FPGA BASED FLIGHT CONTROLLER

Project Report

Submitted in partial fulfillment for the award of the degree
of

BACHELOR OF TECHNOLOGY
IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

Athul John Kurian (MDL17EC030)
Medha Lakshman Rao (MDL17EC078)
Omar Bin Shafi (MDL17EC091)
Sebastian James (MDL17EC109)

Department of Electronics Engineering
Govt. Model Engineering College
Thrikkakara, Cochin - 682 021
APJ Abdul Kalam Technological University

June 2021

FPGA BASED FLIGHT CONTROLLER

PROJECT REPORT

Submitted to
the APJ Abdul Kalam Technological University
in partial fulfillment of the requirement for the award of Degree of
Bachelor of Technology in Electronics and Communication Engineering

BY
ATHUL JOHN KURIAN (MDL17EC030)
MEDHA LAKSHMAN RAO (MDL17EC078)
OMAR BIN SHAFI (MDL17EC091)
SEBASTIAN JAMES (MDL17EC109)

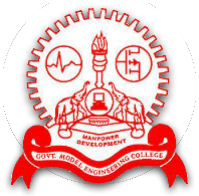
Department of Electronics Engineering,
Model Engineering College,
Thrikkakara, Kochi-682021,
Kerala.

JUNE 2021

DEPARTMENT OF ELECTRONICS ENGINEERING

MODEL ENGINEERING COLLEGE

THRIKKAKARA, KOCHI-682021



CERTIFICATE

This is to certify that this Project entitled ***FPGA BASED FLIGHT CONTROLLER*** is the bonafide record of work carried out by **ATHUL JOHN KURIAN (MDL17EC030), MEDHA LAKSHMAN RAO (MDL17EC078), OMAR BIN SHAFI (MDL17EC091), SEBASTIAN JAMES (MDL17EC109)** in partial fulfillment of the requirements for the completion of Degree of Bachelor of Technology in Electronics and Communication Engineering, at the Department of Electronics Engineering, Model Engineering College, Thrikkakara, Kochi.

Mr. Jagadeesh Kumar P
Project Guide

Ms. Thushara H P
Project Co-ordinator

Dr. Jayasree V K
Head of the Department

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Almighty for helping us to make this possible and complete this work on time. We would like to express our sincere gratitude to everyone who assisted us in completing this project. We express our gratitude towards Prof. (Dr.) Vinu Thomas, Principal, Model Engineering College and Prof. (Dr.) Jayasree V.K, HOD of Electronics Engineering, Model Engineering College. We express our sincere gratitude towards our coordinator Ms. Thushara H P, Assistant Professor, Department of Electronics Engineering, for permitting us to commence this project. Special thanks to our guide Mr. Jagadeesh Kumar P, Assistant Professor, Department of Electronics Engineering, whose guidance and encouragement helped us throughout this endeavor. Last but not the least we would like to thank all our classmates who gave us constant support for completing this venture successfully.

ABSTRACT

This work aims to design and implement a digital flight controller on an FPGA prototype board for stabilizing an unmanned aerial vehicle (UAV) such as a Quadcopter. The brain of any aircraft is the Flight control system (FCS), especially in UAVs. The purpose of the project was to develop and thereby access the feasibility of using an FPGA in the stabilized control of an aerial robotic system. Generally, FPGAs are good for implementing time-critical concurrent processing functions. The project implements PPM decoder, PWM encoder, IMU (Inertial Measurement Unit) interface, and PID controller as the main modules of the flight controller. The FPGA used is Altera Cyclone II and the IDE used is Quartus II. FPGA-based UAVs are better compared to microcontroller-based UAVs due to their low power usage, fast response, less volume, functionality modification, and compatibility with numerous applications.

Contents

List of Tables

List of Figures

ABBREVIATIONS

BLDC Brushless Direct Current

ESC Electronic Speed Controller

FC Flight Controller

FPGA Field Programmable Gate Array

FSM Finite State Machine

GPIO General Purpose Input/Output

HDL Hardware Description Language

I2C Inter Integrated Circuit

IDE Integrated Development Environment

IMU Inertial Measurement Unit

ML Machine Learning

MVP Minimum Viable Product

PID Proportional Integral Derivative

PPM Pulse Position Modulation

PWM Pulse Width Modulation

RC Radio Control

RF Radio Frequency

RTL Register-Transfer Level

SPI Serial Peripheral Interface

UAV Unmmaned Aerial Vehicle

VHDL Very High Speed Integrated Circuit Hardware Description Language

Chapter 1

INTRODUCTION

This chapter gives an introduction to the project by giving details on general background, explaining the motivation, aim, scope and relevance involved.

1.1 General Background

The UAVs (Unmanned Aerial Vehicle) or drones as they are commonly known are aircraft that can be controlled remotely or flown autonomously. These devices have become very popular in recent years and can be used for several applications. Many solutions are based on embedded sequential systems, however, the more processes these systems execute the greater impact they will have on variables such as precision, speed of response, and synchronism [1]. This project presents a solution for the UAV's Flight Controller integrated into an embedded concurrent system such as an FPGA.

1.2 Motivation

The FPGA provides enough power of calculation to extend the intelligence of the system with more autonomy, excellent synchronism, and improved response times. Field Programmable Gate Array (FPGA) provides parallel processing (all the bit at a time) and having large numbers of gates on a single chip (essential for less volume). FPGA also provides a cost savings platform. The software of the systems would also have a huge impact on the

overall design, for which code parallelization provides faster speed in FPGA and other hardware components.

1.3 Aim

The goal of the Flight Controller System is to help the user with automatic compensation in all the maneuvers of the aircraft in case of any disturbance. The inputs of the system are the signals delivered by the RF receiver which delivers the commands coming from the user and the sensor's signals that are on-board which give the direction and position of the aircraft in real-time.

1.4 Scope and relevance

The scope is that UAVs are widely used in real-time applications (e.g. military, disaster rescue operations, civil operations, monitoring of Agricultural area, monitoring of industrial area, photography, coastal surveillance, combat zone surveillance, remote sensing, ecological monitoring, natural resource estimation, patrolling, transport of equipment, etc.).
The relevance lies in the fact that microcontroller-based FC is slower, while FPGA-based FC is faster. FPGAs are more powerful, and precise, with better synchronism, and response leading to more autonomy.

Chapter 2

LITERATURE SURVEY

This chapter involves summarizing the information gathered from reference papers, the objectives of this venture, and the novelty factor.

2.1 Literature review

The insights gained from referring to the list of papers have been mentioned here in this section.
[1] provided an understanding of the range of applications drones are being used for, parallel processing capability of FPGAs enabling integration of various sensor units with the FPGA resulting in fast real-time processing of inputs, the requirement of PWM driver for ESC.
It was understood that the output of the MPU is to be smoothened using filters [2]. The equation necessary to generate PWM in terms of user input values was obtained. An overview of the block of implementing a flight controller, details regarding PID controller, timing, and control of input signals from sensors was also studied and inferred.
An idea on how to design a PWM encoder and decoder was obtained using [3]. Further insights into the basic block diagram of the Flight Controller, mapping output of FPGA to user-defined terms for motor control.

Table 2.1 summarizes the information obtained from the reference papers.

Sl No.	YEAR	TITLE & SOURCE	OBJECTIVES	METHODOLOGY	RESULT	LIMITATIONS
1	2016	TITLE: Design, development and implementation of a UAV flight controller based on a state machine approach using a FPGA embedded system [1]. SOURCE: DASC	Fast Real-time processing of inputs, Sensor Interfacing With FPGA, Combining Various Sensor inputs	Parallel processing, and having large numbers of gates on a single chip and also provides a cost saving platform. Sensors measure the air vehicle's altitude and other functions and compare it to the desired states and the error signal is eliminated.	Parallel Processing	Noise From Sensors Affect stability
2	2016	TITLE: An analytical review on FPGA based autonomous flight control system for small UAVs [2]. SOURCE: ICEEOT	To smoothen out MPU-6050 Output using filters. To generate PWM motor Signals corresponding to roll and pitch from the sensors.	PID Controller offers short transition, good stability, anti-disturbance, good control and fulfills the requirement of real-time and accurate control. The goal of the FCS is to help the user with automatic compensation in all the maneuvers of the aircraft in case of any disturbance.	Timing and Control of FPGA input signals from sensors. Synchronization of signals. Generation of output PWM signals for controlling the motors.	PID - limited stability.
3	2018	TITLE: Design and Implementation of FPGA Based Quadcopter [3]. SOURCE: IJETSR	PWM encoder and decoder design, Study Basic Block diagram of the FC, Mapping unit to produce output for motors from the roll, pitch, yaw, throttle	FPGA Block Diagram, PWM encoder, PWM signal generator, PWM decoder, Mapping unit	Relation between the Motor output, Throttle Roll, pitch and yaw required.	No PPM encoder/decoder is implemented.
4	2019	TITLE: Designing multiple pid controllers based on an fpga for controlling the temperature of tem-cell surfaces [4]. SOURCE: SIBIRCON	Computing PID controllers with different sets of parameters.	Verilog HDL in Quartus, PID controller, I2C protocol	Processing speed of PWM algorithm is important.	RAM, ROM is required.

Table 2.1: Summary of the reference papers

Sl No.	YEAR	TITLE & SOURCE	OBJECTIVES	METHODOLOGY	RESULT	LIMITATIONS
5	2019	TITLE: An Implementation of the System on Chip Control System for a FPGA-Based Computer Vision Accelerator [5]. SOURCE: ISOCC.	To build a high efficiency and compact system, which also provides flexibility and convenience for machine control.	Made use of FPGA and tested on a drone. Programmable logic is used for sensor interfaces such as motor drivers.	Real-time control was feasible. Parallel processing, power-efficient and advantageous for miniaturization.	Heavy, high cost, high-end FPGA has been used.

Table 2.1: Summary of the reference papers (cont.)

2.2 Objectives

The objectives are listed below in this section:

1. Main idea: To design a flight controller (FC) system on an FPGA board using System Verilog to program the FPGA.
2. To develop a PPM decoder unit inside FPGA to receive and decode RC signals.
3. Testing and interfacing MPU-6050 (IMU) using I2C protocol with FPGA board.
4. Implementing PID loop in FPGA.
5. In response to input from MPU and RC transmitter into PID, using motor mixing algorithms for implementing PWM encoder in FPGA for directing the input of ESC to control RPM of each motor.
6. To deploy the code in Altera Cyclone II FPGA development board, & to tune the flight parameters to achieve stable flight of the drone using PID controller.

2.3 Novelty

The code implementations of PPM Decoder, I2C Protocol, MPU Driver in Finite State Machine (FSM) approach, which provides required functionalities, and at the same time making use of the limited number of logic elements. The FC has been realized on a low-cost FPGA board.

Chapter 3

METHODOLOGY

This chapter gives an overview using a block diagram, the theory involved, and the methodology used. Also, provides the study undertaken to realize the feasibility, cost analysis, detailed analysis and modeling, experimentation, and computational works undertaken.

3.1 Block Diagram

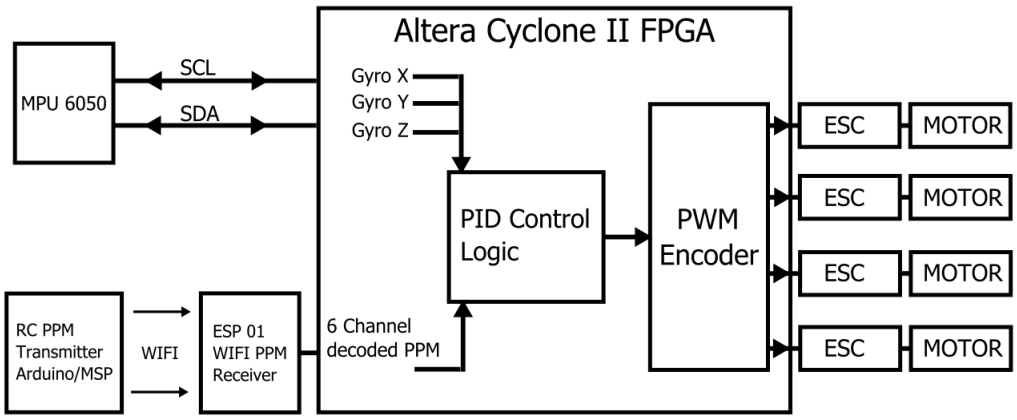


Figure 3.1: Block diagram: Flight Controller on FPGA

From Figure 3.1 the user sends commands intended to control the drone over Wi-Fi, ESP-01 module receives, and generates a corresponding PPM signal. This is send to the PPM decoder code in FPGA to get the user instructions. The Control of Flight controller begins with the receiver, which captures

stick values from the Wifi transmitter used to control the drone [5]. These values are interpreted as "intended" angular velocities for the drone about the three coordinate axes. Inside the FC, a control loop compares the intended velocities to the actual velocities reported by an on-board gyroscope module. Finally, it calculates what motor power setting would most quickly correct the error and communicates the result to the motors.

3.2 Theory & Methodology

This section gives details regarding the various modules involved to complete the system & the theory involved.

3.2.1 Pulse Position Modulation

Pulse Position Modulation (PPM) is an analog modulating scheme in which the amplitude and width of the pulses are kept constant, while the position of each pulse, concerning the position of a reference pulse, varies according to the instantaneous sampled value of the message signal.

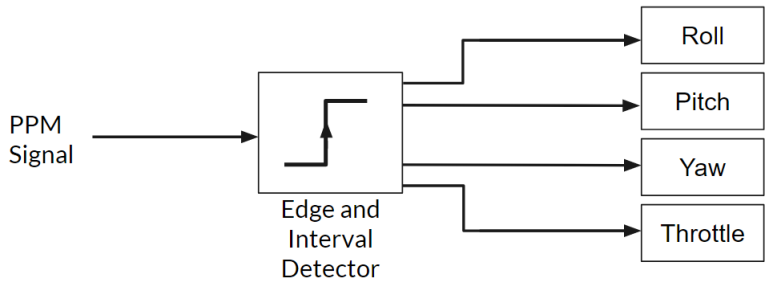


Figure 3.2: PPM - Logical Dataflow

The PPM decoder is used to interpret the pulse position modulation (PPM) signal provided by the radio receiver. Figure 3.2 illustrates the overview of what the PPM decoder is meant to achieve. The encoded PPM which is input to the PPM decoder has an edge and interval detector which gives the user inputs as output.

Each pulse of the PPM signifies each channel for flight movement control which includes roll, pitch, yaw, and throttle control signals from the

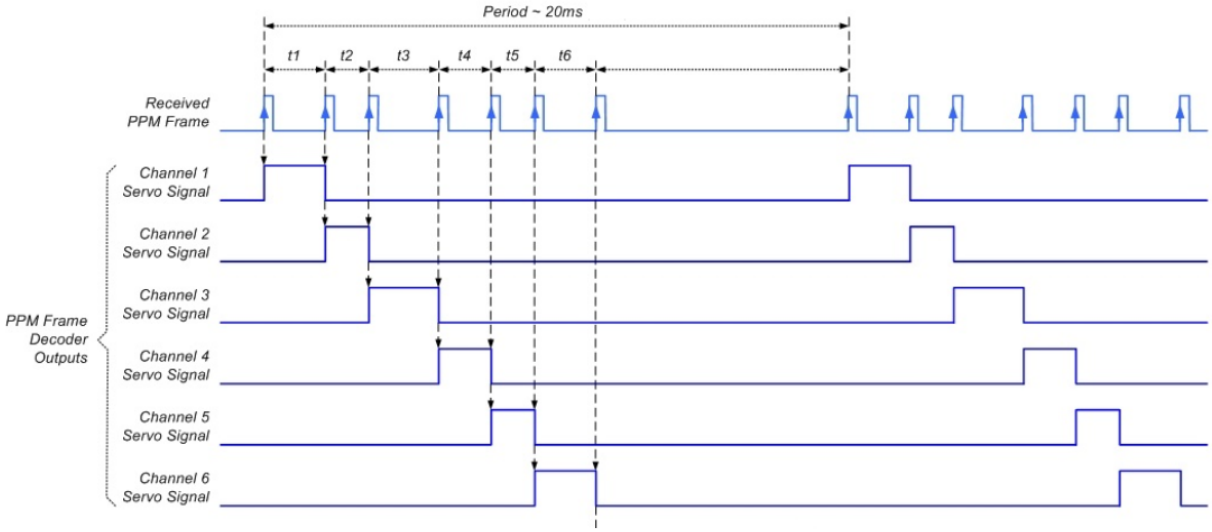


Figure 3.3: Pulse Position Modulation Decoder Dataflow

RC. In a PPM decoder, the time interval between each pulse is measured and is manipulated to the appropriate channels in the output. In an FPGA, this can be done by using a counter to measure the length of signals between two pulses and added with the pulse length of the preceding pulse to obtain each channel signal. This is the value that is set by the user. This signal from Figure 3.3 is given into the PID loop [4] for comparing this user set value with the current value from the sensor and producing the desired result signal to run the motor.

3.2.2 Pulse Width Modulation

Pulse Width Modulation (PWM) or Pulse Duration Modulation (PDM) or Pulse Time Modulation (PTM) is an analog modulating scheme in which the duration or width or time of the pulse carrier varies proportionally to the instantaneous amplitude of the message signal. The width of the pulse varies in this method, but the amplitude of the signal remains constant.

Implementation of the Pulse Width Modulation can be done using the following logic: A free-running counter (one that is continuously incremented) can be used to set an output depending on the counter's state.

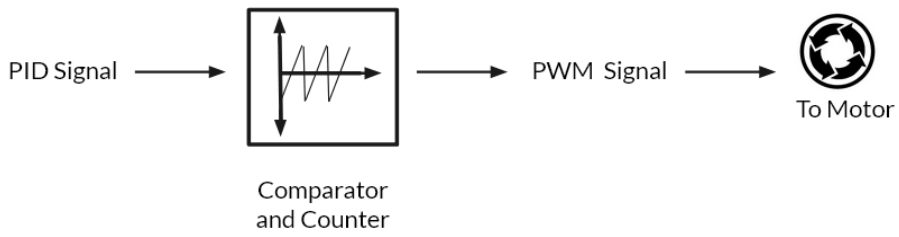


Figure 3.4: PWM - Logical Dataflow

A comparator as in Figure 3.4 is used to compare a set threshold to the counter's current value, can output 1 and 0 otherwise (or vice versa).

3.2.3 I2C (Inter Integrated Circuit)

I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line) [6, 7]. Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

With I2C, data is transferred in messages. Messages are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted.

Start Condition: The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.

Stop Condition: The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high.

The address frame includes a single bit at the end which informs the slave whether the master requires to write data to it or receive data from the same as in Figure 3.5.

Figure 3.5: I2C Frame Structure

If the master wants to send data to the slave, the read/write bit is at a low voltage level. If the master requests data from the slave, the read/write bit is at a high voltage level.

3.2.4 PID Loop

A proportional–integral–derivative controller is a control loop mechanism employing feedback that is widely used in industrial control systems.

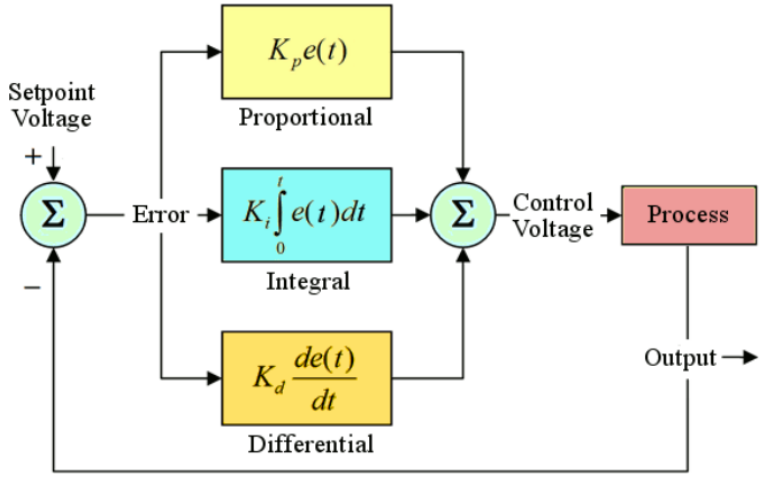


Figure 3.6: PID Controller loop

Figure 3.6 shows that a PID controller [8] continuously calculates an error value as the difference between the desired setpoint and a measured process variable and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name.

Here, PID is used for the stabilization of a drone which is used to regulate its four basic movements: roll, pitch, yaw angles, and altitude [9]. PID controllers offer a simple but effective solution to stabilize the aircraft because they make it possible to treat every variable independently within a limited range in which the behavior of the Quadcopter is approximately linear.

3.3 Detailed Analysis & Modeling

	OBJECTIVES	COMPONENTS	PURPOSE
PPM	Generation of 6 ppm channels Decoding separate channels	Arduino/MSP/ESP01 for PPM generation FPGA (decoder)	For encoding before transmission of RC inputs. For decoding the received values.
I2C	Implement I2C interface in FPGA Drive MPU-6050 using I2C	FPGA FPGA, MPU-6050	To interface with MPU-6050 To get real-time gyro values

Table 3.1: Hardware details

Table 3.1 provides the details of the objectives for choosing the components, which served it's specified purpose. Six channels of PPM have been generated using ESP-01, for encoding user inputs. Separate channels of PPM are decoded using FPGA to get user values. I2C protocol has been implemented in FPGA to interface with MPU-6050. The real-time gyro values from MPU-6050 can be obtained using I2C.

	OBJECTIVES	SOFTWARE USED	PURPOSE
PPM I2C PWM	Create main code and testbench for PPM decoding Create master code for I2C interface Interface MPU-6050 with I2C master code PWM simulation and testbench	Quartus II	Code editor, Altera-modelsim simulator, timing analyzer, synthesizer, Logic Analyzer

Table 3.2: Software details

Table 3.2 provides the details of the objectives for choosing the software, which served it's specified purpose. PPM decoder, I2C master, and PWM encoder has been written in Quartus II, by making use of it's code editor; simulated the testbenches for these using Altera-ModelSim simulator; timing analyzer to correct the timing of the signals; synthesizer to know how the logic block would be & it's resource utilization; logic analyzer for

debugging with hardware.

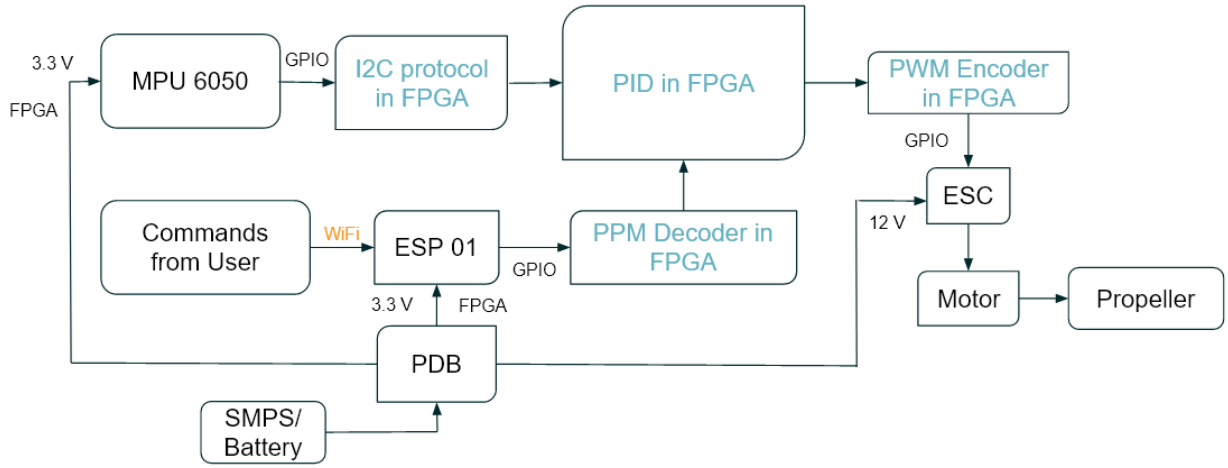


Figure 3.7: Flow Chart of the implementation

Figure 3.7 gives the flow of the whole implementation. SMPS/Battery powers the PDB. PDB powers the FPGA and ESC (12V). FPGA powers MPU-6050 and ESP-01, each 3.3V. The user sends commands via WiFi, ESP-01 receives this and PPM encodes it, then using GPIO send it to the PPM decoder in FPGA, and then to the PID module. MPU-6050 also reads real-time value and using GPIO sends it using I2C protocol to the PID module. PID does the necessary calculations and then by making use of the motor mixing algorithm provides the PWM encoded values. These values are sent to the ESC via GPIO. Based on this, each ESC decides the current to be provided to each of the motors. The propellers attached to the motors in turn give thrust to the drone to fly.

3.4 Feasibility Study

The simulator used for the software realization of the hardware design used is Altera-ModelSim. Table 3.3 shows the options at hand and the choices took. Altera-Cyclone II has been chosen as FPGA development board, as it's low cost and will suffice our requirements. MPU-6050 as the IMU since it has the required functionalities, and not much of unused features. As the FPGA chosen is Altera-Cyclone II, the IDE chosen was Quartus II. System Verilog has been used so that we get used to the latest technology. The control system taken is PID, as it's implementation is simple, and also would stabilize the drone. BLDC has been chosen as it's more durable, and larger torque.

		Hardware/Software available			**Hardware/Software selection**		
Sl No.	**Requirement in the project**	**Name**	**Features relevant to the project**	**Cost (Rs.) / License**	**Name**	**Reason**	**Cost (Rs.)**
1	FPGA Development board	Xilinx Virtex - 6 Xilinx Zynq 7000 Altera Cyclone V Altera Cyclone II	Performance, Scalability SoC, High Scalability SoC Low Cost	20000 80000 25000 2000	Cyclone II	Low Cost, Optimal	2000
2	IMU	MPU-6050 MPU-6500	6-axis IMU, DMP, 8 KHz, I2C 6-axis IMU, DMP, 32 KHz, I2C & SPI	100 200	MPU-6050	Optimal	100
3	IDE	Xilinx Vivado Quartus Prime Quartus II	For Xilinx boards For Altera boards	Free	Quartus II	Cyclone II	Free
4	Language	VHDL Verilog SystemVerilog	Non-C Syntax, HDL C Syntax, Compact, HDL C++ Syntax, HDL, HVL	-	System Verilog	Feasible	-
5	Control System	PID ML	Basic control system Complex	-	PID	Viable	-
6	Motor (4 No.s)	Coreless BLDC	Low Cost, compact Efficiency, Large torque	500 2000	BLDC	Efficiency, Durability	2000

Table 3.3: Summary of feasibility study

3.5 Cost Analysis

Table 3.4 shows the cost distribution of the components used in the project.

Sl No.	COMPONENT NAME	COST
1	POWER DISTRIBUTION BOARD WITH XT60 connector	453.
2	F450 Quadcopter Kit with 4 Pcs. A2212 KV1000 Brushless Motor and 4 Pcs. 30A ESC and 4 Pair 1045 Propeller	3090
3	ALTERA FPGA Cyclone II EP2C5T144 System Development Board	1624
4	USB Blaster ALTERA CPLD/FPGA Programmer	699
5	ESP-01 ESP8266 Serial WIFI Wireless Transceiver Module	208
6	Male To Female Jumper Wires 40 Pcs 10cm	55
	Total Cost	**6129**

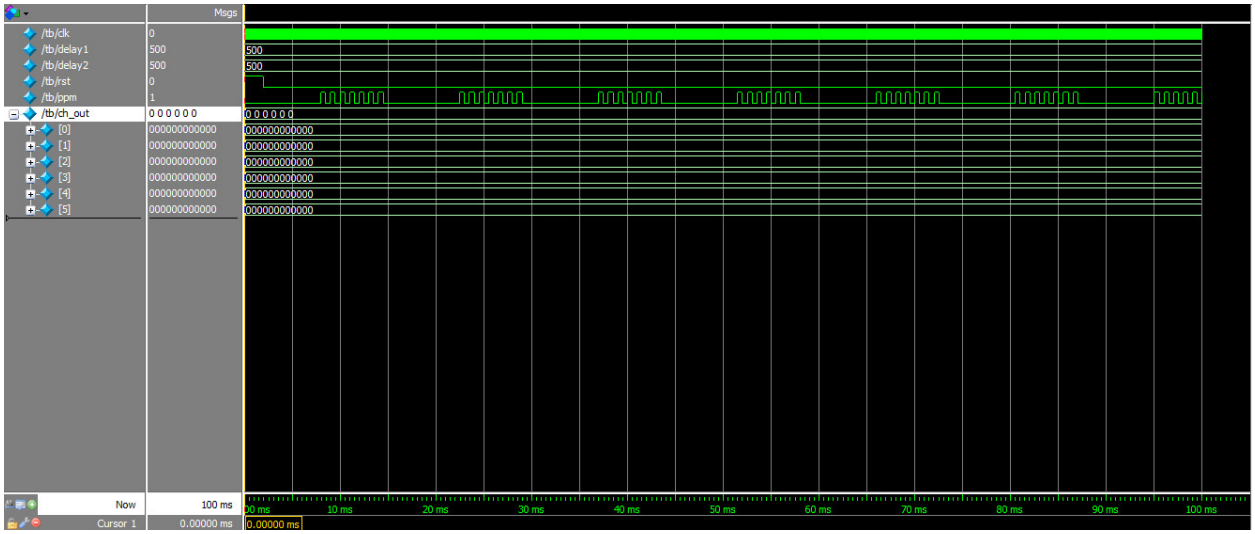
Table 3.4: Cost distribution of components used

3.6 Experimentation & Computational Works

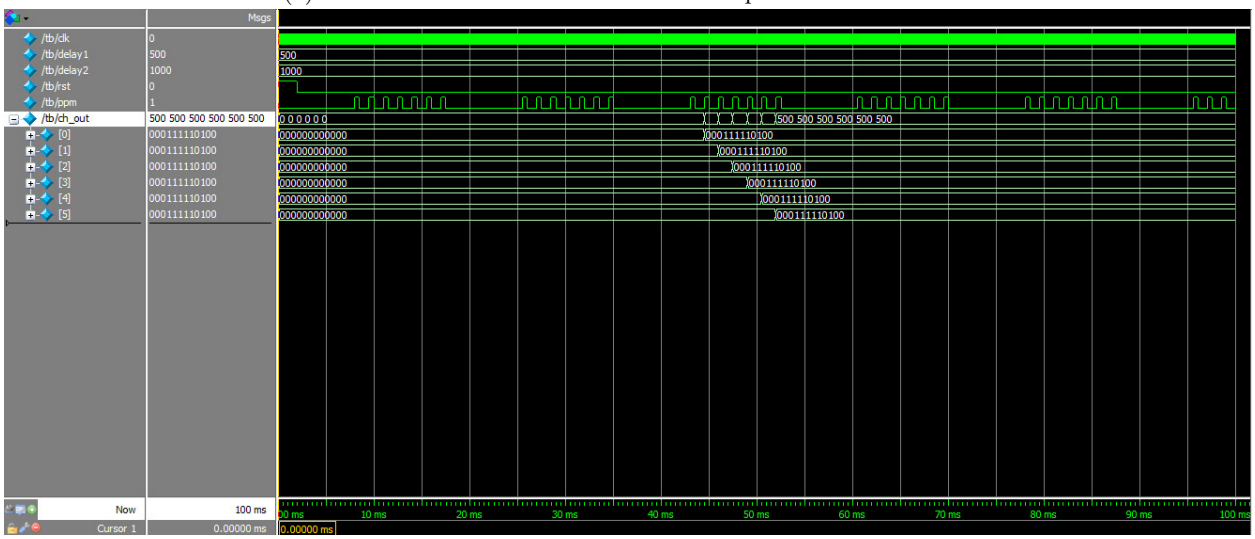
3.6.1 Pulse Position Modulation

The PPM signal is a 6 channel encoded signal consisting of Throttle, roll, pitch, yaw, arm, Custom as the channel values, which are measured by incrementing a counter at each posedge of the incoming PPM signal. The Custom value can be programmed for any functionality. Here, 'clk' is the master clock having 1 MHz frequency, 'ppm' is the PPM signal as input to the receiver, 'ch_out' is a pos-edge incremental counter which counts the clock signal, 'ch0', 'ch1', 'ch2' & 'ch3' are channel values used in Tcl console and 'ch_out' is a bus which consists of the decoded PPM signal as 6 separate channels for flight movement control in the form of servo signals.

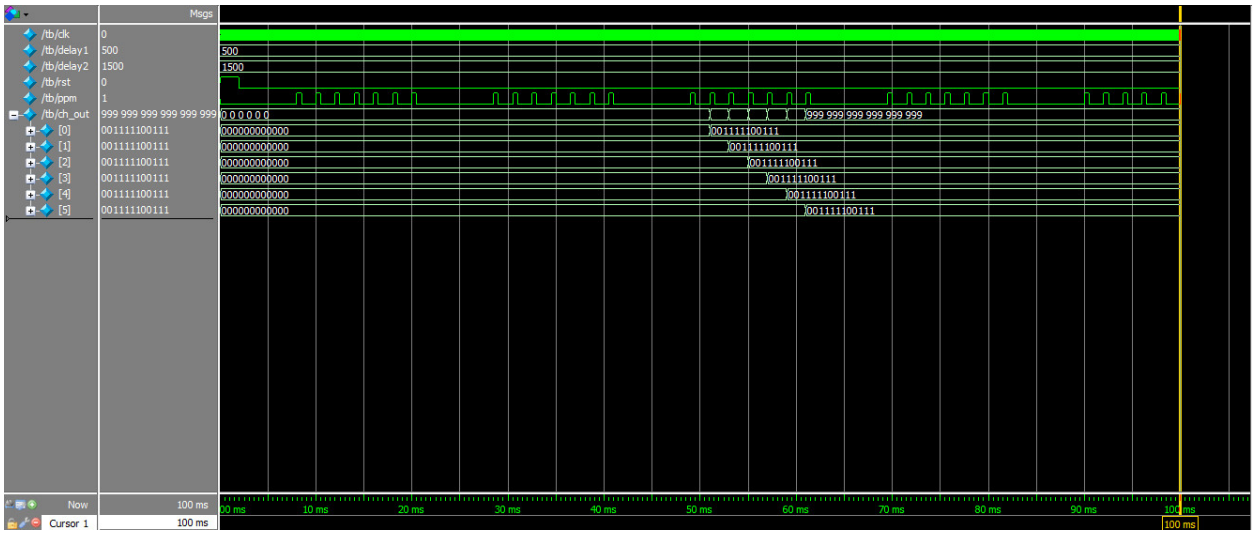
Figure 3.8(a) to 3.8(c) shows the simulation output of the PPM decoder.



(a) PPM decoded for zero values of user input for all 6 channels



(b) PPM decoded for half of the maximum values possible of user input for all 6 channels

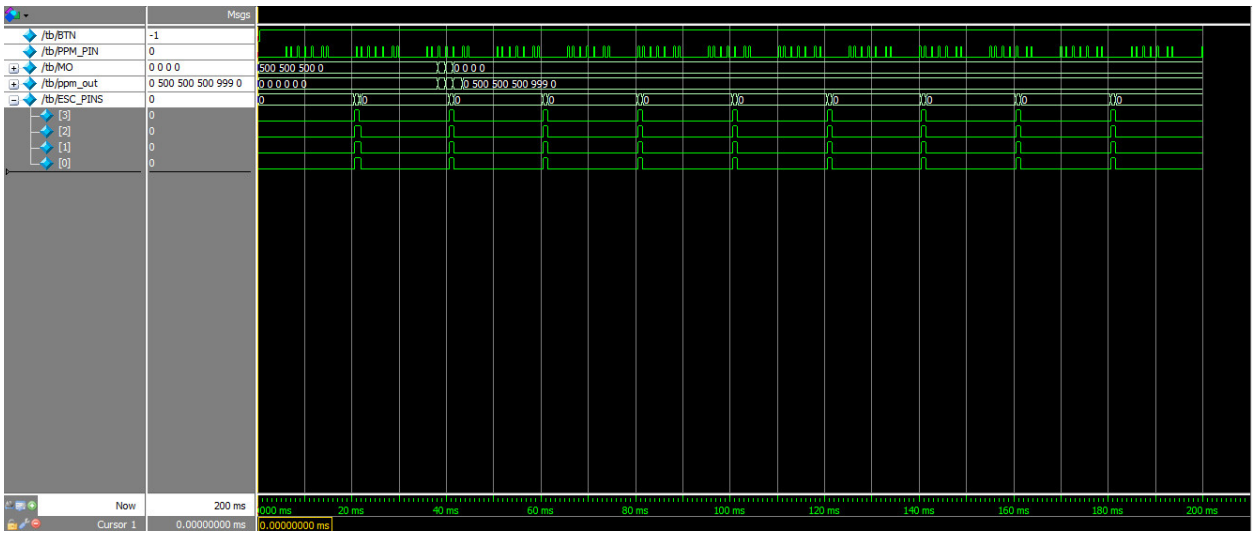


(c) PPM decoded for maximum values possible of user input for all 6 channels

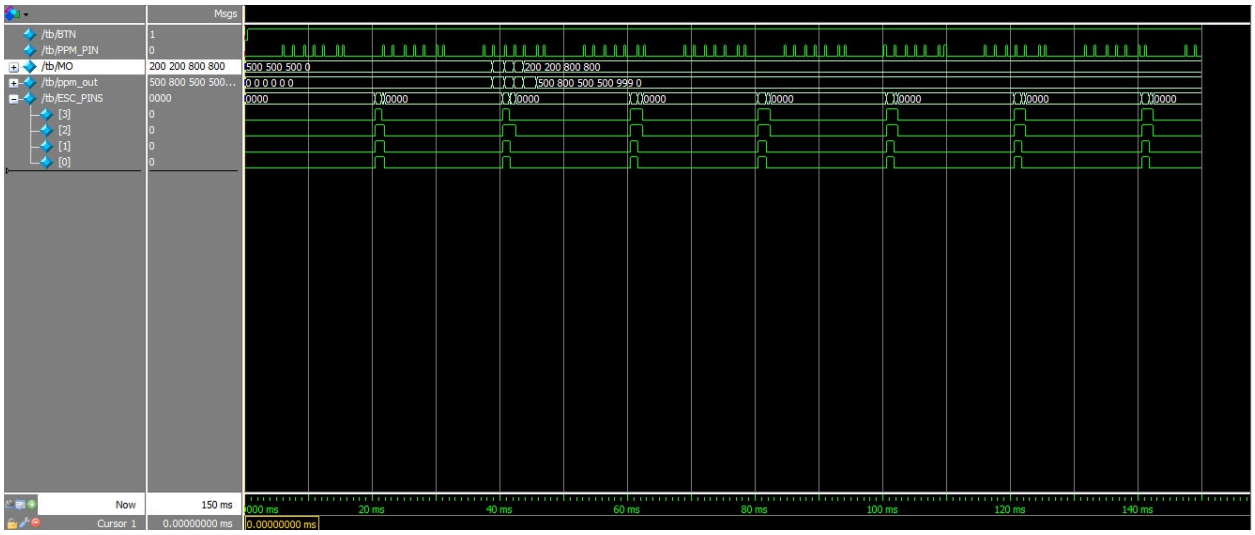
Figure 3.8: Pulse Position Modulation - Simulation Output

3.6.2 Pulse Width Modulation

Figure 3.9(a) to 3.9(c) shows the simulation output of a PWM encoder. Here, master clock runs at 1MHz, ESC pins are incremental pos-edge counter which counts the clock signals that resets after a full PWM cycle is completed and the PWM encoded output signal. The signal shown below is a Pulse Width Modulated signal having a duty cycle of 6% to 8%. The duty cycle of the pulse varies between 1.2 ms and 1.6 ms depending on the amount of throttle, pitch, and yaw required by each motor. The total duration of a single pulse is 20ms.

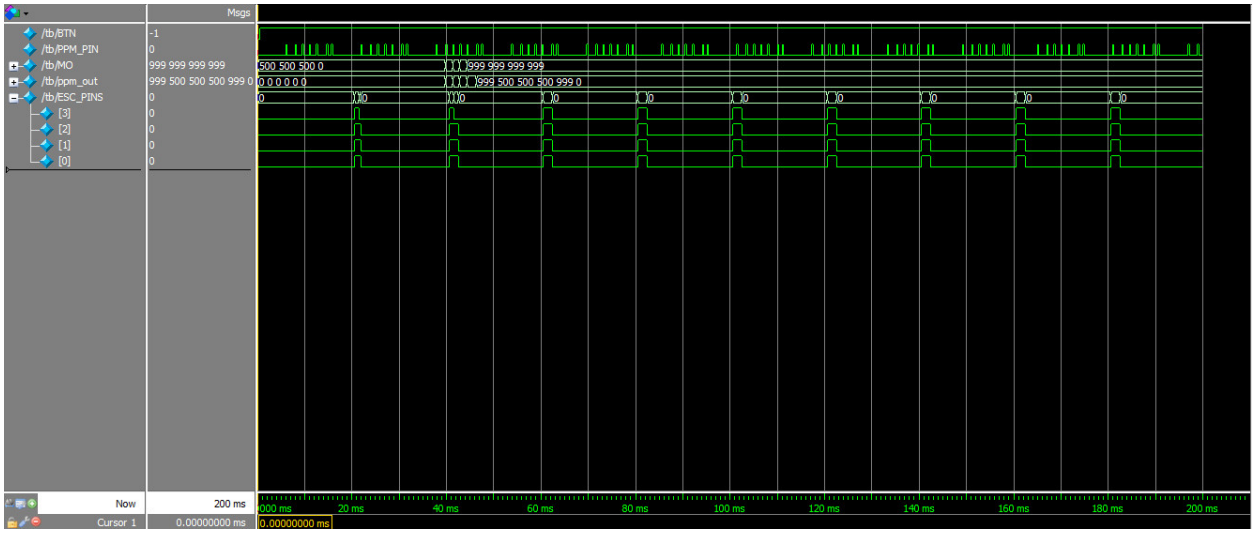


(a) PWM encoded for zero values for all 6 channels of PPM decoded input



(b) PWM encoded for roll value of PPM decoded input

Figure 3.9: PWM Encoder - Simulation Output



(c) PWM encoded for maximum for all 6 channels of PPM decoded input

Figure 3.9: PWM Encoder - Simulation Output (cont.)

3.6.3 I2C

Figure 3.10 shows the simulation output of the I2C protocol. Here, the master clock runs at 50MHz, The first 8 bits after start condition is the slave address bus, 'data_in' is another bus that holds the decoded data transmitted into the FPGA from the sensor, 'enable' shows if the I2C signal line is currently in use or not, 'rw' shows if the mode set is either read or write data, 'data_out' is a bus used to send out the data from the FPGA, 'i2c_sda' is the data line and 'i2c_scl' is the clock signal of the I2C protocol.

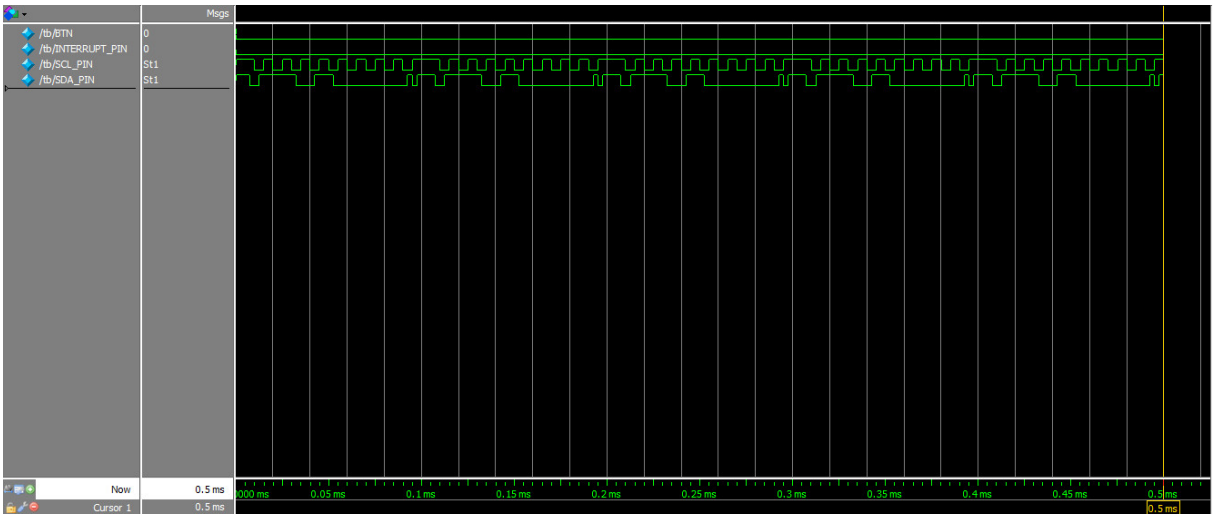


Figure 3.10: I2C - Simulation Output

Chapter 4

RESULTS AND ANALYSIS

This chapter lists the results obtained in software and the validation of the same using hardware.

4.1 Results

- In this work, an FPGA-based flight controller of a Quadcopter was designed.
- Figure 4.1 shows the resource utilization of the Altera Cyclone II FPGA for the design.

Flow Status	Successful - Wed May 19 18:19:45 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ppm_decoder
Top-level Entity Name	main
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Total logic elements	3,715 / 4,608 (81 %)
Total combinational functions	3,475 / 4,608 (75 %)
Dedicated logic registers	1,373 / 4,608 (30 %)
Total registers	1373
Total pins	10 / 89 (11 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	26 / 26 (100 %)
Total PLLs	0 / 2 (0 %)

Figure 4.1: Resource Utilisation of Altera Cyclone II FPGA

- Figure 4.2 shows the RTL block diagram of the design.

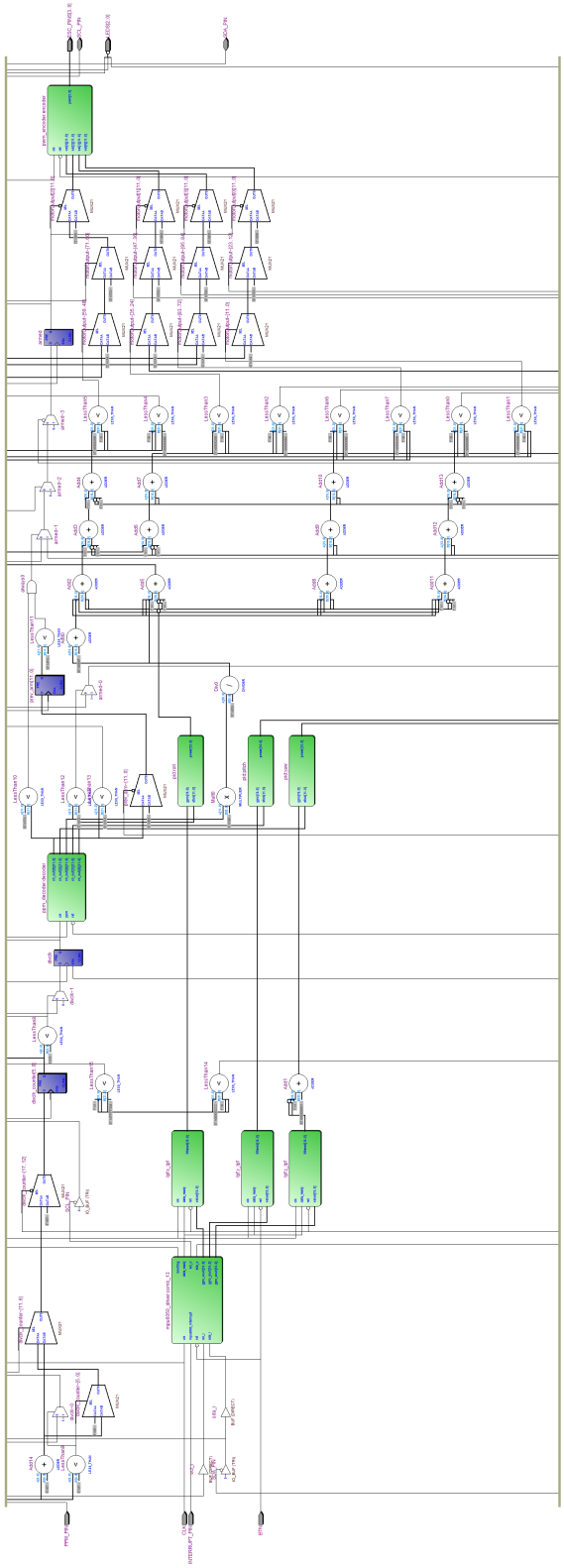


Figure 4.2: RTL Block Diagram

4.2 Validation of Results

The FPGA implemented output provided by Signal Tap II Logic Analyzer present in Quartus II IDE is shown in Figure 4.3. The BTN pin is used to reset the flight controller and is active low. PPM_PIN is the pin used to input the signal from the receiver to FPGA.

Type	Alias	Name	Value
in		BTN	1
in		PPM_PIN	0
R		...r:decoder|ch[0][11..0]	500
R		...r:decoder|ch[1][11..0]	707
R		...r:decoder|ch[2][11..0]	400
R		...r:decoder|ch[3][11..0]	506
R		...r:decoder|ch[4][11..0]	0
R		...r:decoder|ch[5][11..0]	999
io		SCL_PIN	1
io		SDA_PIN	0
out		gyro_xout	14515
out		gyro_yout	1666
out		gyro_zout	-3602
C		...ncoder:encoder|val[0]	555
C		...ncoder:encoder|val[1]	190
C		...ncoder:encoder|val[2]	190
C		...ncoder:encoder|val[3]	567
out		b_right	555
out		f_right	153
out		f_left	141
out		b_left	567

Figure 4.3: Logic analyzer

The channels ch[0] to ch[5] are 12-bit registers used to store the decoded stick values from the PPM signal and are defined as follows:

- ch[0] - Roll register
- ch[1] - Pitch register
- ch[2] - Throttle register

- ch[3] - Yaw register
- ch[4] - Buffer register
- ch[5] - Arm register

The FPGA starts reading the stick values when the 6th channel, ch[5], is HIGH (with value 999). gyro_xout, gyro_yout and gyro_zout are registers used to store the raw gyroscope register measurements of X, Y, and Z axes of the MPU-6050 sensor. The full-scale range is set to 1000 degrees/second. The raw register values vary from -32767 to +32767 and to obtain the measurement in degrees/second, the raw values are divided by the LSB sensitivity of 32.8 LSB/degrees/second. The internal low pass filter of MPU-6050 was turned on, with a cut-off frequency of 5 Hz, to filter out noise.

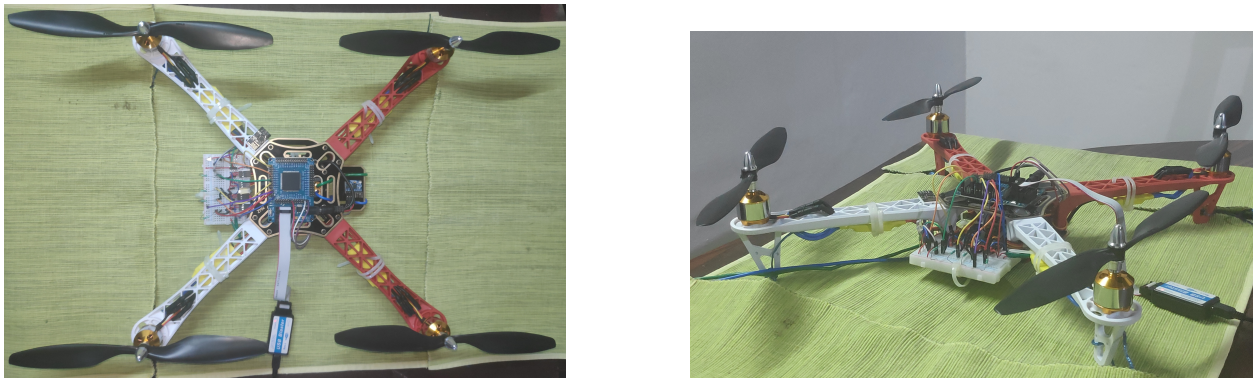
The PID loop takes in the stick values (setpoint) and the current gyroscope values and evaluate a signal to eliminate the error difference between these two values each for roll, pitch, and yaw. These values are used to evaluate the signals to be provided to each motor [1] using a motor mixing algorithm as shown in Figure 4.4.

```
Front Left  = Throttle + Roll - Pitch - Yaw
Front Right = Throttle - Roll - Pitch + Yaw
Back  Left  = Throttle + Roll + Pitch + Yaw
Back  Right = Throttle - Roll + Pitch - Yaw
```

Figure 4.4: Motor Mixing Algorithm

Using the motor mixing algorithm [3], we get the values that are to be provided to the motor to obtain the desired orientation for the quadcopter. These values are shown as b_right, f_right, f_left, and b_left.

The values provided to the motor are limited between 190 and 600. These limited values are stored in the registers val[0] to val[3]. The output to a motor from the FPGA can be given to an Electronic Speed Controller (ESC) for each motor as a servo signal at a rate of 50 Hz. The ESC works at



(a) Top View

(b) Side View

Figure 4.5: FPGA FC on Quadcopter

12 V which can be provided by a 12 V power supply or sufficiently powered 12 V Li-Po battery.

Figure 4.5 shows the drone built from scratch with the proposed FPGA based FC. The PID tuning was achieved using Ziegler-Nichols method. It involves setting the P value to a constant value (typically 1) and I and D values to 0. The P value is increased in steps until a standing oscillation in the output is observed, i.e., until the system becomes unstable. The I value is now increased in steps to dampen out the oscillation within a particular time. The D value is then increased in steps to reduce the overshoot. The resultant will be a stabilised system.

Figure 4.6 shows the remote control interface of the Wifi Transmitter implemented in ESP-01.

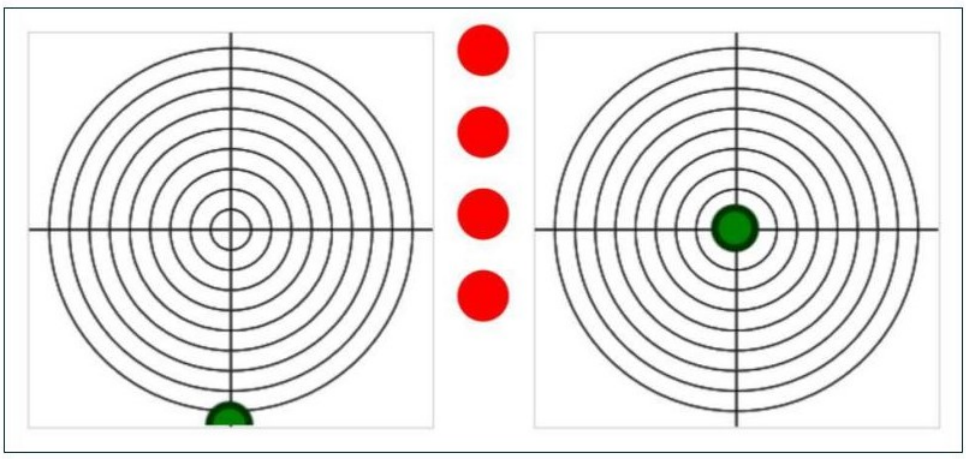


Figure 4.6: Wifi PPM Transmitter Interface

The controls on the left side is used for throttle and yaw in the vertical and horizontal directions respectively. The controls on the right side is used for pitch and roll in the vertical and horizontal directions respectively. The second button out of the four buttons is used for arming the flight controller.

Figure 4.7 shows the FPGA-Based Flight Controller Drone in flight.

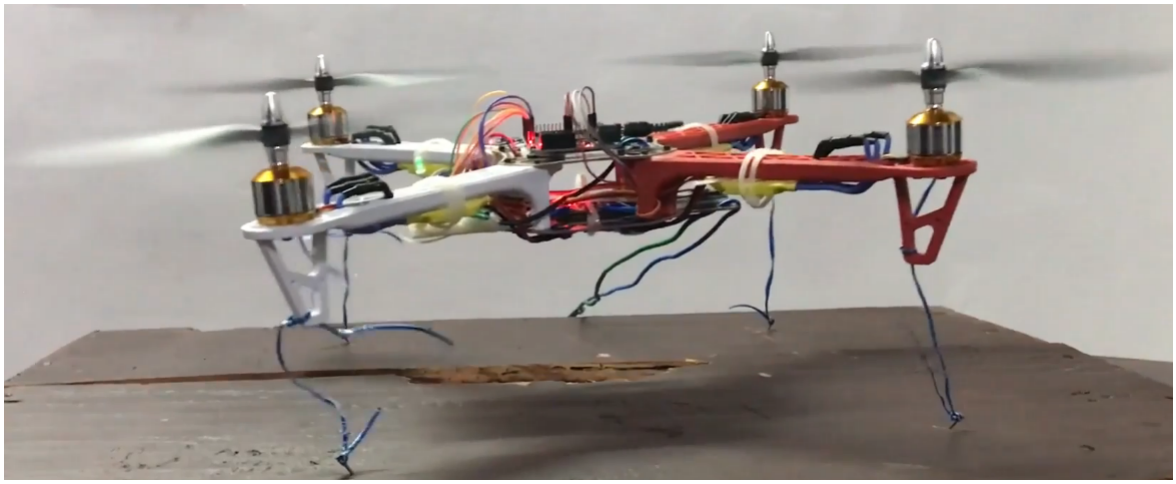


Figure 4.7: FPGA-Based FC Drone in Flight

Chapter 5

CONCLUSION

This chapter provides the conclusion to the work done as part of this project, pros & cons, and the future scope.

5.1 Conclusion

The user inputs which are PPM encoded are received by FPGA, decoded, & fed into the PID. In parallel, the real-time values from the onboard gyro are fed into PID. PID has been tuned for the system, and then makes use of the motor mixing algorithm to provide the PWM encoded values to the ESC, which in turn gives it to the motors which provide the thrust to the drone. Implementing a Flight Controller on FPGA has been successfully achieved.

5.2 Advantages & Disadvantages

Advantages

- A high level of parallelism can be achieved by using FPGA-based designs compared to the serial execution in microcontroller-based designs.
- Increased Response to changing parameters can be achieved.

Disadvantages

- More development time.
- Highly skilled crew required.

5.3 Future Scope

- Integration of various sensors into the flight controller to improve stability.
- Incorporation of ML/Kalman Filter in addition to the PID controller in the system.
- Developing an MVP from the project.

Bibliography

[1] N. Monterrosa, J. Montoya, F. Jarquín, and C. Bran, "Design, development and implementation of a uav flight controller based on a state machine approach using a fpga embedded system," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016, pp. 1–8.

[2] B. L. Sharma, N. Khatri, and A. Sharma, "An analytical review on fpga based autonomous flight control system for small uavs," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, pp. 1369–1372.

[3] G. Premkumar, R. Jayalakshmi, and M. Akramuddin, "Design and implementation of fpga based quadcopter," in *2018 International Journal of Engineering Technology Science and Research, Premkumar2018DesignAI*, vol. 5, no. 3, 2018, pp. 1–5.

[4] V. Dubreuil and A. V. Osintsev, "Designing multiple pid controllers based on an fpga for controlling the temperature of tem-cell surfaces," in *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, 2019, pp. 0194–0198.

[5] E. Lee, S.-J. Jang, and S.-S. Lee, "An implementation of the system on chip control system for a fpga-based computer vision accelerator," in *2019 International SoC Design Conference (ISOCC)*, 2019, pp. 299–300.

[6] L. A. Prasanna Bagdalkar, "Hardware implementation of i2c controller on fpga and validation through interfacing with low-cost adc," in *Proceedings*

of the Fourth International Conference on Inventive Systems and Control (ICISC), 2020.

[7] R. S. S. Kumari and C. Gayathri, "Interfacing of mems motion sensor with fpga using i2c protocol," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1–5.

[8] I. Z. Székely and S. T. Brassai, "Quadcopter control implemented on fpga*," in *2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo)*, 2019, pp. 000 049–000 054.

[9] S. James, "fpga-flight-controller," https://github.com/SebastianJames55/fpga-flight-controller, last accessed on 9 June, 2021.