule Viewer When Corner = -4, Load = 1.49 and Speed = 0

For the set of inputs, corner input was sharp (corner = 3.5), load is heavy (load = 1.49) and speed was rather fast (speed = 0). Referring to Table 6, the rule no. 14 was in active (all inputs are highlighted). The result of the front spring rate show soft and the rear spring rate shows stiff as displayed in Figure 17.

The last set of inputs shows that the corner input was rather sharp (corner = 0), load was heavy (load=1.49) and speed is rather fast (speed = 0). Figure 18 shows that the front spring rate is soft and the rear spring rate is stiff. This verified from the if – then rules from Table 6.

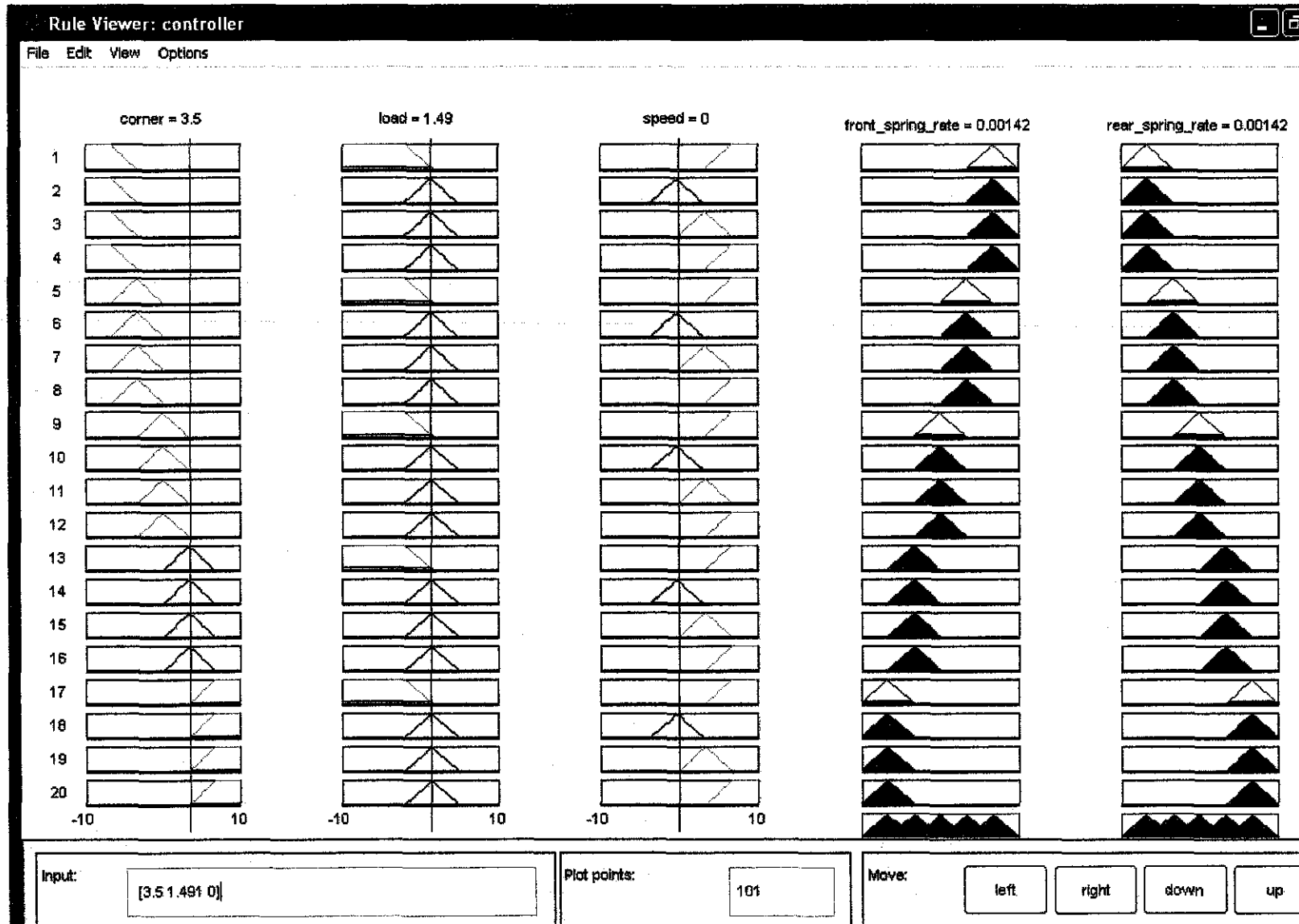


Figure 17: Rule Viewer When Corner = 3.5, Load = 1.49 and Speed = 0

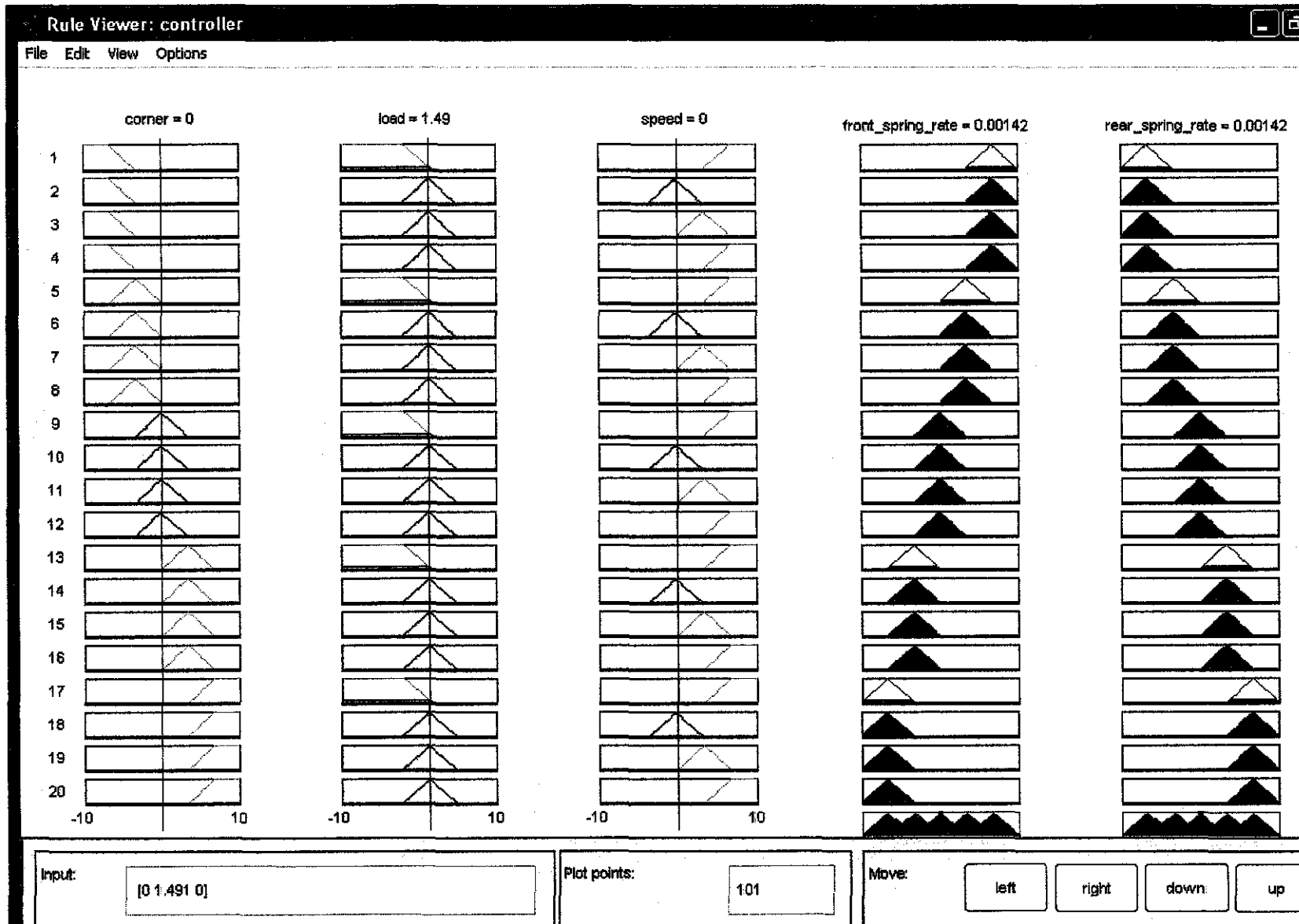


Figure 18: Rule Viewer When Corner = 0, Load = 1.49 and Speed = 0

4.2 The Development of VHDL Coding

The fuzzy logic controller consists of the main blocks which are fuzzification, inference and defuzzification. Due to the complexity of the project, the author would only focus on the fuzzification block. From Figure 9, the fuzzification block is made up of subtraction, comparator and multiplication sub blocks.

In Figure 9, the input that was given to the fuzzification block would be diverting to two different routes. One would enter the comparator. The input would also enter the subtractor. The purpose of the comparator is to determine whether the input given is negative, positive or zero. The subtractor would subtract the input from the centre point of affected fuzzy set [15]. The results from the subtractor would be multiplied by the positive slope [15]. Finally, the output from both the multiplication sub – block and the comparator would determine the output of the overall fuzzification block.

The project was implemented by using VHDL. VHDL can be used to describe the behavior or structure of the digital system. By using the Aldec Active – HDL 5.1, the outcomes from the simulation and design verification of the individual sub – blocks were observed through the output waveforms.

Figure 19 shows the output waveform of the simulation for subtractor sub - block. Since it is individual, thus the input (indicated by ‘a’) was subtracted from a constant. The constant was set to 10. The simulation lasted for 200 ns. The input was increased by 2 for every 10 ns. These specifications were defined in the source code. The ‘result’ indicates the results of the subtraction between ‘a’ and the constant.

Name	Value	Stimulator	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	200 ns
a	40	Binary C...	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	
result	30		-10	-8	-6	-4	-2	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	

Figure 19: The Simulation Results for Subtractor

Figure 20 shows the output waveform of the design simulation for multiplication sub - block. The sub – block is also individual designed. The block had two inputs. The first input (indicated by ‘a’) was multiplied with the second input (indicated by ‘b’).

In this simulation, the duration was also 200 ns. The input 'a' was increased by 2 for every 10 ns. Whereas the 'b' input was increased by 6 for every 20 ns. These specifications were defined in the source code. The 'prod' indicates the results of the multiplication between 'a' and 'b'.

Name	Value	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38
a	40	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38
b	60	0		6		12		18		24		30		36		42		48		54	
prod	2400	0		24	36	96	120	216	252	384	432	600	660	864	936	1176	1260	1536	1632	1944	2052

Figure 20: The Simulation Results for Multiplication

Figure 21 shows the output waveform of the design simulation for comparator sub-block. The sub-block was indeed individual designed. The block had also two inputs. The first input (indicated by 'a') was compared to the second input (indicated by 'b'). The duration for the simulation was 200 ns. The input 'a' was increased by 2 for every 10 ns. Whereas the 'b' input was decreased by 1 for every 10 ns. These specifications are defined in the source code. The 'x1', 'x2' and 'x3' indicate the results of the comparison between 'a' and 'b'. 'x1' is equal 1 when 'a' is larger than 'b'. 'x2' is equal to 1 when 'a' is the same as 'b'. Finally, 'x3' is equal to one when 'a' is smaller than 'b'.

Name	Value	0	2	4	6	8	10	12	14	16	18	20	22									
a	22	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
b	-10	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	
x1	1																					
x2	0																					
x3	0																					

Figure 21: The Simulation Results for Comparator

The VHDL coding for the individual sub-block is included in Appendix A. The next step is to proceed with the integration of all three sub blocks. Figure 22 shows the waveform for design verification for the integrated sub-blocks.

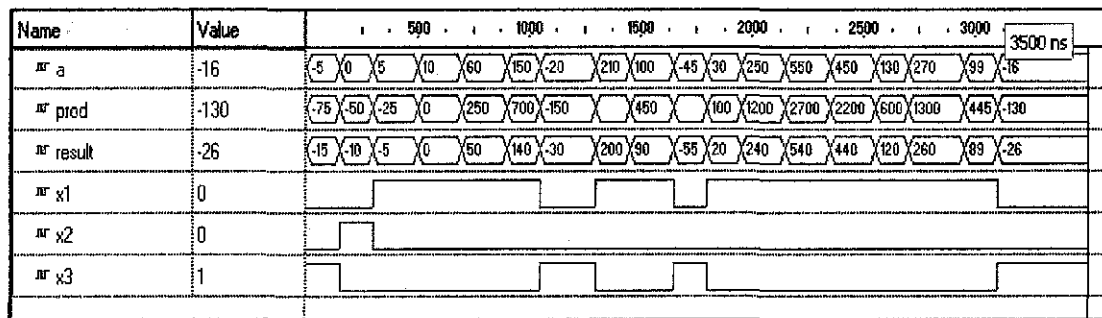


Figure 22: The Output Waveform of Verification for the Integrated Sub – Block

For Figure 22, ‘a’ was given the input according to defined input in the source code for the test bench of the integrated sub – blocks. ‘result’ was the results of the subtraction operation. ‘a’ was subtracted from a constant. The constant was set as 10. Then the difference is multiplied with another constant was set to 15. This is defined in the test bench source code. The author assumed the value of the positive slop to be 15. The assumption was made to reduce the complexity. ‘x1’, ‘x2’ and ‘x3’ were the results for the comparator. This time, the duration of the verification was 3500 ns. The purpose was to observe the outcomes for various inputs given. In this project, the author concentrated in constructing the structure of the fuzzification. Thus to reduce the complexity, the author used integer for number representation.

CHAPTER 5

CONCLUSION & RECOMMENDATIONS

This chapter reviews and concludes the project while highlighting some of the difficulties encountered and how to overcome them. Some recommendations are included to suggest for further improvement and for future progress

5.1 Conclusion

The main component of VHDL coding for the main structure of fuzzification is successfully completed. From the successful simulation results obtained as well as the output waveform from the design verification in Chapter 4, the objective set initially, had already been achieved.

The project had been carried out for two semesters. The first semester was mainly focus on the preliminary research work. The research work deals with a gist of existing car traction control, fuzzy logic and fuzzy logic controller. The second semester work mainly more on developing VHDL coding for specifically fuzzification which to be used eventually for fuzzy logic based car traction control system.

The main challenges faced by the author were familiarizing the concept of fuzzy logic and learning new language VHDL. The course is not included in the syllabus in the university. The author used examples of application of fuzzy logic controller so that the author could understand how fuzzy logic works for car traction control system. Besides that, VHDL is totally different from the programming courses such as C and C++ which were taught in the university. Therefore, the author solely depended on self – study and self – exploration.

Besides that, the author had some difficulties in developing the source code for decimal number representation. Therefore, the author used integer in all source code to reduce the complexity.

5.2 Recommendations for Future Works

Even though the VHDL coding for the main structure is completed, the coding could be improved by modifying the completed coding, such that it can cater for decimal numbers which gives more meaningful for the implementation of the control system.

The development of VHDL coding for fuzzification can also be further improved by developing other main blocks which also included in the fuzzy logic controller. The other two main blocks are inference and defuzzification. Each of these main blocks consists of several other sub blocks.

Eventually, the three completed main blocks can be integrated to form a complete form of fuzzy logic controller algorithm. The algorithm can be used to implement on FPGA in future development.

REFERENCES

- [1] "Oxford Advanced Learner's Dictionary", Oxford University Press
- [2] Scott Memmer, "Safety Tips – Traction Control",
<http://www.edmunds.com/ownership/safety/articles/46352/article.html>
- [3] "Traction Control", Wikipedia - the Free Encyclopedia,
http://en.wikipedia.org/wiki/Traction_control
- [4] Adlan Iqman Bin Suhaimi, "Simulation of Control System for Vehicle Traction Control Using Fuzzy Logic Approach", Thesis, 2002
- [5] Joseph Bih, "Paradigm Shift – An Introduction to Fuzzy Logic", IEEE Potentials, Volume 25, Issue 1, Jan.-Feb. 2006
- [6] "What is Fuzzy Logic? - Getting Started (Fuzzy Logic Toolbox)", Online tutorial,
<http://www.mathworks.com/access/helpdesk/help/toolbox/fuzzy/index.html?access/helpdesk/help/toolbox/fuzzy/fp754.html>
- [7] Daijin Kim, "An Implementation of Fuzzy Logic Controller on the Reconfigurable FPGA System", IEEE Transactions on Volume 47, Issue 3, June 2000
- [8] Yasuhiko Dote, "Introduction to Fuzzy Logic", Industrial Electronics, Control, and Instrumentation, 1995, Proceedings of the 1995 IEEE IECON 21st International Conference on Volume 1, 6-10 Nov. 1995
- [9] Sudhakar Yalamanchili, "Introductory VHDL From Simulation to Synthesis", Prentice Hall, 2001
- [10] "ASR (Automatic Slip Control)", Bentley Publishers Automotive Reference,
<http://www.bentleypublishers.com/gallery.htm?code=GMOB&galleryId=618>
- [11] "Fuzzy Logic: An Overview of the Latest Control Methodology Application Report", Texas Instrument, <http://focus.ti.com/lit/an/spra028/spra028.pdf>
- [12] "Fuzzy Logic Toolbox for Use with MATLAB User's Guide", The MathWorks Inc, 2006
- [13] George J. Klir, "Fuzzy Logic – Unearthing its meaning and significance", Potentials, IEEE Volume 14, Issue 4, Oct-Nov 1995
- [14] Masao Mukaidono, "Fuzzy Logic For Beginners", World Scientific Publishing Co. Pte. Ltd, 2001

- [15] Sameep Singh and Kuldip S. Rattan, "Implementation of a Fuzzy Logic Controller on an FPGA using VHDL", Fuzzy Information Processing Society, 2003
- [16] J. Mirkowski, M. Kapustka, Z. Skowronski, A. Biniszkiewicz, "EVITA Interactive VHDL Tutorial REV.2.0 Active – VHDL Series Book #2", ALDEC

APPENDICES

APPENDIX A
VHDL SOURCE CODES

```
-- ----- --  
-- VHDL Source Code For Subtractor --  
-- Author: Adibah Mohd Ismail --  
-- ID: 4168 --  
-- ----- --
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.ALL;  
USE ieee.std_logic_arith.All;
```

```
ENTITY sub_entity IS
```

```
    PORT
```

```
    (
```

```
        a:    IN INTEGER RANGE -256 TO 255;  
        result: OUT INTEGER RANGE -256 to 255
```

```
    );
```

```
END sub_entity;
```

```
ARCHITECTURE sub_entity OF sub_entity IS
```

```
    CONSTANT b: INTEGER :=10;
```

```
BEGIN
```

```
    result <= CONV_INTEGER(a-b);
```

```
END sub_entity;
```



```
-- -----  
-- VHDL Source Code For Multiplication --  
-- Author      : Adibah Mohd Ismail  --  
-- ID         : 4168                 --  
-- -----
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;
```

```
ENTITY multiplier1 IS  
    PORT (a,b: IN INTEGER RANGE -256 TO 255;  
          prod: OUT INTEGER );  
END multiplier1;
```

```
ARCHITECTURE behavior OF multiplier1 IS  
BEGIN  
    prod <= a * b;  
END behavior;
```

```
-----  
-- Source Code For Comparator --  
-- Author: Adibah Mohd Ismail --  
-- ID : 4168 --  
-----
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;
```

```
Entity comparator IS  
  GENERIC (n: INTEGER);  
  PORT (a,b: IN INTEGER ;  
        x1, x2, x3: OUT STD_LOGIC);  
End comparator;
```

```
ARCHITECTURE signed of comparator IS
```

```
BEGIN
```

```
  x1 <= '1' WHEN a>b ELSE '0';  
  x2 <= '1' WHEN a=b ELSE '0';  
  x3 <= '1' WHEN a<b ELSE '0';
```

```
END SIGNED;
```

```
----- --
-- Source Code For Integrated Subtype - Blocks (Fuzzification) --
-- Author: Adibah Mohd Ismail --
-- ID : 4168 --
----- --
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
```

```
Entity fuzzification IS
```

```
    PORT (a: IN INTEGER RANGE -256 TO 255 ;
          x1, x2, x3: OUT STD_LOGIC;
          result: INOUT INTEGER ;
          prod: OUT INTEGER );
```

```
End fuzzification;
```

```
ARCHITECTURE signed of fuzzification IS
```

```
    CONSTANT b: INTEGER :=10;
    CONSTANT x: INTEGER :=0;
    CONSTANT c: INTEGER :=5;
```

```
BEGIN
```

```
    x1 <= '1' WHEN a>x ELSE '0';
    x2 <= '1' WHEN a=x ELSE '0';
    x3 <= '1' WHEN a<x ELSE '0';
```

```
    result <= CONV_INTEGER(a-b);
    prod <= c * result;
```

```
END signed;
```

APPENDIX B
TEST BENCH SOURCE CODES

```
-----  
-- Test Bench For Subtractor      --  
-- Author: Adibah Mohd Ismail    --  
-- ID: 4168                       --  
-----
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.ALL;  
USE ieee.std_logic_arith.All;
```

```
ENTITY sub_entity IS
```

```
    PORT  
    (  
        a:    IN INTEGER RANGE -256 TO 255;  
        result: OUT INTEGER RANGE -256 to 255  
    );
```

```
END sub_entity;
```

```
ARCHITECTURE sub_entity OF sub_entity IS
```

```
    CONSTANT b: INTEGER :=10;
```

```
BEGIN
```

```
    result <= CONV_INTEGER(a-b);
```

```
END sub_entity;
```

```
entity test_bench_sub is  
end test_bench_sub;
```

```
architecture test_bench_sub of test_bench_sub is  
    component sub_entity  
    port (  
        a:IN INTEGER RANGE -256 TO 255;  
        result: OUT INTEGER RANGE -256 to 255);  
    end component sub_entity;
```

```
    signal a: INTEGER range -30 TO 255 ;  
    signal result: INTEGER range -256 TO 255;  
begin
```

```
    UUT: sub_entity  
    port map(a=>a ,result=>result );
```

```
process  
begin
```

```
a <= 5 ; wait for 50 ns;  
a <= 0 ; wait for 50 ns;  
a <=5;wait for 50 ns;  
a <= 10;wait for 50 ns;  
a <= 50;wait for 50 ns;  
a <= 100;wait for 50 ns;
```

```
wait;
```

```
end process;  
end test_bench_sub;
```

```
-----  
-- Test Bench For Multiplication    --  
-- Author: Adibah Mohd Ismail      --  
-- ID: 4168                         --  
-----
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;
```

```
ENTITY multiplier1 IS  
    PORT (a,b: IN integer RANGE -256 TO 255;  
          prod: OUT integer);  
END multiplier1;
```

```
ARCHITECTURE behavior OF multiplier1 IS  
BEGIN  
    prod <= a * b;  
END behavior;
```

```
entity test_bench_mult is  
end test_bench_mult;
```

```
architecture test_bench_mult of test_bench_mult is
```

```
component multiplier1
```

```
    port (  
        a,b:IN INTEGER RANGE -256 TO 255;  
        prod: OUT INTEGER );
```

```
end component multiplier1;
```

```
signal a: INTEGER ;  
signal b: INTEGER ;  
signal prod: INTEGER;
```

```
begin
```

```
UUT: multiplier1  
port map(a=>a , b=>b, prod =>prod );
```

```
STIM1:process
```

```
begin
```

```
    a <= -5 ; wait for 50 ns;  
    a <= 0 ; wait for 50 ns;  
    a <=5;wait for 50 ns;  
    a <= 10;wait for 50 ns;
```

```

a <= 60;wait for 50 ns;
a <= 150;wait for 100 ns;
a <= 170;wait for 200 ns;
a <= 210;wait for 150 ns;
a <= 100;wait for 200 ns;

wait;

end process;

STIM2:process

begin
b <= -10 ; wait for 50 ns;
b <= 8 ; wait for 50 ns;
b <=0;wait for 50 ns;
b <= -100;wait for 50 ns;
b <= 60;wait for 50 ns;
b <= 2;wait for 50 ns;
b <= -20;wait for 100 ns;
b <= 190;wait for 200 ns;
b <= 500;wait for 150 ns;
b <= 760;wait for 200 ns;

wait;
end process;

end test_bench_mult;

```



```
-----  
-- Test Bench For Comparator --  
-- Author: Adibah Mohd Ismail --  
-- ID : 4168 --  
-----
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;
```

```
Entity comparator IS
```

```
    PORT (a,b: IN INTEGER ;  
          x1, x2, x3: OUT STD_LOGIC);
```

```
End comparator;
```

```
ARCHITECTURE signed of comparator IS
```

```
BEGIN
```

```
    x1 <= '1' WHEN a>b ELSE '0';  
    x2 <= '1' WHEN a=b ELSE '0';  
    x3 <= '1' WHEN a<b ELSE '0';
```

```
END SIGNED;
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;
```

```
entity test_bench_comp is  
end test_bench_comp;
```

```
architecture test_bench_comp of test_bench_comp is
```

```
component comparator
```

```
    port (  
        a,b:IN INTEGER RANGE -256 TO 255;  
        x1, x2, x3: OUT STD_LOGIC);
```

```
end component comparator;
```

```
signal a: INTEGER ;  
signal b: INTEGER ;  
signal x1,x2,x3: STD_LOGIC;
```

```
begin
```

```
UUT: comparator
port map(a=>a , b=>b, x1=>x1, x2=>x2, x3=>x3 );
```

```
STIM1:process
```

```
begin
```

```
    a <= -5 ; wait for 100 ns;
    a <= 8 ; wait for 100 ns;
    a <=5;wait for 100 ns;
    a <= 10;wait for 100 ns;
    a <= 280;wait for 100 ns;
    a <= 150;wait for 100 ns;
    a <= 170;wait for 200 ns;
    a <= 210;wait for 150 ns;
    a <= 100;wait for 200 ns;
```

```
    wait;
```

```
end process;
```

```
STIM2:process
```

```
begin
```

```
    b <= -10 ; wait for 100 ns;
    b <= 8 ; wait for 100 ns;
    b <=0;wait for 100 ns;
    b <= -100;wait for 100ns;
    b <= 60;wait for 100 ns;
    b <= 2;wait for 100 ns;
    b <= 150;wait for 100 ns;
    b <= 190;wait for 200 ns;
    b <= 450;wait for 150 ns;
    b <= 760;wait for 200 ns;
```

```
    wait;
```

```
end process;
```

```
end test_bench_comp
```

```
-----  
-- Test Bench For Integrated Sub – Blocks (Fuzzification) --  
-- Author: Adibah Mohd Ismail --  
-- ID : 4168 --  
-----
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;
```

```
Entity fuzzification IS
```

```
    PORT (a: IN INTEGER RANGE -256 TO 255 ;  
          x1, x2, x3: OUT STD_LOGIC;  
          result: INOUT INTEGER ;  
          prod: OUT INTEGER );
```

```
End fuzzification;
```

```
ARCHITECTURE signed of fuzzification IS
```

```
    CONSTANT b: INTEGER :=10;  
    CONSTANT x: INTEGER :=0;  
    CONSTANT c: INTEGER :=5;
```

```
BEGIN
```

```
    x1 <= '1' WHEN a>x ELSE '0';  
    x2 <= '1' WHEN a=x ELSE '0';  
    x3 <= '1' WHEN a<x ELSE '0';
```

```
    result <= CONV_INTEGER(a-b);  
    prod <= c * result;
```

```
END signed;
```

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_arith.all;
```

```
entity test_bench_fuzzification is  
end test_bench_fuzzification;
```

```
architecture test_bench_fuzzification of test_bench_fuzzification is
```

```
component fuzzification
```

```
    port (  
        a: IN INTEGER ;
```

```

        prod: OUT INTEGER ;
        result: INOUT INTEGER ;
        x1, x2, x3: OUT STD_LOGIC);

end component fuzzification;

signal a: INTEGER RANGE -50 TO 600 ;
signal prod: INTEGER;
signal result: INTEGER ;
signal x1,x2,x3: STD_LOGIC;

begin

UUT: fuzzification
port map(a=>a , prod=>prod, result=>result, x1=>x1, x2=>x2, x3=>x3 );

STIM1:process

begin

    a <= -5 ; wait for 150 ns;
    a <= 0 ; wait for 150 ns;
    a <=5;wait for 200 ns;
    a <= 10;wait for 200 ns;
    a <= 60;wait for 200 ns;
    a <= 150;wait for 150 ns;
    a <= -20;wait for 250 ns;
    a <= 210;wait for 150 ns;
    a <= 100;wait for 200 ns;
    a <= -45 ; wait for 150 ns;
    a <= 30 ; wait for 150 ns;
    a <=250;wait for 200 ns;
    a <= 550;wait for 200 ns;
    a <= 450;wait for 200 ns;
    a <= 130;wait for 150 ns;
    a <= 270;wait for 250 ns;
    a <= 99;wait for 150 ns;
    a <= -16;wait for 200 ns;

    wait;

end process;

end test_bench_fuzzification;

```