FPGA Applications in Hybrid Energy Storage System and Field-oriented Motor Control

by

Fanning Jin

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering (Electrical Engineering) in the University of Michigan-Dearborn 2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis demonstrates the utilization of FPGA chip in hybrid energy storage system (HESS) and motor control, which contains four sections. 1. In order to compensate for such mismatches, a HESS composed of battery and ultracapacitor (UC) is presented in this thesis. To optimize the power allocation in the HESS, a global-level fuzzy logic-based power management strategy is proposed. It has the following objectives: 1) manage the power distribution between the battery and the UC to achieve power balance without over charging/discharging the two sources; 2) make full use of the nature of the two sources to smooth out power fluctuation and extend the lifespan of batteries. Case studies are provided under various scenarios. 2. This thesis not only considered the PV-involved system, it also considered hybrid electric vehicles (HEV) system issues, which means we also developed two fuzzy logic power management strategies for HEV-involved system. The goal is to investigate the effect of two fuzzy logic strategy when considering battery degradation. In that case, the battery degradation model is presented and compared. The simulation results verified that one of the proposed strategy can greatly reduce the battery degradation. 3. Due to the fluctuation and instability of photovoltaic (PV) output power, mismatches between the power supply and the power demand may occur in a stand-alone PV system. In order to test the potential of FPGA chip, both fuzzy algorithm and field-oriented control (FOC) algorithm are implemented in FPGA. The digital design of each fuzzy logic stage is presented. The results validate the effectiveness of the proposed fuzzy logic-based power management strategy. 4. For FOC algorithm implementation, the goal is to utilize the rapidity of the FPGA chip to generate high-frequency PWM signals for power converter board. Both simulation and hardware results are presented, which shows the advancement of FPGA chip.

Chapter 1 Introduction

1.1 Review of HESS system in EV application*

Hybrid Electric Vehicles (HEVs) and Plug-in Hybrid Electric Vehicles (PHEVs) are receiving more attention than ever before because of their low emissions of greenhouse gases and high fuel economy [1]-[2]. We can expect a major energy demands shift from fossil fuels to electricity for road transportation in the next 20 years [3]. The energy storage system (ESS) is an indispensable component for the adoption of HEV and PHEV technology. For our study, we choose the battery and ultracapacitor combined hybrid energy storage system (HESS) given its varieties of benefits over single-source ESS [4]. The advantages of battery/ultracapacitor HESS over the traditional battery-based ESS can be summarized as below.

1. The capacity requirement for batteries or the number of batteries can be lowered becau se it does not need to be sized to satisfy the peak value of the power profile, thus the system cost is reduced;

2. The efficiency of HESS is higher than single-source ESS. Based on the RTDS simulation results presented in [5], the Joule loss of HESS is only 65% of those of battery-only ESS in a time period of 200 seconds, the size of the HESS will be reduced due to less batteries are needed.

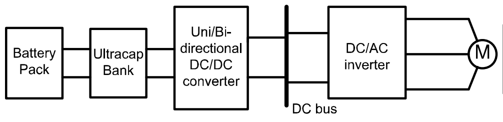
* These sections or chapters are part of my published work with title "A fuzzy logic based power management strategy for hybrid energy storage system in hybrid electric vehicles considering battery degradation"

3. The life of the batteries will be extended. It is the nature that the high-frequency high-magnitude pulse power will shorten the cycle life of batteries. With ultracapacitors providing and absorbing the pulsed power for vehicle acceleration and breaking, the battery power can be viewed as "filtered" since di/dt of charging or discharging process will be limited and the SOC will be controlled. This will, as a result, extend the life span of the HESS.

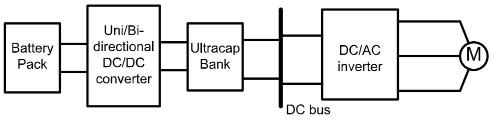
1.2 Review of HESS configurations in EV application*

The system configuration of HESS can be grouped into three major categories: cascade, parallel and multi-input configurations. For cascade configuration, it can be further divided into one-converter cascade and two-converter cascade structures.

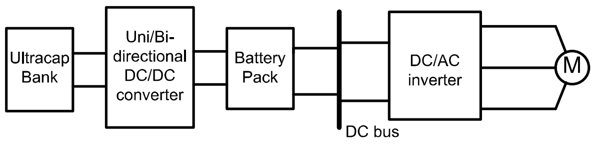
There are there variations for one-converter cascaded configuration as shown in Fig. 1(a)-(e). For the Battery-UC-DC/DC structure (Fig. 1(a)), although the DC bus voltage can be regulated by the DC/DC converter, the battery voltage has to follow the UC voltage due to the lack of interface between those two. The battery current also has to charge the UC before provide power to the motor [6]. Instead, if we put the DC/DC converter in between the battery pack and UCwith either the battery pack or the UC directly interfacing the DC bus (Fig. 1(b) and Fig. 1(c)), the number of cells is large in order to match the high DC bus voltage. Thus, cell balancing issue may occur and additional balancing circuit or BMS is needed. In addition, the UC cannot be fully utilized for the structure shown in Fig. 1(b). The three structures mentioned above can be improved by adding another converter as shown in Fig. 1(d) and (e). The trade-off is that two full-size converters are needed and the control is more complicated.



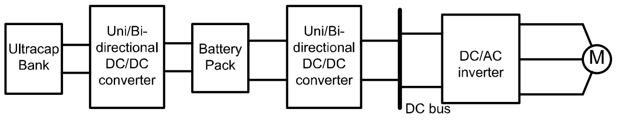
(a)



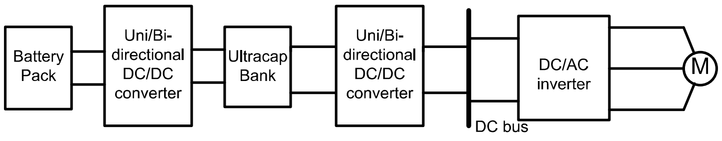
(b)



(c)


Ultracap Bank
Uni/Bi-directional DC/DC converter
Battery Pack
Uni/Bi-directional DC/DC converter
DC bus
DC/AC inverter
M

(d)


Battery Pack
Uni/Bi-directional DC/DC converter
Ultracap Bank
Uni/Bi-directional DC/DC converter
DC bus
DC/AC inverter
M

(e)

Figure 1. HESS configuration in EV application

For the parallel configuration, both the battery pack and the ultracapacitor are equipped with two converters to interface with the DC bus. Although two full-size converters are needed, the power flow of battery and ultracapacitor can be controlled separately, allowing flexibility for power management. The system can also be easily scaled up. Thus, we placed our scope on the parallel structure as shown in Fig. 2.

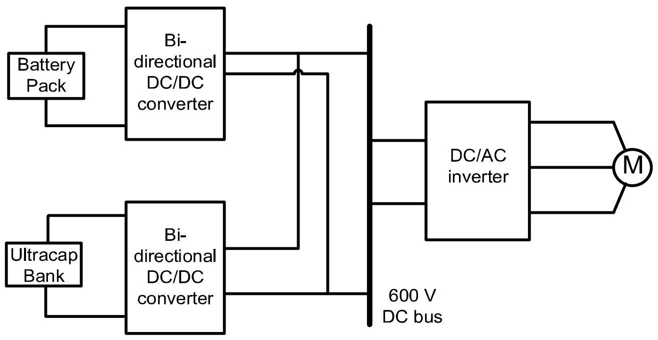

Battery Pack
Bi-directional DC/DC converter
Ultracap Bank
Bi-directional DC/DC converter
DC/AC inverter
M
600 V DC bus

Figure 2. Battery/Ultracapacitor parallel configuration

1.3 Review of PV involved HESS system

Sustainable energy resources, especially solar power, have drawn tremendous interests in recent decades. However, the stability of the photovoltaic (PV) output power cannot be guaranteed because of the variation of the operating conditions such as temperature, solar radiation and partial shading effects [7]. Fig. 3 shows a typical 24-hour PV output power and residential load power profile. It shows that the PV output power curve is changing dramatically and the load power curve is relatively placid. In addition, the mismatch between the power supply and the power demand is

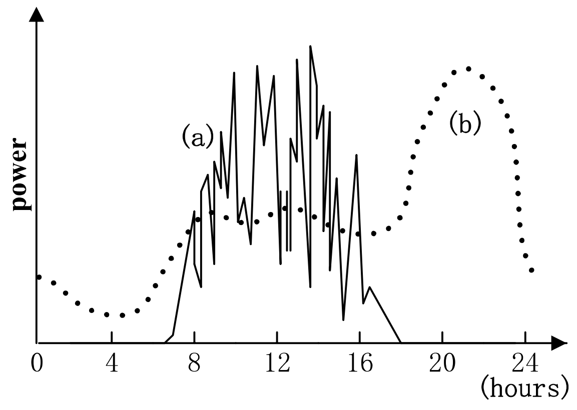


Figure 3. Conceptual 24-h (a) PV output power and (b) residential load profile [8]

unavoidable, especially for the cloudy days. Therefore, it is necessary to compensate for such mismatches in an optimal way, which is one of the motivation of our work. The energy storage system (ESS) is an indispensable component for the PV system. For our study, we choose the battery and ultracapacitor combined hybrid energy storage system (HESS).

1.4 Review of PV system configuration

The PV system configuration can be grouped into four types in general. The features of each type are listed as below. 1) PV only system [9]: the instability of PV power will stress the main power bus in that there is no power grid connected. The wide range input caused by DC or AC bus fluctuation will bring more challenges for the power electronics devices. Besides, the surplus power or deficient power cannot be preserved or compensated due to the lack of storage device. The PV only configuration can be seen in fig. 4(a) 2) PV-battery system [10]: Given the nature of the battery which is high energy density, it is a good candidate for smoothing out low frequency power [11]. In other words, it can effectively preserve and compensate slow moving surplus power and deficient power. However, batteries may suffer from the degradation issue when exposed to high fluctuating power profile. In this case, the battery lifetime tends to be shortened [12]. The PV-battery configurations are shown in fig. 4(b)(c) (3) PV-ultracapacitor (UC) system [13]: Given the nature of the UC which is high power density, it is a good candidate for smoothing out the pulsed power, which will alleviate the voltage stress on the main bus and converter. However, UCs can deplete or saturate very quickly due to the low energy density. Thus, it is not suitable for providing long-time lasting and constant power. The PV-UC configuration is shown in fig. 4(d) 4) PV-battery-UC system [14]: It is a PV system in combination of high energy density battery and high-power density UC, which has the following advantages.

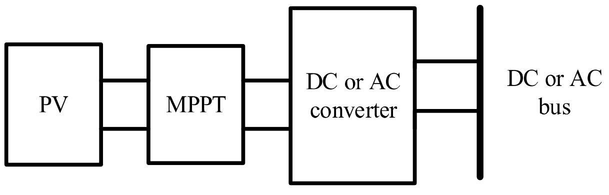
1. It can effectively smooth out the high frequency and low frequency flu ctuating PV power;

2. The main voltage bus is easier to stabilize;

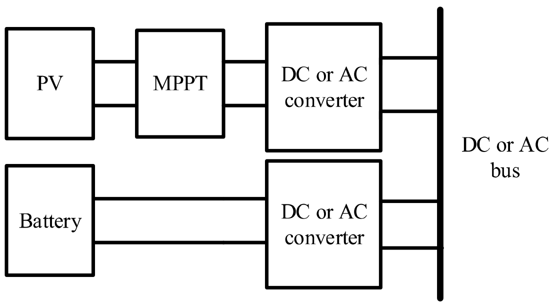
3. Because the surplus power can be stored, it will make full use of PV output power;

4. With UC, the battery power can be viewed as "filtered" since di/dt of charging/discharging rate will be limited. This will, as result, extend the life span of the battery.

The PV-battery-UC configuration can be seen in fig. 4(e)(f). Through carefully comparison, in this thesis, the battery and UC combined HESS is selected for the stand-alone PV system.



(a)



(b)

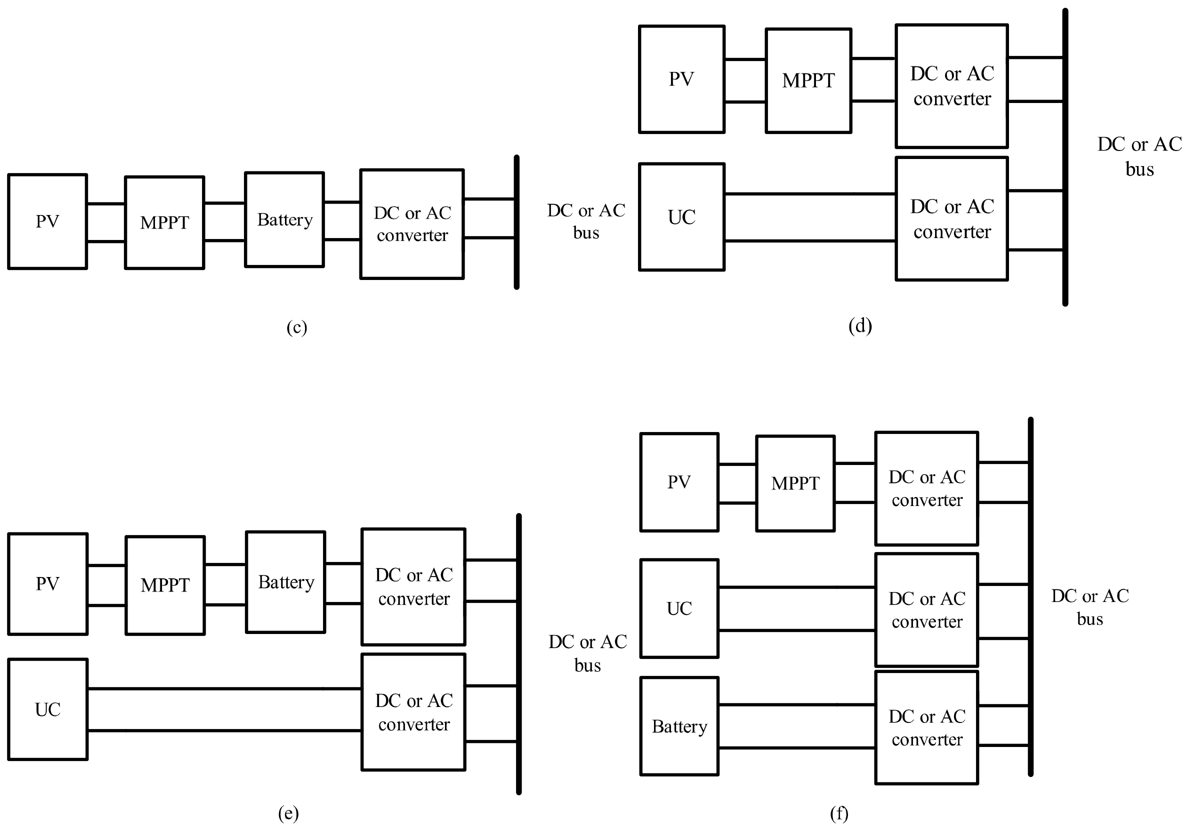


Figure 4. The configurations of PV system

1.5 Review of Structure of Fuzzy Logic System

Basically, fuzzy control is a type of digital control. Therefore, the structure of the fuzzy logic system is similar with the normal digital control system, which has the diagram shows in Fig. 5.

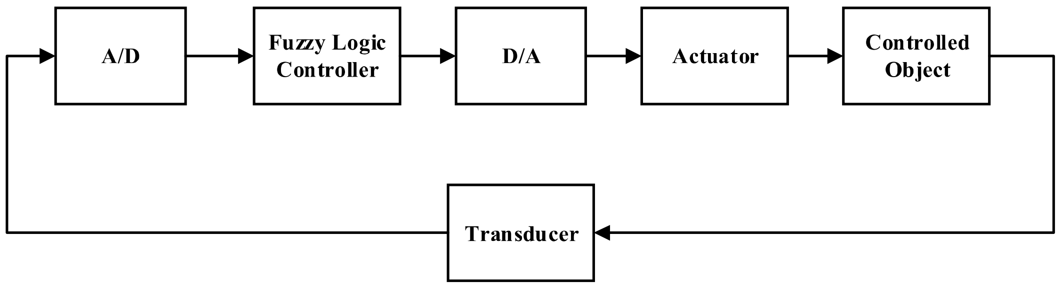


Figure 5. Structure of fuzzy logic system

From the figure, we can see that the fuzzy control system can be divided into four parts, which shows below:

1. The fuzzy logic controller. It is the core part of the whole fuzzy control system. According to the requirements of the system, the digital realization can be fulfilled by MCU, DSP or FPGA.

2. The inputs/outputs interface. It contains the Analog-to-Digital converter (A/D) and Digital-to Analog converter (D/A). A/D converts the analog input to digital output and send it to the fuzzy logic controller, the output of fuzzy logic controller will send the value to D/A and the analog output will be executed by actuator. In inputs/outputs interface, besides the A/D and D/A, the level shift circuit is required too.

4. Actuator & Controlled object. Typically, the actuator and the controlled object are integrated together. The type of controlled object can be varying: linear or nonlinear, time variant or time invariant, single variable or multiple variables, with delay or without delay, with turbulence or without turbulence. In our design, the controlled object are batteries and UCs.

5. Transducer. Transducers are often employed at the boundaries of automation, measurement, and control systems, where electrical signals are converted to and from other physical quantities. It plays a vital role in fuzzy system. The precision and resolution of the transducer can determine the performance of whole system.

1.6 Proposed fuzzy logic power management strategy*

In order to optimize the power distribution between the battery and the UC, the fuzzy algorithm, which was developed based on Zadeh's fuzzy set theory [15], is utilized here. Fuzzy system is suitable for complex nonlinear scenario with multiple variables. Its robustness and superior real-time performance make it a practical choice for various applications. Many researchers use fuzzy concept to design the controller in converter level. The associated power electronics design can follow [16] and [17], which use "error" and "change rate in error" as the input of fuzzy logic controller. The controller outputs determine the value of the control variables in the plants, such as duty ratio, pulse density, or phase shift, etc., depending on the topologie s. A comparative study between boost converter controllers is been presented in [18], which demonstrates the efficiency and robustness of fuzzy logic control compared to traditional linear control. However, few works implemented fuzzy logic in global level design. In [19] and [20], the authors developed math equations to fulfill the power flow control in global level. This method needs precise mathematical models and in nonlinear and unpredictable process, it is tedious and time consuming to find the

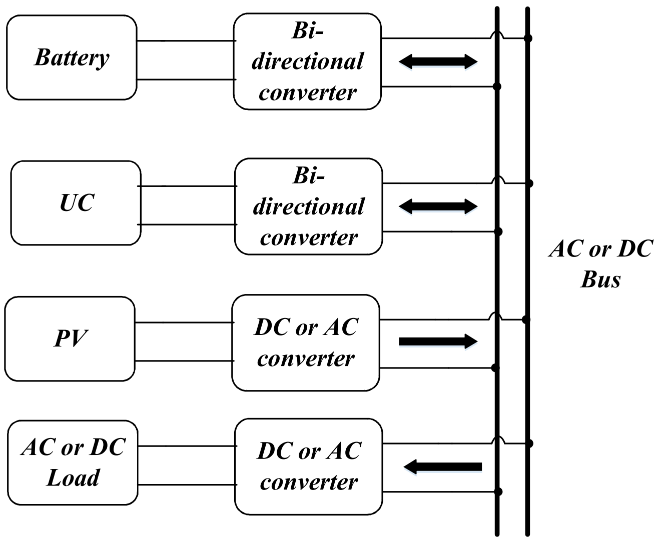


Figure 6. Suggested system configuration

precise mathematical model, let alone developing equations based on the model. Fuzzy-logic, however, is more suitable for global level design. Since it uses the experiences and intuition of human, it can deal with complex cases with the system has multiple inputs and multiple outputs. The suggested topology of proposed fuzzy-logic strategy is shown in Fig. 6, which can be utilized in either DC or AC power distribution configurations as long as the structure of system is parallel. The load can be either AC or DC as well. This demonstrates the compatibility of proposed fuzzy-logic strategy.

1.7 Digital realization of fuzzy logic algorithm using FPGA technology

With the increasing of the development of the Electronic Design Automation (EDA) technology, FPGA is receiving more attention than ever before due to its high compactness and high speed. In addition, the feasibility of modular design makes FPGA capable of new and complex algorithms like fuzzy logic algorithm [21-22]. There are researchers implemented FPGA on variable speed generators [23], MPPT controller [24], air-conditioning system [25], damped pendulum system [26] and so on. All of them are using single variable single output fuzzy system. There are few literatures addresses multiple inputs multiple outputs fuzzy system in global-level, let alone for HESS. This is another motivation of our work.

When developing a FPGA control system, there is an unavoidable procedure: function verification. Typically, the verification is carry out by Matlab/Simulink environment and the majority of simulation is executed by Modelsim software. In this fuzzy logic algorithm, for example, the design procedure can be divided into five parts: 1. Build the fuzzy logic model in the Simulink and do the simulation; 2. Write the HDL code for corresponding FPGA chip; 3. Synthesis the code by using specific software (i.e. QuartusPrime) and transfer it into RTL code; 4. Do gate-level or timing simulation by using Modelsim software; 5. Compare the results between Matlab and Modelsim, and then download the code into the FPGA chip. Fig. 7. Shows the flow chat of the design procedure. The FPGA-in-the-loop (FIL) technology is a derivative of Hardware-in-the-loop (HIL) technology, which can achieve communication between FPGA and Simulink. The advantages of adopting FIL is that the step 3 and 4 in design procedure can be skipped. The testing signal can be directly generated by Simulink. Through Ethernet or JTAG interface, the testing signals are directly transferred back and forth between FPGA and Simulink. Therefore, FIL can shorten the development cycle and reduce the research expenses.

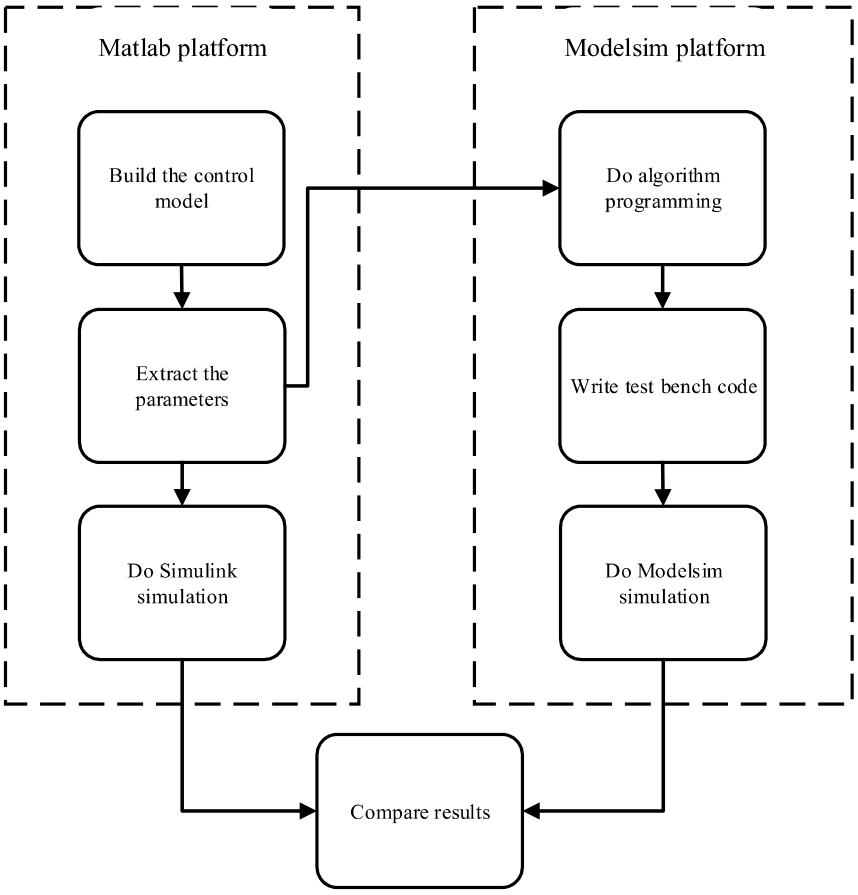


Figure 7. The flow chat of design procedure

1.8 Model-based FOC algorithm using FPGA technology

Due to the good characteristics of PMSM, it is widely used in many fields from daily life to sophisticate technology. The control algorithm is the key factor that affects the performance of PMSM. FOC algorithm is commonly regarded as the best method to control the PMSM. However, given the complexity of FOC algorithm and it requires real time feedback of position and angle information, traditional MCU cannot meet this requirement. Thus, people use DSP to fulfill the FOC algorithm for a long time. However, in some cases which require abundant I/O resources such as multi-motor control, the single DSP chip cannot meet the requirement either. In this scenario, people have to use two DSP chips, one for control algorithm and another for floating calculation, to improve the performance of the system. This in result lower the cost-effectiveness of the whole system.

Since the nature of FPGA is high flexibility and abundant resources, it is a good candidate for FOC algorithm. Especially, by utilizing PFGA technology, the frequency of output PWM can be increased to 100 kHz. For traditional DSP chip, the switching frequency is only around 10 kHz. Typically, the PWM signals are sent to the inverter board and the high frequency PWM makes the SiC-based inverter possible. This will bring us the following advantages:

- Increased switching frequency
- Low total harmonic distortion in machine phase current
- Increased fundamental frequency
- Increased machine pole numbers
- Reduction in machine weight

1.9 Organization of Thesis

Chapter 2 gives an analysis on the principle of the proposed fuzzy-logic management strategy in PV application. Chapter 3 provides the fuzzy logic power management strategy in EV application. Chapter 4 provides the digital realization of fuzzy algorithm by using VHDL language. Chapter 5 shows the model-based FOC algorithm design using. Chapter 6 concludes the contributions and the future work of this thesis.

Chapter 2 A Fuzzy Logic Power Management Strategy for HESS in PV Application

2.1 The Overall Scheme*

Given the nature of ultracapacitor is high power density and being able to endure fast charging and discharging rates and the nature of batteries are high energy density and being able to constantly provide or absorb power, it is an optimal choice that the battery provides average power and ultracapacitor will engaged during high magnitude high frequency power. A basic control strategy limits power provided by the battery. When below certain threshold, the battery works unaided. When demanded power crosses the threshold, ultracapacitor starts to charge/discharge. It is worth pointing out that the minimization of the additional cost of battery storage can be achieved by an appropriate power distribution strategy. The guidelines of power flow management are summarized as follows: 1. When the absolute value of power demand is constant and small, the power is only provided by the batteries unless they are over discharged (<25%) or over charged (>80%); 2. When the absolute value of power demand fluctuates dramatically, the power is mainly provided/absorbed by UC unless they are over discharged (<20%) or over charged (>85%); 3. If the UC is fully charged/discharged, batteries will be in charge for smoothing out the powerfluctuation. Here, the demanded power is defined as the difference between PV output power and load power. The equation can be expressed as (1).

$$P_{dem} = P_{load} - P_{pv} \tag{1}$$

The details of the power distribution are shown in Fig. 8. The arrows represent the direction of

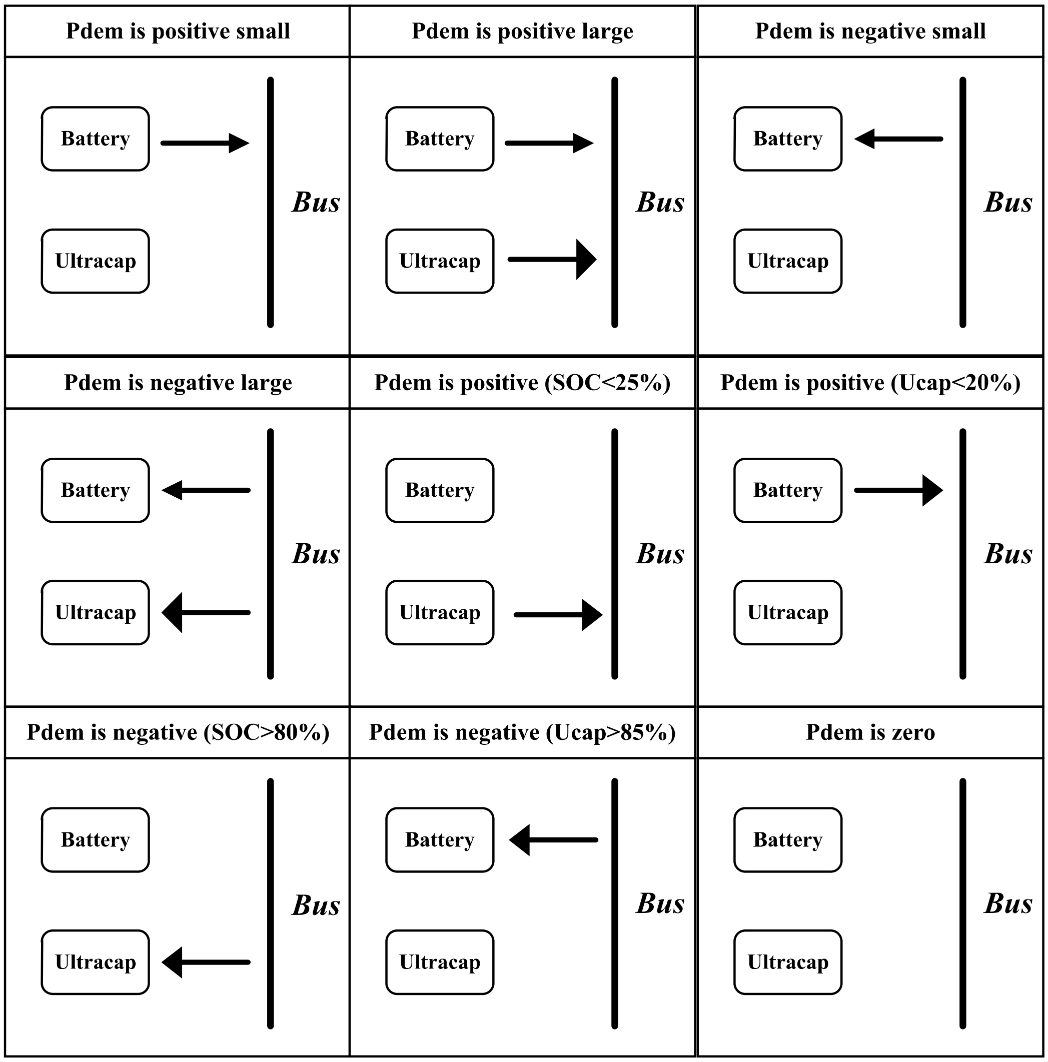


Figure 8. The power flow diagram

power flow and the size of arrows indicates the relative amount of power. A general schematic of fuzzy system is shown in Fig. 9. It can be observed that the fuzzy logic controller has four inputs and two outputs. The inputs are: the error between the actual bus voltage and the reference value (Vbuserr), power demand (Pdem), the UC's voltage (Ucap) and the state of charge of the battery (SOC). The outputs are: the reference for the battery output power (Pbat) and the reference for the

UC output power (Pcap). The Vbuserr information is used to stabilize the bus voltage. The Mamdani-type model is implemented in this Chapter.

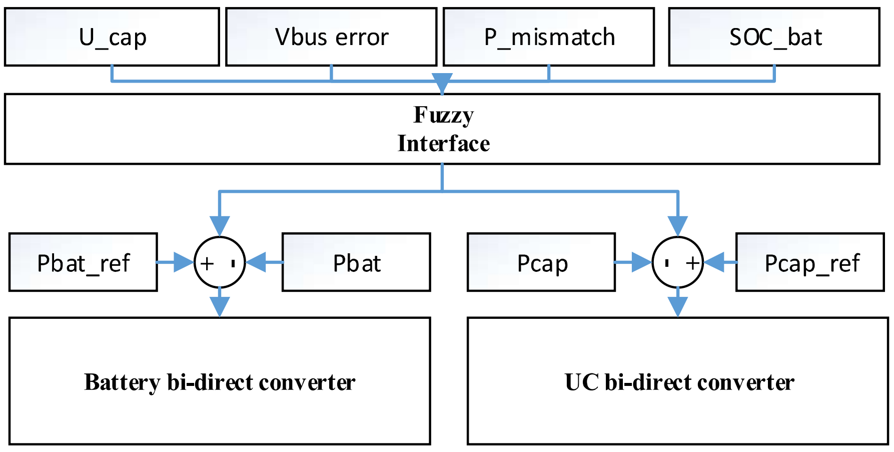


Figure 9. Overall scheme of fuzzy system

2.2 The Design of Proposed Fuzzy Logic Strategy

A fuzzy logic controller can achieve mapping from input variables to output variables, which is the core part of a fuzzy system. Typically, a fuzzy logic system includes three parts: the fuzzifier (fuzzification), the rule evaluator (rule inference) and the defuzzifier (defuzzification), as shown in Fig. 10. I will illustrate each part in detail in the following.

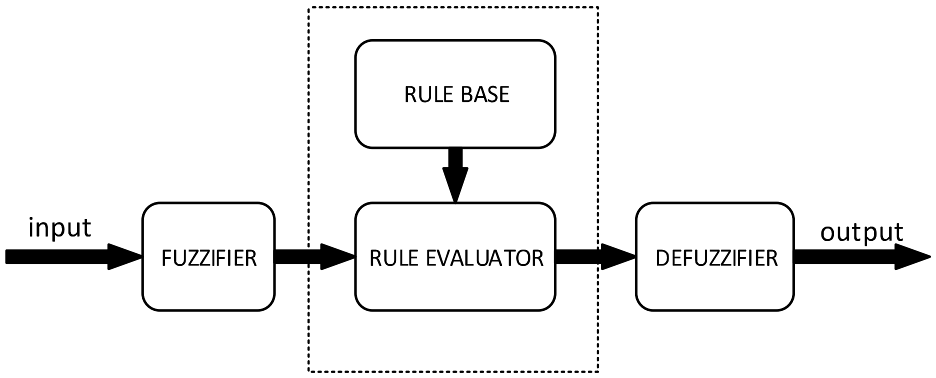


Figure 10. The composition of fuzzy system

2.2.1 Fuzzification

The fuzzification is the first stage of fuzzy logic system. Given the fact that the value received from sensor is always crispy value, we need to convert this crispy value into fuzzy sets before carrying out the rule evaluation. In this stage, the fuzzy sets of inputs and outputs should be defined. Let us take the fluctuation of DC bus voltage as an example. In our design, the fluctuation range of DC bus voltage is set to -30 V to 30 V. This fluctuation range is called the universal disclosure of the corresponding fuzzy sets. Typically, the universal disclosure is divided into five parts: negative large (NL), negative small (NS), zero (Z), positive small (PS) and positive large (PL). In this case, we define “NL” as -30 V, “NS” as -15 V, “Z” as 0 V, “PS” as 15 V and “PL” as 30 V. After defined the fuzzy sets of inputs and outputs, the membership function for each input and output should be formed. There are many different types of membership functions. Among them, the triangle type, the trapezoidal type, the gauss type and pendulum type are commonly used. However, the gauss type and pendulum type function are hard to implemented in digital world. Besides, the performance of trapezoidal type and triangle type function is almost the same as the gauss type and pendulum type function. For simplicity and effectivity consideration, the triangle

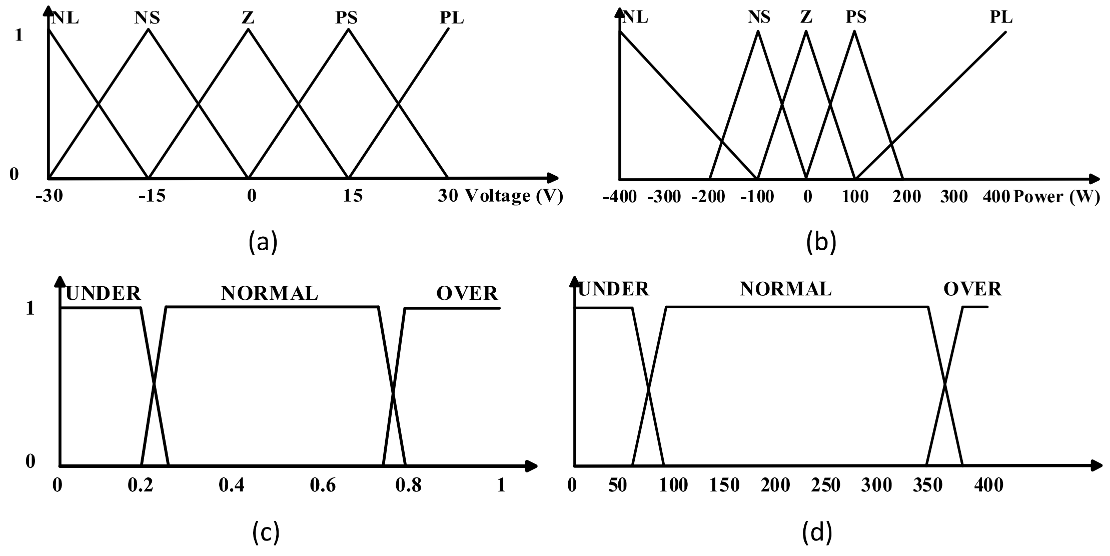


Figure 11. The membership functions of inputs: (a) Vbuserr, (b) Pdem, (c) SOC, (d) Ucap

shape membership function is utilized to describe “Vbuserr” fuzzy set, which can be seen from Fig. 11.

2.2.2 Fuzzy Inference

The fuzzy inference is the second stage of the fuzzy logic system. There are many fuzzy inference method, the Mamdani and Zadeh methods are commonly used. The difference between these two methods is the determination of fuzzy relation. However, when do the aggregation, they follow the same discipline. Here, the Mamdani method is utilized in our design, which contains a rule base with 20 rules. Typically, AND, OR and Not operators are used for the antecedent of rules. The IF-THEN rules with antecedent operators can have the form shows below:

IF x is A AND y is B THEN z is C

IF x is NOT A OR y is NOT B THEN z is C

Here, AND means the minimum of two items, OR means the maximum of two items. For our fuzzy logic strategy, all the three operators are utilized.

The details of the fuzzy rules are listed in table I. The weight of each rule is 1. For rules 1-9, they are independent from inputs “Pdem” and are designed for stabilizing the AC or DC bus; For rules 10-14, they are independent from the inputs “Buserr” and are designed for compensating the mismatch power; Rules 15-18 described the case when ultracapacitors are overcharged/discharged and batteries are under normal status, then batteries will be forced to provide/absorb power. For the last two rules 19-20, they are designed for the extreme scenario. In this case, both the battery and the ultracapacitor will make their effort to guarantee the power balance.

TABLE I. Fuzzy Rules of The Rule Base

IF								THEN	
Rule	***Buserr***	***Logical Operator***	***Pdem***	***Logical Operator***	***SOC***	***Logical Operator***	***Ucap***	***PBAT***	***PCAP***
1	NL	AND	-	AND	NOT OVER	AND	NOT OVER	PS	PL
2	NS	AND	-	AND	NOT OVER	AND	-	PS	Z
3	Z	AND	-	AND	-	AND	-	Z	Z
4	PS	AND	-	AND	-	AND	NOT UNDER	NS	Z
5	PL	AND	-	AND	NOT UNDER	AND	NOT UNDER	NS	NS
6	NL	AND	-	AND	NOT OVER	AND	-	PL	Z
7	NS	AND	-	AND	-	AND	NOT OVER	Z	PS
8	PS	AND	-	AND	NOT UNDER	AND	-	Z	NS
9	PL	AND	-	AND	-	AND	NOT UNDER	Z	NL
10	-	AND	NL	AND	-	AND	NOT OVER	NS	NS
11	-	AND	NS	AND	NOT OVER	AND	-	PS	Z
12	-	AND	Z	AND	-	AND	-	Z	Z
13	-	AND	PS	AND	NOT UNDER	AND	-	NS	Z
14	-	AND	PL	AND	-	AND	NOT UNDER	Z	NL
15	-	AND	NL	AND	NOT OVER	AND	OVER	PL	Z
16	-	AND	NS	AND	NOT OVER	AND	OVER	PS	Z
17	-	AND	PS	AND	NOT UNDER	AND	UNDER	NS	Z
18	-	AND	PL	AND	NOT UNDER	AND	UNDER	NL	Z
19	PL	AND	PL	AND	NOT UNDER	AND	NOT UNDER	NL	NL
20	NL	AND	NL	AND	NOT OVER	AND	NOT OVER	PL	PL

2.2.3 Defuzzification

The defuzzification is the third stage of the fuzzy logic controller. It converts the fuzzy values into a crisp value for output. The methods for defuzzification are various. They are centroid method, height method, weighted average method, middle of maximum method, center of sums method, center of largest area method and smallest (largest) of maximum method. The most prevalent and physically appealing of all the defuzzification methods is Center of Gravity (COG), which is shown in Fig. 12, the equation is shown in (2). The merits of this method are as following: 1. It in line with the human intuition; 2. It consists most of the information in fuzzy sets so that the results are more promising; 3. In Matlab/Simulink, COG method is a built-in method thus it is easy to evaluate. Therefore, COG method is selected in this chapter.

$$COG = \frac{\int_{i=1}^{no.\ of\ rules} grade_i \times center\ of\ grade_i}{\int_{i=1}^{no.\ of\ rules} grade_i} \tag{2}$$

For the outputs membership function, we determined the universal disclosure of the "Pbat" and "Pcap". Since the UC will absorb/release more power than batteries, the disclosure range of "Pcap" is larger than "Pbat". Here, the triangle type function is selected, which shows in Fig. 13.

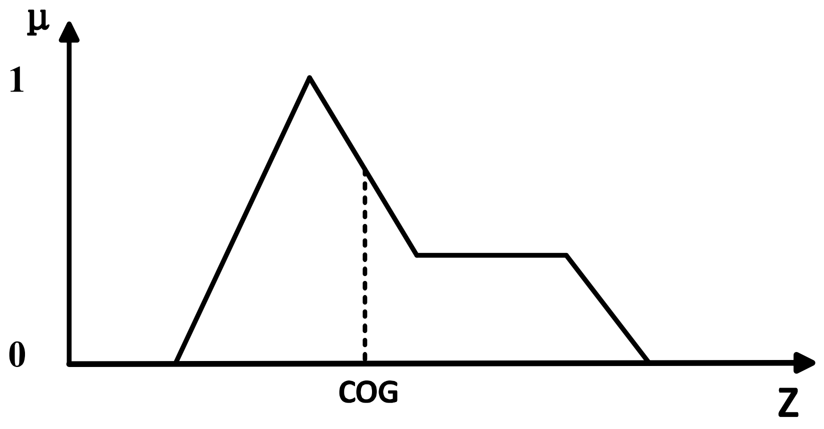


Figure 13. The concept of COG method

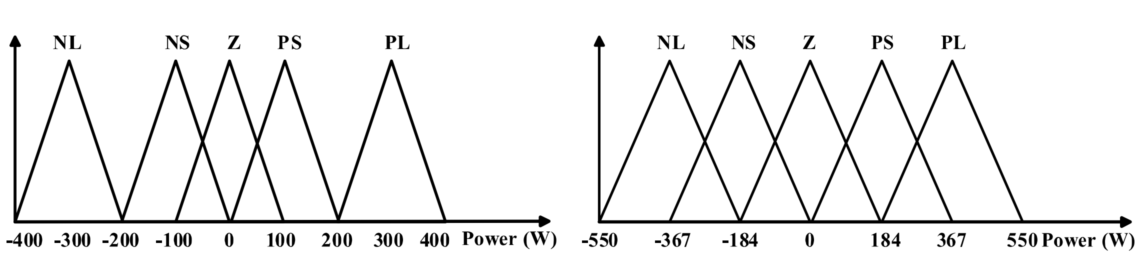


Figure 12. The outputs membership function: (a) Pbat, (b) Pcap

2.3 Simulation Results

In order to verify the proposed power management strategy, the fuzzy logic system model has been developed in Matlab/Simulink. The power level of a scaled-down PV system is set at 1 kW. Five scenarios have been studied: 1. When SOC of the batteries and the voltage of UC are below normal status; 2. When the batteries are overcharged; 3. When the batteries are over discharged; 4. When the UC is over discharged; 5. When the UC is overcharged. The assumptive “Pload” and “Ppv”

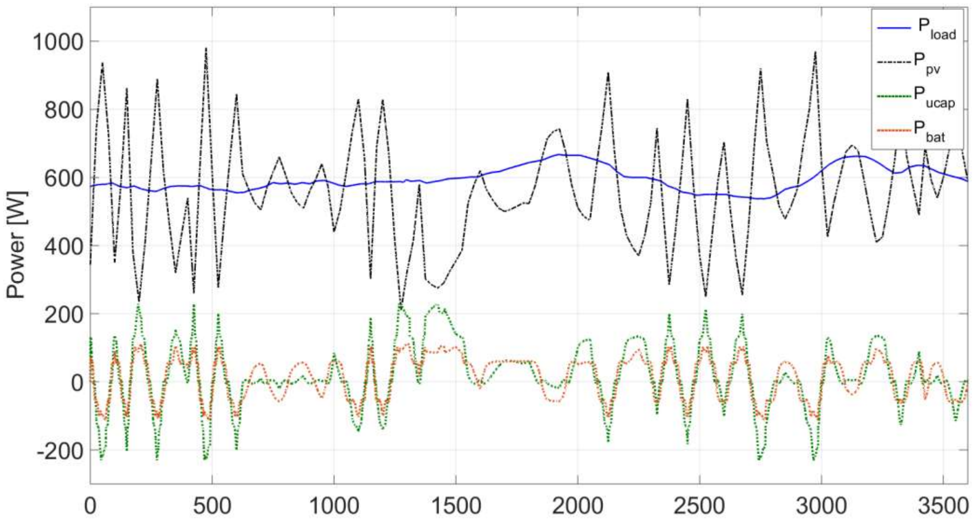


Figure 14. The power distribution when two sources are under normal status

profiles, which are based on 1000s-scale, are generated to simulate the actual PV power characteristics. Fig. 14 shows the normal scenario. In this case, the power range of battery is well limited to make the “Pbat” curve placid. At the meantime, batteries are able to smooth out low frequency and low magnitude power demands such as shown from 250s to 400s.

For Fig. 15, because the SOC of the battery is very low, the battery only absorbs power rather than providing power at the beginning. Fig.16 shows the opposite scenario compared with Fig. 15, which means the SOC of battery is too high and cannot absorb power any more. Fig.17 shows that the UC's voltage is too high to absorb the power and Fig.18 shows the cases when UC's voltage is too low to release.

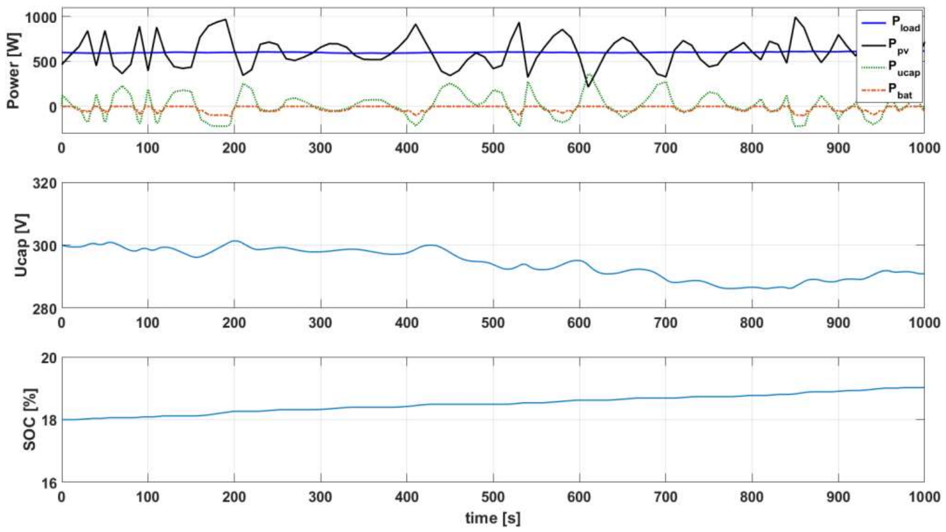


Figure 15. The power distribution when batteries are over-discharged

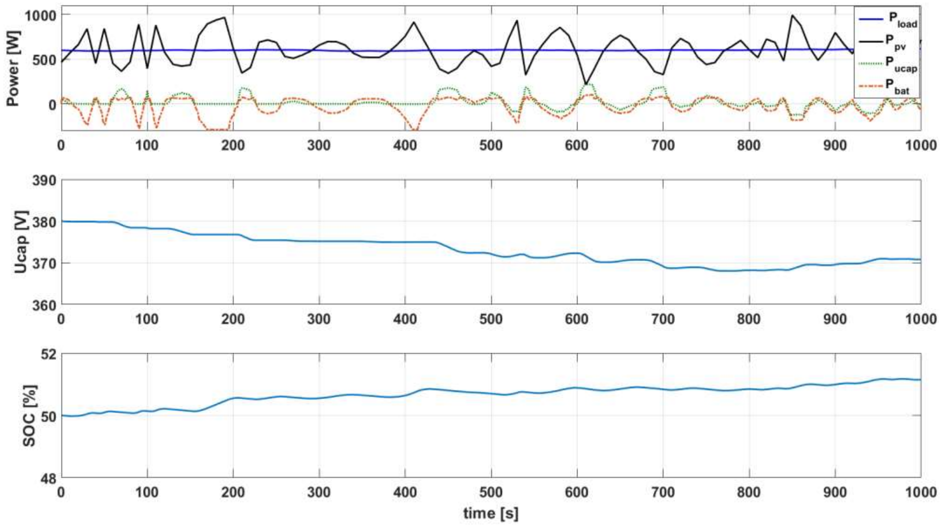


Figure 16. The power distribution when UC are over-charged

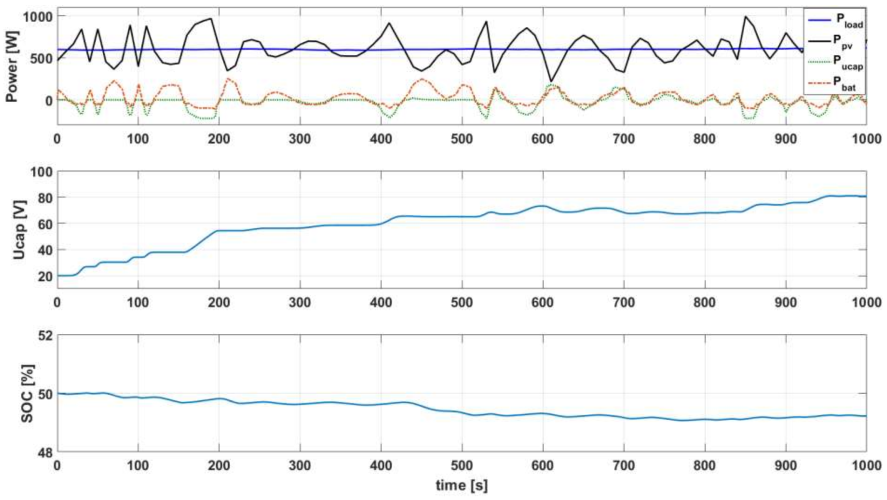


Figure 17. The power distribution when UC are over-discharged

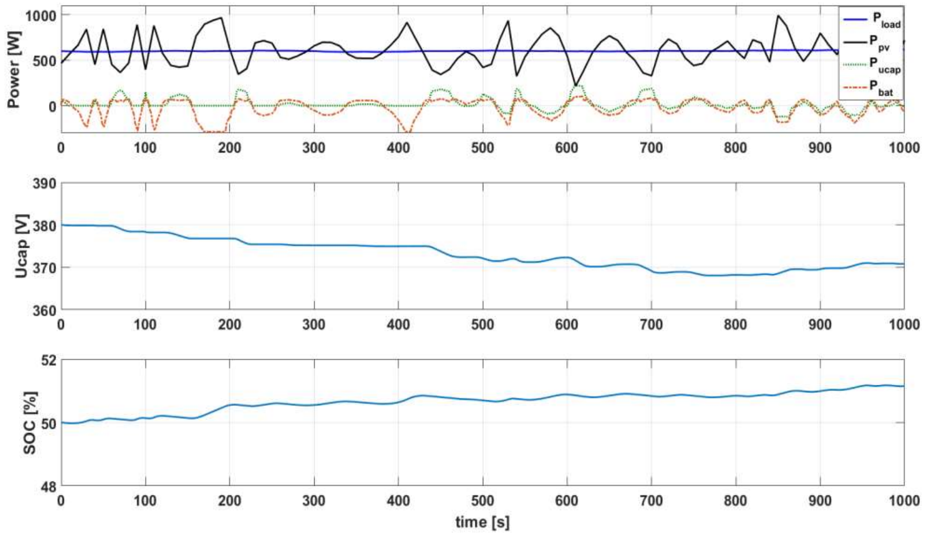


Figure 18. The power distribution when UC are over-charged

The simulation results demonstrate that the PV power is well smoothed out by the storage devices. Besides, the batteries absorb/release the least pulse power, which greatly helps extend the life span of HESS.

Chapter 3* A Fuzzy Logic based Power Management Strategy for HESS in HEVs Application Considering Battery Degradation

3.1 Analysis of Fuzzy Logic Power Management Strategy in HEVs

A general schematic of fuzzy logic control is shown in Fig. 19. From the control scheme, the fuzzy interface has five inputs and two outputs variables, which is a MIMO system. In [27], the author gave us a good example of using fuzzy logic controller to do power management in HEV application. However, [27] did not consider the effect of fuzzy logic strategy when considering battery degradation.

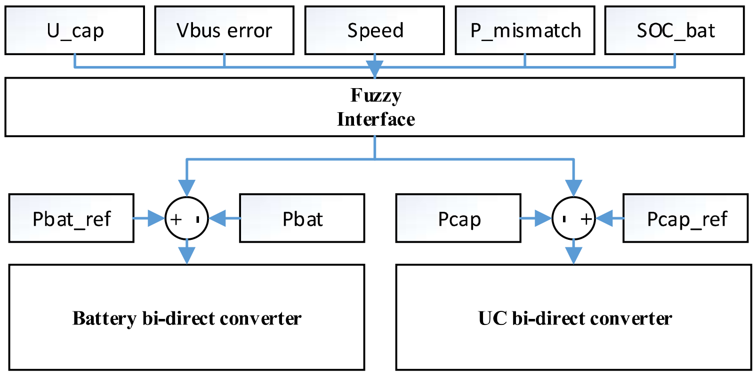


Figure 19. Overall scheme for HESS in HEV application

For our study, the input variables are: output voltage of the hybrid DC link error (Udc_err), power load (P_load), ultracapacitors voltage (Ucap), normalized speed (V2) range from 0 to 1 and the UC's state of charge (SOC). The membership function of the inputs is shown in Fig. 20. Since the zero-order Sugeno model is implemented, the output membership functions are constant values for both "Pbat" and "Pcap": neg_L=-40, neg_S=-20, zero=0, pos_S=20, pos_L=40.

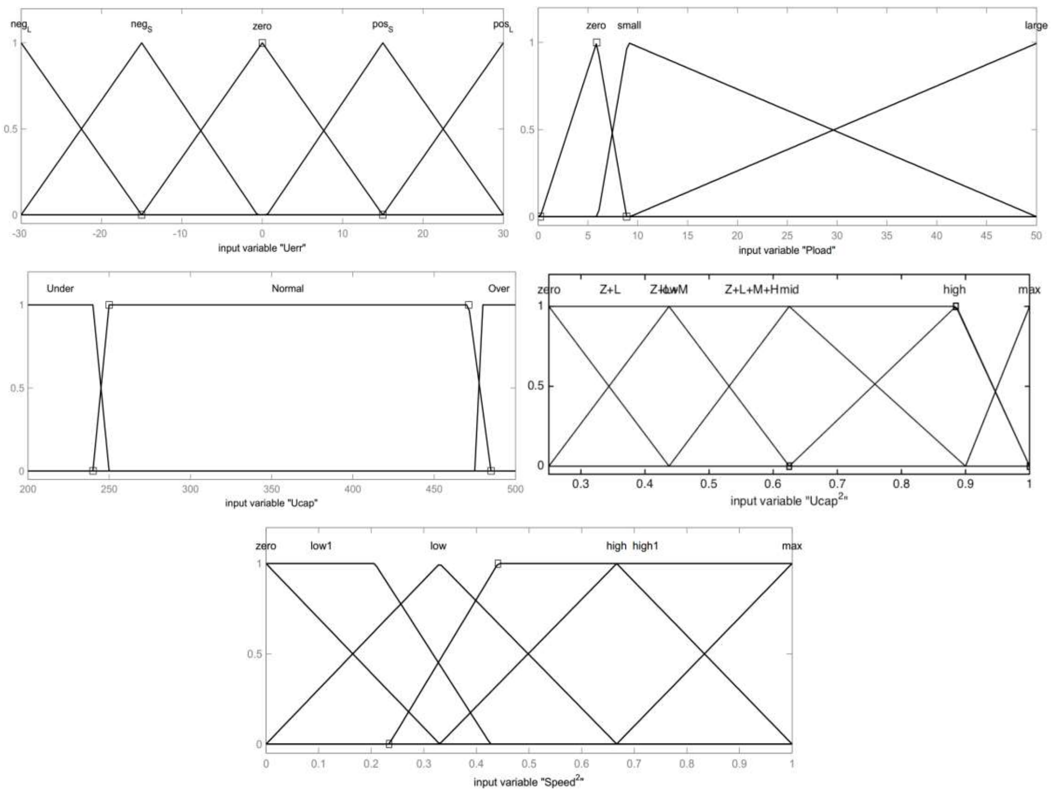


Figure 20. The membership function of inputs

Here, two types of fuzzy logic are presented. Firstly, a fuzzy logic-based "balanced" strategy is proposed. In this case, the power flow between battery and ultra-capacitor obeys the rules as follows: 1.Batteries will provide the required power when the speed of car is constant; 2. Ultracapacitors provide required power during acceleration unless they are over discharged (<45%); 3. Ultracapacitors absorb power during braking unless they are over charged (>90%); 4. If ultracapacitors are fully charged/discharged, battery will provide the power for acceleration and capture the energy from regenerative braking. The detailed set of rules for fuzzy logic controller are shown below. The end value of each rule indicates the weight.

1. If (*Uerr* is pos_L then (*Pbat* is neg_L)(*Pcap* is zero) (1)

2. If (*Uerr* is pos_S) then (*Pbat* is neg_S)(*Pcap* is zero) (0.5)

3. If (*Uerr* is zero) then (*Pbat* is zero)(*Pcap* is zero) (1)

4. If (*Uerr* is neg_S) and (*Ucap* is not over) then (*Pbat* is zero)(*Pcap* is pos_S) (1)

5. If (*Uerr* is neg_L) and (*Ucap* is not over) then (*Pbat* is zero)(*Pcap* is pos_L (1)

6. If (*Uerr* is neg_S) and (*Ucap* is over) then (*Pbat* is pos_S)(*Pcap* is zero) (0.5)

7. If (*Uerr* is neg_L) and (*Ucap* is over) then (*Pbat* is pos_L)(*Pcap* is zero) (1)

8. If (*Pload* is large) and (*Ucap* is not under) then (*Pbat* is zero)(*Pcap* is neg_L)(1)

9. If (*Pload* is small) and ($speed^2$ is high1) and (*Ucap* is not under) then (*Pbat* is zero)(*Pcap* is neg_S) (0.25)

10. If (*Pload* is neg_zero) and ($speed^2$ is low1) and (*Ucap* is not under) then (*Pbat* is zero)(*Pcap* is neg_S) (0.05)

11. If ($Ucap^2$ is max) and ($speed^2$ is low) then (*Pbat* is Pos_L)(*Pcap* is neg_L) (0.125)

12. If ($Ucap^2$ is not zero+low+mid) and ($speed^2$ is high) then (*Pbat* is Pos_L)(*Pcap* is neg_L) (0.125)

13. If ($Ucap^2$ is not (zero+low) and ($speed^2$ is max) then (*Pbat* is Pos_L)(*Pcap* is neg_L) (0.125)

14. If ($Ucap^2$ is zero+low) and ($speed^2$ is max) then (*Pbat* is pos_L)(*Pcap* is Neg_L) (0.05)

15. If ($Ucap^2$ is high) and ($speed^2$ is zero) then (*Pbat* is neg_L)(*Pcap* is pos_L) (0.1)

16. If ($Ucap^2$ is zero+low+mid) and ($speed^2$ is low) then (*Pbat* is neg_L)(*Pcap* is pos_L) (0.125)

17. If ($Ucap^2$ is zero+low) and ($speed^2$ is high) then (*Pbat* is neg_L)(*Pcap* is pos_L) (0.125)

18. If ($Ucap^2$ is mid) and ($speed^2$ is high) then (*Pbat* is neg_L)(*Pcap* is Pos_L) (0.08)

19. If ($Ucap^2$ is zero+low+mid) and ($speed^2$ is zero) then (*Pbat* is neg_L)(*Pcap* is pos_L) (0.2)

20. If ($Ucap^2$ is zero+low) and ($speed^2$ is low) then (*Pbat* is neg_L)(*Pcap* is Pos_L) (0.1)

21. If ($Ucap^2$ is zero) and ($speed^2$ is high) then (*Pbat* is neg_L)(*Pcap* is pos_L) (0.1)

For rules 1-8, they are independent from inputs "SOC" and "Speed^2". The rest of the rules are designed to maintain the SOC of ultracapacitor at a reasonable value. However, since lithium-ion (Li-ion) batteries take up a good percentage of vehicle cost, which was already discussed in section I, we always want to make the degradation of battery as less as possible. In this case, an optimized fuzzy logic-based "cost-effective" strategy is developed by sacrificing the life time of ultracapacitor. The detailed set of rules are shown below. The rules 15-21 are working as the voltage limiter of UC. Thus, these rules were changed to meet the requirement.

15. If ($Ucap^2$ is high) and ($speed^2$ is zero) then (*Pbat* is neg_L)(*Pcap* is zero) (0.1)

16. If ($Ucap^2$ is zero+low+mid) and ($speed^2$ is low) then (*Pbat* is neg_L)(*Pcap* is zero) (0.125)

17. If ($Ucap^2$ is zero+low) and ($speed^2$ is high) then (*Pbat* is neg_L)(*Pcap* is zero) (0.125)

18. If ($Ucap^2$ is mid) and ($speed^2$ is high) then (*Pbat* is neg_L)(*Pcap* is zero) (0.08)

19. If ($Ucap^2$ is zero+low+mid) and ($speed^2$ is zero) then (*Pbat* is neg_L)(*Pcap* is zero) (0.2)

20. If ($Ucap^2$ is zero+low) and ($speed^2$ is low) then (*Pbat* is neg_L)(*Pcap* is zero) (0.1)

21. If ($Ucap^2$ is zero) and ($speed^2$ is high) then (*Pbat* is neg_L)(*Pcap* is zero) (0.1)

The results of these two fuzzy logic strategies will be discussed in simulation results. Furthermore, for the given five inputs, the behaviors of "SOC" and "speed^2" are with the most concern. On the one hand, the SOC of ultracapacitor and the speed of car are two significant factors when doing power distribution; on the other hand, the nonlinearity of the system makes them very complicated to control. However, the fuzzy logic control is designed to address the issue mentioned above. In addition, it has significant benefits over traditional control method, i.e., PID. Fig. 21 shows the relationship of "SOC", "speed2" and "P_bat".

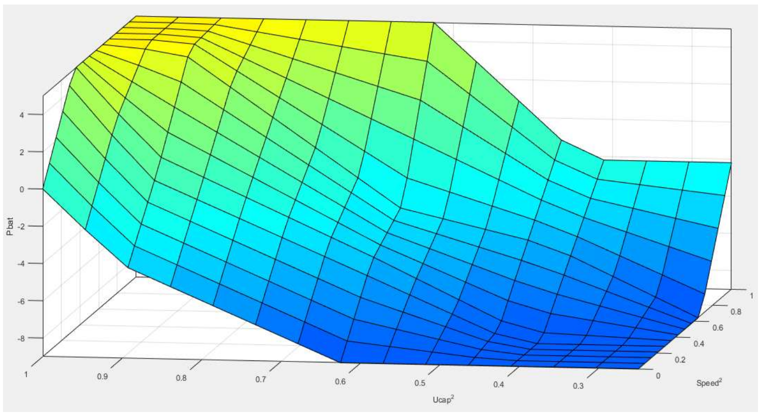


Figure 21. The relationship among "SOC", "speed^2", "P_bat"

3.2 Analysis of Lithium-ion Battery Degradation Model

It is one of our objectives to justify the impact of fuzzy logic control strategy on the battery lifetime. For battery choosing, we have compared the characteristics of different types of batteries. The requirements of electrified vehicle brought us challenge in energy and power density, SOC window and cycle life, charge/discharge rate, operating temperature and safety. Table II lists the theoretical and practical performance of battery technologies [28].

TABLE II. BATTERY TECHNOLOGIES

System	Neg Elec	Pos Elec	OCV	Theoretical Ah/kg	Theoretical Wh/kg	Practical Wh/kg
Pb-acid	Pb	PbO_2	2.1	83	171	20-40
Nickel-Cadmium	Cd	NiOOH	1.35	162	219	40-60
Ni-metal Bydride	MH	NiOOH	1.35	178	240	60-80
Sodium-Metal Chloride (300C)	Na	$NiCL_2$	2.58	305	787	80-100
Lithium-Ion	Li_xC_6	Li_{1x} MO_2 M=C, Ni,Mn	4.2 (4.0) to 3.0	95@ x=0.6	380	150-200
Lithium polymer	Li	VOx	3.3 (2.6) to 2.0	340	884	150

It can be observed that Lithium-ion battery has larger energy density and power density. For example, $LiNiCoAlO_2$ 4.xV/cell is already used for automotive applications. Lithium-ion batteries have good discharge power performance across some nominal temperature band centered at room temperature [28]. Thus, we choose lithium-ion battery as one source of the HESS.

Given that lithium-ion (Li-ion) batteries take up a good percentage of vehicle cost [29-30], the battery degradation during charge/discharge optimization is extremely important. [30] has predicted that in 2017, the cost of the ultracapacitor-based system is less than the cost of the current best solution (double absorbed glass mat (2xAGM) batteries), and is only about 20% more expensive than the traditional dual starting-lighting-ignition (2xSLI) solution. Thus, a combination of battery and ultracapacitor would lower the system cost in a large extent compared to battery-only storage system.

Models to predict battery life from limited test data have fallen into two types [31]. The first type is the model of the detailed chemistry of the cell, which requires extensive computation facilities. It can predict certain time and temperature effects accurately, but do not agree well with cycle life test data. However, it is very hard to build the accurate model even we have all the characteristics of battery. The second type is the empirical model, which fits test data to equivalent circuit models. Besides, it uses power law relationships that fit a limited range of parameters. Although it lacks theoretical basics, it is more conceptual and has computational simplicity and relative high accuracy [32]. We have implemented the battery degradation model proposed in [33], which is an empirical model. Typically, there are three types of degradation. They are temperature-related degradation [34], SOC-related degradation [35] and depth of discharge (DOD) degradation [36]. For temperature-related degradation, the temperature change produced by a given charge profile is approximated as a linear function of charge power. The equation is defined as below:

$$\frac{\Delta L}{L} = \int \frac{1}{8760 \times L_p(T_{amb} + R_{th} \times |P(t)|)} dt \tag{3}$$

Where ΔL is the lifetime degradation due to the charge cycle being evaluated per unit time, and L is the total battery lifetime if the charge cycle under evaluation were repeated until the battery's end of life. ΔL/L is battery life expended as a fraction of total lifetime, R_{th} is the thermal resistance of the battery pack and T_{amb} is the ambient temperature. For SOC-related degradation, the following linear fit equation is derived from the data in [38] for the cost of one hour during which the average SOC is SOC_{avg}:

$$\frac{\Delta L}{L} = \frac{m \times SOC_{avg} - d}{CF_{max} \times 15 \times 8760} \tag{4}$$

Here CF_{max} is the capacity fade at the battery end of life (EOL) which we have taken above to be 100% - Q_{EOL}/Q_0 = 20%. The SOC_{avg} are limited by the vehicle's battery protection controls to a desired range (30%-90%). For DOD-related degradation, [38] define ETL as the lifetime energy throughput (a function of ΔSOC_{avg}), $E_{T,used}$ as the total change in the remaining energy throughput due to a cycle, and $E_{T,base}$ as the minimum energy throughput required to recharge the battery. The battery degradation is then:

$$\frac{\Delta L}{L} = \frac{E_{T,used} - E_{T,base}}{E_{TL}} \tag{5}$$

The performance and expected lifetime of a battery are greatly impacted by the temperature at which the battery operates [39]. Basically, the higher the temperature, the shorter the battery life time. Since the temperature takes up a major percentage of battery degradation in our system, we selected "P_bat", which reflects the charge/discharge behavior of battery, as a reference to see how the temperature fluctuation of battery influence the battery lifetime. Thus, the temperature-related degradation is selected.

3.3 Simulation Results

In order to acquire the performance of the proposed control scheme, a simulation model has been developed in Matlab/Simulink. The driving cycles we chose are provided by United States Environmental Protection Agency (US EPA) and were derived using statistical method from the vehicle speed profile captured in major cities. Here, we selected the New York speed cycle and the Urban speed cycle for case study. Specifically, the New York speed cycle indicates the low speed scenario and the Urban speed cycle indicates the high-speed scenario. Table III presents the parameters regarding vehicle studied.

TABLE III. VEHICLE PARAMETERS

Vehicle Parameters	
Mass (vehicle only)	1500 kg
Max speed	45km/h
Lithium battery storage	10kW/h
Ultracapacitor value	1.76F (initial voltage: 475V)
Aerodynamic coefficient	0.37
Rolling friction coefficient	0.01
Grade/Slope	0

The simulation results of power distribution between the battery and ultracapacitor and the battery degradation curve are shown in Fig. 22-24. Two types of control strategy as mentioned previously are compared. The battery degradation under New York speed cycle is presented here. For Fig. 23, we limited the battery power and make ultracapacitor reaching its full potential. Specifically, the power range of battery is well limited from -10kW to 10kW to make the "P_bat" curve as placid as possible. For the battery degradation, the end value of curve shows the *$\Delta L/L$* with one-year (8760 hours) driving under specific driving cycle.

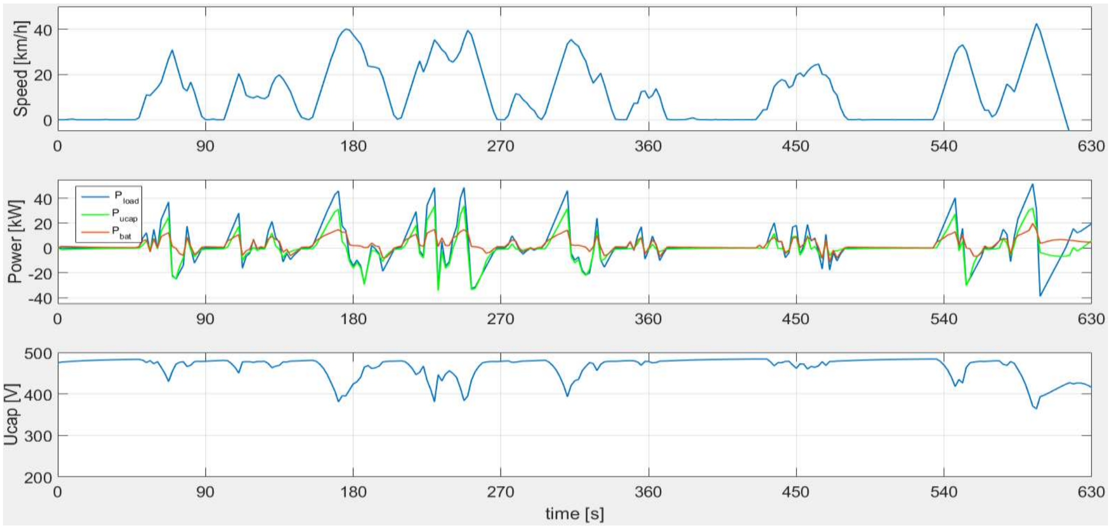


Figure 22. The power distribution under New York speed cycle (balanced strategy)

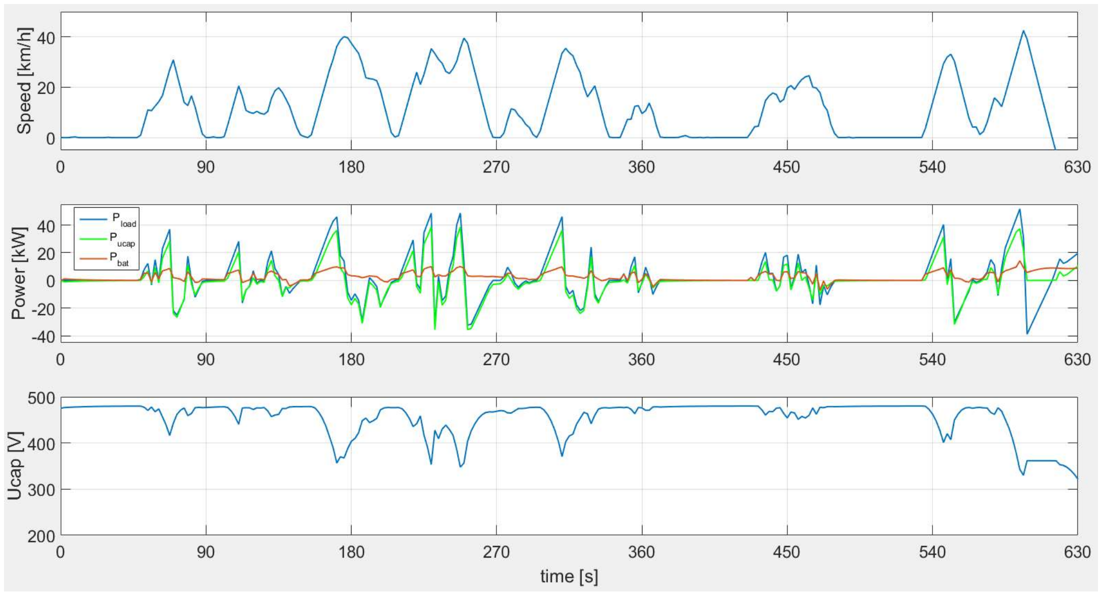


Figure 23. the power distribution under ECE speed cycle (cost-effective)

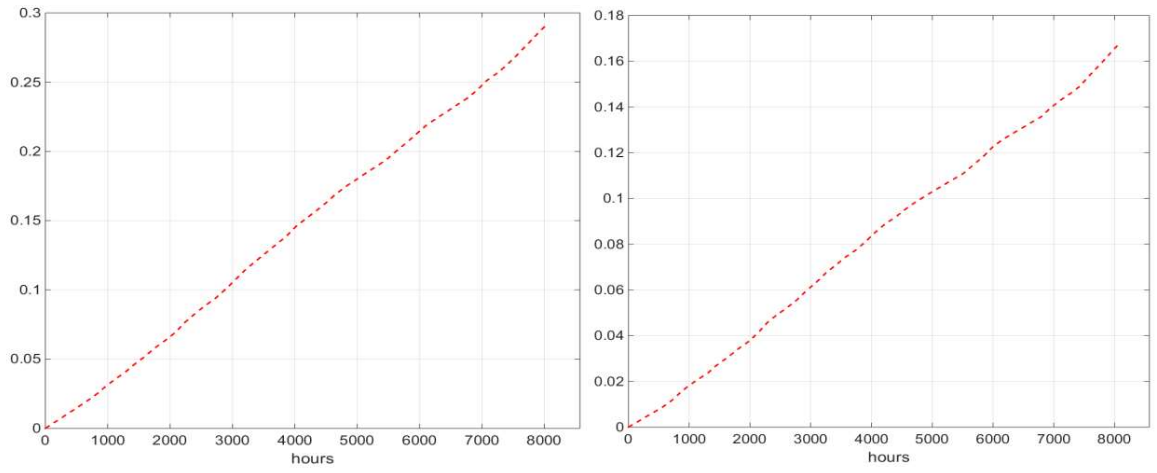


Figure 24. The battery degradation under New York speed cycle (left is balanced strategy, right is cost effective strategy)

It can be observed from the simulation results that the optimized fuzzy logic control strategy has less battery degradation (17%) than original fuzzy logic control (29%). To further test the performance of our proposed fuzzy logic control strategy, we implemented several high-speed cycles from US EPA. Fig. 25-27 shows the simulation results. For the Fig. 25, we can see that the system works very well at low speed range (0-150s). However, when the speed of car continues to go up (150-300s), some extremely high and low power occurs (marked by red cycle). In this case, neither battery nor ultracapacitor can provide or absorb these powers. According to the power flow equation from (1), the engine of car starts working at this time to provide the rest of power needed. Fig. 26-27 shows the similar condition as Fig. 25. Although our system has positive effects on both low speed scenarios and high-speed scenarios, it is more suitable for the low speed cases.

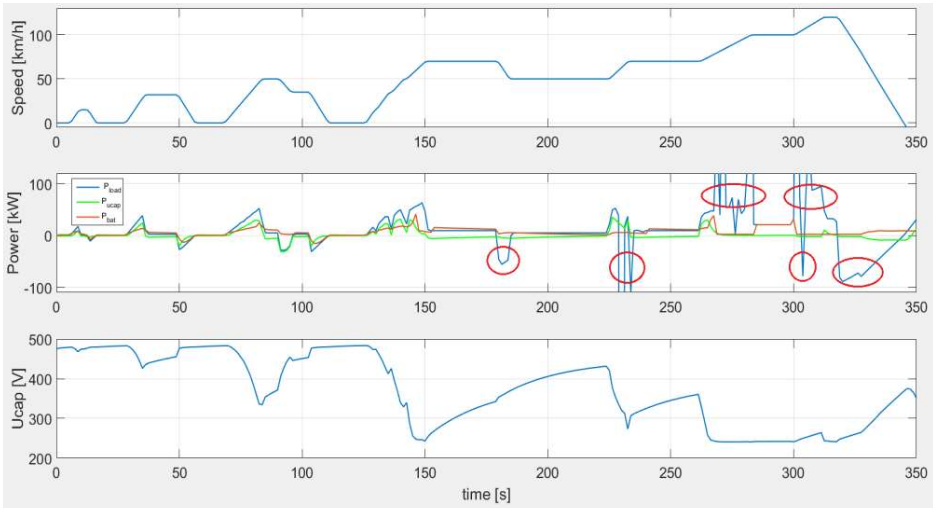


Figure 25. The power distribution under ECE speed cycle

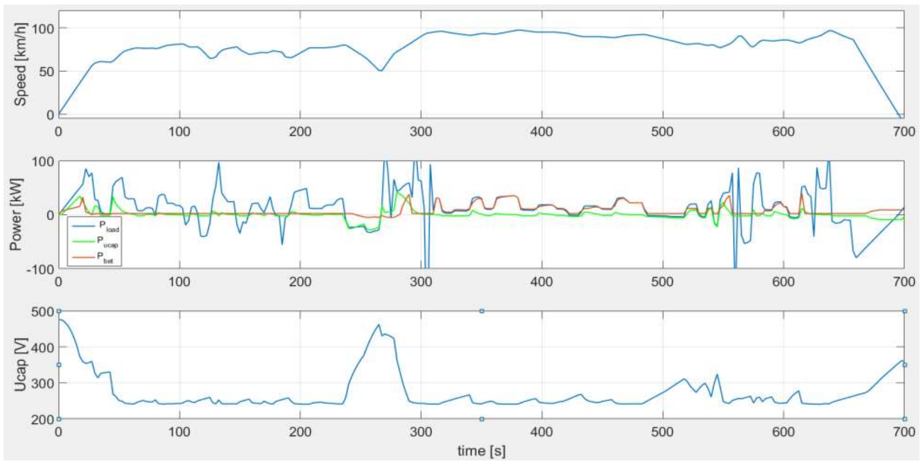


Figure 26. The power distribution under highway speed cycle

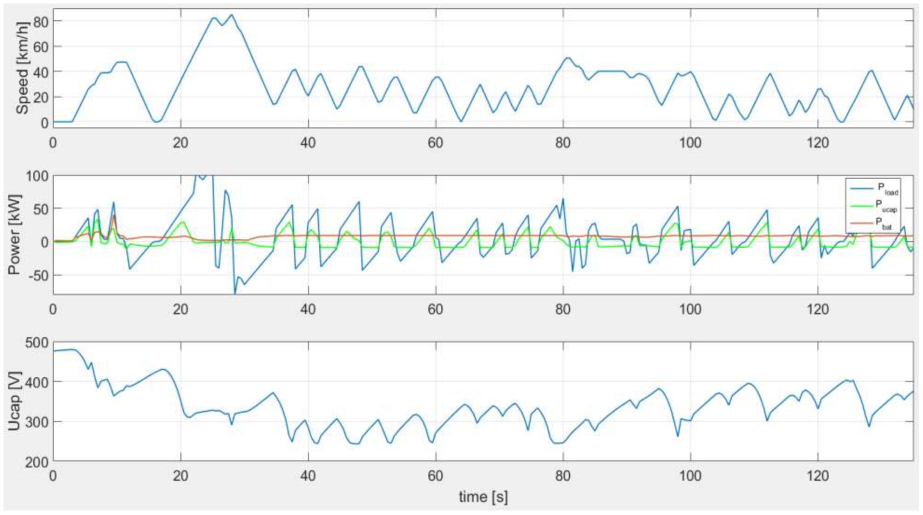


Figure 27. The power distribution under urban speed cycle

Chapter 4 Digital Realization of Fuzzy Logic Algorithm

4.1 The digital design flow

Typically, there are two types of design methods in digital world: the top-down method and the bottom-up method. The majority of traditional design method is bottom-up, which has the following procedures: 1. Determine the bottom-level modules as well as the structure and the function of fundamental components; 2. Based on the requirements of the main system, each basic module is composed in a specific manner to satisfy the higher-level specifications; 3. Repeat procedure 1 and 2 until meet the requirements of the whole system. The feature of this method is that the accessibility of bottom hardware should be addressed and the characteristics of each basic module should be clarified. Therefore, it lacks efficiency. The procedures of the top-down method are listed as following: 1. Determine the main function of the whole system and divide the whole system into different modules; 2. Realize each basic module and compose them to a higher-level module until the whole system is built. The top-down method is more efficient than the bottom-top method and more applicable in the modern control. The flow chat of designing fuzzy logic power management strategy using FPGA are shown in Fig. 28.

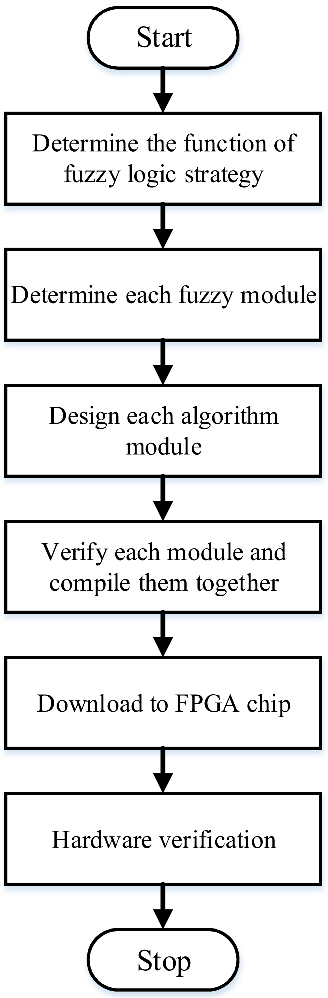

Start
Determine the function of fuzzy logic strategy
Determine each fuzzy module
Design each algorithm module
Verify each module and compile them together
Download to FPGA chip
Hardware verification
Stop

Figure 28. The flow chat of fuzzy logic algorithm realization

4.2 The Architecture of Fuzzy System

In digital world, all the values are represented in binary form. In our study, the 8 -bit binary number which representing “0-255” is used for describing the proposed FLC [40]. In this case, the disclosure of universe is restricted from X”00” to X”FF” (the “X” sign indicates hexadecimal number representation). In this Chapter, the proposed fuzzy logic power management strategy for PV application is taken for an example.

4.2.1 Fuzzification

The fuzzification is the first stage of the fuzzy logic system. For our design, it contains four input membership functions and each membership function outputs two corresponding grades when input value is detected. Fig. 29 shows the membership functions of inputs. As can be seen from

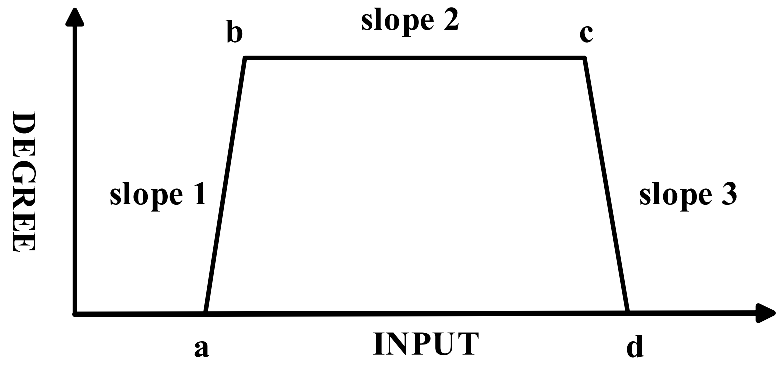


Figure 30. Formation of the membership function

the figure, all the universe of discourses and the membership grades are limited from X"00" to X"FF", which makes the VHDL coding possible. "NL", "NS", "Z", "PS" and "PL" represents "negative large", "negative small", "zero", "positive small" and "positive large". Fig. 30 shows the formation of the membership function. Here, the trapezoidal type is studied. The formation

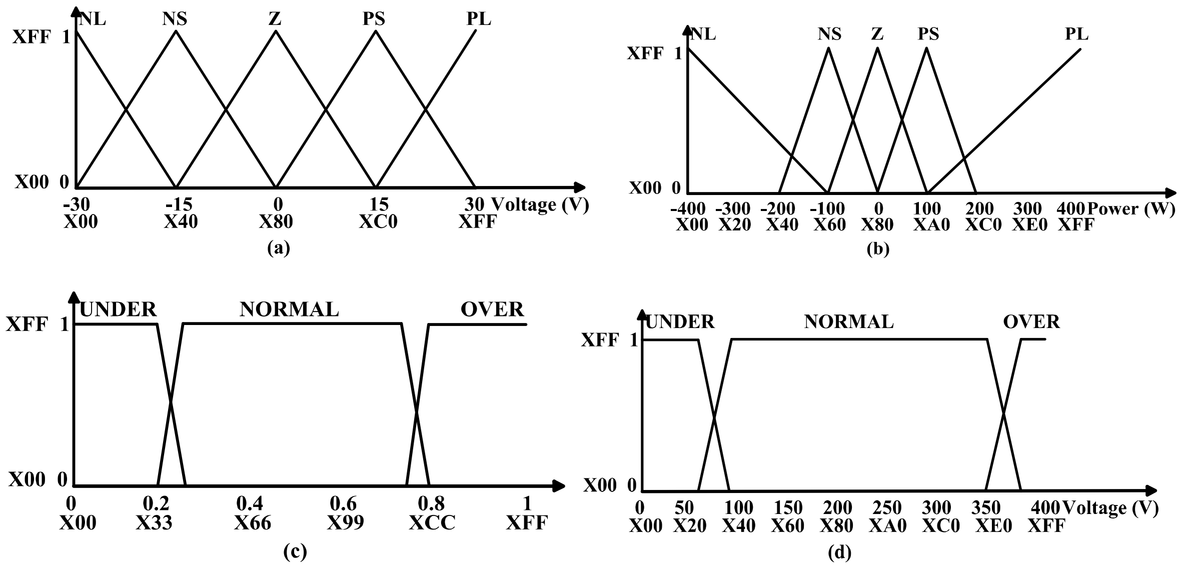
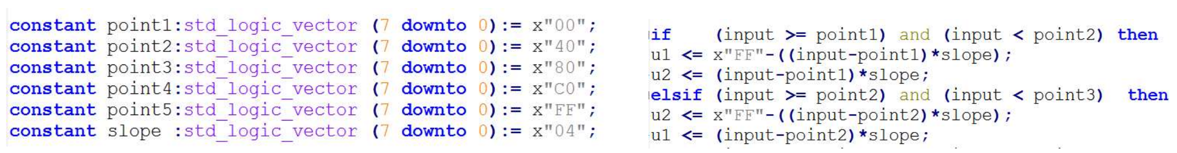


Figure 29. The membership functions of inputs: (a) Vbuserr, (b) Pdem, (c) SOC, (d) Ucap

procedures can be divided into two steps. First step is to define the location of the points (a, b, c and d). Specifically, if the point b and c are at the same position, then it becomes the triangle type. The second step is to define the corresponding degree of the membership when input detected. For example, if the input is located on a-b section. The grade of this input is defined as below:

$$grade = X\text{FF} - (input_{value} - a) * slope1 \tag{6}$$

The core code of forming the membership function is shown in Fig. 31.



```
constant point1:std_logic_vector (7 downto 0):= x"00";
constant point2:std_logic_vector (7 downto 0):= x"40";
constant point3:std_logic_vector (7 downto 0):= x"80";
constant point4:std_logic_vector (7 downto 0):= x"C0";
constant point5:std_logic_vector (7 downto 0):= x"FF";
constant slope :std_logic_vector (7 downto 0):= x"04";
```

```
if    (input >= point1) and (input < point2) then
u1 <= x"FF"-((input-point1)*slope);
u2 <= (input-point1)*slope;
elsif (input >= point2) and (input < point3)  then
u2 <= x"FF"-((input-point2)*slope);
u1 <= (input-point2)*slope;
```

Figure 31. VHDL Code for membership funtion's position definition (left) and degree definition (right)

The RTL of the fuzzifier is presented in Fig. 32, the four inputs value and the eight membership grades value will be collected by the rule evaluator.

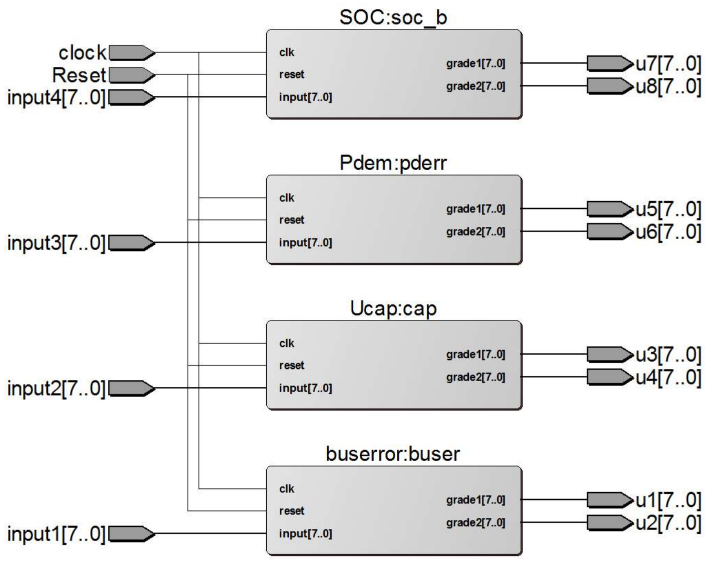
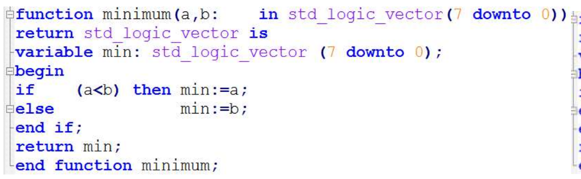


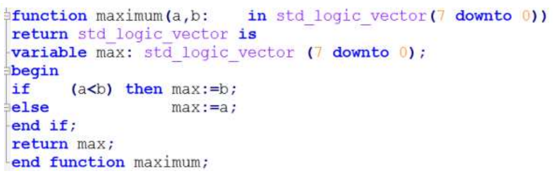
Figure 32. The architecture of fuzzifier

4.2.2 Rule Evaluator

The Rule evaluator is the second stage of the fuzzy logic controller. It contains a rule base with 20 rules. Here, AND and NOT operators are used for the antecedent of rules. Thus the "find max" function and "find min" function should be built at first place for the rule evaluator. The "not" function is a build-in function and can be called directly. The core code of "find max" and "find min" functions is shown in Fig. 33. The core code for building IF-THEN rules is shown in Fig. 34.

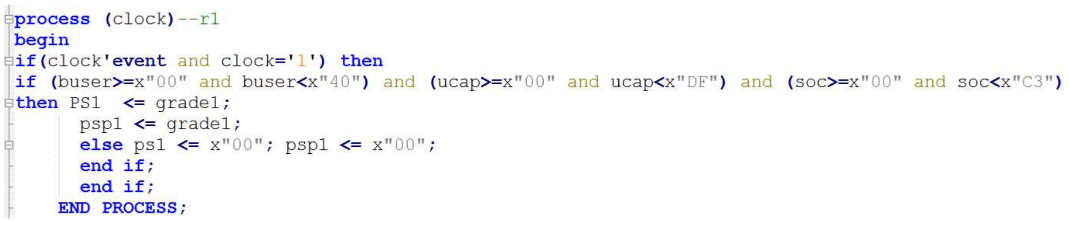


```
function minimum(a,b:    in std_logic_vector(7 downto 0))
return std_logic_vector is
variable min: std_logic_vector (7 downto 0);
begin
if    (a<b) then min:=a;
else             min:=b;
end if;
return min;
end function minimum;
```



```
function maximum(a,b:    in std_logic_vector(7 downto 0))
return std_logic_vector is
variable max: std_logic_vector (7 downto 0);
begin
if    (a<b) then max:=b;
else             max:=a;
end if;
return max;
end function maximum;
```

Figure 34. VHDL Code for "find max" function (left) and "find min" functrion (right)



```
process (clock)--r1
begin
if(clock'event and clock='1') then
if (buser>=x"00" and buser<x"40") and (ucap>=x"00" and ucap<x"DF") and (soc>=x"00" and soc<x"C3")
then PS1  <= grade1;
       psp1 <= grade1;
       else ps1 <= x"00"; psp1 <= x"00";
       end if;
       end if;
     END PROCESS;
```

Figure 33. VHDL Code for IF-THEN rules

Here, each rule is carried out by each process and each process is triggered by the same system clock. In that case, all the rules can be evaluated simultaneously, which is the key characteristic of FPGA chip. The outputs of each rule will be compared in a specific aggregation method. The RTL of rule evaluator is shown in Fig. 35.

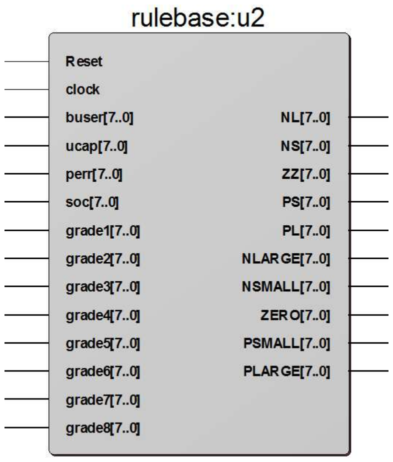


Figure 35. The architecture of rule evaluator

4.2.3 Defuzzification

The defuzzifier is the third stage of the fuzzy logic controller. It converts the fuzzy values into a crisp value for output. In Chapter 2, the COG method is utilized for defuzzification. The COG method can make full use of the information of fuzzy sets and it is a comprehe nsive consideration. However, the drawback of this method is that the integral operator will take up too much resources of FPGA chip and lower the calculation speed. Therefore, an advanced method called Weighted Average (WA) is used for our study, as shown in Fig. 36. The equation for WA method is expressed as (7).

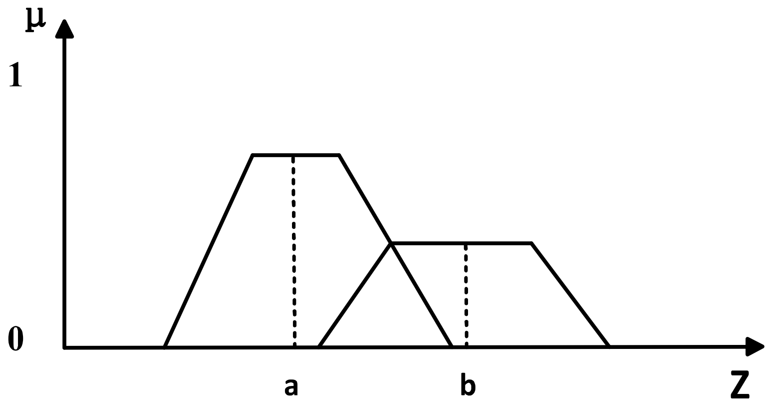


Figure 36. The concept of MA method

$$MA = \frac{\sum_{i=1}^{no.\ of\ rules} grade_i \times center\ of\ grade_i}{\sum_{i=1}^{no.\ of\ rules} grade_i} \tag{7}$$

This method only involves additive and division, which makes the VHDL coding more applicable. The only limitation is that it is valid for symmetrical output membership functions, as shown in Fig. 37.

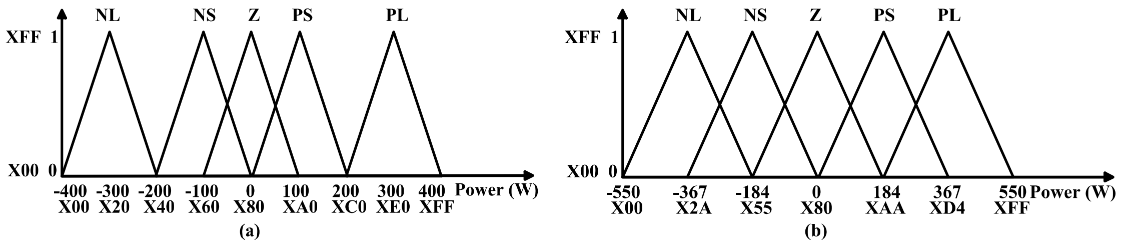
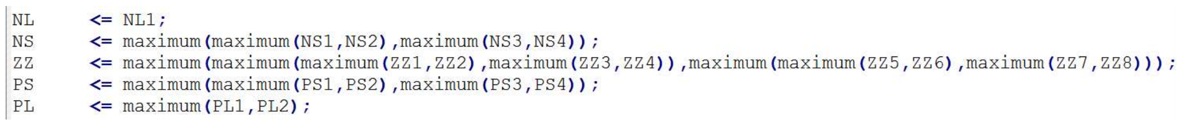


Figure 37. The membership functions of outputs: (a) Pbat, (b) Pcap

The defuzzifier contains one aggregator and two dividers. The method for aggregation is MAX method and two dividers output “Pbat” and “Pcap” respectively. The core code for aggregation is shown in Fig. 38. The RTL of defuzzifier is shown in Fig. 39.



```
NL      <= NL1;
NS      <= maximum(maximum(NS1,NS2),maximum(NS3,NS4));
ZZ      <= maximum(maximum(maximum(ZZ1,ZZ2),maximum(ZZ3,ZZ4)),maximum(maximum(ZZ5,ZZ6),maximum(ZZ7,ZZ8)));
PS      <= maximum(maximum(PS1,PS2),maximum(PS3,PS4));
PL      <= maximum(PL1,PL2);
```

Figure 38. VHDL Code for aggregation

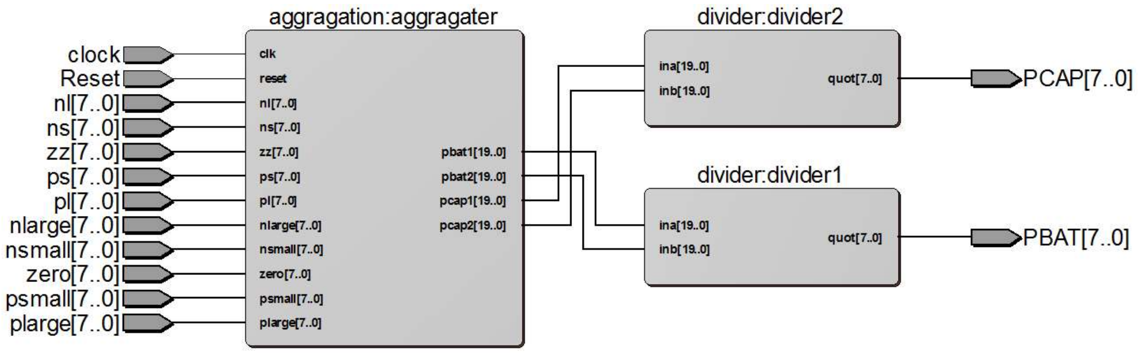


Figure 39. The architecture of defuzzifier

4.2.4 Top level of fuzzy system

After done the individual module design, the final step is to connect them together. Fig.40 shows

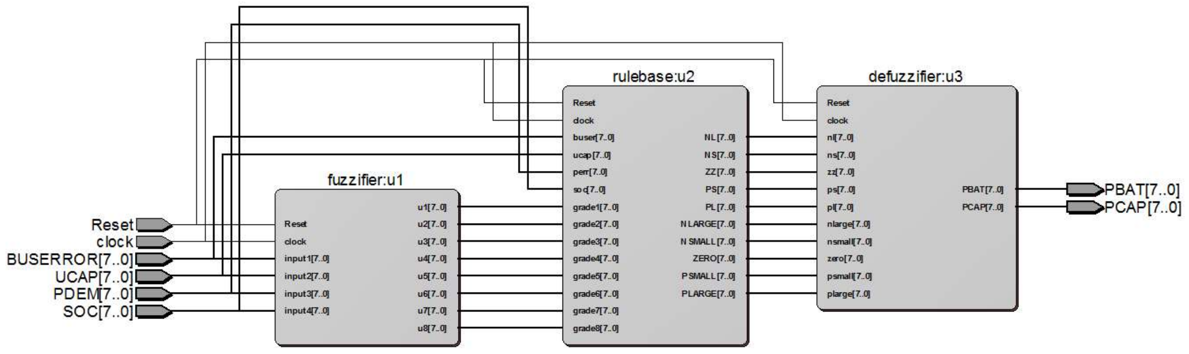


Figure 41. The top-level architecture of fuzzy interface

the top-level representation of the fuzzy system. The VHDL code is successfully synthesized. The Modelsim simulation software is used to verify the digital realization of proposed fuzzy logic strategy. Here, several sets of inputs value which based on each scenario are selected. Fig.41 shows the FPGA simulation results. The comparison between Matlab results and FPGA results are listed in table IV ("M" means "Matlab", "F" means "FPGA"). The max error is 3%, which is a reasonble value.

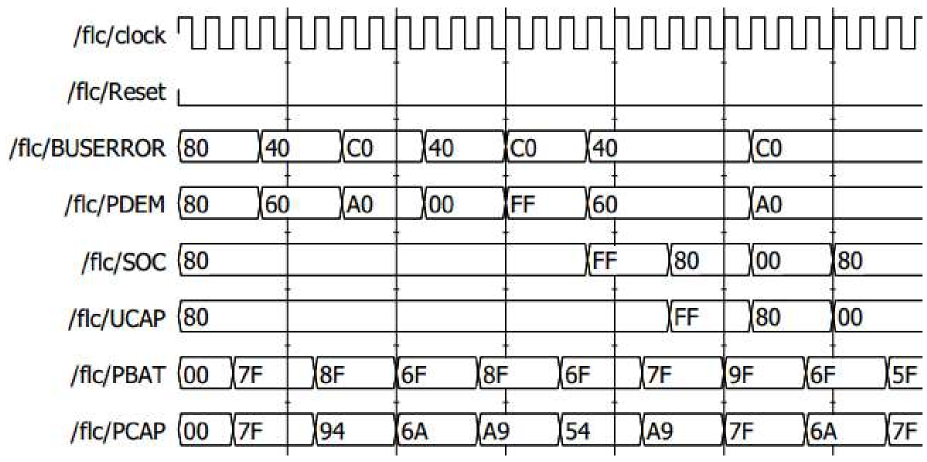


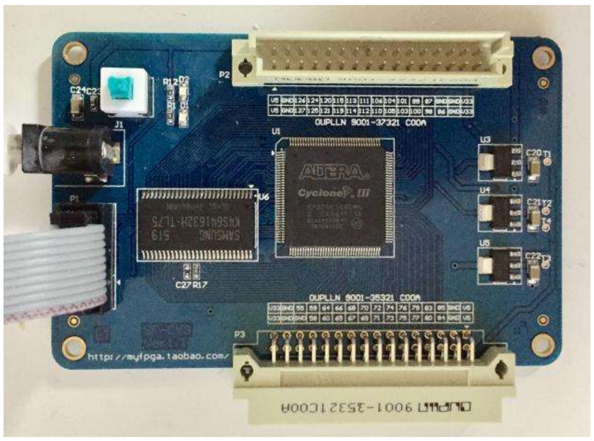
Figure 40. Simulation results of FPGA

TABLE IV. THE COMPARISION RESULTS

Inputs				**Outputs**				**Error (%)**	
buserr	*pdem*	*SOC*	*Ucap*	*Pbat (M)*	*Pbat (F)*	*Pcap (M)*	*Pcap (F)*	*Pbat*	*Pcap*
0	0	0.5	200	0	0	0	0	0	0
-15	-100	0.5	200	49.9	50	91.6	90	0.2	1.7
30	100	0.5	200	-49.9	-50	-91.6	-89	0.2	2.8
-15	-400	0.5	200	49.9	50	183	180	0.2	1.6
15	400	0.5	200	-49.9	50	-183	-180	0.2	1.6
-15	-100	1	200	0	0	183	180	0	1.6
-15	-100	0.5	400	100	103	0	0	3	0
15	100	0	200	0	0	-183	-180	0	1.6
15	100	0.5	0	-100	-103	0	0	3	0

4.3 FIL simulation results

In Chapter 2, the proposed fuzzy logic power management strategy has been verified. In this section, the FIL simulation results are presented. Fig. 42 is the EP3C5E144C8 board used for FIL simulation and the right figure represents the FIL block in Simulink. The communication between computer and FPGA chip is achieved by JTAG interface. Similar to the simulation in Chapter 2,



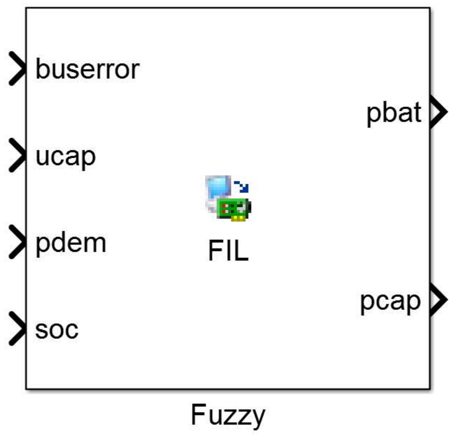


Figure 42. the FPGA board (left) and the FIL block in Simulink (right)

five scenarios have been studied here: 1. When SOC of the batteries and the voltage of UC are below normal status; 2. When the batteries are overcharged; 3. When the batteries are over discharged; 4. When the UC is over discharged; 5. When the UC is overcharged. The assumptive “Pload” and “Ppv” profiles, which are based on 1000s-scale, are generated to simulate the actual

PV power characteristics. The results are shown in Fig. 43-47, which indicated that the difference between Matlab Simulation results and FPGA results is minors (see Chapter 2 for comparison). Thus, the digital realization of proposed fuzzy logic power management strategy has been done.

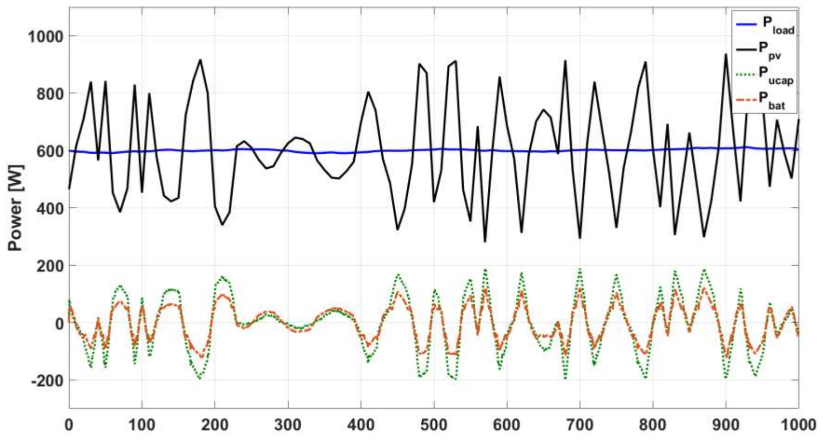


Figure 43. the power distribution when two sources are under normal status (FIL)

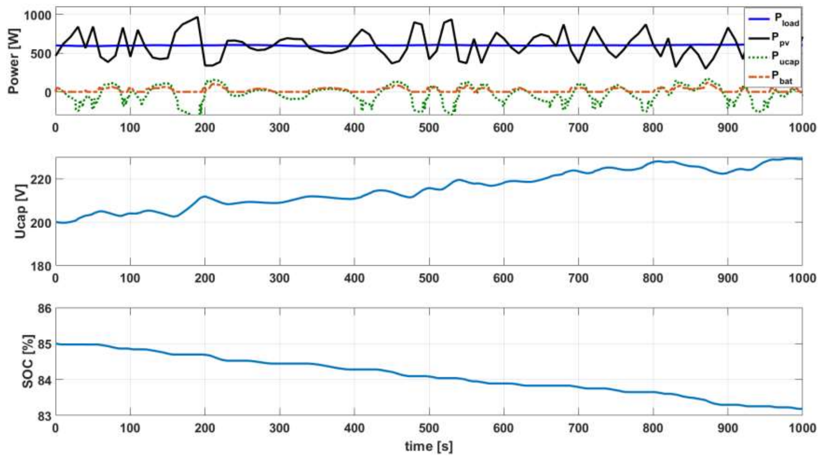


Figure 44. the power distribution when batteries are over-charged (FIL)

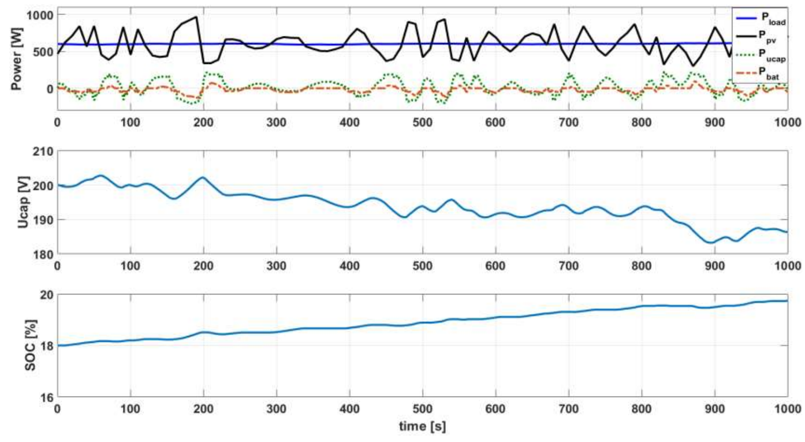


Figure 45. the power distribution when batteries are over-discharged (FIL)

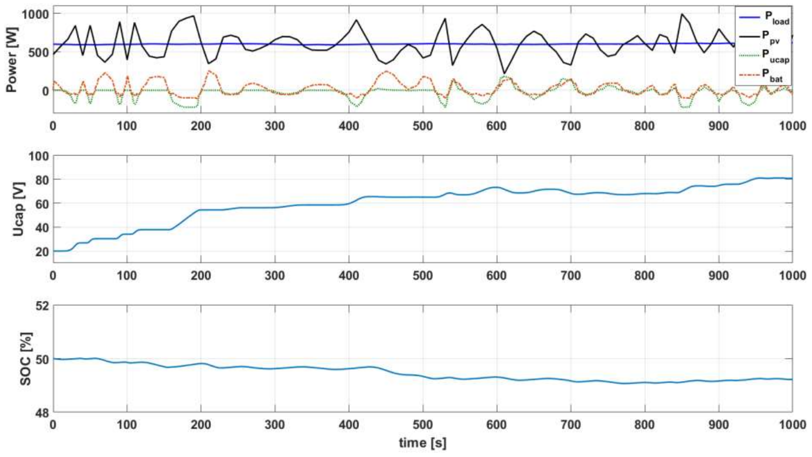


Figure 46. The power distribution when UC are over-discharged

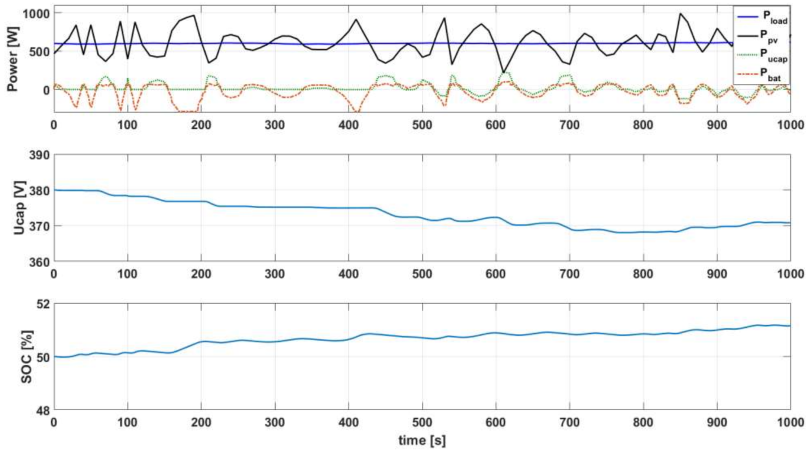


Figure 47. The power distribution when UC are over-charged

Chapter 5 The Model-based FOC design using FPGA chip

5.1 Scheme of Simulink model-based design

When utilizing FPGA technology, programming is unavoidable for researchers and engineers. The traditional workflow of the design using hand-coding to program the FPGA chip, which is time-consuming and tedious. Besides, the debugging process is annoying and sometimes requires simulation tool (Modelsim) to do the verification. Thus, the automatic code generation feature is among the top concern for developers and designers.

HDL Coder is an Add-on which provided by Matlab. It can generate portable, synthesizable Verilog and VHDL code from MATLAB functions, Simulink models, and Stateflow charts. The generated HDL code can be used for FPGA programming or ASIC prototyping and design [41]. In Chpater 3 we introduced the FIL function for direct communication between FPGA chip and Simulink. With HDL Coder, we can even build the model of algorithm in Simulink and automatically generate the HDL code. In this case, all the design procedures can be carried out in Matlab/Simulink environment, which achieves the seamless design flow. In addition, to reduce the resource utilization and improve the performance, HDL Coder provide us the optimization method such as pipeline optimization. This method can guarantee the time is constrained and the speed requirement is met.

It needs to be mentioned that HDL Coder only support fixed-point data type. Therefore, we have to convert the floating-point model into fixed-point model by using Fixed-point Designer. The goal of Fixed-point Designer is to determine the proper word length and fraction length for data stream.

In short, by using HDL Coder (or similar tools) can greatly shrink down the time span of the design and put more emphasis on algorithm optimization and parameters tuning process, which free programmers from hand-coding.

5.2 PMSM Modeling

A mathematical model of the PM in the rotor reference frame is described by (1),

$$\begin{cases} V_{ds} = R_s i_{ds} + L_d \dfrac{di_{ds}}{dt} - \omega_r L_q i_{qs} \\ V_{qs} = R_s i_{qs} + L_q \dfrac{di_{qs}}{dt} + \omega_r L_d i_{ds} + \omega_r \psi_m \end{cases} \tag{8}$$

where V_{ds}, V_{qs}, i_{ds}, i_{qs} are the stator d and q axis voltage, current in the rotor reference frame, respectively. R_s stands for the stator resistance, L_d, L_q stand for the d and q axis inductance, ω_r is the angular velocity, and ψ_m is the permanent magnet flux leakage.

The current loop control of PMSM drive based on a FOC approach. If i_d is forced to 0, the PMSM will be decoupled and controlling a PMSM is like controlling a DC motor. After decoupling, the motor's torque is proportional to i_q

$$T_e = \frac{3N_p}{4} \lambda_f i_q \tag{9}$$

5.3 Structure of FOC

FOC, also called Vector Control, is a variable frequency drive (VFD) control method that the stator currents are treated as two orthogonal components by de-coupling process [42]. One of the components is the torque of the motor and another is the magnetic flux. By controlling the two components independently, the speed or the torque can be regulated. Proportional-Integral (PI) controller is utilized to controlling the speed and the torque. Typically, the dual close loop is utilized for speed mode control, in which the inner loop is torque loop and the outer loop is the speed loop. Basically, The FOC control of PMSM contents four stages: Clarke Transform, Park Transform, Inverse Park Transform and Space-Vector PWM (SVPWM), which is shown in Fig. 48. The quadrature encoder interface is used for position and speed feedback.

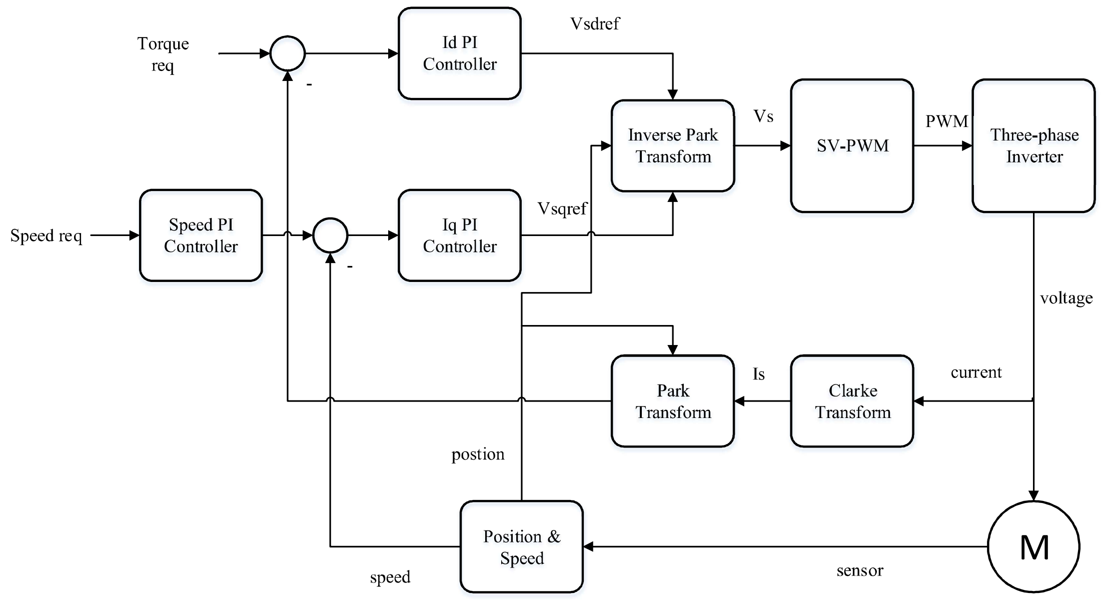


Figure 48. The structure of FOC

5.3.1 Clarke Transform

Clarke transform, also called abc-αβ transform, is the first stage of the FOC. It projects the three - phase current into two stationary d and q axis. The relationship between input currents and output currents is shown in (8).

$$\begin{bmatrix} i\alpha \\ i\beta \end{bmatrix} = \frac{\sqrt{2}}{\sqrt{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} ia \\ ib \\ ic \end{bmatrix} \tag{10}$$

In Simulink, it is convenient to build this mathematical model by using Constant, Product and Add components in HDL Coder tool box. The Clarke transform model can be seen in Fig. 49.

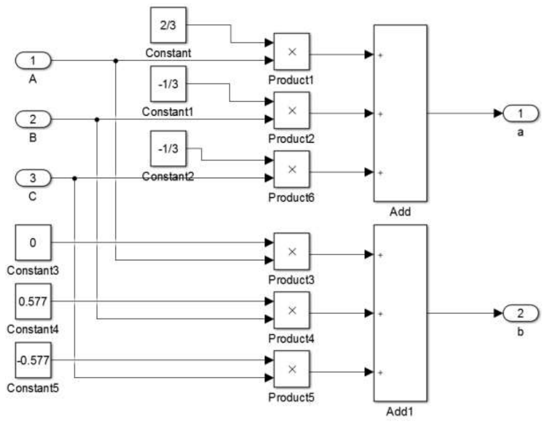


Figure 48. The model of Clarke Transform

5.3.2 Park Transform

Park Transform is the second stage of FOC, the goal of Park transform is to transfer the stationary d and q axis into rotary d and q axis without changing the angle relationship between them. The expression of this transform is shown as (9).

$$\begin{bmatrix} i\mathrm{d} \\ i\mathrm{q} \end{bmatrix} = \begin{bmatrix} \cos\theta & sine\ \theta & 0 \\ -sine\ \theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} i\alpha \\ i\beta \end{bmatrix} \tag{11}$$

In order to implement the Park Transform in Simulink, besides the Product and Add math operators, the Trigonometric Function has to be used for axis rotation. The range of this function is from 0 to 2π in radius unit.

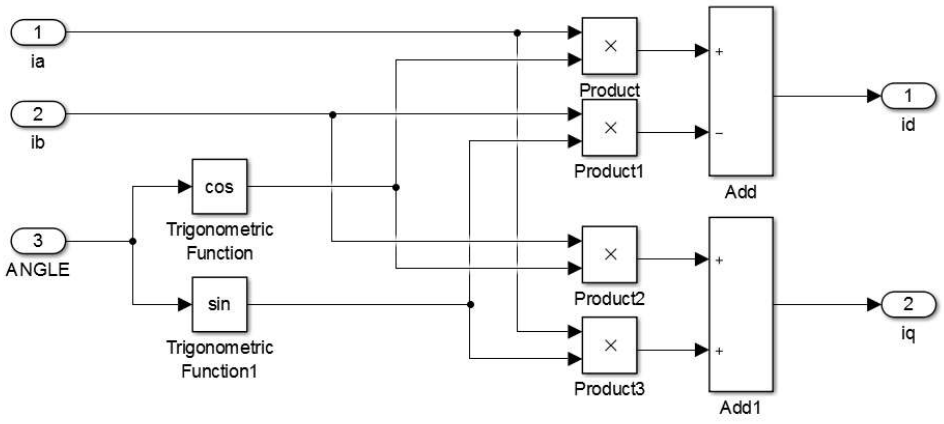


Figure 49. The model of Park Transform

5.3.3 Inverse Park Transform

Inverse Park Transform, also called αβ-abc Transform, is the third stage of FOC. The goal of this stage is to transfer the stationary-relative rotor into stationary-relative stator by inverse transformation of Park Transform. The equation can be seen in (10).

$$\begin{bmatrix} Va \\ Vb \end{bmatrix} = \begin{bmatrix} \cos\theta & sine\ \theta & 0 \\ -sine\ \theta & \cos\theta & 0 \end{bmatrix}^{-1} \begin{bmatrix} id \\ iq \end{bmatrix} \tag{12}$$

The model for Invert Park Transform is similar to Park Transfrom, as shown in Fig. 51.

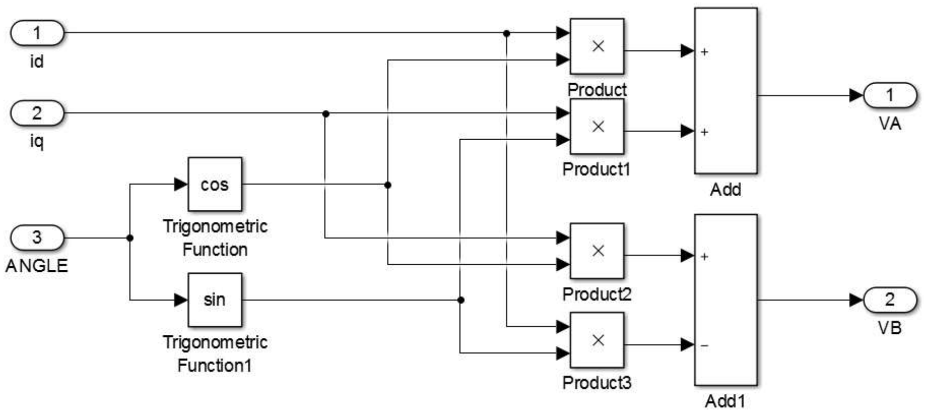


Figure 50. The model for Inverse Park Transform

5.3.4 SVPWM

SVPWM is the fourth stage of FOC, which uses the principle of mean equivalency. In one switching period, the average voltage is equal to the given voltage vector by combining the basic voltage vectors. At the specific point, the voltage vector rotates into an area that vector can be obtained in different combinations between two adjacent non-zero vectors and zero vectors. The action time of the two vectors is applied several times in a sampling period to control the action time of each voltage vector. In this case, each voltage vector is controlled so that the voltage space vector rotates in a way close to the circle trajectory. Furthermore, the actual magnetic flux generated by the different switching states of the inverter can approach the ideal flux circle. The goal of this stage is to calculate the duty cycle of three upper bridge PWM of inverter. The HDL model is shown in Fig. 52. The calculated duty cycle will be sent to the PWM module for generating PWMs.

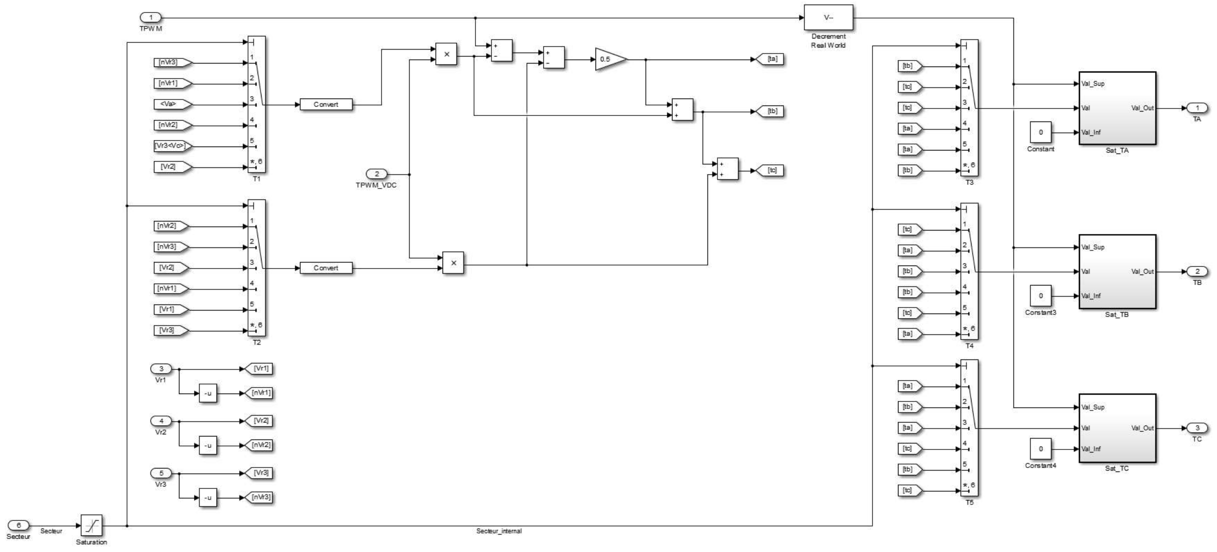


Figure 51. The model of SVPWM

5.4 Simulation Results

Before generating the Verilog code for PFGA chip, the performance of FOC algorithm model should be test. The simulation results are presented here. The top level of the model is shown in Fig. 53. The inputs are position information, speed information, three-phase current feedback. The outputs are 6 PWMs. Fig. 54 shows the three-phases feedback current. The glitch at the beginning is due to the transient state. Fig. 55 shows the calculated duty cycle. The range of modulation is from 0 to 1000. However, the duty-cycle cannot be too large or too small in case of pulse PWM signal. Therefore, the modulation range is limited. Fig. 56 shows the stator current of id and iq, where id represents the flux, iq represents the torque. Based on the simulation results, the FOC algorithm model is successfully built.

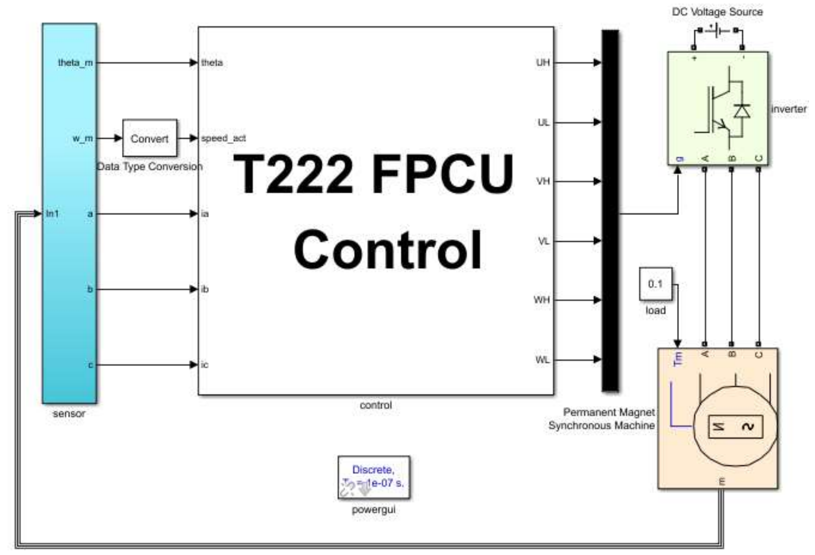


Figure 52. The top level of the FOC model

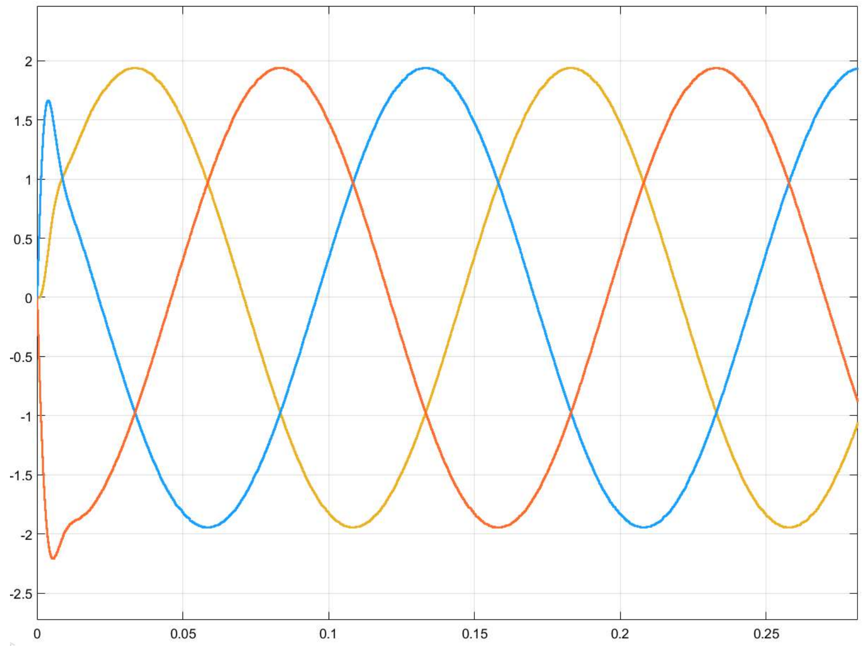


Figure 53. The three-phase current feedback

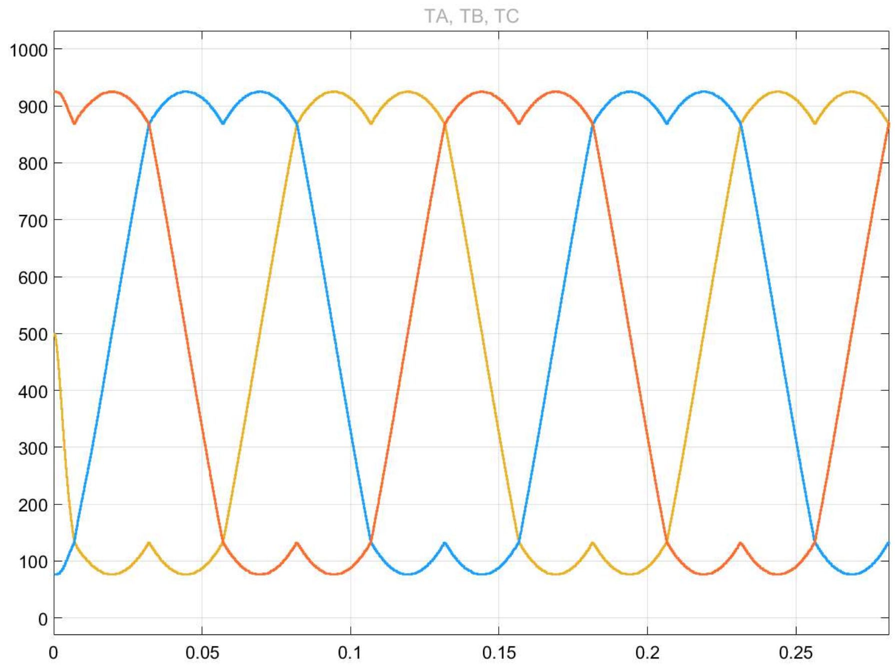


Figure 54. The calculated duty-cycle

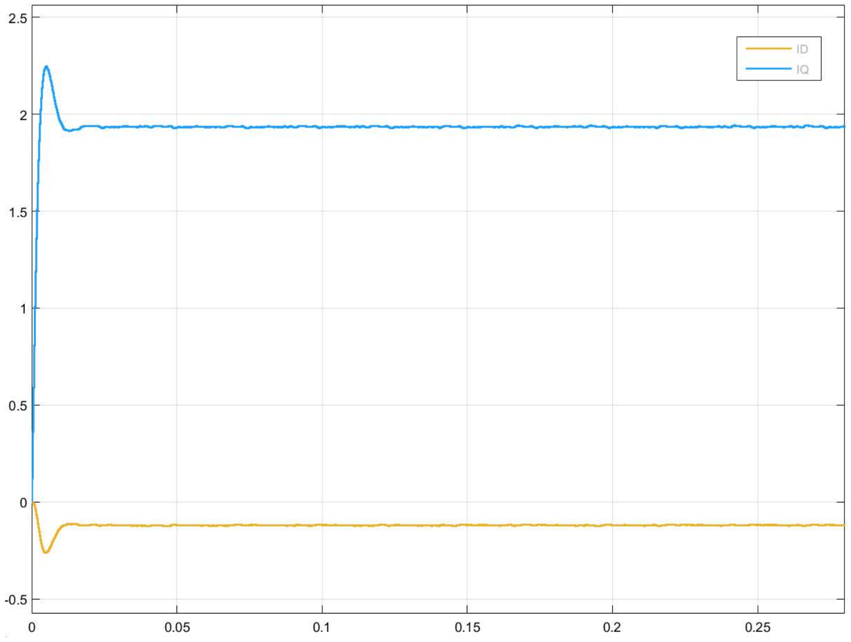


Figure 55. The stator current

5.5 Hardware Implementation Results

In order to evaluate the performance of this model-based FOC algorithm design, the hardware implementation results are presented in this section. The evaluation board for this thesis is T222 Starter Kit from Silicon Mobility, as shown in Fig. 57.

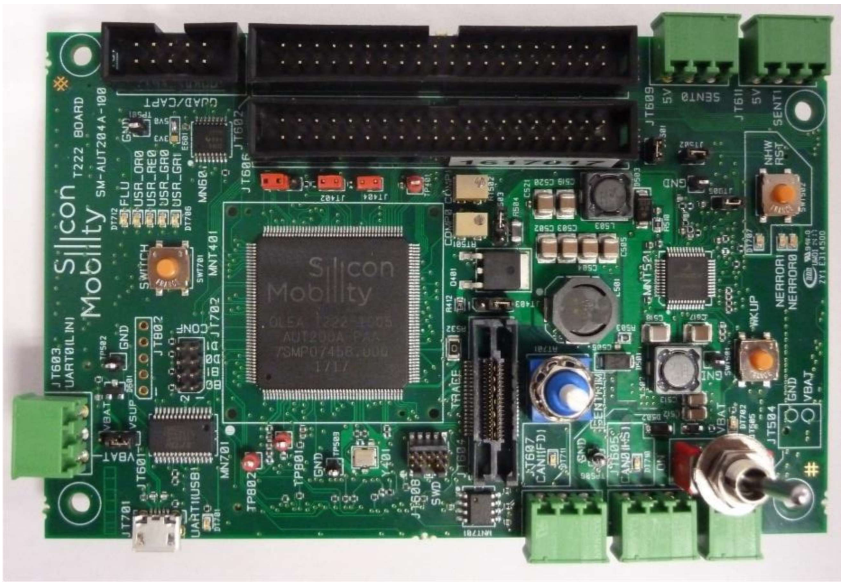


Figure 56. T222 Starter Kit

5.5.1 Resolver to Digital Converter

Since the PMSM we used is a resolver-type motor, a resolver to digital converter (RDC) is needed in our design. Besides, since T222 starter kit only has quadrature encoder interface, the position information should be convert in to phase A, phase B and Index signal for T222 starter kit. Therefore, the EVAL-AD2S1210SDZ evaluation board which from ANALOG DEVICES is utilized for RDC. The functional block diagram of AD2S1210 chip is shown in Fig. 58. The reference oscillator generates excitation signal to the resolver of PMSM, then resolver will generate sine and cosine signals to AD2S1210 chip. After done the calculation of sine and cosine signal, the position register will send the position information to encoder emulator. Through the encoder emulation process, the A, B and Index signal for encoder interface will be generated. Fig. 59 shows the waveform of Encoder signal.

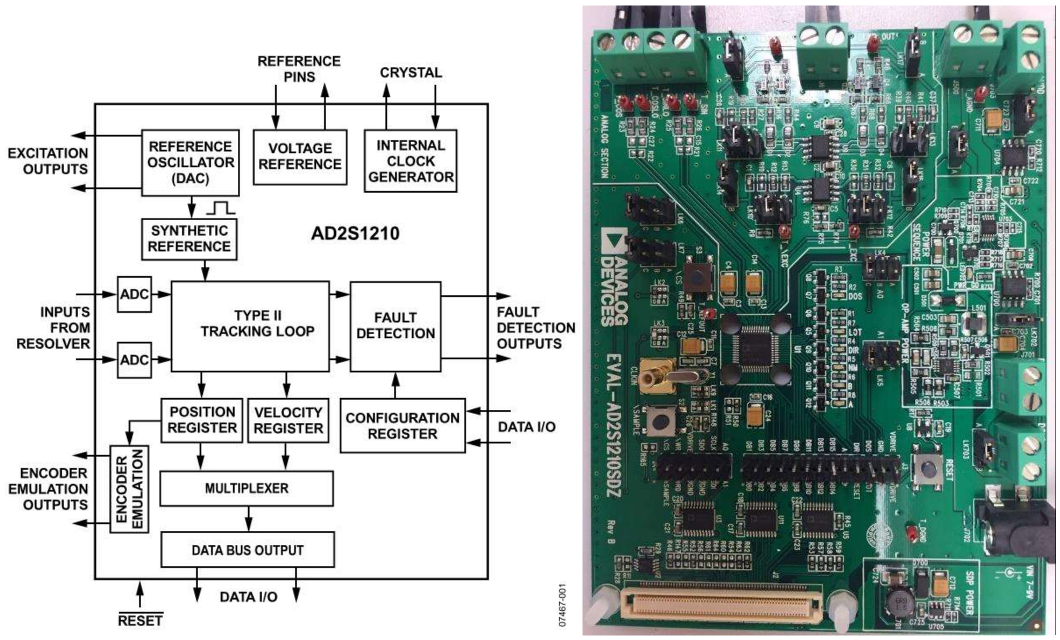
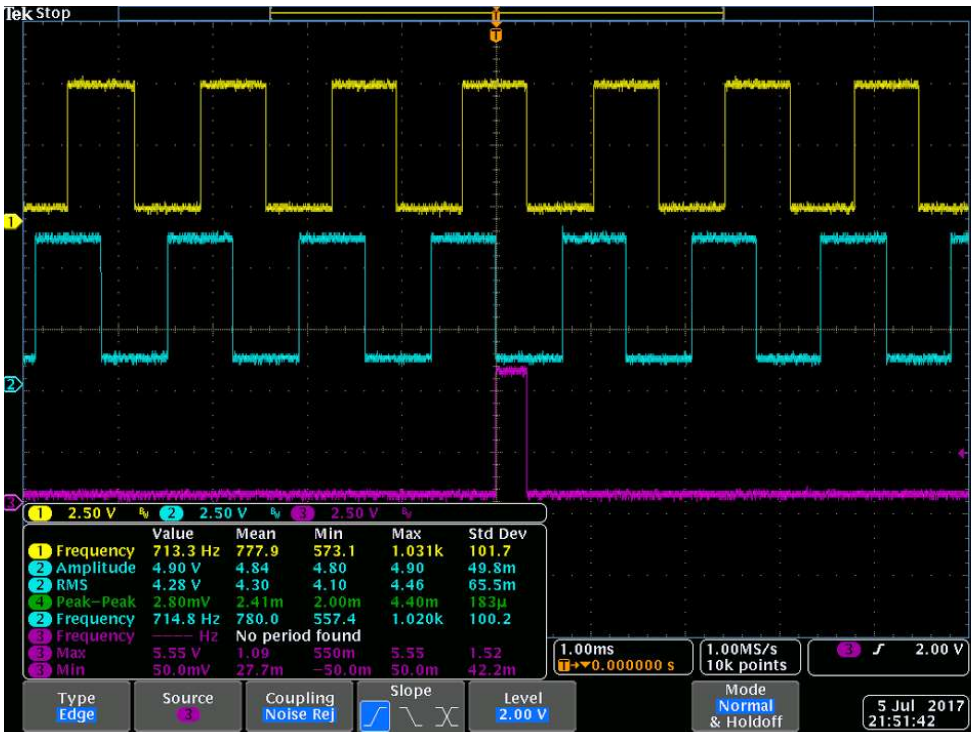


Figure 57. The functional block diagram of AD2S1210 (left) and the evaluation board (right)

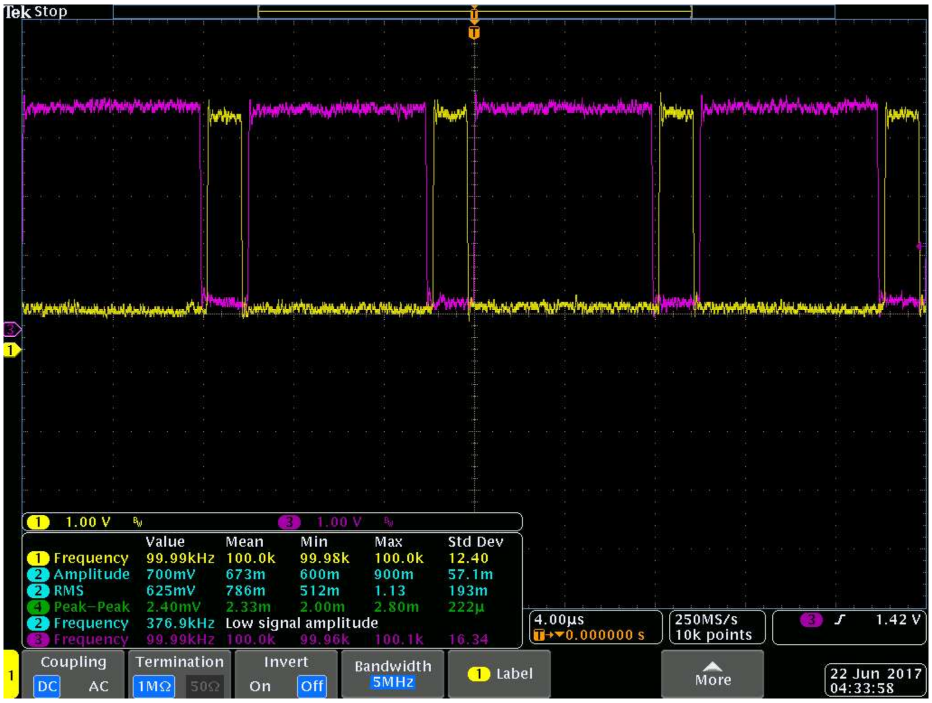


```
                Value     Mean   Min     Max     Std Dev
1 Frequency     713.3 Hz  777.9  573.1   1.031k  101.7
2 Amplitude     4.90 V    4.84   4.80    4.90    49.8m
2 RMS           4.28 V    4.30   4.10    4.46    65.5m
4 Peak-Peak     2.80mV    2.41m  2.00m   4.40m   183µ
2 Frequency     714.8 Hz  780.0  557.4   1.020k  100.2
3 Frequency     ---- Hz   No period found
3 Max           5.55 V    1.09   550m    5.55    1.52
3 Min           50.0mV    27.7m  -50.0m  50.0m   42.2m
```

Figure 58. Phase A (yellow), Phase B (blue) and Index (red) signal

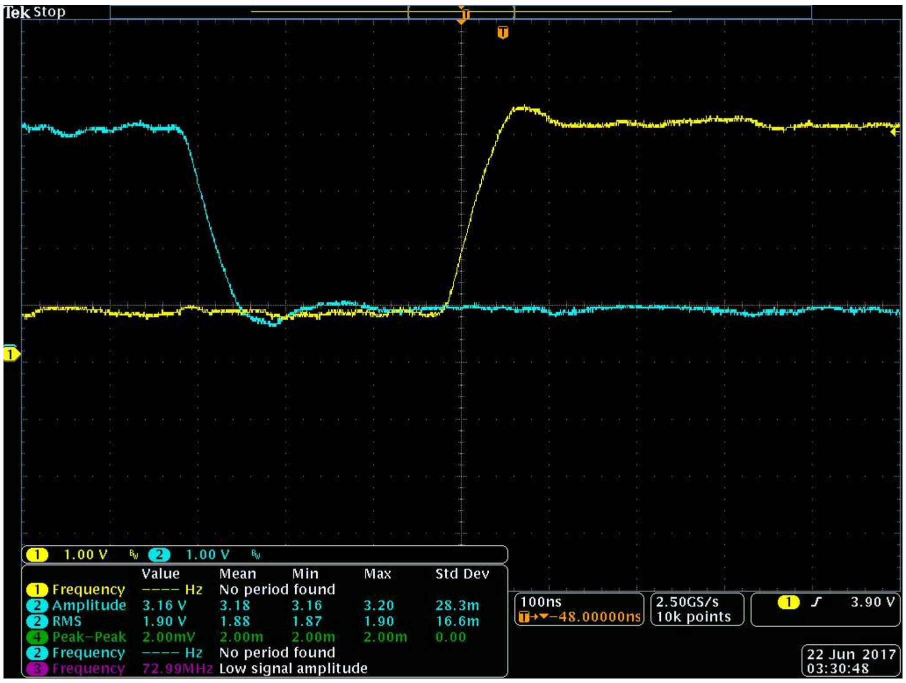
5.5.2 T222 Starter Kit implementation results

Since one of our goals is to achieve 100 kHz switching frequency, the PWM output signal is among the top concern. Here, the SiC-based three-phase inverter is utilized. It is worth to point out that the dead-band time for SiC devices is required. Fig. 60 shows the complementary PWM signal for one inverter bridge. From the figure, we can observe that the frequency is 100 kHz. Fig. 6 1 shows the zoom-in figure of the dead-band time, the dead-band time is set to be 300 ns, which meet the requirements. Fig. 62 shows the three-upper bridge PWM, which are center aligned.



	Value	Mean	Min	Max	Std Dev
1 Frequency	99.99kHz	100.0k	99.98k	100.0k	12.40
2 Amplitude	700mV	673m	600m	900m	57.1m
2 RMS	625mV	786m	512m	1.13	193m
4 Peak−Peak	2.40mV	2.33m	2.00m	2.80m	222µ
2 Frequency	376.9kHz	Low signal amplitude			
3 Frequency	99.99kHz	100.0k	99.96k	100.1k	16.34

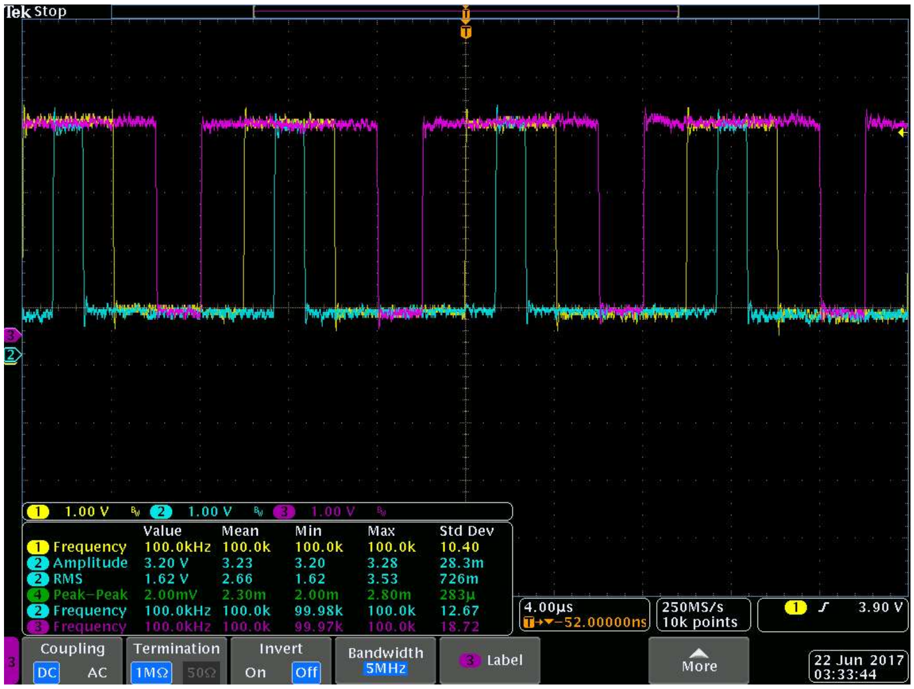
Figure 59. Complementary PWM signal



```
              Value     Mean      Min     Max     Std Dev
1 Frequency   ---- Hz   No period found
2 Amplitude   3.16 V    3.18      3.16    3.20    28.3m
2 RMS         1.90 V    1.88      1.87    1.90    16.6m
4 Peak-Peak   2.00mV    2.00m     2.00m   2.00m   0.00
2 Frequency   ---- Hz   No period found
3 Frequency   72.99MHz  Low signal amplitude

100ns           2.50GS/s      1 ∫ 3.90 V
T→▼-48.00000ns  10k points
                              22 Jun 2017
                              03:30:48
```

Figure 60. The zoom-in figure of dead-band time



```
              Value     Mean      Min     Max     Std Dev
1 Frequency   100.0kHz  100.0k    100.0k  100.0k  10.40
2 Amplitude   3.20 V    3.23      3.20    3.28    28.3m
2 RMS         1.62 V    2.66      1.62    3.53    726m
4 Peak-Peak   2.00mV    2.30m     2.00m   2.80m   283μ
2 Frequency   100.0kHz  100.0k    99.98k  100.0k  12.67
3 Frequency   100.0kHz  100.0k    99.97k  100.0k  18.72

4.00μs          250MS/s       1 ∫ 3.90 V
T→▼-52.00000ns  10k points
                              22 Jun 2017
                              03:33:44
```

Figure 61. Three upper bridge PWM

Fig. 63 shows the implementation results of torque mode close loop control. The first row shows the three-phase current. They have 120-degree phase shift with each other. The peak to peak value of current is 6 A. The second row shows the position feedback of PMSM. Here, a 16-bit counter is utilized to represent the position information. Here, counter up means the inverse direction and counter down means the forward direction. The third row shows the output of flux PI controller and torque PI controller. Since we only need to control torque, the reference value of flux is set zero. The behavior of Vq is corresponding to the position information, which means when the torque command is positive, the motor run at forward direction and when the torque command is negative, the motor run at inverse direction. The last row shows the duty cycle for each PWM generator.

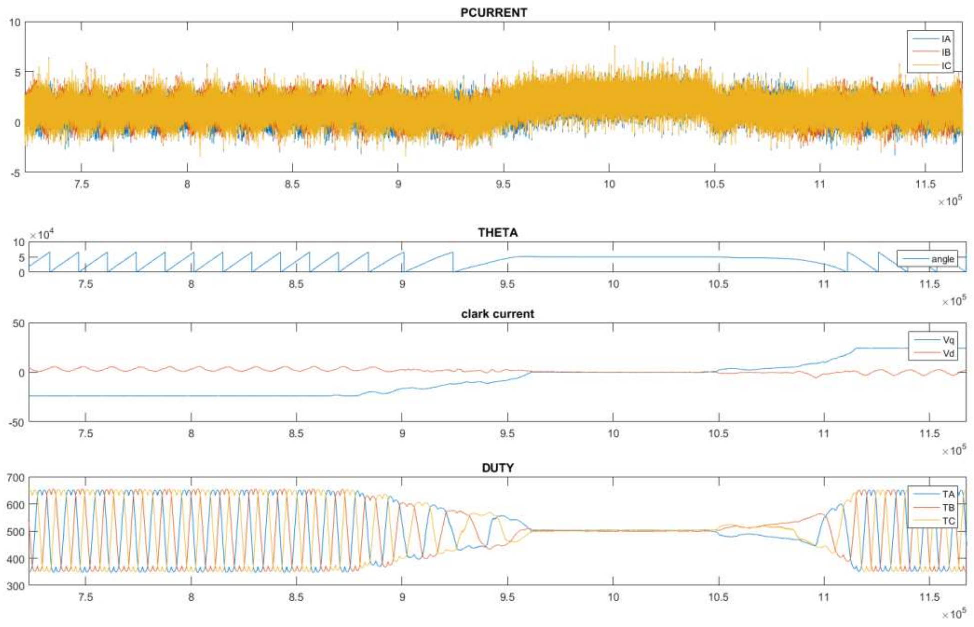


Figure 62. Results of torque mode close loop control

Chapter 6 Conclusions and Future Work

6.1 Conclusions

This thesis discussed and addressed several issues regarding the HESS power management strategy and FPGA implementations. The following shows the details:

1. A global-level fuzzy-logic based power management strategy for a standalone PV system with hybrid energy storage devices is presented, which has the following targets: 1) smooth out the PV power; 2) optimize the power distribution between the battery and the UC; 3) extend the battery lifespan. Furthermore, system configurations of stand-alone PV systems with or without HESS are reviewed and compared in depth. Control algorithms for the power management strategy are reviewed and compared. Simulation results are given to evaluate the performance of proposed strategy.

2. In order to realize optimal power distribution in double-sources energy storage system of HEVs or PHEVs, a fuzzy logic control strategy is proposed. In order to evaluate the performance of the proposed energy management strategy, the simulation models are developed. The models are tested under several typical driving cycles provided by US EPA. Furthermore, in order to evaluate the influence of the proposed control strategy on the battery life, the temperature-related battery degradation model is implemented. The results show that the proposed fuzzy logic control can effectively reduce battery degradation to 17 percentages.

3. The fuzzy logic power management algorithm has been successfully implemented in EP3C5E144C8 FPGA with the purpose of fast and reliable performance. The VHDL code for fuzzy logic algorithm is designed. FIL verifications have been provided to validate the effectiveness of the proposed control algorithm. The comparison between Modelsim simulation results and FPGA results is presented to show the effectiveness of digital realization.

4. In order to test the potential of the FPGA chip, a model-based FOC algorithm for PMSM is presented. The HDL Coder tool box is utilized for building the model and generating Verilog code for PFGA. The simulation results verified the functionality and effectiveness of FOC algorithm. The high switching frequency shows the power of FPGA technology and makes the SiC devices possible.

6.2 Future Work

This thesis still has the potential for improvement. Researchers who are interested in HESS and FPGA technology should take the following aspects into consideration.

1. **Sizing issue**. In chapter 2, the fuzzy logic power management strategy for HESS is presented. It is no doubt that the optimal power distribution will minimize the additional cost of battery storage and in result reduce the cost of whole system. Therefore, the sizing issue of HESS should be addressed. By choosing the proper specifications of energy storages (batteries, UCs and PVs), the performance as well as the cost-effectiveness of HESS will be improved.

2. **Race and hazard issue**. In Chapter 3, the digital realization of fuzzy-logic power management strategy is presented. The FIL results show the effectiveness of the design. However, there are some spikes and glitches in the FIL results. Basically, this is caused by race and hazard, which is the behavior of digital circuit where the output is dependent on the sequence or timing of other

uncontrollable events. It becomes a bug when events do not happen in the order the programmer intended. To prevent this issue, the output signal of each design stage should be strictly synchronized. In that case, the timing-constraint tool is required and the gate-level simulation of the digital circuit should be carried out.

3. **Common mode noise issue.** In Chapter 4, the FOC algorithm for PMSM with 100 kHz switching frequency is presented. The high switching frequency can shrink down the size and weight of PMSM without jeopardize the performance. However, when the switching frequency goes up, the common mode noise becomes larger too. Thus, the common mode choke for compensating such noise is required and the noisier the common mode noise, the larger the common mode choke is needed. In that case, the advantage of small PMSM will be neutralized by the big common mode choke. Therefore, the good combination of switching frequency and common mode size will improve the performance of whole system.

Reference

[1] Lewis, A.M.; Kelly, J.C.; Keoleian, G.A., "Evaluating the life cycle greenhouse gas emissions from a lightweight plug-in hybrid electric vehicle in a regional context," in Sustainable Systems and Technology (ISSST), 2012 IEEE International Symposium on , vol., no., pp.1-6, 16-18 May 2012

[2] Rahman, S.A.; Nong Zhang; Jianguo Zhu, "A comparison on fuel economy and emissions for conventional hybrid electric vehicles and the UTS plug-in hybrid electric vehicle," in Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on , vol.5, no., pp.20-25, 26-28 Feb. 2010

[3] Pill-Soo Kim, "Cost modeling of battery electric vehicle and hybrid electric vehicle based on major parts cost," in Power Electronics and Drive Systems, 2003. PEDS 2003. The Fifth International Conference on , vol.2, no., pp.1295-1300 Vol.2, 17-20 Nov. 2003

[4] Ostadi, A.; Kazerani, M., "A Comparative Analysis of Optimal Sizing of Battery-Only, Ultracapacitor-Only, and Battery–Ultracapacitor Hybrid Energy Storage Systems for a City Bus," in Vehicular Technology, IEEE Transactions on , vol.64, no.10, pp.4449-4460, Oct. 2015

[5] J. Cao and A. Emadi, “A new battery/ultra-capacitor hybrid energy storage system for electric, hybrid and plug-in hybrid electric vehicles,” IEEE Trans. Power Electronics, vol. 27, issue 1, pp. 122-132, Dec. 2011.

[6] A. Khaligh and Z. Li, “Battery, ultracapacitor, fuel cell, and hybrid energy storage systems for electric, hybrid electric, fuel cell, and plug-In hybrid electric vehicles: state of the art,” IEEE Trans. Vehicular Technology, vol. 59, no. 6, pp. 2806-2010, Jul. 2010.

[7] C. A. Hill, M. C. Such, D. Chen, J. Gonzalez, and W. M. Grady, “Battery energy storage for enabling integration of distributed solar power generation,” IEEE Trans. Smart Grid, vol. 3, no. 2, pp. 850–857, Jun. 2012.

[8] H. Zhou, T. Bhattacharya, D. Tran, T. S. T. Siew, and A. M. Khambadkone,“Composite energy storage system involving battery and ultracapacitor with dynamic energy management in microgrid applications,” IEEE Trans. Power Electron., vol. 26, no. 3, pp. 923–930, Mar. 2011.

[9] E. Dallago, A. Liberale, D. Miotti and G. Venchi, "Direct MPPT Algorithm for PV Sources W ith Only Voltage Measurements," in IEEE Transactions on Power Electronics, vol. 30, no. 12, pp. 6742-6750, Dec. 2015.

[10] J. Li, Z. Wu, S. Zhou, H. Fu and X. P. Zhang, "Aggregator service for PV and battery energy storage systems of residential building," in CSEE Journal of Power and Energy Systems, vol. 1, no. 4, pp. 3-11, Dec. 2015.

[11] J. M. Miller, "Energy storage system technology challenges facing strong hybrid, plug-in and battery electric vehicles," Vehicle Power and Propulsion Conference, 2009. VPPC '09. IEEE, Dearborn, MI, 2009, pp. 4-10.

[12] D. N. Wong, D. A. Wetz, A. M. Mansour and J. M. Heinzel, "The influence of high C rate pulsed discharge on lithium-ion battery cell degradation," 2015 IEEE Pulsed Power Conference (PPC), Austin, TX, 2015, pp. 1-6.

[13] A. R. Aarathi and M. V. Jayan, "Grid connected photovoltaic system with super capacitor energy storage and STATCOM for power system stability enhancement," Advances in Green Energy (ICAGE), 2014 International Conference on, Thiruvananthapuram, 2014, pp. 26-32.

[14] M. M. Patankar, R. G. Wandhare and V. Agarwal, "A high performance power supply for an Electric Vehicle with solar PV, battery and ultracapacitor support for extended range and enhanced dynamic response," 2014 IEEE 40th Photovoltaic Specialist Conference (PVSC), Denver, CO, 2014, pp. 3568-3573.

[15] L. A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, pp. 338-353, 1965

[16] X. Hao, T. Zhou, J. Wang and X. Yang, "A hybrid adaptive fuzzy control strategy for DFIG-based wind turbines with super-capacitor energy storage to realize short-term grid frequency support," Energy Conversion Congress and Exposition (ECCE), 2015 IEEE, Montreal, QC, 2015, pp. 1914-1918.

[17] S. T. Lee and H. A. F. Almurib, "Control techniques for power converters in photovoltaic hybrid energy storage system," Clean Energy and Technology (CEAT) 2014, 3rd IET International Conference on, Kuching, 2014, pp. 1-6.

[18] N. F. Nik Ismail, N. Hashim and R. Baharom, "A comparative study of Proportional Integral Derivative controller and Fuzzy Logic controller on DC/DC Buck-Boost Converter," 2011 IEEE Symposium on Industrial Electronics and Applications, Langkawi, 2011, pp. 149-154.

[19] D. Zhu, Y. Wang, N. Chang and M. Pedram, "Optimal design and management of a smart residential PV and energy storage system," 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2014, pp. 1-6.

[20] Jianhua Yuan, Feng Gao, Houlei Gao, Hua Zhang and Jiang Wu, "An adaptive control strategy for parallel operated photovoltaic inverters," Power Electronics and Motion Control Conference (IPEMC), 2012 7th International, Harbin, China, 2012, pp. 1522-1526.

[21] Monmasson, E.; Cirstea, M.N., "FPGA Design Methodology for Industrial Control Systems—A Review," IEEE Transactions on Industrial Electronics, vol. 54, no. 4, 2007, pp. 1824-1842.

[22] Y. Bai et al., "FPGA vs DSP: A throughput and power efficiency comparison for Hierarchical Enumerative Coding," 2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC), Istanbul, 2013, pp. 318-321.

[23] M. Cirstea, J. Khor and M. McCormick, "FPGA fuzzy logic controller for variable speed generators," Control Applications, 2001. (CCA '01). Proceedings of the 2001 IEEE International Conference on, Mexico City, 2001, pp. 301-304.

[24] L. Bouselham, M. Hajji, B. Hajji, A. E. Mehdi and H. Hajji, "Hardware implementation of fuzzy logic MPPT controller on a FPGA platform," 2015 3rd International Renewable and Sustainable Energy Conference (IRSEC), Marrakech, 2015, pp. 1-6.

[25] P. T. Vuong, A. M. Madni and J. B. Vuong, "VHDL Implementation For a Fuzzy Logic Controller," 2006 World Automation Congress, Budapest, 2006, pp. 1-8.

[26] Y. Li, D. Han, and B. Sarlioglu, “Design of high-speed machines using silicon-carbide-based inverters,” in Proc. IEEE Energy Conversion Congress and Expo. (ECCE), Montreal, Canada, 2015, pp. 3895-3900.

[27] M. Michalczuk, B. Ufnalski and L. Grzesiak, "Fuzzy logic control of a hybrid battery-ultracapacitor energy storage for an urban electric vehicle," 2013 Eighth International Conference and Exhibition on Ecological Vehicles and Renewable Energies (EVER), Monte Carlo, 2013, pp. 1-7.

[28] J. M. Miller, "Energy storage system technology challenges facing strong hybrid, plug-in and battery electric vehicles," Vehicle Power and Propulsion Conference, 2009. VPPC '09. IEEE, Dearborn, MI, 2009, pp. 4-10. doi: 10.1109/VPPC.2009.5289879

[29] J. Belt, “Long Term Combined Cycle and Calendar Life Testing,”214th Electrochemical Society Meeting, Honolulu, HI, INL/CON-08-14920, Oct. 2008.

[30] http://www.edn.com/design/automotive/4424041/Hybrid-automotive-use-of-ultracapacitors

[31] A. Millner, "Modeling Lithium Ion battery degradation in electric vehicles," Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES), 2010 IEEE Conference on, Waltham, MA, 2010, pp. 349-356.

[32] J. Smekens et al., "Influence of pulse variations on the parameters of first order empirical Li-ion battery model," Electric Vehicle Symposium and Exhibition (EVS27), 2013 World, Barcelona, 2013, pp. 1-6.

[33] Hoke, A.; Brissette, A.; Maksimovic, D.; Pratt, A.; Smith, K., "Electric vehicle charge optimization including effects of lithium-ion battery degradation," in Vehicle Power and Propulsion Conference (VPPC), 2011 IEEE , vol., no., pp.1-8, 6-9 Sept. 2011

[34] J. Jaguemont; L. Boulon; P. Venet; Y. Dube; A. Sari, "Lithium Ion Battery Aging Experiments at Sub-Zero Temperatures and Model Development for Capacity Fade Estimation," in IEEE Transactions on Vehicular Technology, vol.PP, no.99, pp.1-1

[35] Li Xue, Jiang Jiuchun, Zhang Caiping, Zhang Weige and Sun Bingxiang, "Effects analysis of model parameters uncertainties on battery SOC estimation using H-infinity observer," Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on, Istanbul, 2014, pp. 1647-1653.

[36] H. Dai, X. Zhang, W. Gu, X. Wei and Z. Sun, "A Semi-Empirical Capacity Degradation Model of EV Li-Ion Batteries Based on Eyring Equation," Vehicle Power and Propulsion Conference (VPPC), 2013 IEEE, Beijing, 2013, pp. 1-5.

[37] T. Markel, K. Smith, and A. Pesaran, "Improving Petroleum Displacement Potential of PHEVS Using Enhanced Charging Scenarios," EVS-24 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium, Stravanger, Norway, NREL/CP-540-45730, May 2009.

[38] X. Xu, Z. Li and N. Chen, "A Hierarchical Model for Lithium-Ion Battery Degradation Prediction," in IEEE Transactions on Reliability, vol. 65, no. 1, pp. 310-325, March 2016.

[39] A. A. Hussein, "Experimental modeling and analysis of lithium-ion battery temperature dependence," Applied Power Electronics Conference and Exposition (APEC), 2015 IEEE, Charlotte, NC, 2015, pp. 1084-1088.

[40] E. H. Mamdani, "Twenty years of fuzzy control: experiences gained and lessons learnt," Fuzzy Systems, 1993., Second IEEE International Conference on, San Francisco, CA, 1993, pp. 339-344 vol.1

[41] https://www.mathworks.com/solutions/hdl-code-generation-verification.html

[42] https://en.wikipedia.org/wiki/Vector_control_(motor)

Appendix 1. Core Code for Fuzzy Logic Algorithm

```
--buserror membership function--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity buserror is
port(
clk     : in std_logic;
reset   : in std_logic;
input   : in std_logic_vector  (7 downto 0);
grade1  : out std_logic_vector (7 downto 0);
grade2  : out std_logic_vector (7 downto 0)
);
end buserror;

architecture behave of buserror is

constant point1:std_logic_vector (7 downto 0):= x"00";
constant point2:std_logic_vector (7 downto 0):= x"40";
constant point3:std_logic_vector (7 downto 0):= x"80";
constant point4:std_logic_vector (7 downto 0):= x"C0";
constant point5:std_logic_vector (7 downto 0):= x"FF";
constant slope :std_logic_vector (7 downto 0):= x"04";
signal u1: std_logic_vector (15 downto 0);
signal u2: std_logic_vector (15 downto 0);
```

```
begin

process (reset, clk)

begin

if reset='1' then u1 <= x"0000"; u2 <=x"0000";

else

--if clk'event and clk='1' then

if    (input >= point1) and (input < point2) then
u1 <= x"FF"-((input-point1)*slope);
u2 <= (input-point1)*slope;
elsif (input >= point2) and (input < point3)  then
u2 <= x"FF"-((input-point2)*slope);
u1 <= (input-point2)*slope;
elsif (input >= point3) and (input < point4)  then
u1 <= x"FF"-((input-point3)*slope);
u2 <= (input-point3)*slope;
elsif (input >= point4) and (input < point5)  then
u2 <= x"FF"-((input-point4)*slope);
u1 <= (input-point4)*slope;
else
u2<= x"0000";
u1<= x"FFFF";
end if;
end if;
--end if;

end process;
        grade1 <= u1 (7 downto 0);
        grade2 <= u2 (7 downto 0);


        end architecture;
--pdem membership function--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity Pdem is
port(
clk     : in std_logic;
reset   : in std_logic;
input   : in std_logic_vector  (7 downto 0);
grade1  : out std_logic_vector (7 downto 0);
grade2  : out std_logic_vector (7 downto 0)
);
end Pdem;

architecture behave of Pdem is

constant point1:std_logic_vector (7 downto 0):= x"00";
constant point2:std_logic_vector (7 downto 0):= x"40";
constant point3:std_logic_vector (7 downto 0):= x"60";
constant point4:std_logic_vector (7 downto 0):= x"80";
constant point5:std_logic_vector (7 downto 0):= x"A0";
constant point6:std_logic_vector (7 downto 0):= x"C0";
constant point7:std_logic_vector (7 downto 0):= x"FF";
constant slope1:std_logic_vector (7 downto 0):= x"03";
constant slope2:std_logic_vector (7 downto 0):= x"08";



signal u1: std_logic_vector (15 downto 0);
signal u2: std_logic_vector (15 downto 0);

begin

process (reset, clk)

begin

if reset='1' then u1 <= x"0000"; u2 <=x"0000";

else

--if clk'event and clk='1' then

if    (input >= point1) and (input < point2) then
u1 <= x"FF"-((input-point1)*slope1);
u2 <= x"0000";
elsif (input >= point2) and (input < point3)  then
u1 <= x"FF"-((input-point2)*slope1);
u2 <= (input-point2)*slope2;
elsif (input >= point3) and (input < point4)  then
u2 <= x"FF"-((input-point3)*slope2);
u1 <= (input-point3)*slope2;
elsif (input >= point4) and (input < point5)  then
u1 <= x"FF"-((input-point4)*slope2);
u2 <= (input-point4)*slope2;
elsif (input >= point5) and (input < point6)  then
u2 <= x"FF"-((input-point5)*slope2);
u1 <= (input-point5)*slope1;
elsif (input >= point6) and (input < point7)  then
u1 <= (input-point6)*slope1;
u2 <= x"0000";
else
```

```
u1 <= x"FFFF";
u2 <= x"0000";
end if;
end if;



end process;
        grade1 <= u1 (7 downto 0);
        grade2 <= u2 (7 downto 0);


        end architecture;
--SOC membership function--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity SOC is
port(
clk     : in std_logic;
reset   : in std_logic;
input   : in std_logic_vector  (7 downto 0);
grade1  : out std_logic_vector (7 downto 0);
grade2  : out std_logic_vector (7 downto 0)
);
end SOC;

architecture behave of SOC is

constant point1:std_logic_vector (7 downto 0):= x"00";
constant point2:std_logic_vector (7 downto 0):= x"33";
constant point3:std_logic_vector (7 downto 0):= x"3E";
constant point4:std_logic_vector (7 downto 0):= x"C3";
constant point5:std_logic_vector (7 downto 0):= x"CC";
constant point6:std_logic_vector (7 downto 0):= x"FF";
constant slope1:std_logic_vector (7 downto 0):= x"17";
constant slope2:std_logic_vector (7 downto 0):= x"1C";
signal u1: std_logic_vector (15 downto 0);
signal u2: std_logic_vector (15 downto 0);

begin

process (reset, clk)

begin

if reset='1' then u1 <= x"0000"; u2 <=x"0000";

else

--if clk'event and clk='1' then

if    (input >= point1) and (input < point2) then
u1 <= x"FFFF";
u2 <= x"0000";
elsif (input >= point2) and (input < point3)  then
u1 <= x"FF"-((input-point2)*slope1);
u2 <= (input-point2)*slope1;
elsif (input >= point3) and (input < point4)  then
u2 <= x"FFFF";
u1 <= x"0000";
elsif (input >= point4) and (input < point5)  then
u2 <= x"FF"-((input-point4)*slope2);
u1 <= (input-point4)*slope2;
elsif (input >= point5) and (input < point6)  then
u1 <= x"FFFF";
```

```
u2 <= x"0000";
else
u1 <= x"FFFF";
u2 <= x"0000";

end if;
end if;



end process;
            grade1 <= u1 (7 downto 0);
            grade2 <= u2 (7 downto 0);


            end architecture;
--Ucap membership function--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity Ucap is
port(
clk     : in std_logic;
reset   : in std_logic;
input   : in std_logic_vector  (7 downto 0);
grade1  : out std_logic_vector (7 downto 0);
grade2  : out std_logic_vector (7 downto 0)
);
end Ucap;

architecture behave of Ucap is


constant point1:std_logic_vector (7 downto 0):= x"00";
constant point2:std_logic_vector (7 downto 0):= x"29";
constant point3:std_logic_vector (7 downto 0):= x"37";
constant point4:std_logic_vector (7 downto 0):= x"DF";
constant point5:std_logic_vector (7 downto 0):= x"EF";
constant point6:std_logic_vector (7 downto 0):= x"FF";
constant slope1:std_logic_vector (7 downto 0):= x"17";
constant slope2:std_logic_vector (7 downto 0):= x"1C";
signal u1: std_logic_vector (15 downto 0);
signal u2: std_logic_vector (15 downto 0);

begin

process (reset, clk)

begin

if reset='1' then u1 <= x"0000"; u2 <=x"0000";

else

--if clk'event and clk='1' then

if    (input >= point1) and (input < point2) then
u1 <= x"FFFF";
u2 <= x"0000";
elsif (input >= point2) and (input < point3)  then
u1 <= x"FF"-((input-point2)*slope1);
u2 <= (input-point2)*slope1;
elsif (input >= point3) and (input < point4)  then
u2 <= x"FFFF";
u1 <= x"0000";
elsif (input >= point4) and (input < point5)  then
u2 <= x"FF"-((input-point4)*slope2);
```

```
u1 <= (input-point4)*slope2;
elsif (input >= point5) and (input < point6)  then
u1 <= x"FFFF";
u2 <= x"0000";
else
u1 <= x"FFFF";
u2 <= x"0000";

end if;
end if;



end process;
            grade1 <= u1 (7 downto 0);
            grade2 <= u2 (7 downto 0);


            end architecture;
--findmax function--
library ieee;
use ieee.std_logic_1164.all;

package findmax is


function maximum(a,b:    in std_logic_vector(7 downto 0))
return std_logic_vector;
end findmax;


package body findmax is
function maximum(a,b:    in std_logic_vector(7 downto 0))
return std_logic_vector is
variable max: std_logic_vector (7 downto 0);
begin
if    (a<b) then max:=b;
```

```
else            max:=a;
end if;
return max;
end function maximum;
end findmax;
--findmin function--
library ieee;
use ieee.std_logic_1164.all;

package findmin is

function minimum(a,b:    in std_logic_vector(7 downto 0))
return std_logic_vector;
end findmin;

package body findmin is
function minimum(a,b:    in std_logic_vector(7 downto 0))
return std_logic_vector is
variable min: std_logic_vector (7 downto 0);
begin
if    (a<b) then min:=a;
else            min:=b;
end if;
return min;
end function minimum;
end findmin;
--rulebase--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use work.findmin.all;
use work.findmax.all;
```

```
entity rulebase is

port(
clock                                        : in std_logic;
Reset                                        : in std_logic;
buser, ucap, perr, soc                           : in std_log ic_vector (7 downto 0);
grade1,grade2,grade3,grade4,grade5,grade6,grade7,grade8  : in std_logic_vector (7 downto 0);

NL,NS,ZZ,PS,PL,NLARGE,NSMALL,ZERO,PSMALL,PLARGE          : out std_logic_vector(7  downto 0)
);
end rulebase;

architecture behave of rulebase is

signal NL1,NS1,NS2,NS3,NS4,ZZ1,ZZ2,ZZ3,ZZ4,ZZ5,ZZ6,ZZ7,ZZ8,PS1,PS2,PS3,PS4,PL1,PL2:
std_logic_vector (7 downto 0);
signal nlp1,nlp2,nsp1,nsp2,zzp1,zzp2,zzp3,zzp4,zzp5,zzp6,zzp7,zzp8,zzp9,zzp10,zzp11,psp1,psp2,plp1,plp2:
std_logic_vector (7 downto 0);

BEGIN
process (clock)--r1
begin
if(clock'event and clock='1') then
if (buser>=x"00" and buser<x"40") and (ucap>=x"00" and ucap<x"DF") and (soc>=x"00" and soc<x"C3")
then PS1  <= grade1;
          psp1 <= grade1;
          else ps1 <= x"00"; psp1 <= x"00";
          end if;
          end if;
         END PROCESS;

PROCESS (clock)--r2
BEGIN
```

```
if(clock'event and clock='1') then
if (buser>=x"00" and buser<x"80") and (soc>=x"00" and soc<x"C3")
then PS2 <=grade2;
     zzp1 <=grade2;
               else ps2 <= x"00"; zzp1<= x"00";
               end if;
               end if;
               END PROCESS;

PROCESS (clock)--r3
BEGIN
if(clock'event and clock='1') then
if buser>=x"40" and buser<x"C0"
then ZZ1 <= grade1;
     zzp2 <= grade1;
                else zz1 <= x"00";zzp2<= x"00";
               end if;end if;
              END PROCESS;

PROCESS (clock)--r4
BEGIN
if(clock'event and clock='1') then
if (buser>=x"80" and buser<x"FF") and (soc>=x"3E" and soc<x"FF")
then NS1 <= grade2;
     zzp3 <= grade2;
               else ns1<= x"00"; zzp3<= x"00";
               end if;
               end if;
              END PROCESS;

PROCESS (clock)--r5
BEGIN
if(clock'event and clock='1') then
```

```
if (buser>=x"C0" and buser<x"FF") and (ucap>=x"37" and ucap<=x"FF") and (soc>=x"3E" and soc<=x"FF")
then NS2 <= grade1;
    nsp1 <= grade1;
            else ns2<= x"00";nsp1<= x"00";
            end if;
            end if;
END PROCESS;

PROCESS (clock)--r6
BEGIN
if(clock'event and clock='1') then
if (ucap>=x"00" and ucap<x"DF") and (perr>=x"00" and perr<x"40")
then ZZ2 <= grade5;
    plp1 <= grade5;
             else zz2<= x"00";plp1<= x"00";
            end if;end if;
           END PROCESS;

PROCESS(clock)--r7
BEGIN
if(clock'event and clock='1') then
if          (perr>=x"00" and perr<x"80") and (soc>=x"00" and soc<=x"C3")
then PS3 <= grade6;
    zzp4 <= grade6;
             else ps3<= x"00";zzp4<= x"00";
             end if;end if;
             END PROCESS;

PROCESS (clock)--r8
BEGIN
if(clock'event and clock='1') then
if perr>=x"40" and perr<x"C0"
then ZZ3 <= grade5;
```

```
    zzp5 <= grade5;
            else zz3<= x"00";zzp5<= x"00";
            end if;end if;
            END PROCESS;

PROCESS (clock)--r9
BEGIN
if(clock'event and clock='1') then
if(perr>=x"80" and perr<x"FF") and (soc>=x"3E" and soc<x"FF")
then NS3 <= grade6;
    zzp6 <= grade6;
            else ns3<= x"00";zzp6<= x"00";
            end if;end if;
            END PROCESS;

PROCESS (clock)--r10
BEGIN
if(clock'event and clock='1') then
if (ucap>=x"37" and ucap<x"FF") and (perr>=x"C0" and perr<=x"FF")
then ZZ4 <= grade5;
    nlp1 <= grade5;
            else zz4<= x"00";nlp1<= x"00";
            end if;end if;
            END PROCESS;

PROCESS (clock)--r20
BEGIN
if(clock'event and clock='1') then
if (buser>=x"00" and buser<x"40") and (ucap>=x"00" and ucap<=x"DF")
then ZZ5 <= grade1;
    plp2 <= grade1;
            else zz5<= x"00";plp2<= x"00";
            end if;end if;
```

```
END PROCESS;

PROCESS (clock)--r11
BEGIN
if(clock'event and clock='1') then
if (buser>=x"00" and buser<x"40") and (soc>=x"00" and soc<=x"C3")
then PL1 <= grade1;
     zzp7 <= grade1;
               else pl1<= x"00";zzp7<= x"00";
               end if;end if;
             END PROCESS;

PROCESS (clock)--r12
BEGIN
if(clock'event and clock='1') then
if (buser>=x"00" and buser<x"80") and (ucap>=x"00" and ucap<x"DF")
then ZZ6 <= grade2;
     psp2 <= grade2;
               else zz6<= x"00";psp2<= x"00";
               end if;end if;
               END PROCESS;

PROCESS (clock)--r13
BEGIN
if(clock'event and clock='1') then
if (buser>=x"80" and buser<x"FF") and ( ucap>=x"37" and ucap<x"FF")
then ZZ7 <= grade2;
     nsp2 <= grade2;
               else zz7<= x"00";nsp2<= x"00";
               end if;    end if;
               END PROCESS;

PROCESS (clock)--r14
```

```
BEGIN
if(clock'event and clock='1') then
if (buser>=x"C0" and buser<x"FF") and (ucap>=x"37" and ucap<x"FF")
then ZZ8 <= grade1;
   nlp2 <= grade1;
          else zz8<= x"00";nlp2<= x"00";
          end if;end if;
          END PROCESS;

PROCESS (clock)--r15
BEGIN
if(clock'event and clock='1') then
if (buser>=x"C0" and buser<x"FF") and (ucap>=x"3E" and ucap<x"FF")
then NL1 <= grade1;
   zzp8 <= grade1;
          else nl1<= x"00";zzp8<= x"00";
          end if;end if;
          END PROCESS;

PROCESS (clock)--r16
BEGIN
if(clock'event and clock='1') then
if (ucap>=x"DF" and ucap<x"FF") and (perr>=x"00" and perr<x"40") and (soc>=x"00" and soc<x"C3")
then PL2 <= minimum(grade3,grade5);
   zzp9 <= minimum(grade3,grade5);
          else pl2<= x"00";zzp9<= x"00";
          end if;end if;
          END PROCESS;

PROCESS (clock)--r17
BEGIN
if(clock'event and clock='1') then
if (ucap>=x"DF" and ucap<x"FF") and (perr>=x"00" and perr<x"80") and (soc>=x"00" and soc<x"C3")
```

```
then PS4 <= minimum(grade3,grade6);
     zzp10 <= minimum(grade3,grade6);
                else ps4<= x"00";zzp10<= x"00";
                end if;end if;
END PROCESS;


PROCESS (clock)--r18
BEGIN
if(clock'event and clock='1') then
if (perr>=x"80" and perr<x"FF") and (soc>=x"3E" and soc<x"FF")
then NS4 <= grade6;
     zzp11 <= grade6;
                else ns4<= x"00";zzp11<= x"00";
                end if;end if;
                end process;


process (clock)
begin
--if (clock'event and clock='1') then
NL     <= NL1;
NS     <= maximum(maximum(NS1,NS2),maximum(NS3,NS4));
ZZ     <=
maximum(maximum(maximum(ZZ1,ZZ2),maximum(ZZ3,ZZ4)),maximum(maximum(ZZ5,ZZ6),maximum(ZZ7,Z
Z8)));
PS     <= maximum(maximum(PS1,PS2),maximum(PS3,PS4));
PL     <= maximum(PL1,PL2);


NLARGE <= maximum(nlp1,nlp2);
NSMALL <= maximum(nsp1,nsp2);
PSMALL <= maximum(psp1,psp2);
PLARGE <= maximum(plp1,plp2);
ZERO   <=
maximum(maximum(maximum(maximum(zzp1,zzp2),maximum(zzp3,zzp4)),maximum(maximum(zzp5,zzp6),max
imum(zzp7,zzp8))),maximum(maximum(zzp9,zzp10),maximum(zzp11,zzp11)));
--end if;
```

```
end process;

end architecture;
--aggregation--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity aggragation is
port(
clk                          : in std_logic;
reset                        : in std_logic;
nl,ns,zz,ps,pl                : in std_logic_vector  (7 downto 0) ;
nlarge,nsmall,zero,psmall,plarge : in std_logic_vector  (7 downto 0) ;
pbat1,pbat2                   : out std_logic_vector (19 downto 0);
pcap1,pcap2                   : out std_logic_vector (19 downto 0)
);
end aggragation;

architecture behave of aggragation is

signal pbat, pcap                        : std_logic_vector (19 downto 0);
signal nlb,nsb,zzb,psb,plb,nlp,nsp,zzp,psp,plp : std_logic_vector (19 downto 0);


begin

process (clk,reset)

begin

pbat <= nl*x"020"+ns*x"060"+zz*x"080"+ps*x"0A0"+pl*x"0E0";
```

```
pcap <= nlarge*x"02A"+nsmall*x"055"+zero*x"080"+psmall*x"0AA"+plarge*x"0D4";
nlb   <= x"000" & nl; nlp <= x"000" & nlarge;
nsb   <= x"000" & ns; nsp <= x"000" & nsmall;
zzb   <= x"000" & zz; zzp <= x"000" & zero;
psb   <= x"000" & ps; psp <= x"000" & psmall;
plb   <= x"000" & pl; plp <= x"000" & plarge;

end process;

pbat1 <= pbat; pbat2 <= nlb+nsb+zzb+psb+plb;
pcap1 <= pcap; pcap2 <= nlp+nsp+zzp+psp+plp;

end architecture;
```