PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

NEELAMBUR, COIMBATORE-641 062

EE3031- Intelligent Control of Electric Vehicles

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ACADEMIC YEAR 2023-2024

NAME	
REGISTER No.	
BRANCH / YEAR	**EEE / III year**
SEMESTER	**V**

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

NEELAMBUR, COIMBATORE-641 062

BONAFIDE CERTIFICATE

Certified that this is a Bonafide Record of work done by ……... …… of III-year B.E (Electrical and Electronics Engineering) in the **EE3031- Intelligent Control of Electric Vehicles** conducted in this institution, as prescribed by Anna University, Chennai, for the fifth semester, during the academic year 2023-2024.

Faculty in-charge **Head of the Department**

Date:

University Register Number: _____

Submitted on **:** _____

Internal Examiner **External Examiner**

List of Experiments

S. No.	Title	Marks	Signature
1	Design and simulate speed controller for induction motors in EV for both dynamic and steady state performance		
2	Simulate a fuzzy logic controller-based energy storage system for EV		
3	Fuzzy logic control of BLDC motor using FPGA in real time		

Ex. No.: 1	**Design and Simulate Speed Controller for Induction Motors in EV for both Dynamic and Steady State Performance**
Date:	

Aim

To design and simulate a speed controller for induction motors in EV using PI Controller and Space Vector Pulse Width Modulation (SVPWM)

Introduction

In electric vehicle (EV) applications, the speed control of induction motors is crucial for efficient and responsive operation. A PI (Proportional-Integral) controller is commonly employed to achieve precise speed regulation. The PI controller adjusts the motor's input voltage based on the error signal, which is the difference between the desired and actual speeds.

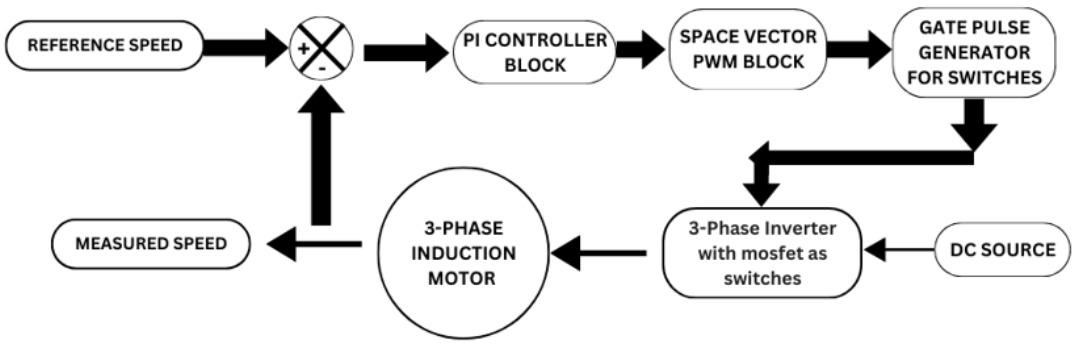
The Proportional component responds to the present error, providing a control action proportional to the speed deviation. The Integral component adds a corrective action based on the accumulated past errors, minimizing any steady-state speed errors.

The PI controller helps maintain a stable and accurate speed, enhancing overall performance and energy efficiency. This control strategy ensures that the induction motor operates at the desired speed under varying load and driving conditions, contributing to a smoother and more reliable electric vehicle experience.

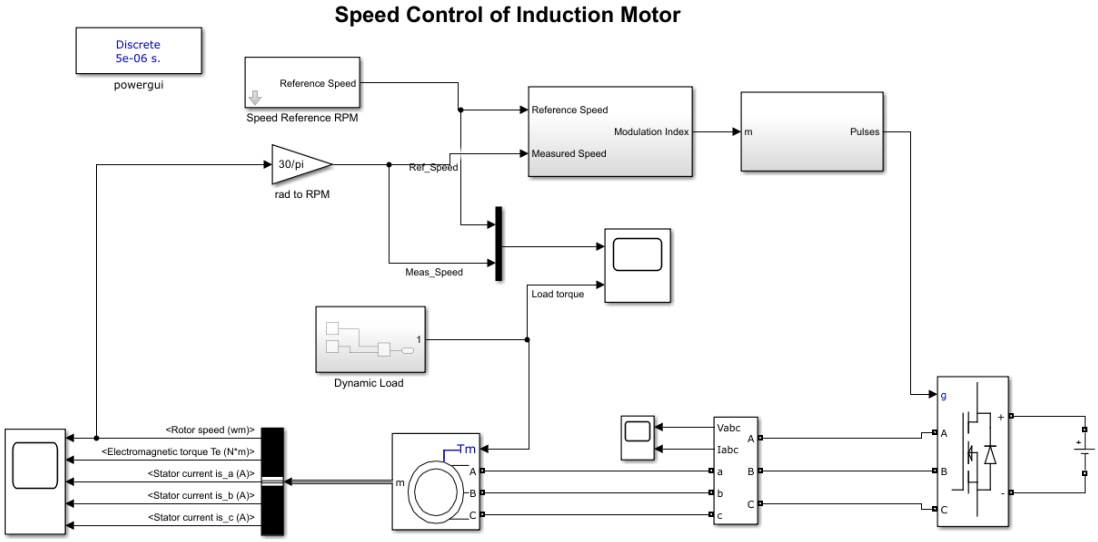
ALGORITHM

The algorithm for speed control of an induction motor involves setting a reference speed, measuring the actual speed using a sensor, and calculating the speed error. The PI controller adjusts the modulation index based on the speed error. This modulation index is then used in the SVM (Space Vector Modulation) block, which generates three sine waves 120 degrees out of phase to represent reference voltages for the motor phases. A repeating sequence block ensures the cyclical and repetitive nature of the SVM output. Relational operators compare SVM output with a carrier signal to determine gate pulses for a three-phase inverter. These gate pulses control the inverter switches, modulating the voltage applied to the induction motor. The closed-loop system continuously adjusts the modulation index via the PI controller to minimize speed error, resulting in precise speed control of the induction motor.

BLOCK DIAGARAM



CIRCUIT:



WAVEFORMS:

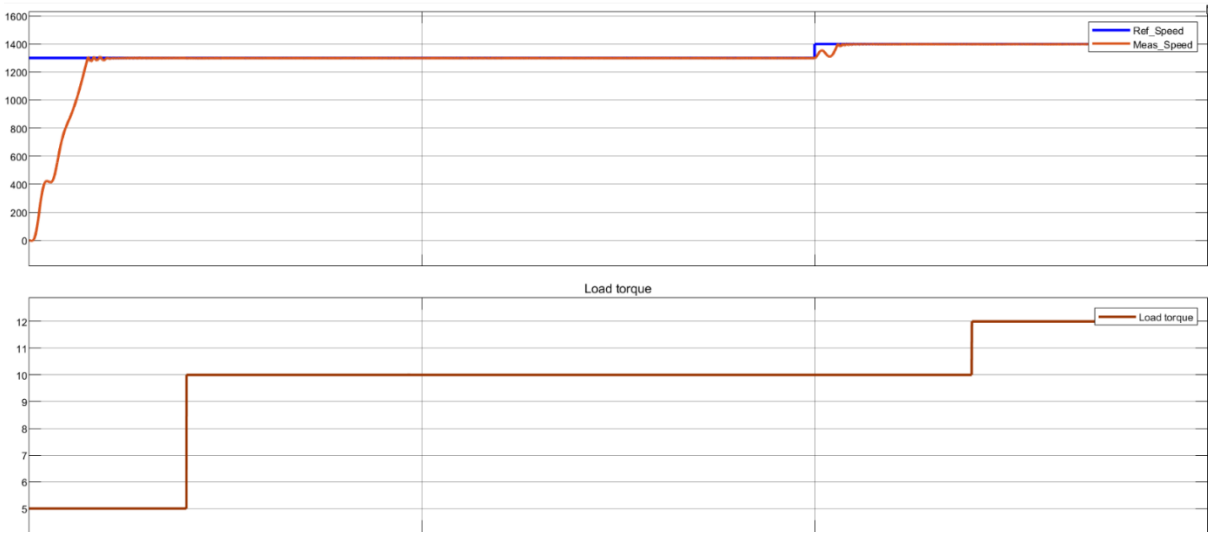


Fig.1: Plot of reference speed and measured speed for different load torques.

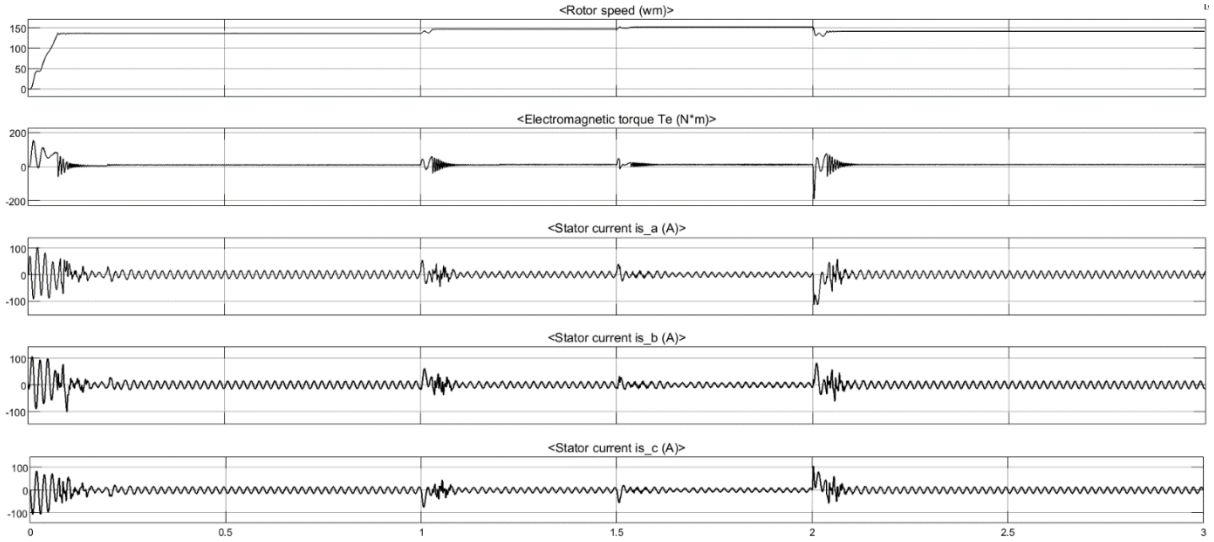


Fig.2: Plot of stator currents and Electromagnetic Torque

INFERENCE

The simulation shows that the motor responds well to changes. The actual speed closely follows the desired speed, indicating effective speed control. The load torque graph illustrates the motor's ability to handle different loads smoothly. In summary, the simulation confirms the reliability of the control system for maintaining accurate motor speed, even with dynamic load variations.

RESULT

Thus, the design and simulation of the speed controller for induction motors for EV application is simulated using Simulink.

Ex. No.: 2	**Simulate a Fuzzy Logic Controller Based Energy Storage System for EV**
Date:	

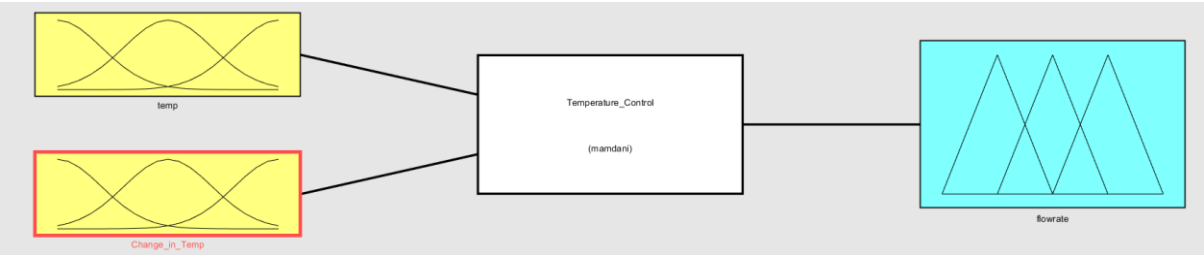
Aim

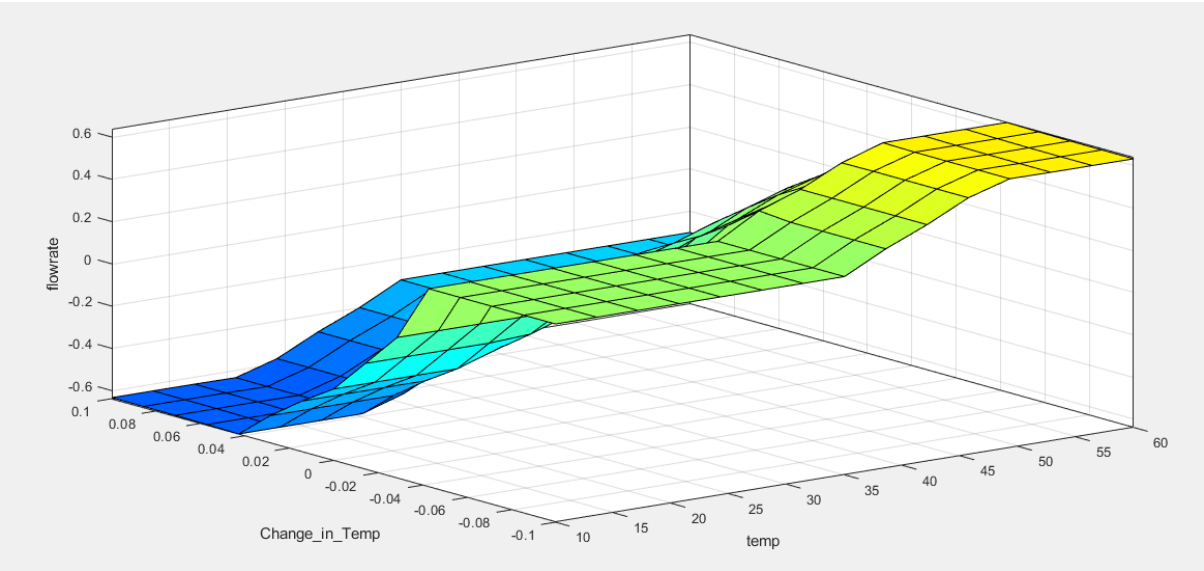
To Simulate a fuzzy logic controller to control the temperature of the battery pack.

Introduction

A Fuzzy Logic Controller (FLC) can be a used for managing energy storage systems in Electric Vehicles (EVs). Energy storage systems play a crucial role in enhancing the overall performance, efficiency, and range of EVs. The temperature management of electric vehicle (EV) batteries is crucial for their performance, efficiency, and lifespan. Fuzzy Logic Controllers (FLCs) provide a flexible and adaptive approach to control systems, making them well-suited for the dynamic and uncertain nature of battery temperature regulation in EVs. The temperature of the battery pack is measured and given as input to fuzzy system to predict the flow rate of coolant to maintain the temperature in nominal value. The fuzzy model and the fuzzy rule surface is shown in the following figures

Fuzzy model and surface plot of rules

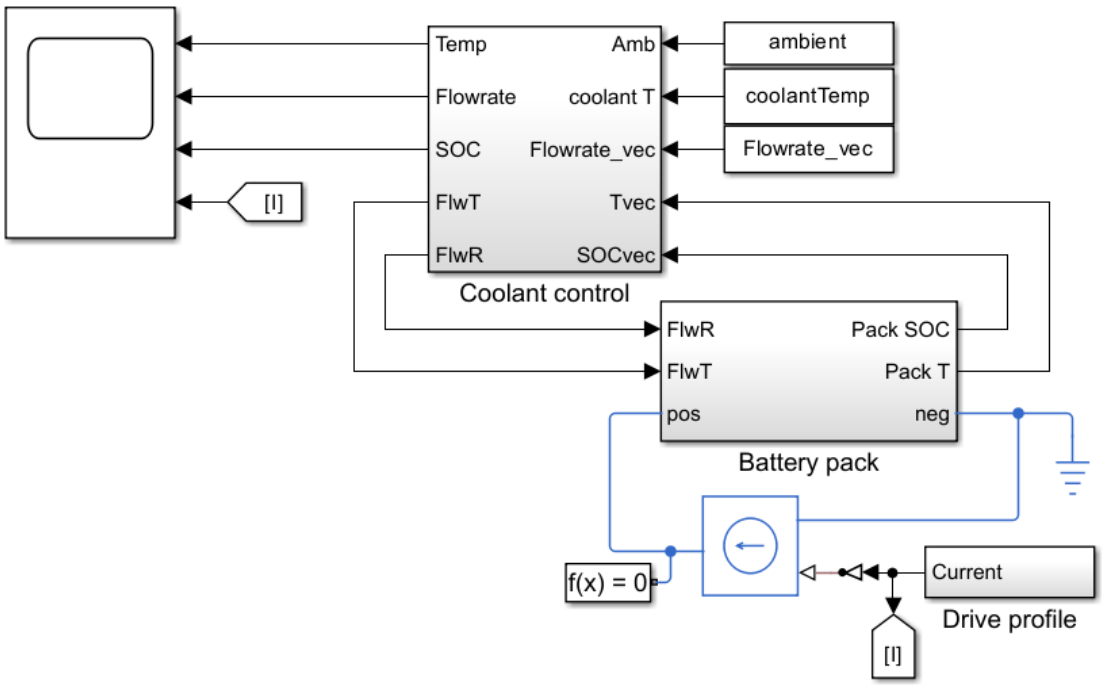




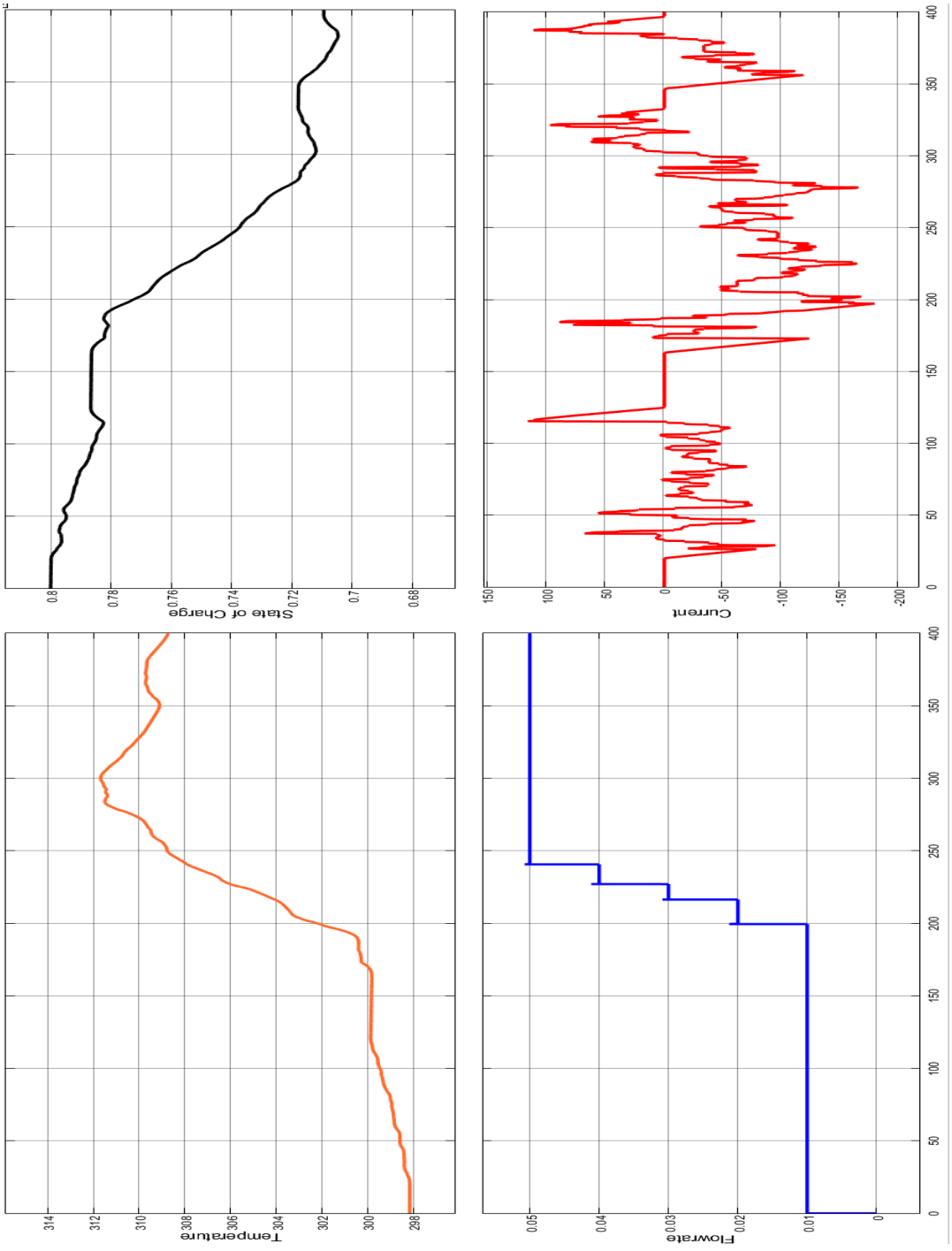
Procedure:

Step 1: Build the battery pack model and configure the thermal properties
Step2: Develop a fuzzy controller using the fuzzy toolbox in Matlab
Step3: Use EV drive cycle as a load to the battery pack.
Step4: Import the fis file to the fuzzy controller block.
Step6: The measured temperature and change in temperature are given as input to the fuzzy logic controller and output is connected to the flow controller.
Step7: Simulate the model and verify the results

Matlab model



Output Waveform.



RESULT

Thus, the fuzzy logic controller has been implemented to control the temperature of the battery pack.

Ex. No: 3 Date:	**Fuzzy Logic Control of BLDC Motor using FPGA in Real Time**

Aim

To implement real time fuzzy based BLDC motor control using FPGA.

Introduction

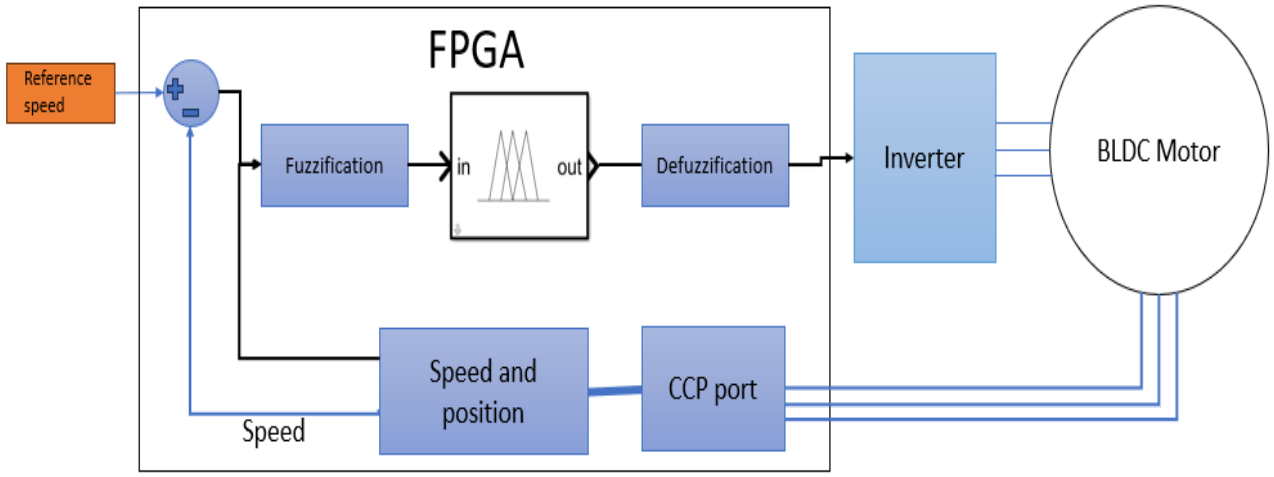
Brushless DC (BLDC) motors are known for their widespread use in various applications due to their high efficiency and reliability. The implementation of a fuzzy logic controller on a Field-Programmable Gate Array (FPGA) offers the advantage of parallel processing, making it an ideal platform for real-time control applications. The experiment aims to combine the strengths of BLDC motors, fuzzy logic control, and FPGA technology to achieve precise and responsive motor control.

The fuzzy logic controller is designed to regulate the motor's speed and position. Fuzzy rules are defined based on the error (difference between the desired and actual position), change in error, and speed of the motor. The linguistic variables such as "low," "medium," and "high" are used to describe the input and output membership functions. The fuzzy rules are established through experimentation and tuning to achieve optimal motor performance.

The objective is to achieve precise and efficient control over the motor's speed and position through the utilization of fuzzy logic principles. The experiment involves the design and integration of a fuzzy logic controller into the FPGA, interfacing with sensors for real-time feedback, and controlling the BLDC motor's phases.

The fuzzy logic controller is translated into a hardware description language, such as Verilog or VHDL, for implementation on the FPGA. The FPGA interfaces with the sensors to receive real-time feedback on the motor's position. Logic is developed to control the three phases of the BLDC motor based on the fuzzy logic controller's output. The parallel processing capabilities of the FPGA are harnessed to ensure real-time execution of the control algorithm.

Block Diagram



Procedure

Step1: Measure the speed of the BLDC motor using Hall sensor

Step2: Compare the speed of motor with the reference speed

Step3: Calculate the error and change in error and input these values to fuzzy inference system (FIS)

Step4: The output of the FIS is converted into duty cycle

Step 5: The duty cycle value is passed to the PWM generator block to generate PWM signal to the driver circuit of converter to change the voltage input to BLDC motor to control the speed.

Fuzzy Truth Table

e	Δ*e*	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
NB	NB	NB	NB	NB	NS	NS	NS	NM
NM	ZE	NB	NM	NS	NS	NS	NM	NM
NS	PB	NS	NS	NS	NS	NM	PM	PM
ZE	NS	NS	NM	NS	NS	NS	PM	NM
PS	NS	NS	NS	NM	NM	NM	PM	NS
PM	ZE	NM	NM	PM	PM	PM	PM	NM
PB	NM	NM	NS	NM	NM	NS	NM	NM

In this table, e represents the speed error linguistic variable, Δe represents the change in speed error linguistic variable, and the columns Rule 1 through Rule 7 represent the fuzzy output based on the specified rules

VHDL Code: Fuzzy

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FuzzySpeedController is
   Port ( e : in STD_LOGIC_VECTOR(2 downto 0);
          de : in STD_LOGIC_VECTOR(2 downto 0);
          dDC : out STD_LOGIC_VECTOR(2 downto 0));
end FuzzySpeedController;

architecture Behavioural of FuzzySpeedController is
   type MembershipFunction is array (0 to 7) of STD_LOGIC_VECTOR(2 downto 0);

   constant NB : MembershipFunction := ("100", "011", "010", "001", "001", "000", "000", "000");
   constant NM : MembershipFunction := ("000", "100", "011", "010", "001", "001", "000", "000");
   constant NS : MembershipFunction := ("000", "000", "100", "011", "010", "001", "001", "000");
   constant ZE : MembershipFunction := ("000", "000", "000", "100", "011", "010", "001", "001");
   constant PS : MembershipFunction := ("001", "001", "000", "000", "100", "011", "010", "001");
   constant PM : MembershipFunction := ("001", "001", "000", "000", "000", "100", "011", "010");
   constant PB : MembershipFunction := ("010", "001", "001", "000", "000", "000", "100", "011");

   signal rule_outputs : STD_LOGIC_VECTOR(6 downto 0);

begin

   process (e, de)
   begin
      -- Fuzzy rule evaluation
      case e is
         when "000" =>
            case de is
               when "000" => rule_outputs <= NB;
               when "001" => rule_outputs <= NM;
               when "010" => rule_outputs <= NS;
               when "011" => rule_outputs <= NS;
               when "100" => rule_outputs <= NS;
               when "101" => rule_outputs <= NM;
               when others => rule_outputs <= "000";
            end case;

         when "001" =>
            case de is
               when "000" => rule_outputs <= NB;
               when "001" => rule_outputs <= NM;
               when "010" => rule_outputs <= NS;
               when "011" => rule_outputs <= NS;
               when "100" => rule_outputs <= NS;
```

```
                when "101" => rule_outputs <= NM;
                when others => rule_outputs <= "000";
            end case;

        -- Add cases for other values of e

        when others =>
            -- Default case
            rule_outputs <= (others => '0');
    end case;

     -- Calculate weighted sum and total weight for centroid defuzzification
    for i in 0 to 6 loop
        weighted_sum <= weighted_sum + (to_integer(unsigned(rule_outputs(i))) * (i + 1));
        total_weight <= total_weight + to_integer(unsigned(rule_outputs(i)));
    end loop;

    -- Centroid defuzzification
    if total_weight /= 0 then
        dDC <= std_logic_vector(to_unsigned(weighted_sum / total_weight, 3));
    else
        dDC <= "000";
    end if;
  end process;

end Behavioural;
```

VHDL CODE: PWM

```
//Inputs: Clk, DUTY_CYCLE
//Output: PWM_OUT
module PWM_Generator (
input clk,        // Clock input
input [7:0] DUTY_CYCLE, // Input Duty Cycle
output PWM_OUT         // Output PWM
);
reg out;
assign PWM_OUT =out;
reg [7:0] count=0;
always @ (*) begin
out=((DUTY_CYCLE/2)>=count)? 1:0;
end
always @ (posedge clk) begin
if(count==50) count=0;
count=count+1;
end
end module
```

Result

Real time fuzzy based BLDC motor control using FPGA has been implemented and verified.