Hardware Design and Implementation of a MAVLink Interface for an FPGA-Based Autonomous UAV Flight Control System

Blake Fuller, Jonathan Kok, Neil Kelson, Felipe Gonzalez
Queensland University of Technology, Australia
{b5.fuller,j.kok,n.kelson,felipe.gonzalez}@qut.edu.au

Abstract

This paper details the initial design and planning of a Field Programmable Gate Array (FPGA) implemented control system that will enable a path planner to interact with a MAVLink based flight computer. The design is aimed at small Unmanned Aircraft Vehicles (UAV) under autonomous operation which are typically subject to constraints arising from limited on-board processing capabilities, power and size. An FPGA implementation for the design is chosen for its potential to address such limitations through low power and high speed in-hardware computation. The MAVLink protocol offers a low bandwidth interface for the FPGA implemented path planner to communicate with an on-board flight computer. A control system plan is presented that is capable of accepting a string of GPS waypoints generated on-board from a previously developed in-hardware Genetic Algorithm (GA) path planner and feeding them to the open source PX4 autopilot, while simultaneously responding with flight status information.

1 Introduction

Traditionally thought of as purely within the realm of military applications, now unmanned aerial vehicles (UAV) have their place in civilian applications [Cox *et al.*, 2004] [Konyushko, 2013]. Examples of commercial and industrial settings where UAV are effectively utilised include, agriculture surveillance [Barrientos *et al.*, 2011], emergency services [DeBusk, 2010] and power line inspection [Li *et al.*, 2012]. However, flight computing systems on current UAVs lack the capability for on-board processing of autonomous real-time processing tasks, such as image processing, path planning and collision avoidance. Field programmable gate arrays (FPGAs) are advanced reconfigurable processing hardware capable of high-performance computing through parallel and logic-level processing. Being compact in size, light in weight and low in power-consumption, FPGAs are well suited for integration as an on-board flight computer on UAVs [Guanglin *et al.*, 2007].

Modern autopilots are intelligent flight system controllers that, together with position data from Global Positioning System unit and attitude data from inertial measurement unit, convert high-level navigation commands into servo signals to bring an aircraft from its current position and attitude to a new desired position and attitude [Xiao *et al.*, 2011] [Ortner, 2009]. From a hardware platform perspective, an FPGA-based flight computer needs to be physically connected to the autopilot of the UAV via serial connections. To communicate effectively with the autopilot, data has to be sent and received according to pre-defined protocols of the autopilot. This work proposes a hardware design architecture for an FPGA-based flight computer that allows *Internal Components* (such as path planning) to communicate with the autopilot via a *Control System* interface. For the purpose of this work, the target UAV is an AR.Drone platform with a PX4 autopilot that communicates via MAVLink protocol. The design of the proposed architecture encompasses the interaction between *Internal Components* of a path planner with a *Control System* that transforms the data into communication packets compatible of MAVLink protocol.

The proposed architecture was encoded in Very High Speed Integrated Circuit Hardware Description Language (VHDL). Synthesis results show that the FPGA-based flight computer architecture utilised roughly 1,000 logic cells with a maximum operating frequency of 900 MHz. The results obtained thus far are encouraging and suggest that further work is warranted towards realising a working small UAV prototype that incorporates the design presented here.

This paper is organised as follows: Section 3 provides the details of both the UAV and FPGA chosen for this investigation. Section 4 describes the proposed hardware design architecture for an effective FPGA-based on-board flight computer. Section 5 concludes with implementation results and warrant for future work.

2 Related Work

The use of FPGAs to provide on-board functionality relating to UAV autopilots or high level control systems in the open literature has not been explored in any great detail. Konyushko [Konyushko, 2013] details the usage of FPGAs and Microcontrollers on UAVs. He concludes that while Microcontrollers can be implemented and developed easily, FPGAs are much more suited for tasks such as DSP and path planning.

Guanglin *et al.* [Guanglin *et al.*, 2007] proposes a Linux system integrated with an Intel FPGA chip that functions as an autopilot for a small UAV. The authors state that the Linux autopilot allocates a high amount of resources to tasks such as navigation and control algorithms and conclude that if these tasks are transferred to an on-board FPGA the system will experience greatly increased efficiency. They then detail the synthesis of an interface between the two processors. The design is quite limited by its inability to be expanded beyond sensor device driving.

Kok [Kok *et al.*, 2013] designed a Genetic Algorithm (GA) path planner and implemented it on an FPGA, where simulated results saw vast speed increases when compared to similar microcontroller implementations. The VHDL design of Kok *et al.* was chosen in this work as a suitable path planner for use with the proposed VHDL MAVlink design.

3 Target UAV details

A small rotary-wing UAV was chosen to explore the feasibility of the proposed design, as detailed below.

3.1 UAV and Autopilot

The fairly inexpensive and widely used Parrot AR.Drone was chosen here with a PX4 autopilot installed. The PX4 autopilot consists of the electrical and mechanical adapter board PX4IOAR and autopilot/flight management unit PX4FMU. The autopilot is implemented on a 32 bit ARM9 RISC core on-board the PX4FMU and interfaces with a selection of sensors, including: a 3 axis gyroscope, a 3 axis accelerometer and an ultrasound sensor. Although the interaction of an on-board path planner with the PX4 autopilot via a FPGA-based module is the main focus of the present work, the information produced by these sensors could also be requested by the FPGA module via MAVLink commands with appropriate future extensions.

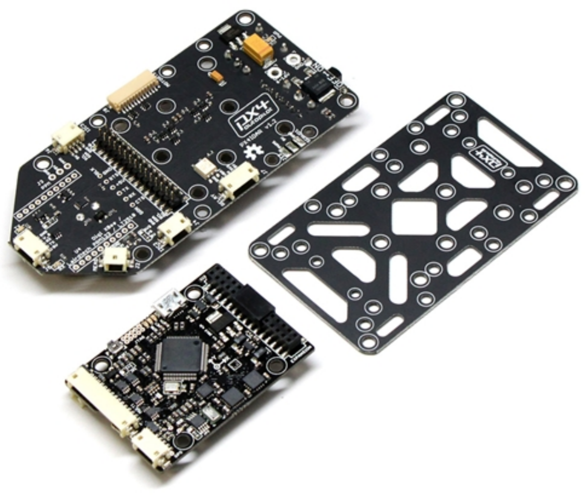

Figure 1: PX4 Autopilot Board[1]

3.2 MAVLink

MAVLink (see www.qgroundcontrol.org) is a lightweight header-only communications protocol that allows up to 256 UAVs to communicate on the same frequency band. Released under the GNU LGP license in 2009, the protocol provides the means to control UAVs from a high level rather than by interacting with low level elements such as orientation and thrust. **This strongly contributed to the choice to use the protocol as it aligns with the design aims.**

MAVLink messages can broadly be split into two different categories: information requests and mission commands. To explore our design proposals, a limited set of MAVLink information requests and mission commands were selected from the complete library which were deemed to be sufficient to control the UAV operating autonomously under the guidance of an on-board path planner (see Table 1).

A serial UART connection is used for the FPGA to PX4 interface as the bandwidth required to transmit the MAVLink messages is quite low as computations are performed on the FPGA.

3.3 FPGA

A Xilinx Zynq 7-Z010 system-on-chip was chosen as the initial target hardware for the design implementation. This FPGA is used on the MicroZed board (see www.microzed.org) which should be of sufficiently small size and weight for use in an AR.Drone based prototype. The Zynq FPGA has fairly limited resources of 28k logic cells and 17.6k LUTs, which also provides a practical constraint for a design which is ultimately intended to

[1]Image from PixHawk: `pixhawk.org/modules/px4fmu/` last visited 30/08/2014

[2]Image from Parrot: `ardrone2.parrot.com/support-ardrone-1/` last visited 30/08/2014

Figure 2: AR.Drone Quadcopter[2]

be both resource and power efficient. If necessary, the proposed design could be translated onto other FPGAs for example the Xilinx Zynq-7000 or Virtex 7 family with greater amounts of logic resources.

4 Proposed Hardware Design Details

4.1 Architecture

The proposed modular design for interaction of *Internal Components*, which in this work is an on-board path planner, with the *Control System* serial connected to the PX4 autopilot via MAVLink communications protocol is illustrated in Fig. 6. Note that the various modules

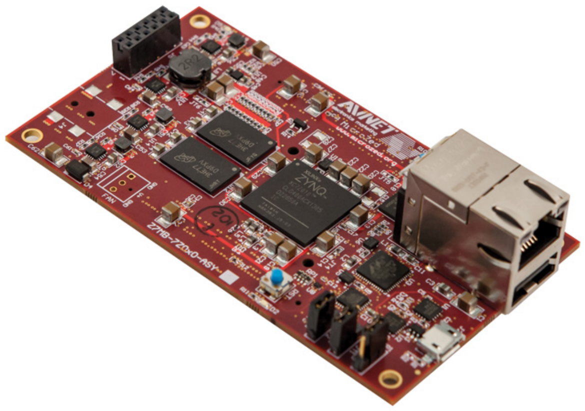

Figure 3: MicroZed Board[3]

shown are all intended for implementation on a single FPGA, which communicates with the PX4 autopilot via serial connection. Given the project heavily relies on inter-module communications, a reliable and effective handshaking protocol was implemented throughout the design, as illustrated in Fig. 4.

Two natural groupings of modules are depicted in Fig. 6, relating to either the MAVLink based Control System or *Internal Components* that provide on-board services requiring communication with the PX4 autopilot for operations such as path planning. Brief descriptions for some of the key modules are given below.

[3]Image from ZedBoard: `http://www.zedboard.org/product/microzed` last visited: 20/08/2014

Table 1: Selected MAVLink information requests and mission commands

Information Requests	Mission Commands
Pitch angle	Navigate to GPS location
Yaw angle	Loiter at *location* for infinite
Roll angle	Loiter at *location* for *time*
GPS latitude	Return to launch location
GPS longitude	Take off
Altitude from ground	Set *system mode*
Altitude from sea level	Change home location
Latitude speed	Calibrate sensors
Longitude speed	Shut down *component*
Altitude speed	
Compass heading	
Time since system boot	

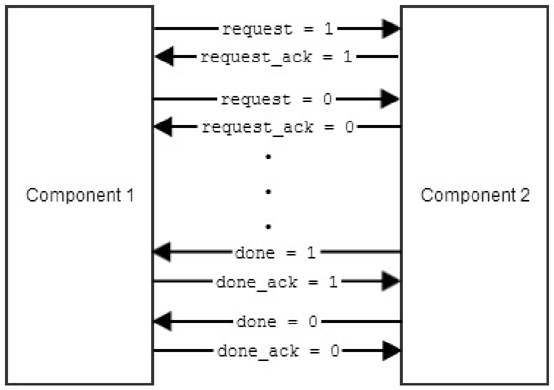

Figure 4: Handshaking Protocol

4.2 Internal Components

Referring to Fig. 6, a fully FPGA based design for the internal Path Planner component module such as the one developed previously by the authors [Kok *et al.*, 2013] could be used here. At the present stage a simplified path planner module has been included that produces sample waypoints.

4.3 Control System

Control System Management

Within the Control System, the Control System Management (CSM) module ports key modules of the design together through a system of handshaking. The architecture is akin to that of a state machine, wherein a process waits in an idle state until a module makes a request. Once received, this results in a series of further states which perform handshakes with other modules to carry out a specific function. Multiple processes handling each separate stream of data are included so as to exploit the parallel architecture of the FPGA.

Internal Component Manager

The role of the Internal Component Manager (ICM) module is to act as an interface between the on-board path planner and the Control System. In the present case, the module is able to both receive a series of waypoints generated by the path planner and generate information requests, then reformat these into a form suitable for the Control System. Using handshaking, these messages are then transferred to the CSM module which either retrieves and responds with the requested information or proceeds with further processing. Additionally, the ICM can also process initialisation data from the path planner to prepare and arm the UAV for flight. In the present design proposal, the ICM interacts only with an on-FPGA path planner. However, in future work it is envisaged that this module could also be used for managing data flow to and from the Control System for other on-board services that need to interact with the flight computer via MAVLink.

Header Generator

The Control System uses the Header Generator module to translate message information into the MAVLink specific format. The design manages the MAVLink specific functionalities such as message formatting, variable message lengths, checksum generation and target device definition. When generation is complete, the formatted message is returned as a string to the Control System.

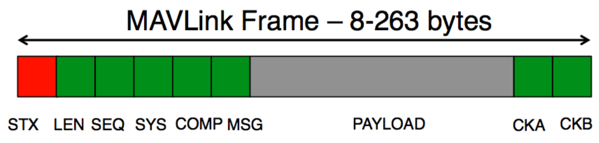

Figure 5: MAVLink Header Format [ref]

[ref] Image from QGroundControl: `http://qgroundcontrol.org/mavlink/start` last visited: 29/08/2014

Message Receiving System

Within the Message Receiving System module, messages received by the Control System are read into a buffer before further processing. The checksum of the message is generated on-FPGA, then the checksum within the message is extracted and the two are compared to ensure message integrity. Should the test pass, the message ID is extracted from the buffer and the receiving system will then proceed with further processing. The current design recognises message acknowledgements, heartbeats, information updates and mission completion messages.

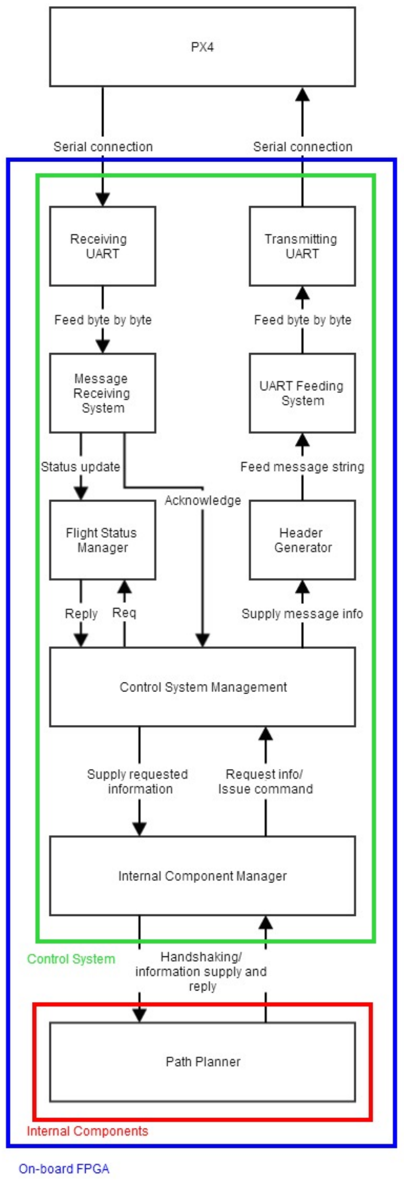

Figure 6: Proposed Modular Design Architecture

Flight Status Manager

All flight information is stored within this module, including UAV status, UAV mode, flight vector, flight position and mission tracking. This information is constantly updated by the flight computer through the on-FPGA message receiving system. The module responds to information requests from the Internal Component Manager. When a new command is issued from the FPGA or a mission completion message is received from the flight computer, the module tracks which missions are current or complete. This enables the system to send a string of commands in a burst.

5 Results

The proposed design was encoded in the VHDL hardware description language and synthesised successfully using the Xilinx ISE software with the Zynq 7010 as the target device. To ensure correctness, the integrity of the MAVLink messages in the design was tested extensively via waveform generation in the Xilinx ISE platform and expected vs simulated comparison with MATLAB. It was vital that these tests were performed on both platforms, to increase the reliability of the simulated results.

The Xilinx simulation was performed by utilising waveform generation to ensure handshaking, data transfer and state machine integrity. It demonstrated that each module dynamically communicated messages that were acknowledged utilising the handshaking protocol outlined in section 4.1. Test cases were performed on all possible flight conditions with this method initially, which followed on to output text log generation of the simulated autopilot communication. An example test case of the path planner interfacing with the Internal Component Manager can be seen in figure 7.

The MATLAB simulation produced a replica of the expected message logs locally and then performed comparisons between them and the Xilinx ISE outputs. A script ran through each test case and presented a pass or fail result and message differences.

Comparing the two simulation methods, module to module communication and message integrity was found to be reliable.

The results of the simulation indicated that the design could be easily implemented on the Zynq 7010 FPGA with theoretical clock of 866 MHz and with less than 5% of the rather limited configurable logic resources available on the Zynq chip actually being used. The results obtained thus far are encouraging and suggest that further work is warranted towards realising a working small UAV prototype which incorporates the design presented here.

6 Conclusion

The FPGA based high level control system for a small UAV using the MAVLink communication protocol pre-

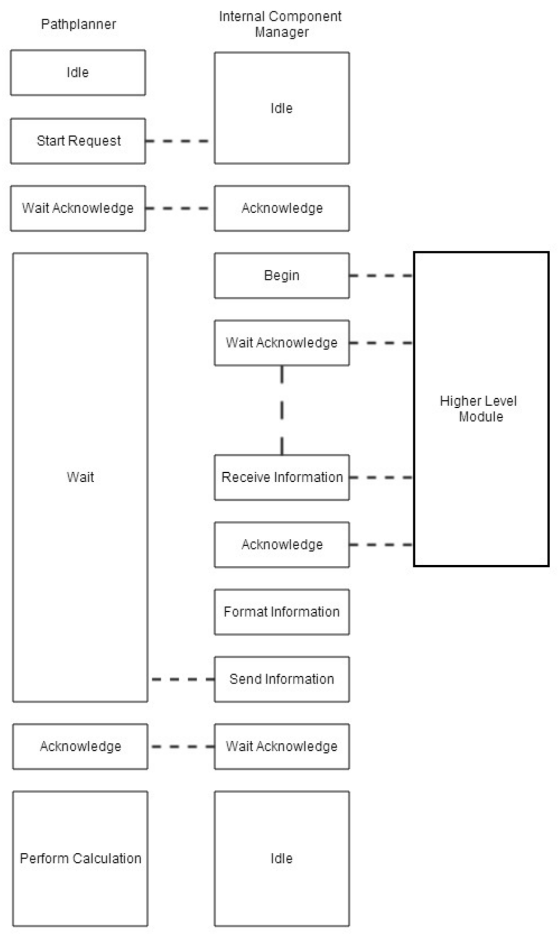

Figure 7: Path planner and Internal Component Manager communication test case

sented in this paper addresses the need for such an architecture within the aerospace community. While there is a degree of further work to be completed, the design will accelerate research projects in the same field once fully implemented.

6.1 Further Work

The system simulation is fully complete and the main avenue of expansion is on-board flight testing. Furthermore to this primary goal, additional functionality, such as on-board image processing and dynamic terrain mapping, could be implemented.

The FPGA with control system and path planner on-board will be mounted to the AR.Drone. This will require a mechanical mount, power supply and communications between the FPGA and PX4 autopilot. Extensive flight testing will be required to ensure safe flight dynamics, message integrity and correct operation.

Image processing could be implemented on the FPGA to create a truly dynamic control system. An obstacle detection algorithm could be implemented to communicate to the path planner and terrain to update on-the-fly so that missions into changing environments would be possible.

A Radio Frequency (RF) connection between the FPGA and ground control station that also utilises the MAVLink protocol will be implemented. It will be capable of enabling and disabling the entire system, receiving heartbeat and status updates and feeding an end location to the path planner.

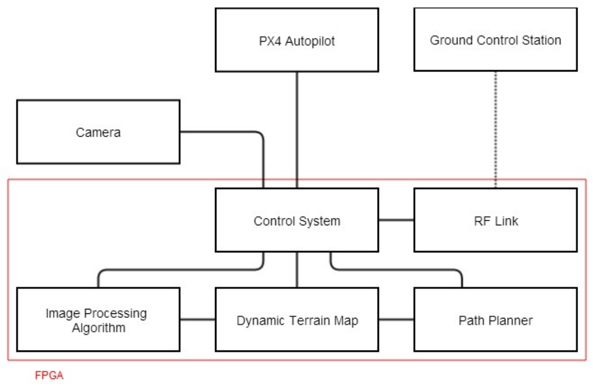

Figure 8: Block Diagram of Proposed Further Work

References

[Barrientos *et al.*, 2011] Antonio Barrientos, Julian Colorado, Jaime del Cerro, Alexander Martinez, Claudio Rossi, David Sanz, and João Valente. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689, 2011.

[Cox *et al.*, 2004] T. H. Cox, C. J. Nagy, M. A. Skoog, and I. A. Somers. Civil UAV capability assessment. NASA technical report, December 2004.

[DeBusk, 2010] W. M. DeBusk. Unmanned aerial vehicle systems for disaster relief: Tornado alley. In *Proc. AIAA Infotech@Aerospace 2010 Conference*, pages Paper AIAA 2010–3506, Atlanta, Georgia, USA, April 2010.

[Guanglin *et al.*, 2007] He Guanglin, Guo Rujun, and Yang Shil. Application of FPGA in small UAV autopilot based on embedded linux system. In *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2007.

[Kok *et al.*, 2013] J. Kok, L. F. Gonzalez, and N. A. Kelson. FPGA implementation of an evolutionary algorithm for autonomous unmanned aerial vehicle on-board path planning. *IEEE Transactions on Evolutionary Computation*, 17(2):272–281, 2013.

[Konyushko, 2013] V.M. Konyushko. Features of uav control systems assessing. In *2013 IEEE 2nd International Conference Actual Problems of Unmanned Air Vehicles Developments Proceedings*, 2013.

[Li *et al.*, 2012] Zhengrong Li, Troy S. Bruggemann, Jason J. Ford, Luis Mejias, and Yuee Liu. Toward automated power line corridor monitoring using advanced aircraft control and multisource feature fusion. *Journal of Field Robotics*, 29(1):4–24, 2012.

[Ortner, 2009] Luigi Ortner, Peter; Del Re. Autopilot design comparison and flight experiments for a small UAV. In *Proceedings of 2009 7th Asian Control Conference, ASCC 2009*, pages p 314–319, Johannes Kepler University Linz, Austria, August 2009.

[Xiao *et al.*, 2011] Yahui Xiao, Xinmin Wang, and Xiaoyan Wang. An effective controller design of formation flight of unmanned aerial vehicles (UAV). In *Xibei Gongye Daxue Xuebao/Journal of Northwestern Polytechnical University*, pages p 834–838, Xi'an 710072, China, December 2011.