FPGA BASED FLIGHT CONTROLLER

Athul John Kurian
Electronics & Communication
Govt. Model Engineering College, Thrikkakara
Ernakulam, India
athuljohnkurian.mec@gmail.com

Medha Lakshman Rao
Electronics & Communication
Govt. Model Engineering College, Thrikkakara
Ernakulam, India
medhalakshmanrao.mec@gmail.com

Omar Bin Shafi
Electronics & Communication
Govt. Model Engineering College, Thrikkakara
Ernakulam, India
omarbinshafi.mec@gmail.com

Sebastian James
Electronics & Communication
Govt. Model Engineering College, Thrikkakara
Ernakulam, India
sebastianjames.mec@gmail.com

Abstract— **This work aims at designing and implementing a flight controller on an FPGA (Field Programmable Gate Array) which is used to control an aerial vehicle. The brain of any aircraft is the Flight control system (FCS), especially in UAVs. The purpose of the project was to develop a stable and feasible UAV like a quadcopter by using an FPGA. FPGAs are in general good for implementing parallel processing. 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, functionality modification, and compatibility with numerous applications.**

Keywords— **Channel, Comparator, Counter, Hardware Description Language, Logic Analyzer, Motor Mixing Algorithm, Servo Signal**

I. Introduction

Recently, several popular applications make use of remote-controlled or autonomously flown aircraft knows as drones. Numerous solutions rely on embedded sequential systems, however, the greater the number of processes these systems execute the more they contribute inversely towards variables such as precision, speed of response, and synchronism [1]. Here, a solution for the UAV's Flight Controller integrated into an FPGA, which is an embedded concurrent system is in focus. FPGA has the capacity for the calculation to extend the intelligence of the system with more autonomy, excellent synchronism, and improved response times.

In the event of a disturbance, the Flight Controller System (FCS) will assist the user with automatic compensation in all drone maneuvers. FPGA allows parallel processing (via pipelining) and has a large number of gates per area, resulting in less volume and a lower-cost platform. The systems' software provides code parallelization, resulting in faster speeds in the FPGA and other hardware components, with a significant impact on the overall design [2]. The Radio Frequency (RF) receiver receives the commands sent by the user. These system inputs and the sensor signal provide direction for the aircraft and position in real-time.

II. Related Work

FCS based on the State Machine approach [1] involves acting on outputs from the gyroscope, thereby moving from one state to another. The control signals sent by the user, through sticks in the remote control (RC) are PPM encoded and transmitted via an RF transmitter. These values are decoded at the drone end and sent to the FCS. The values from the sensors like a gyroscope as part of the Inertial Measurement Unit (IMU), provides the real-time position of the drone to the FCS. IMU is interfaced with FPGA by using the I2C protocol [3]. The FCS will have control algorithms like PID (Proportional Integral Derivative) [2,3] to calculate the states. FCS can also be implemented by using Kalman Filter in addition to PID or by using a fuzzy logic-based controller, thereby improving performance. FCS acts based on the state it is in, with respect to the set of values received. Further integration of sensors like accelerometer, ultrasonic, altimeter, etc. leads to accurate determination of states. Based on the states, the FCS output is PWM encoded which are inputs to the motors providing thrust to the drone [1,3].

III. Method

This section explains the block diagram and the methodology of implementation of the flight controller.

A. Block Diagram

The control of the Flight Controller begins with the radio transmitter, which sends encoded signals of each channel. These are received by the receiver and the decoded signal of the stick values from the radio transmitter are used to control the drone [1-3]. These values are interpreted as and correspond to the orientation required along the three axes of the UAV. The FC has a control loop that compares the intended orientation to the actual orientation obtained by an onboard gyroscope

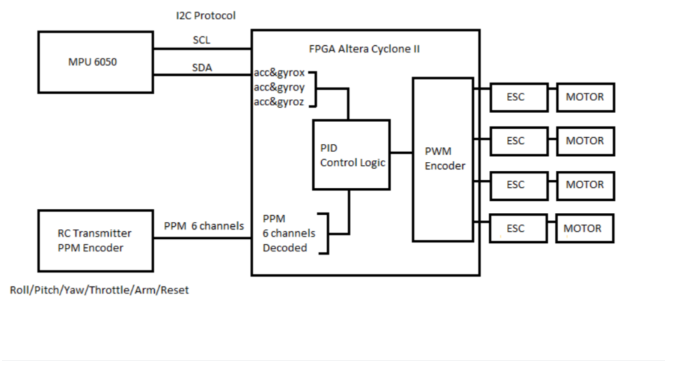


Fig. 1. Block diagram: Flight Controller on FPGA

module and produces an error. Finally, the required motor power setting is calculated and the error is corrected and the resultant value is given as input to the motors, as shown in Fig. 1.

B. Methodology

1) Pulse Position Modulation:

PPM (Pulse Position Modulation) is a modulating technique in which the width and amplitude of the pulses are held constant whereas the pulse position varies based on an instantaneous sampled message signal, usually a PWM signal.

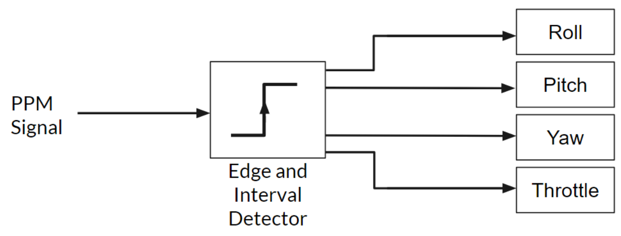


Fig. 2. PPM - Logical Dataflow

Fig. 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.

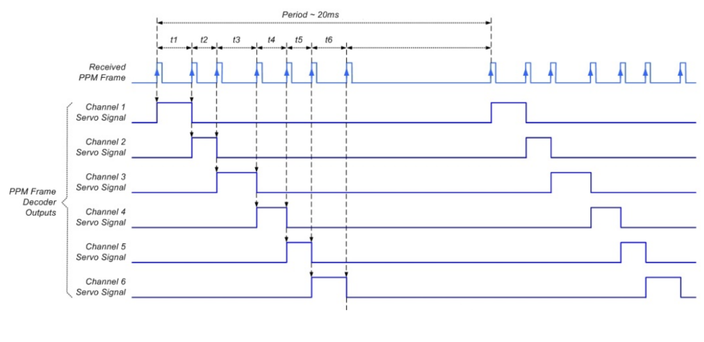


Fig. 3. Pulse Position Modulation Decoder Dataflow

From Fig. 3, we can observe that each pulse of the PPM signifies each channel for flight movement control which includes roll, pitch, yaw, and throttle control signals from the 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 a pulse length of the preceding pulse to obtain each channel signal. This value is set by the user. This signal is given into the PID loop for comparing this user set value with the current value from the sensor and producing the desired result signal to run the motor.

2) Pulse Width Modulation:

Implementation of the Pulse Width Modulation (PWM) encoder can be done using the following logic: The signals representing each channel from the PID loop are compared against a counter until which the output of the encoder is set to HIGH.

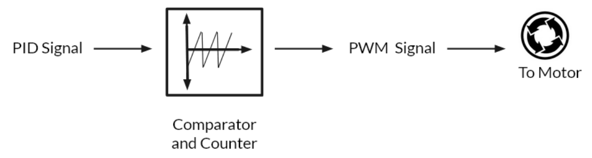


Fig. 4. PWM - Logical Dataflow

From Fig. 4, we can see that a counter (one that is continuously incremented up to a set-point) can be used to set an output depending on the counter's state. Using a comparator to compare a set threshold to the counter's current value, we can output HIGH or LOW (or vice versa).

This signal is given to the motors via Electronic Speed Controllers (ESCs) which accept servo signals as their input and provides necessary output signals to drive the motors [1,3].

3) I2C (Inter Integrated Circuit):

I2C is a widely used serial communication protocol in various electronic devices. It transfers data bit by bit on a single path called the SDA line. I2C, like SPI, is synchronous, which means that the output is synchronized to the sampling of bits via a common clock signal between the master and slave devices. The master controls the clock signal. Data transferred in I2C are in the form of messages at high frequencies like 100 KHz or 400 KHz [4,5].

Messages are broken down into data frames. Each I2C message has an address that specifies the slave device's binary address, as well as one or more data frames that specify the data being exchanged between master and slave.

Start Condition: When the SDA (serial data) line switches from a high voltage level (5V) to a low voltage level (0V) before the SCL (serial clock) line switches from high to low, I2C communication begins.

Stop Condition: When the SDA (serial data) line switches from a low voltage level (0V) to a high voltage level (5V) after the SCL (serial clock) signal line switches from low to high, the I2C communication comes to an end.

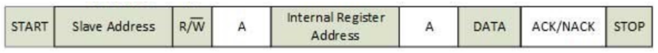


Fig. 5. I2C - Frame Structure

Fig. 5 shows that the address frame includes a Read/Write (R/W) bit at the end which informs the slave whether the master requires a data write to the slave or data received from the slave. The R/W bit is written to a low voltage level if the master wants to send data to the slave or it is written to a high voltage level when the master requests data from the slave.

4) *PID Controller:*

A PID controller is a control loop mechanism providing feedback that is extensively used in industrial control systems and also in a variety of other applications that requires continuously modulated control [6,7]. Here, the PID control loop is used to stabilize the quadcopter that is used to regulate its four basic control movements: roll, pitch, yaw, and throttle [8]. As it considers each variable independently in a limited range in which the quadcopter's behavior is approximately linear, PID controllers provide a simple but effective solution for stabilizing the UAV. A PID controller continuously calculates an

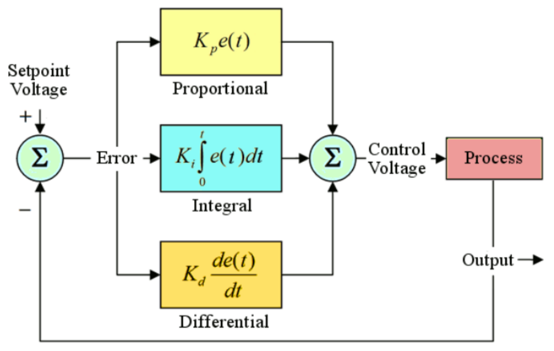


Fig. 6. PID Controller Loop

error value as the difference between the desired setpoint and a measured process variable (denoted P, I, and D, respectively) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D, respectively), as shown in Fig. 6.

IV. Implementation

This section explains the feasibility study, simulations, and the FPGA hardware implementation of the flight controller.

A. *Feasibilty study*

1) For the FPGA development board, the board chosen is Altera Cyclone II.
2) MPU 6050 Inertial Measurement Unit is chosen as it is a 6 Axis IMU, has I2C communication, and is cost-effective.
3) Arduino UNO is used to generate PPM signals which provide the input stick values to the flight controller.
4) PID control loop is used for the control system as it is a simple and viable solution.
5) IDE used is Quartus II and the simulator used for the software realization of the hardware is Altera-ModelSim.
6) System Verilog is chosen as the hardware description language for implementing the logic of the project.

B. *Simulations*

1) *Pulse Position Modulation:*

Fig. 7 shows the simulation output of a PPM decoder.

Here, 'clk' is the master clock having 1 MHz frequency, 'ppm' is the PPM signal as input to the receiver, 'ch_count' is a pos-edge incremental counter which counts the clock signal, 'ch0', 'ch1', 'ch2' & 'ch3' are test variables used in Tcl console and 'ch' is a bus which consists of the decoded PPM signal as 4 separate channels for flight movement control in the form of servo signals.

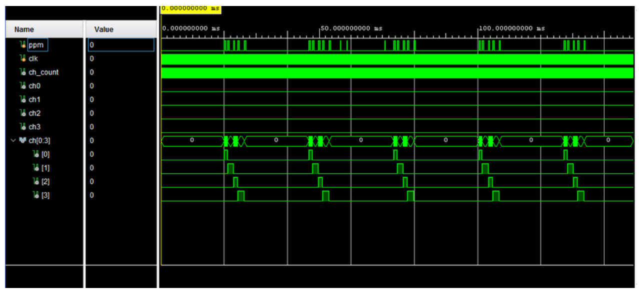


Fig. 7. Pulse Position Modulation - Simulation Output

2) *Pulse Width Modulation:*

Fig. 8 shows the simulation output of a PWM encoder.

Here, 'clk' is the master clock, 'counter' is an incremental pos-edge counter which counts the clock signals that resets after a full PWM cycle is completed and 'led' is the PWM encoded output signal. The currently shown signal is a PWM signal with a duty cycle of 20%.

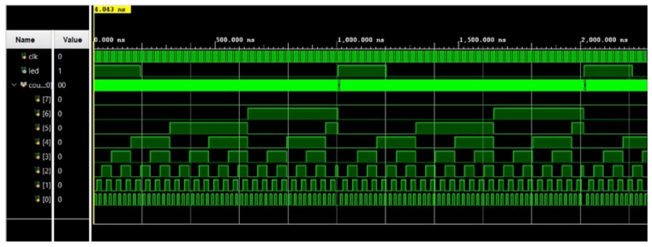


Fig. 8. Pulse Width Modulation - Simulation Output

3) I2C (Inter Integrated Circuit):

Fig. 9 shows the simulation output of the I2C protocol.

Here, ‘clk’ is the master clock, ‘addr’ is the decoded slave address bus, ‘reg_addr’ is the bus holding the decoded register address of the register inside the MPU-6050 sensor, ‘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.

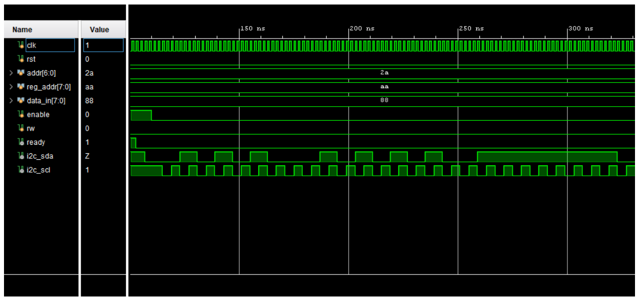


Fig. 9. I2C - Simulation Output

C. FPGA Implementation - Logic Analyzer

All the modules - PPM decoder, I2C, IMU interface, PID controller, and PWM encoder were implemented on Altera Cyclone II FPGA board.

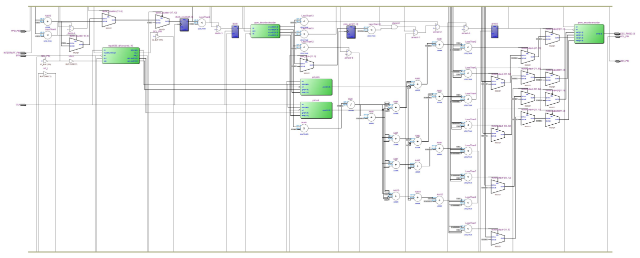
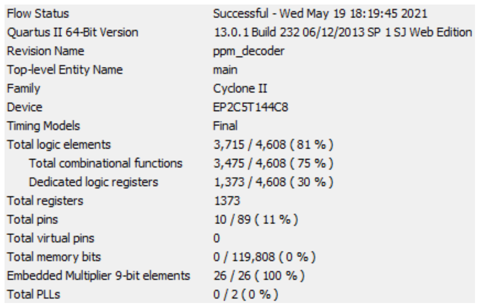


Fig. 10. FPGA Synthesis Result - Quartus II

The synthesis result is provided in Fig. 10.



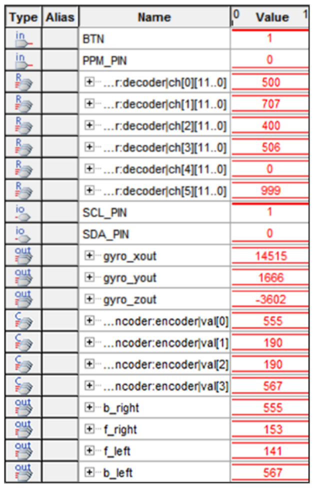
Flow Status	Successful - Wed May 19 18:19:45 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ppm_decoder
Top-level Entity Name	main
Family	Cyclone II
Device	EP2C5T144C8
Timing Models	Final
Total logic elements	3,715 / 4,608 (81 %)
Total combinational functions	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 %)

Fig. 11. FPGA Resource Utilization - Quartus II

Fig. 11 lists the resource utilization on synthesis.

The FPGA implemented output provided by Signal Tap II Logic Analyzer present in Quartus II IDE is shown in Fig. 12. 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. 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



Type	Alias	Name	0 Value 1
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

Fig. 12. Signal Tap II Logic Analyzer Output

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. Internal low pass filter of MPU-6050 was turned on, with a cut off frequency of 42 Hz, to filter out noise.

The PID loop takes in the stick values (setpoint) and the current gyroscope values and evaluate a signal so as 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 Fig. 13.

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

Fig. 13. 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 12 V which can be provided by a Li-Po battery or any other power source for testing the working of the drone.

V. Result and Conclusion

The flight controller was designed and implemented in the Altera Cyclone II FPGA board. The PPM signals were generated using an Arduino UNO micro-controller board which was provided to the FPGA. The IMU used was an MPU-6050 sensor. PID controller was the control system used. The simulations were used for testing purposes. The hardware was implemented successfully and desired results were achieved. If techniques like machine learning are implemented, we can extend the use of the drone for performing automated operations without human control. Integration of various sensors into the flight controller to improve stability and developing a minimum viable product (MVP) from the project is also one of the future scopes of this project.

References

[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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] K. Black, “fpga-flight-controller,” https://github.com/kvablack/fpga-flight-controller, 2019.