

FPGA Based Flexible Autopilot Platform for Unmanned Systems

W. Alvis, S. Murthy, K. Valavanis, W. Moreno, S. Katkoori
CSE and EE, Unmanned Systems Laboratory
University of South Florida
Tampa, FL 33620, USA

Abstract- Diversity in unmanned vehicle designs and control as well as diversity in existing autopilots lead to major compatibility issues among different platforms. The small number of different autopilot designs currently available in the market and their limitations, justifies the rationale for proposing design of a new modular, flexible, easy to integrate autopilot, suitable for multiple platforms. A discussion and comparison of the state of the art in autopilot research is presented, followed by the introduction of an autopilot design that will provide a novel and well-needed processing platform for unmanned systems. The design is fully integrated with MATLAB/Simulink allowing for both simplified high level programming and hardware-in-the-loop capabilities, and provides a flexible FPGA-based platform that maximizes parallel processing and allows for analog signal conditioning that can be modified through digital communication from FPGA processor.

Keywords – Unmanned vehicles, autopilot, hardware-in-the-loop, FPGA, MATLAB / Simulink

I. INTRODUCTION

Research in unmanned vehicles has seen unprecedented levels of growth in recent years. Although this is the case, the diversity of existing platforms, from their mechanical design, to size, to range and endurance, to functionality and operational profile, to navigation system, to electronics and sensors, to controllers, to hardware used to build such systems, to utilized software libraries, to payloads, to power consumption requirements, to level of autonomy and to application domains, has resulted in total lack of standards, common requirements and specifications for such vehicles. There exist major compatibility issues among different platforms, while cooperation and coordination of multiple heterogeneous vehicles is an almost impossible task.

When focusing on unmanned aerial vehicle (UAVs), fixed- or rotary- wing, vertical take off and landing (VTOL) ones, an additional challenge that needs to be overcome is that of the on-board autopilot. This challenge becomes even more important when UAV certification issues arise, which require, among other things, that the on-board autopilot is at least FAA certified and approved.

Concentrating in civilian domain applications where cost effectiveness is also an important factor, the diversity of existing giant, large, small and miniature size UAVs, and the small number of different autopilot designs currently available in the market, mostly custom made, justifies the rationale for proposing design of a new modular, flexible, easy to integrate

autopilot, suitable for multiple platforms, complying with FAA rules, procedures and regulations.

When proposing an as flexible as possible and easy to program autopilot, one should consider at least, among other things, high level mission planning capabilities, sensor fusion, sensor integration and diagnostics, in addition to the conventional servo and actuator control, as well as switching among, or modifying, control techniques if and when necessary. In other words, consideration should be given to implementing different controllers based on mission profiles or flight scenarios.

Thus, any proposed design must include an interface module that allows for simulation, validation and verification before actual implementation; by default, such a design should be fully interfaced / integrated with *MATLAB* and *Simulink* allowing for both simplified high level programming and hardware-in-the-loop capabilities.

Flexibility and modularity requires a separate computer processing system (separate processor) that handles servos, analog and TTL-type sensor inputs / outputs and actuator signals, and a second one using FPGAs to maximize parallel processing capabilities and analog signal conditioning that can be modified through digital communication from FPGA processors.

Regardless, flying an unmanned aerial vehicle requires computationally intensive calculations and massive sensor data and signal processing, rendering the need for a full computing system on-board (such as a mini-ITX or PC-104) along with components utilizing DSP processors or RISC architecture processors. A generic and at the same time flexible autopilot should be a 'stand alone system' that does not require in depth knowledge of analog / digital design and low level programming languages like VHDL to utilize its full functionality and capabilities.

Considering vehicle payload limitations, power consumption and requirements, cost-effectiveness and available 'space' on the unmanned vehicle, and given the fact that real-time control requires very strict and fixed timing for stability purposes, the embedded system approach is preferred in designing a stand alone autopilot, because it can be implemented in a much lighter package, making suitable even for miniature vehicles.

The next Section offers a very comprehensive review of existing designs, while Section III presents fundamentals of the proposed design. Section IV compares this design with others, while the last Section concludes the paper.

II. LITERATURE REVIEW

An overview of autopilots currently available is presented first. Then, processing systems are discussed as a generality of the different hardware types used, while the few proposed and developed autopilots within the research community are discussed in detail. Advantages and disadvantages are stated.

A. On-the-Market Autopilots

There are several commercially available autopilots. Some are fully or partially proprietary, while others are open source and have been implemented using DSP processors. Only one incorporates an FPGA processor for handling input / output signals in parallel.

Two autopilots specifically developed to be used on vehicles designed by the same company, are the Generation II by BAI [1, 2] allowing for very minor modifications, and the one designed by Rotomotion [3] allowing for no modifications. Minor details about these designs are available because of the proprietary nature of the autopilots. Neither of these is suitable for research purposes, or even for an end user with more than one airframe, due to the built in dependency for airframe specific modifications to the software.

The Kestral autopilot designed by Procerus [4] and the MP2028 designed by Micropilot [2, 5] have included user flexibility into the system, but they are both still proprietary in nature. They have additional input / output ports, allowing for reprogramming and hardware-in-the-loop capabilities, but only through proprietary software. Both autopilots implement a single processor; the Kestral utilizes a 29MHz, 8-bit rabbit processor; the MP2028 utilizes a 20MHz 32-bit Motorola processor.

The Ezi-Nav autopilot built by Autonomous Unmanned Air Vehicles [2, 6] is designed to be a low cost autopilot with minimal capabilities, containing eight microprocessors that share computational load. The disadvantage of this unit is that it only works with hand held GPS units; it has no additional ports for communications with an external processors or additional sensors.

The Phoenix autopilot by O-Navi [7, 8] is an open source, fully reprogrammable autopilot that utilizes a 32MHz, 32-bit Motorola processor. It is programmable through a provided flash kit only, and does not provide built in hardware-in-the-loop capabilities.

The Piccolo II by Cloudcap [2, 9] is an open source autopilot designed specifically for fixed wing vehicles [2, 9] but has enough flexibility that implementation with rotary wing vehicles appears reasonable. This autopilot is popular with the research community, most likely due to its flexibility and ability to be programmed through *Simulink* Real Time Workshop. In addition, it does allow for hardware-in-the-loop capabilities, but only if the computer running *Simulink* is equipped with CAN interface card. This unit utilizes a single 40 MHz, 32-bit Motorola processor which lacks the parallel processing capabilities of an FPGA processor.

The Microbot autopilot designed by Microbotics [10] has the most built in flexibility. It is the only open source design on the market that has included an FPGA to allow for reconfiguration of up to 32 I/O ports. An expansion board allows for two asynchronous serial

ports and twelve analog inputs to be included in the design. Unfortunately, the FPGA is only utilized for the input and output logic. Most programming resides in a single microprocessor, which does not allow for any parallel processing of the main functions. Another major disadvantage of this unit is the lack of design for rapid prototyping. While the unit is fully reprogrammable, it does not allow for programming through *Simulink*, nor does it have any hardware-in-the-loop capabilities incorporated.

B. Microprocessor/DSP Low Power Autopilots

The majority of publications in this area discuss the processing system as a brief portion of a larger research project. Two popular methods dominate the research: either implementing a low power DSP/microprocessor chip, or a full motherboard type system, such as a mini-ITX board or PC-104 system. Only recently the advantages of FPGA-based parallel processing and reconfigurability are being considered

In [11], a simplified autopilot is designed for use with a specific fixed wing aircraft, the Goldberg Decathlon ARF, as a learning lab tool for undergraduate students at the Georgia Institute of Technology. A Rabbit 3000 is used along with several sensors. This microprocessor meets requirements for easy implementation of simple algorithms that are used for teaching basic theory; however, it does not have much flexibility or processing power for use across platforms. A similar design was developed by Brigham Young University with a fixed wing aircraft fabricated by foam [12].

An autopilot system was designed using Microchips 16F877 microcontroller and used in a kite style micro air vehicle [13]. Because of the low dynamics of this system and minimal sensors used, very little processing power was needed.

Preliminary designs are presented for an MC68HCS12 microcontroller based autopilot in [14]. The design focuses on providing low cost, easy to modify system. The specific UAV and sensors are not mentioned.

The payload capacity and power requirements of micro air vehicles prevent the use of full computing systems. Algorithms are basically implemented on microcontrollers, such as microchips line [15-17]. Several publications discuss sensor design or vehicle design, but without implementing the processing on-board. Other methods have been used, such as utilizing a ground station [18] for control processing, verification of design by simulating the system [19-24], flown by radio control for verification of design [25, 26] ,or a tethered system connected to a DSP board and *MATLAB* [27].

C. Full Computer Implementations

Most published research discussing design of VTOL processing systems implements full motherboard systems without dedicated hardware for signal processing and low level control. The most popular one is the PC-104 board either running a real-time operating system such as VxWorks® [28] or QNX [29-31]. In [31] a full data acquisition card is also included in the design. Various other computing systems have been used with real time operating systems as well [32, 33].

Timing issues have been considered in [34] by implementing a two processor system running on RT-Linux. The software was designed with a layered approach, with a main board running an x86 compatible motherboard for the wireless and GPS communications and mission planning, and an ATM Mega 163 chip for the real time flight control processes. This is the same concept of using a dedicated autopilot for low level controls, which further argues the need for a dedicated autopilot for low level controls.

D. Implementations using FPGAs

FPGAs are very slowly gaining popularity due to the recent advances in the increased number of and simplification of design by the manufacture's providing intellectual properties, IP, which provide built in functions. Because these advances are fairly recent, there are only a few publications that are following the same philosophy of utilization of FPGAs [8, 35-39].

In [38] a 40K FPGA is combined with an 8 bit microprocessor for control of a fixed wing aircraft. This system is designed for the control of a fixed wing aircraft with GPS as the only sensor for position. The FPGA array is utilized for the FM aircraft receiver and the servo control.

A proposed FPGA based design to allow for a system capable of integrating a propulsion health system with a control system for VTOLs is presented [35]. This design has recognized strength of both FPGA architecture and integration with *Simulink* for programming. The proposed design, however, intends to implement the algorithms running on the hardware under a real time operating system, VxWorks®. In order to make this work, however, the intention is to create an ICD along with third party software for programming. The system will implement a Vibe Card for receiving some sensor data. With some simple front end analog signal conditioning and A/D converters, this can be eliminated allowing all the signal processing can easily be implemented with the FPGA chip. In addition, the design also includes a Geode DSP processor that will run a majority of the processes, ultimately limiting the system to sequential processing for the majority of the processes.

A flexible FPGA/DSP based autopilot has been developed by Georgia Institute of Technology [36, 37]. This project has been completed and tested on two separate platforms, working in conjunction with a more powerful processing system on the GTMax, and also as a stand alone device in the GTSpy. A flexible, hardware proven design is defined, but the full strength of the Xilinx line is not taken advantage of. While the design does implement parallel processing for some of the tasks, much of the processing on board the Xilinx chip is done with a soft core DSP running on a real time operating system, MicroC/OS II. This means that many of the tasks can not be divided into smaller sequential tasks running parallel. In addition, the system includes an addition DSP chip to run any high level processing. This prevents the entire system from being developed with *Simulink* using System Generator.

Virginia Commonwealth University has recently demonstrated a successful in flight test of an FPGA based autopilot [8]. This utilized a Suzuku V board containing a Xilinx II FPGA chip, 32 M Bytes of

SDRAM, 8 M Bytes of flash memory and an Ethernet interface. The FPGA's on chip PowerPC was running a Linux kernel for implementation of the majority of the processing. This research demonstrated that the development of the control software could be developed in the commercial off the shelf hardware and then ported to any other hardware running the same Linux kernel. Since the focus of the research was not a complete hardware autopilot design, this research did not utilize the ability to run processes in parallel other than I/O processes, nor did it take advantage of the Virtex II *Simulink* capabilities of hardware-in-the-loop testing and high level programming. It does, however, demonstrate the capabilities of the Virtex II as the processing hardware for an autopilot.

Ongoing work in the design of an FPGA based control system for a Micro-Satellite has demonstrated the potential benefits of FPGAs for autopilots [39]. The Xilinx series of FPGAs is implemented for the processing system, and full parallel processing is taken advantage of. While in flight tests have not yet been demonstrated, lab tests have indicated that good timing and parallel communication with the devices have been obtained.

E. Comparison: Advantages & Disadvantages

Full computing systems do have the advantage of excellent processing power. For unmanned vehicles with the payload capacity to utilize them, they are an excellent choice. Most implementations under this category follow the same basic design principle: external sensor and hardware, with a single processor powerful computing system. This method has been proven to work successfully, but great care must be taken in programming the controls system, or the real time requirements will not be met. This requires, not only a great deal of knowledge of control systems, but also knowledge of the real time programming language. Each function must be given a priority; those with the least priorities run only when high priority ones are not. For a final implementation that is designed once, with perhaps some upgrades later on, this is certainly more than acceptable. However, whenever the control system is entirely changed, for example from PID controllers to H-infinity feedback controllers, the entire low level control program changes and the timing issue must be entirely reconsidered leading to a longer design time between deriving new theory and implanting it in hardware. A way out of this problem may be to include a separate processor for providing an off board system that follows a given trajectory while handling the tight timing constraints required for sensor integration and control of the vehicle dynamics. This allows for the utilization of the main computer for the computational complex calculations that do not require strict timing.

The majority of the systems presented implement a single processor. The processing power varies depending on the specific chip selected. All designs are at a disadvantage compared to processors implementing either a full FPGA or a hybrid DSP/FPGA design. Because single processor systems can not operate with parallel processing, care must be taken to be sure that each of the asynchronous sensor inputs are sampled at the correct time, while also updating the servo outputs. In addition, the majority of the implementations did not

allow for *Simulink* integration or hardware-in-the-loop.

Several of the FPGA implementations have benefited from the flexibility and parallel processing capabilities of the FPGA in regards to managing I/O functions, but failed to carry the parallel processing capabilities into the majority of the processes by either implementing a real time OS on the FPGA, or incorporating a separate DSP or microcontroller in addition. The only work that has utilized the full parallel capabilities for flight controls was for the design of a Micro-Satellite, which indicated the good timing and parallel capabilities of the FPGA implementation, but, it did not explicitly design *Simulink* integration into the system.

The next Section discusses the proposed autopilot.

III. PROPOSED DESIGN

The functional block diagram given in Fig. 1 displays a generalized overview of the autopilot's design and ability.

A Xilinx Virtex II Pro FPGA is selected as the autopilot primary processing platform. The Virtex-II Pro FPGA includes two PowerPC™ 405, 32-bit RISC processor cores in a single device. These processors deliver DSP capability and can operate with clock frequencies that can exceed 300 MHz while operating in parallel with the FPGA logic slices. Xilinx, working in conjunction with *Mathworks*, has developed the System Generator toolbox that gives these FPGAs the ability for full integration with *MATLAB / Simulink*. Because of this functionality, high level abstractions may be directly compiled into an FPGA. In addition, the toolbox allows for hardware-in-the-loop simulation directly with *Simulink* through a standard USB connection by synchronizing the FPGA clock to *Simulink* time.

The autopilot is fully interfaced / integrated with *Simulink* and it is capable of handling several types of communication protocol, accepting various ranges of analog sensor input, data acquisition by including additional memory, measuring altitude and forward speed through on board pressure sensors, and allowing for releasing control of the physical system to either a master computer, or a human pilot.

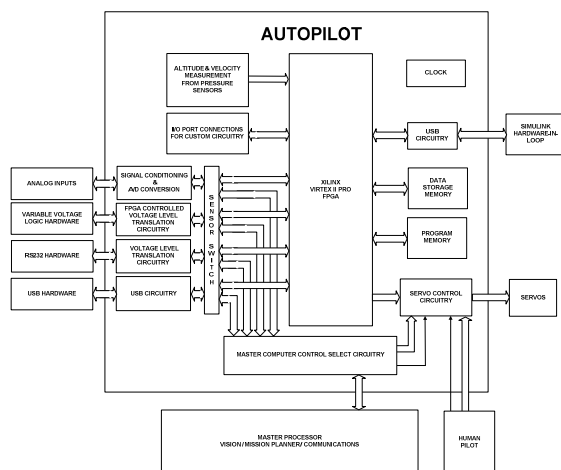


Figure 1: Autopilot Functional Diagram

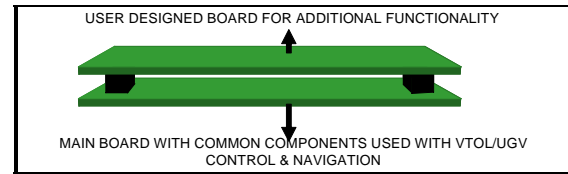


Figure 2: Stacked Board Design

In order to both minimize size and allow for custom analog and MEMS sensor to be developed for use with the autopilot, a stacked board design consisting of a main board and a secondary daughter board is implemented as shown in Fig. 2. The daughter board may be used for inclusion of application specific hardware. This is a necessary requirement for micro air vehicles because the small payload capacity requires extremely small on board sensors to be utilized.

When a mini-ITX or PC-104 type processing system is used, a second board, the secondary safety board, is designed to allow for this second processor to act as the “master”. The master processor and autopilot communicate through a standard USB connection. This custom daughter board allows for master processor to take control of half the servos for camera pan and tilt control. This results in ability to integrate with existing well-developed vision systems [40].

If the master computer detects a failure of the autopilot, it takes control of the sensor information and the flight control servos in order to gain full control of the vehicle dynamics and to prevent a total loss of control of the vehicle. In order to allow the master computer to receive sensor inputs, sensor signals are collected using the various digital communication protocols and then sent to the master computer utilizing the USB connection. For the master processor to gain control of the servos, the desired servo effort must be sent from the master computer and then converted to the PWM control signal required by the servos. While the FPGA on the main board is capable of acting as the intermediate between the master computer and these peripherals, utilizing it in this regard would essentially remove the additional safety layer, since if the FPGA were to fail, the ability of the master computer to take control would be lost. For this reason, a separate circuit is utilized with an on board DSP chip. The DSP has control of a multiplexer that selects whether sensor inputs are connected to the FPGA or the DSP chip. For this reason, the same I/O connections are repeated on the second board with the additional multiplexer interface. The circuitry for the master computer take over is now an entirely separate circuit for an additional failsafe behavior.

The designer of the unmanned vehicle control system may now interface critical sensors with the daughter board, and utilize the interfaces on the main board for additional non-critical equipment or sensors. Detailed specifications of the main board are shown in Table 1 and specifications for the secondary safety board are shown in Table 2.

The size of the board, based on the selected circuitry, is accurately approximated: the entire autopilot circuit is no larger than 2 by 3 inches, which is approximately the size of a credit card.

TABLE 1
MAIN BOARD SPECIFICATIONS

I/O ports and sensors
On-board pressure sensors for altitude and forward speed Two large signal, single ended analog inputs Two small signal differential analog inputs Twenty-four variable voltage logic inputs <ul style="list-style-type: none"> Input voltage set in blocks of four 1.8V to 5V range 5 Tx and 5 Rx RS232 lines <ul style="list-style-type: none"> Allows for 1 full duplex or two half duplex connections Three USB ports
Capabilities
Integration with master computer <ul style="list-style-type: none"> requires second board for emergency take over of sensors and servos On-board data acquisition memory Twelve servo outputs Safety switch for servo control <ul style="list-style-type: none"> Six critical servos can be taken over by pilot All twelve, selectable in sets of six by second board (if present) Simulink programming and hardware-in-the-loop 60 I/O FPGA connections to daughter board

TABLE 2
SECONDARY SAFETY BOARD SPECIFICATIONS

I/O ports
Two large signal, single ended analog inputs Two small signal differential analog inputs Twenty-four variable voltage logic inputs <ul style="list-style-type: none"> Input voltage set in blocks of four 1.8V to 5V range 5 Tx and 5 Rx RS232 lines <ul style="list-style-type: none"> Allows for 1 full duplex or two half duplex connections Three USB ports
Capabilities
Communication with master computer Takeover of control of vehicle dynamics Takeover of six servos for camera pan and tilt control Separate circuit design for additional failsafe mechanism

A. Analog, I/O Interface and Servo Control

The proposed autopilot requires designing of dedicated hardware surrounding the FPGA. In order to meet the specifications given in Table 1 and Table 2, there must be flexible interfaces between TTL/logic inputs and FPGA, and analog inputs and FPGA, custom signal conditioning and A/D conversion with respect to the pressure sensors, additional memory, RS232 communication, USB connections and dedicated separate circuitry for the selection of the servo control.

The FPGA is powered by a 3.3 volt supply and it is capable of I/O voltages that vary between 1.5 and 3.3 volts, but it cannot accept 5 volt logic. Since there are devices, such as ultrasonic sensors, encoders and A/D converters, that require 5 volt logic, the autopilot must be able to handle this level as well. In order to allow I/O ports that have a range including 5 volt logic, the Texas Instrument's TXB0104 bi-directional voltage translator is utilized. The TTL logic levels of the translator are set by a voltage provided to two specified pins. By

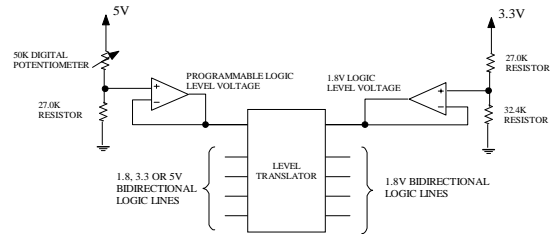


Figure 3 Adjustable Logic Level Circuitry

utilizing the FPGA's 1.8V logic level ports and treating the input as the higher voltage side, any voltage level between 1.8 and 5 volts can be accepted. A digital potentiometer that is programmed through the FPGA processor is used to set the variable logic level. Fig. 3 displays the entire circuit.

The analog input ports are useful to interface any form of a sensor that provides a voltage output or to monitor battery levels directly. Because of this, the signal condition circuitry must be programmable for such things as voltage division on the battery voltages being monitored, filtering of noise and gain on small signal inputs. This design offers such flexibility through the use of a Field Programmable Analog Array (FPAA), which can be reprogrammed in circuit through the FPGA processor. The selected FPAA, Anadigm's AN221E04, includes 8-bit internal A/D converters that can directly convert the conditioned signal into the TTL required by the processor, comes with user friendly software that allows for the design to be simulated and either a C-code or HEX file to be generated that can be utilized directly by the processor for reprogramming and, in addition, this reprogramming can be done dynamically while the chip is in use in the circuit.

The chips do have some limitations at the input signal. The chip utilizes a +2 volt reference for circuit common in order to allow for A/C signal inputs and the input is limited to 4 volts. This creates an issue with ground referenced signals, but with some initial voltage division and offset circuitry this can be overcome in order to allow for the two large signal inputs and the two small differential signals. The analog input circuitry is displayed in Fig. 4.

V_{in1} and V_{in2} allow for two input voltage that are ground referenced and less than 20 volts. This configuration requires a 470 KHz low pass filter to be utilized within the FPAA, which also places a limitation on the frequency input. This is not a substantial

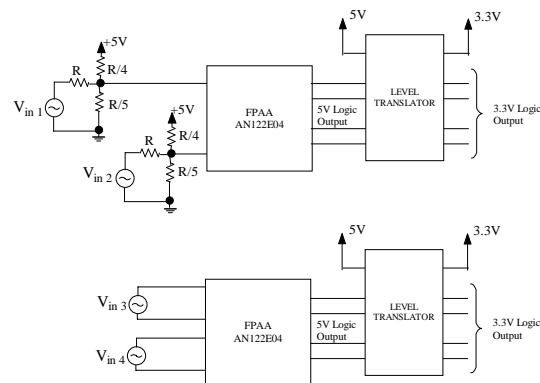


Figure 4: Analog Input Circuitry

limitation, since most sensors utilized in VTOL applications do not provide an AC signal. V_{in3} and V_{in4} allow for two small signal differential voltages to be measured. This magnitude of these signals must be less than 2 volts.

Most available pressure sensor are small, circuit board mounted and require custom signal conditioning, for this reason they will be included onboard. Two separate pressure sensor are utilized; one for use with a pitot tube to measure forward velocity and another to measure altitude. Both the altitude and velocity measurements involve signal conditioning from the output of a pressure sensor. In both cases, there was both an offset that needed to be removed and increase of the voltage in order to gain better accuracy from the measured pressure sensor output. The same circuit was used for both but with different values for the resistors, Fig. 5.

The values of R_f and R_{in} were selected in order to obtain the correct gain using (1).

$$Gain = 1 + \frac{R_f}{R_{in}} \quad (1)$$

A voltage reference of 4.1V was used along with voltage division, in (2), R_1 and R_2 , in order to the remove the offset.

$$V_{offset} = \frac{R_2}{R_1 + R_2} V_{ref} \quad (2)$$

The calculation of altitude is based on the fact that pressure decreases as the altitude of an aerial vehicle increases. A pressure sensor can be used to measure this relationship and the altitude calculated.

The selected pressure sensor relates 15 to 155 kPa to 0.2 to 4.8 volts. In order to achieve more bits/ft at the output of the A/D converter, the range of the output used is limited to 1.8 to 4.2 volts which relates to approximately 200 feet below sea level to 14,900 feet. The A/D converter, Linear Technology's LTC1865, full range input is 0 to 5 volts. The circuit displayed in Fig. 5, was used to convert the 1.8 to 4.2 voltage range to the 0 to 5 volts range used by the A/D converter. The resistor values were calculated from (1) and (2), resulting in R_1 equal to 11.0K, R_2 equal to 62.0K, R_{in} equal to 10K and R_f equal to 10.7K.

Calculating the speed using a pressure measurement works slightly different; a pitot tube is used to generate a pressure that is the difference between the static pressure, with no velocity, and the dynamic pressure, generated from the wind entering the tube from the aircraft's forward velocity. This differential pressure is

measured by the pressure sensor is proportional to the indicated forward air speed of the vehicle. For this measurement a pressure sensor was selected that measured from 0 to 3.92kPa, with an output between 1 and 4.9 volts. As with the altitude sensor, the full range of the sensor is not used in order to obtain better accuracy in the measurement. The designed voltage range is limited to 1 to 3 volts, which is relative to 0 mph to approximately 100mph. As with the previous circuit, (1) and (2) were used to calculate the correct resistor values for the circuit in Fig. 5. The calculated resistor values are R_1 equal to 82.0K, R_2 equal to 56.2K, R_{in} equal to 10K and R_f equal to 15K.

Due to the fact that the proposed design is oriented towards the research community, it is expected that large amounts of data are needed to be collected for both system identification and diagnosis of flight response analysis. For this reason an additional flash memory is included onboard. This will be separate from the static RAM program memory for future use where the program memory can be encased in a plug and play type case for a simple switch between vehicles.

USB requires a 5V bus to the device, a ground connection and two 3.3Volt logic lines. There are three data rates given in the USB specifications, 1.5Mbps, 12Mbps and 480Mbps. The third is the high speed data rate specified by USB 2.0. However, to be compliant with 2.0 USB, this high speed is not required. A full speed device (12Mbps) is still compatible with 2.0USB devices. Because the logic level is compatible with the 3.3V logic of the FPGA processor, no level translator is required on the logic ports. The USB port requires a 5 volt supply, which can be directly connected to the 5V power supply.

RS232 communication logic is older than TTL and does not operate at the standard 5V, 3.3V or 1.8V logic levels that are now much more popular with processors. The high level is between +5 to +15 volts and the low between -5 and -15 volts. There are several standard ICs that contain charge pumps to allow for the negative voltage levels from a 5 volt power supply. The MAX205E allows for 5 inputs and 5 outputs, while only requiring one external capacitor. In order to optimize board space, this chip in the tiny SSOP package was the best selection.

It is standard in small scale vehicles to include a safety switch that will allow a pilot to take control of the servos controlling the aircraft at any time. Not only will this be built into the design, but the switch will also allow an external processor to take control as well. This is to allow an additional layer of safety in case of an autopilot failure when the craft is out of the line of sight of the pilot.

The autopilot will be able to control up to servos by outputting a 3.3 volt PWM signal to each servo. In addition, the pilot, or when in use, the additional safety board will also be able to generate these signals controlling the servos. The highest priority will be set to the pilot, then the safety board, and finally the FPGA. In order to achieve this without excessive use of analog switches, an additional FPGA is utilized in order to receive each of the servo control lines from the three sources and then select which is outputted to the servos, Fig. 6.

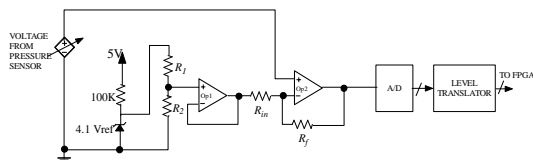


Figure 5: Pressure Sensor Signal Conditioning

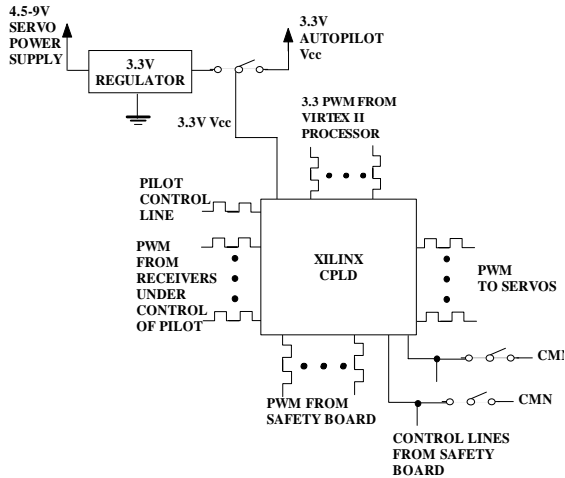


Figure 6 Actuator Control

The power to both the control switch and the servos can either be supplied by a separate battery running on 4.5V to 9V or the same 3.3V supply that is powering the autopilot, selected by a jumper. The control line from the pilot is a PWM coming from a standard servo receiver. The CPLD will monitor the receiver frequency, for the pilot's request for control. This input will be given the highest priority for selection of control of the servos controlling the vehicle's dynamics.

There are two logic control lines from the safety board that request control of the servos in groups of six. Logic high on the control lines for the flight control servos will be acknowledged only if the pilot has relinquished control. Two jumpers will be included on the main board to set these control line to logic low when a safety board is not present. The CPLD will be preprogrammed as part of the autopilot design, and will not need to be modified by the end user.

B. Software Overview

The software within the autopilot is built from both provided libraries and the standard System Generator building blocks, in addition to various functions running on a user selectable operating system within the PowerPCs, as shown in Fig. 7.

Sensor and actuator interfaces to the FPGA are written in VHDL code to efficiently manage the internal hardware clock. This code is made available to the user as building blocks within the provided library. These blocks include I/O protocols such as frequency measurement, I²C, SPI and RS232 and the required PWM output for servo control. Other high level signal processing functions, such as filtering, sensor integration, and controllers can be developed using standard System Generator functions. In addition, the Virtex II Pro provides two PowerPCs that can contain a small operating system, such as the Slackware version of Linux, or VxWorks®. While this does seem contradictory in nature to argue the case for parallel processing, this ability to utilize a DSP structure makes it possible implement many algorithms, such as wireless networking protocol, that have been developed by various programming methods to operate within a specific operating system such as Linux or VxWorks®.

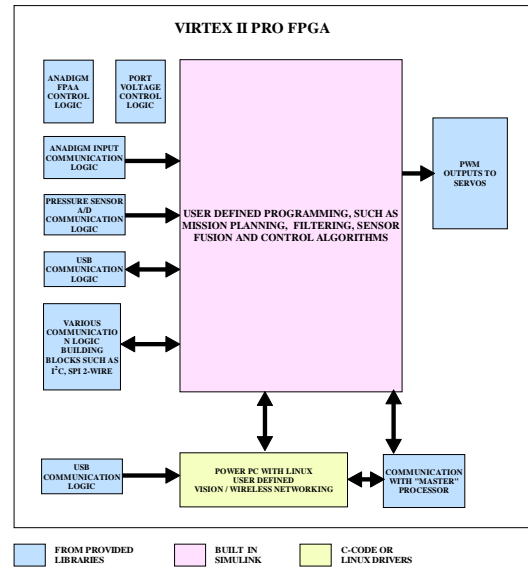


Figure 7: Software Overview

In addition hardware may be designed into the system that utilizes Linux drivers without the additional work of developing custom software. VxWorks® has been FAA approved as a real time operating systems, for those that have developed software compliant to this operating system will still have use of all the previous work, while taking advantage of parallel processing for the I/O and newly developed functionality.

IV. COMPARISON OF PROPOSED DESIGN WITH EXISTING ONES

The proposed design has not only included all of the best features of various autopilot designs, such as integration with *Simulink*, open source to allow any modification needed, and FPGA implementation, but in addition to all of the above, it demonstrates improvement by including additional features that, as far as the authors are aware, are unique.

Full FPGA implementation lends itself to this type of application far better than DSP processors or even hybrid DSP/FPGA because of the flexibility and the ability to process different algorithms in parallel, such as wireless networking, vision algorithms and the sensor integration and control. Because the processor hardware architecture is reconfigurable, each signal and variable can be represented using a different number of bits as required. This contributes for higher sampling rates, better accuracy, high computation speed with low power consumption, unlike the DSP processors which have fixed architecture and word length. FPGAs operate at a very high frequency that when combined with parallel computational structures to provide computational speeds as much as 100 times greater than those possible with digital signal processors which are sequential in operation [41, 42].

An additional advantage of a full FPGA design is that there is a natural migration to micro-air vehicles. Once the prototype is developed and the design verified, the power and size of the processing system can be reduced by implementing the tested algorithm in a system-on-chip design.

Many of the designs implementing FPGAs still utilized a separate DSP/microcontroller processor for the majority of the processing, such as the Microbot and the GTSpy, once again neglecting the benefits of parallel processing. Two of the full FPGA designs require the programmer to implement the majority of the programming in a PowerPC [8, 35], rather than allowing the programmer has the option of implementing some or all of the processing utilizing a real time operating system, or taking full advantage of parallel processing capabilities. Although the presented work in [39] implemented a full FPGA processing system for control of a Satellite rather than a small scale or micro vehicle, the full parallel processing capabilities of the Xilinx chip was utilized and analysis indicated good timing and parallel communication could be maintained.

The only design found that allowed for programming directly through *Simulink* is the Piccolo autopilot, which utilized a DSP processor and required additional hardware to implement the CAN interface protocol.

The proposed design not only has the capability of implementing hardware-in-the-loop simulation through the single USB connection, but will also have all the advantages of an FPGA.

By implementing full FPGA processing design and full *Simulink* integration, the benefits of rapid system prototyping, tight timing control and flexibility across platforms and sensors can be realized. In addition, none of the presented related work has built in analog flexibility or the ability to allow a “master” computer to take control off some or all of the servos.

CONCLUSION

The design presented not only improves upon the processing capabilities of small DSP/microcontroller designs, but it also provides programming and hardware-in-the-loop capabilities in an environment familiar to researchers in many areas of engineering that will allow for rapid prototyping of new ideas. The design also presents an unrivaled flexibility due to the programmable analog interface and the inherent flexibility of the FPGA processor. While any processor that will meet the small size and power requirements of micro air vehicles can not possible match the processing capabilities of a mini-ITX or PC104 design, this autopilot, instead, compliments these systems by allowing for dedicated hardware for real time controls of the system dynamics while giving the off board computer the role of a “master” computer when necessary. The combined functionality and flexibility of this research will lead to a novel and well-needed processing platform for the unmanned systems community.

ACKNOWLEDGEMENT

This research was supported in part by an appointment to Student Research Participation Program at U.S. Army Research Laboratory administered by the Oak Ridge Institute for Science and Education through interagency agreement between the U.S. Department of Energy and US ARL. This work was also supported partially by two grants ARO W911NF-06-1-0069 and SPAWAR N00039-06-C-0062.

REFERENCES

- [1] <http://www.baiaerosystems.com>.
- [2] D. N. Borys and R. Colgren, "Advances in Intelligent Autopilot Systems for Unmanned Aerial Vehicles," presented at AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, California, 2005.
- [3] <http://www.rotomotion.com>.
- [4] <http://www.procerusuav.com>.
- [5] <http://www.micropilot.com>.
- [6] <http://www.ezi-nav.com>.
- [7] <http://www.o-navi.com>.
- [8] R. H. Klenke, W. C. S. IV, and M. A. Motter, "A High-Throughput Processor for Flight Control Research Using Small UAVs," presented at 25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference, San Francisco, California, 2006.
- [9] <http://www.cloudcaptech.com>.
- [10] <http://www.microboticsinc.com>.
- [11] D. Jung, E. J. Levy, D. Zhou, R. Fink, J. Moshe, A. Earl, and P. Tsiortras, "Design and Development of a Low-Cost Test-Bed for Undergraduate Education in UAVs," *44th IEEE Conference on Decision and Control, and the European Control Conference*, pp. 2739-2744, 2005.
- [12] D. Kingston, R. Beard, T. McLain, M. Larsen, and W. Ren, "Autonomous Vehicle Technologies for Small Fixed Wing UAVs," presented at 2nd AIAA "Unmanned Unlimited" Systems, Technologies, and Operations, San Diego, California, 2003.
- [13] A. D. Kahn and J. C. Kellogg, "Low Complexity, Low Cost, Altitude Heading Hold Flight Control System," *IEEE AESS Systems Magazine*, pp. 14-18, 2003.
- [14] A. Sagahyoon, M. A. Jarah, A. Al-Ali, and M. Hadi, "Design and Implementation of a Low Cost UAV Controller," *2004 IEEE International Conference on Industrial Technology* pp. 1394-1397, 2004.
- [15] P. Y. Oh and W. E. Green, "CQAR: Closed Quarter Aerial RobotDesign for Reconnaissance, Surveillance and Target Acquisition Tasks in Urban Areas," *International Journal of Computational Intelligence*, vol. 1, pp. 353-360, 2004.
- [16] R. J. Wood, S. Avadhanula, E. Steltz, M. Seeman, J. Entwistle, A. Bachrach, G. Barrows, S. Sanders, and R. S. Fearing, "Design Fabrication and Initial Results of a 2g Autonomous Glider," *IEEE*, pp. 1870-1877, 2005.
- [17] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor," *RSJ International Conference on Intelligent Robots and Systems*, pp. 2451-2456, 2004.
- [18] S. Todorovic and M. C. Nechyba, "A Vision System for Intelligent Mission Profiles of Micro Air Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 53, pp. 1713-1725, 2004.
- [19] Y.-J. Yang, J.-P. Chen, J.-S. Cheng, C. Zhang, and Y.-L. Xiao, "Autonomous Micro-Helicopter Control Based on Reinforcement Learning with Replacing Eligibility Traces," *Proceedings of the First International Conference on Machine Learning and Cybernetics*, pp. 860-864, 2002.
- [20] M. D. Bugajska and A. C. Schultz, "Coevolution Form and Function in the Design of Micro Air Vehicles," *IEEE Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware*, 2002.
- [21] S. H. McIntosh, S. K. Agrawal, and Z. Khan, "Design of a Mechanism for Biaxial Rotation of a Wing for a Hovering Vehicle," *IEEE/ASME Transactions on Mechatronics*, vol. 11, pp. 145-153, 2006.
- [22] S. E. Lyshevski, "Distributed Control of MEMS-Based Smart Flight Surfaces," *Proceedings of the American Control Conference*, pp. 2351-2356, 2001.
- [23] A. S. Wu, A. C. Schultz, and A. Agah, "Evolving Control for Distributed Micro Air Vehicles," *IEEE*, pp. 174-179, 1999.
- [24] J. M. Pflimlim, P. Soueres, and T. Hamel, "Hovering Flight Stabilization in Wind Gusts for Ducted Fan UAV," *43rd IEEE Conference on Decision and Control*, pp. 3491-3496, 2005.
- [25] D. Sun, H. Wu, R. Zhu, and L. C. Hung, "Development of

- Micro Air Vehicles Based on Aerodynamic Modeling Analysis in Tunnel Tests," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2235-2240, 2005.
- [26] H.-y. Wu, D. Sun, Z.-y. Zhou, S.-s. Xiong, and X.-h. Wang, "Micro Air Vehicle: Architecture and Implementation," *Proceedings of the 2003 International Conference on Robotics & Automation*, pp. 534-539, 2003.
- [27] F. Ruffier, S. Viollet, S. Amic, and N. Franceschini, "Bio-Inspired Optical Flow Circuits for the Visual Guidance of Micro-Air Vehicles," *IEEE*, pp. III-846-III-849, 2003.
- [28] S. Taamallah, A. J. C. d. Reus, and J.-F. Boer, "Development of a Rotorcraft Mini-UAV System Demonstrator," *IEEE*, vol. 2005, pp. 11.A.2-1-11.A.2-15, 2005.
- [29] J. Evans, G. Inalhan, J. S. Jang, R. Teo, and C. J. Tomlin, "Dragonfly: A Versatile UAV Platform for the Advancement of Aircraft Navigation and Control," *IEEE*, pp. 1.C.3-1-1.C.3-12, 2001.
- [30] J. L. Campbell and J. T. Kresge, "Brumby Uninhabited Aerial Vehicle Flight Dynamics-Instrumentation and Flight Test Results," *IEEE*, 2003.
- [31] G. Cai, K. Peng, B. M. Chen, and T. H. Lee, "Design and Assembling of a UAV Helicopter System," *2005 IEEE International Conference on Control and Automation*, pp. 697-702, 2005.
- [32] S.-J. Lee, S.-P. Kim, T.-S. Kim, H.-K. Kim, and H.-C. Lee, "Development of Autonomous Flight Control System for 50m Unmanned Airship," *IEEE*, pp. 457-462, 2004.
- [33] E. N. Johnson, S. G. Fontaine, and A. D. Kahn, "Minimum Complexity Unnhabited Air Vehicle Guidance and Flight Control System," *AIAA Digital Avionic Conference*, pp. 1-9, 2001.
- [34] W. E. Hong, J. S. Lee, L. Rai, and S. J. Kang, "RT-Linux based Hard Real-Time Software ARchitecture for Unmanned Autonomous Helicopters," *11th IEEE Conference on Embedded and Real-Time Compute Systems an Applications*, 2005.
- [35] T. Brotherton, R. Luppold, P. Padykula, and S. L. Richard Wade, "Generic Integrated PHM / Controller System," *IEEE*, 2005.
- [36] H. B. Christopherson, W. J. Pickell, A. A. Koller, S. K. Kannan, and E. N. Johnson, "Small Adaptive Flight Control Systems for UAVs using FPGA/DSP Technology," *American Institute of Aeronautics and Astronautics*.
- [37] A. A. Proctor, B. Gwin, S. K. Kannan, and A. A. Koller, "Ongoing Development of an Autonomous Aerial Reconnaissance System at Georgia Tech."
- [38] R. H. Klenke, "A UAV-Based Computer Engineering Capstone Senior Design Project," *IEEE International Conference on Microelectronic Systems Education*, 2005.
- [39] G. Grillmayer, M. Hirth, F. Huber, and V. Wolter, "Development of an FPGA Based Attitude Control System for a Micro-Satallite," presented at AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Keystone, Colorado, 2006.
- [40] M. Kontitsis, K. P. Valavanis, and R. Garcia, "A simple low cost vision system for small unmanned VTOL vehicles," presented at IEEE/RSJ International Conference on Intelligent Robots and Systems 2005, Edmonton, Alberta, Canada, 2005.
- [41] F. Krach, B. Frackelton, J. Carletta, and R. Veillette, "FPGA-Based Implementation of Digital Control for a Magnetic Bearing," presented at Proceedings of the American Control Conference, Denver, Colorado, 2003.
- [42] Z. Fang, J. E. Carletta, and R. J. Veillette, "A Methodology for FPGA-Based Control Implementation," *IEEE Transactions on Control Systems Technology*, vol. Vol 13, pp. 977-987, 2005.