COVER SHEET

A HIGH PERFORMANCE FUZZY LOGIC ARCHITECTURE FOR UAV DECISION MAKING

Paul Wu[1], Pritesh Narayan[1], Duncan Campbell[1], Michael Lees[2], Rodney Walker[1]
[1]School of Engineering Systems, Queensland University of Technology
[2]Department of Electrical and Electronic Engineering, University of Melbourne
Australia
p.wu@qut.edu.au

ABSTRACT
The majority of Unmanned Aerial Vehicles (UAVs) in operation today are not truly autonomous, but are instead reliant on a remote human pilot. A high degree of autonomy can provide many advantages in terms of cost, operational resources and safety. However, one of the challenges involved in achieving autonomy is that of replicating the reasoning and decision making capabilities of a human pilot. One candidate method for providing this decision making capability is fuzzy logic. In this role, the fuzzy system must satisfy real-time constraints, process large quantities of data and relate to large knowledge bases. Consequently, there is a need for a generic, high performance fuzzy computation platform for UAV applications. Based on Lees' [1] original work, a high performance fuzzy processing architecture, implemented in Field Programmable Gate Arrays (FPGAs), has been developed and is shown to outclass the performance of existing fuzzy processors.

KEY WORDS
Fuzzy systems, hardware implementation, unmanned aerial vehicle, pipelining, decision support systems, parallel computing

1. Introduction

In recent times, Unmanned Aerial Vehicles (UAVs) have been employed in an increasingly diverse range of applications. Numerous UAV market forecasts portray a burgeoning future, including predictions of a USD10.6 billion market by 2013 [2]. Within the civilian realm, UAVs are useful in performing a wide range of airborne science missions such as disaster monitoring, search and rescue, and atmospheric observation [3]. However, operation of UAVs in the National Air Space (NAS) requires an equivalent level of safety to that of a human pilot [4]. Achieving higher levels of onboard autonomy helps to address this safety requirement. At the same time, it also reduces the susceptibility to communications failure (with less reliance on a remote pilot), lowers the operational costs, and decreases operator workload.

There are many components required to replicate the capabilities of a human pilot onboard a UAV. One significant component of this is replicating the decision making capabilities. A candidate approach to achieving this includes the application of fuzzy logic.

1.1 Fuzzy Logic

There has been increasing interest in employing fuzzy logic in applications that have large rule bases, real-time computational constraints and require processing of large quantities of data. These applications present a computational challenge as the fuzzy processor must meet stringent performance requirements. Examples include such diverse areas as fuzzy image processing, data mining, decision support, trajectory generation and trajectory tracking [5,6].

1.2 UAV Decision Making

The motivation for the work described in this paper is the implementation of decision making systems within UAVs. Such systems are required to direct UAV responses to various scenarios equivalent to those of a human pilot. Examples include path planning, collision avoidance and forced landings [7,8]. This level of reasoning and decision making onboard a UAV can be modelled hierarchically as: *deliberation*, *sequencing*, and *primitive actions* [9].

A deliberator (the decision maker) prescribes tasks (to achieve some goal) to a task sequencing layer. This layer resolves high level tasks into a sequence of individual primitive tasks to a reactive action layer which executes real world actions. [9]

Performing such cognitive activities inherently requires the evaluation of multiple and possibly conflicting objectives. Factors such as unanticipated weather changes, errors in terrain databases and vehicle subsystem failures are difficulties that human pilots routinely deal with, but are beyond the capability of most current UAVs. Intelligent control, and more specifically fuzzy logic, is seen as a potential method for encapsulating pilot behavior onboard a pilot-less aircraft. As these systems are expected to perform in near real-time due to the rapid dynamics of fixed wing aircraft, it is imperative that the fuzzy system meets real-time processing requirements.

Computational platforms for fuzzy logic are an important consideration in terms of implementing UAV decision systems, as well as other control, navigation and path planning systems in real-time. In this regard, reconfigurable logic, in the form of Field Programmable Gate Arrays (FPGAs) has emerged as a candidate computational platform for these applications.

2. Background

The application of high performance fuzzy logic to UAV decision making is not new. Lundell [10] demonstrated the applicability of fuzzy logic to UAV decision making with an implementation of a fuzzy search and strike decision making algorithm. Multiple parameters (or dimensions in decision space) can be evaluated when deciding on a course of action. These include spatial, temporal and other aircraft related (e.g. fuel) constraints. It can be seen that such a fuzzy multi-criteria decision making system needs to process great quantities of data using a large array of rules.

2.1 Current Application of Fuzzy Logic in UAVs

Intelligent control architectures implemented on board UAVs are generally variations of the three tiered robotics architecture pioneered by Bonasso [9]. Boskovic [11] presents a variation of this multi-layered framework for UAVs. (Figure 1) This framework makes provisions for the entire range of onboard UAV functionalities, from decision making and path planning to low level reactive control for preservation of vehicle safety.

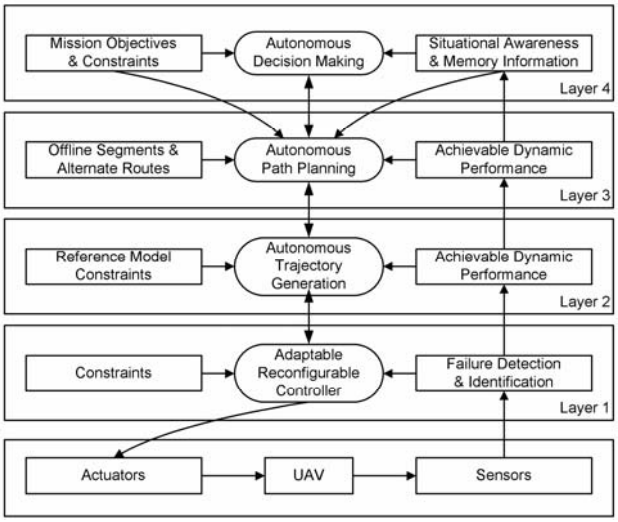

Figure 1: UAV Conceptual Framework [11]

2.1.1 Autonomous Path Planning

Fuzzy logic based techniques have been applied to various components within intelligent control architectures for UAVs (see Figure 1). Autonomous path planning involves the generation of waypoints based on mission requirements. Shi [12] presents a novel technique for navigation and path planning of a rotary UAV in unknown environments through fuzzy reasoning.

2.1.2 Autonomous Trajectory Generation

The trajectory generation layer creates an optimal path through the waypoints with respect to a particular cost function (such as fuel or distance). Nojima [13] uses a fusion of fuzzy logic and genetic algorithms to generate trajectories for mobile robots. As UAVs are essentially mobile robots that fly, it is conceivable that fuzzy trajectory generation could be applied to UAVs.

2.1.3 Fuzzy Control

Fuzzy control is a popular application of fuzzy theory that has been extensively applied to UAV platforms. For example, Dong [14] describes a fuzzy controller that performs trajectory tracking (manoeuvring a UAV and track a generated trajectory).

2.1.4 Computational Considerations

The decision making and path planning layers do not have explicit real-time requirements as initial mission planning is generally computed offline. Updates take place during the mission as required. A mission update (or re-plan) requires the generation of a new set of mission waypoints that take into consideration the achievable dynamic performance of the aircraft. The planner must process a multitude of data each time a re-planning exercise occurs. These could include offline data in the form of mapped terrain or airspace boundaries, sensor data, and mission objectives and constraints. In this situation a fuzzy processing capability with a high throughput could greatly accelerate the mission re-planning phases and potentially decrease mission completion times.

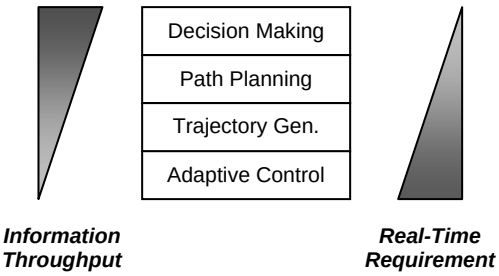

Figure 2: Throughput and Real-Time Requirements of the Different Decision Making Layers

Lower layers perform the task of trajectory tracking, whilst avoiding any unforeseen obstacles and ensuring dynamic stability of the UAV (through the use of adaptive fuzzy controllers). These tasks must occur in real-time (Figure 2) as a processing unit with insufficient computational capacity could lead to inadequate control of the aircraft. This can be avoided with the use of a fuzzy processor with low latency.

2.2 Fuzzy Real-Time Implementation

First introduced in 1965 by Zadeh [15], fuzzy logic essentially provides a method for mapping input space to output space in a non-linear, rule based fashion. Fuzzy sets form the foundation of fuzzy reasoning and are a

form of multi-valued logic where an element can belong to a set (or Membership Function MF) with varying degrees of membership between 0 and 1 (typically). Generally, a fuzzy calculation involves 3 main stages:

1. Fuzzification – where non-fuzzy (or crisp) real world data is transformed into fuzzy degrees of membership
2. Inferencing – where the rules are applied to determine the output fuzzy variables
3. Defuzzification – where the output fuzzy variables are resolved into crisp values. [6]

A fuzzy processor needs to execute these three stages in sequence to create the input-output mapping. There have been many fuzzy processor implementations in the literature with examples of both digital and analogue fuzzy processing techniques. Typically, analogue fuzzy processors consume less power than their digital counterparts but are limited in terms of data precision [6]. Dualibe [16] presents an analogue processor that achieved a throughput of 5.26MFLIPS (Mega Fuzzy Logic Inferences Per Second). This architecture is limited to Sugeno inferencing and only offers the equivalent of 5 or 6 bits of precision.

By contrast, digital fuzzy processors offer more resolution (8 bits or more) and are common due to the ubiquity and ease of use of development tools [6]. Some relatively recent architectural implementations include that by Salcic [17] which attains 50 MFLIPS using true Centre of Gravity (CoG) defuzzification. Samman [18] presents another architecture employing Sugeno inferencing that has a pipeline delay of 100ns. Aranguren [19] describes a balanced implementation (with regards to performance and resource usage) that has a system latency of 423ns. Similarly, Daijin [20] implemented an 8 bit processor that has a system latency of 5.14ms. However, Lees' [1] implementation arguably provides the highest performance in terms of throughput (1.2GFLIPS with max defuzzification and 25MFLIPS with CoG defuzzification) and system pipeline delay (28.7ns). This also surpasses the performance obtained by the Motorola 68HC12 microcontroller which makes fuzzy computations in the order of microseconds (given a clock cycle duration of 125ns) [21].

Daijin, Aranguren, Samman and Lees all made use of FPGAs. Besides the flexibility offered through reconfigurability, modern FPGAs also boast high resource availability in terms of on-chip logic elements, dedicated signal processing elements (such as multipliers) and streamlined circuit development tools (often referred to as Electronic Design Automation or EDA software). Furthermore, modern FPGAs also feature on-chip memory elements (e.g. Cyclone II, Stratix II, Virtex 4, Spartan 3) for caching data. [22,23]

3. Design and Methodology

Based on Lees' [1] architecture, a high performance fuzzy processor is described. This synchronous, 8-bit, fully parallel and pipelined architecture is targeted at programmable logic devices and especially modern FPGAs. The architecture and implementation presented by Lees is further developed with the view of potential adoption in UAV onboard decision making. By adapting the architecture to newer technology (notably the Altera Cyclone II FPGA), even greater performance outcomes were achieved**.**

3.1 Parameters

Fuzzy systems are characterised by several important design parameters. These are:

- Number of inputs (or input variables)
- Number of outputs
- Number of input membership functions
- Number of output membership functions
- Number of rules
- Type of fuzzification
- Type of inferencing
- Type of defuzzification

The architecture is extensible to systems with any number of rules, inputs and outputs. It employs singleton fuzzification and Mamdani inferencing (derived from Zadeh's compositional rule of inference) using Generalised Modus Ponens (or forward chaining) reasoning:

Implication statement:	IF X is A THEN Y is B
Premise:	X is A'
Conclusion:	Y is B'

where A, A' and B, B' are input and output MFs respectively. In this case, X is A is the antecedent and Y is B is the consequent; note that in general, each fuzzy rule would take the form of:

$$\text{IF } X_1 \text{ is } A_1 \text{ and } X_2 \text{ is } A_2 \text{ and... THEN } Y_1 \text{ is } B_1 \text{ and } Y_2 \text{ is } B_2 \text{ and...} \quad (1)$$

With Mamdani inferencing, the output set B' for a single input single output system is calculated as:

$$\mu_{B'}(z) = \sup \min \left[\mu_{A'}(x), \max_{i=1..n} \min \left(\mu_{Ai}(x), \mu_{Bi}(z) \right) \right] \quad (2)$$

where A' is the input MF, A and B are input and output MFs respectively associated by n rules on universes of discourse x and z respectively. Extension to multiple input single output (MISO) systems can be done by performing conjunction on the antecedents. Multiple Input Multiple Output (MIMO) systems, on the other hand, can be broken down into multiple parallel MISO systems [6].

Min-max inferencing was implemented where min was the conjunctive (or t-norm) and max the disjunctive (or s-

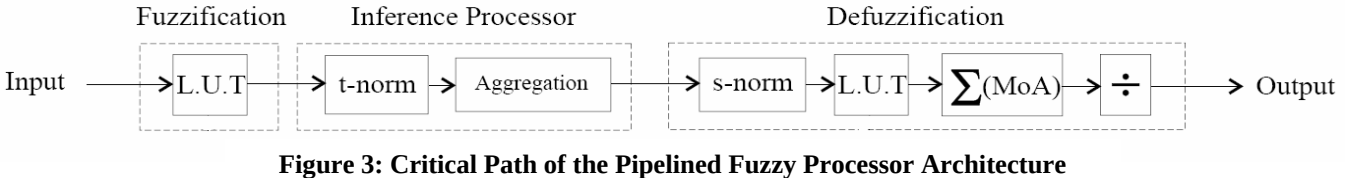

Figure 3: Critical Path of the Pipelined Fuzzy Processor Architecture

norm) operator. As the architecture is modular, other conjunctive and disjunctive operators could be implemented such as product-sum. Note that when more than one rule implicates the same output MF, the maximum implication value is taken; this process is called aggregation.

With regards to defuzzification, both true CoG defuzzification (refer equation (3)) and max defuzzification (where the output z_o corresponds to the highest point in $\mu_{B'}(z)$) were implemented. Again, as the architecture is modular, other forms of defuzzification could also be implemented.

$$z_o = \frac{\int z\mu_{B'}(z)dz}{\int \mu_{B'}(z)dz} \tag{3}$$

3.2. Architecture

An overview of the pipelined fuzzy processing architecture is shown in Figure 3.

3.2.1 Fuzzification

Singleton fuzzification was implemented using Look Up Tables (LUT) as LUTs are the fastest single element on an FPGA [24]. Each 8-bit input value is interpreted as the address to a LUT to obtain the degree of membership to that MF. In this way, the degree of membership to every MF for every input variable is obtained in parallel. It can be seen that this reduced runtime processing is offset by increased memory resource usage. However, this is not a problem given the availability of extensive memory resources in modern FPGAs.

3.2.2 Inferencing

Based on equation (2), a fully parallel and pipelined inferencing structure was derived as shown below.

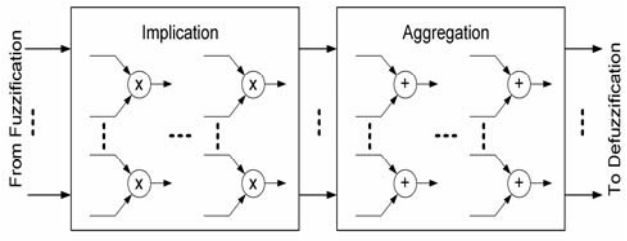

Figure 4: Inferencing Architecture [1]

For each input in implication (or each rule in aggregation), a tree structure was employed using the relevant min or max operator to arrive at the resulting output as shown in Figure 5.

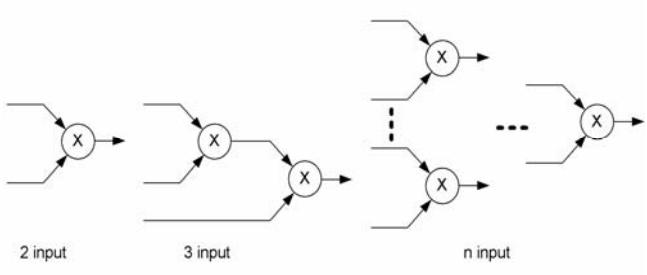

Figure 5: Implication/Aggregation Tree Structure

Unlike Lees [1], each min or max block was sub-pipelined into two sub-stages to further reduce the system pipeline latency. This was possible due to advancements in FPGA technology resulting in reduced clock skew, memory register propagation and setup delays. Additionally, a more effective design of the internal logic for min and max blocks also helped in reducing the pipeline latency [25].

3.2.3 Defuzzification

Both max defuzzification and CoG defuzzification were implemented. Max defuzzification can be performed with a tree of disjunctive elements similar to that shown in Figure 5.

On the other hand, CoG defuzzification, which is often touted as the most computationally expensive process in a fuzzy system, was implemented using a partially pre-calculated approach. First proposed by Lees [1], this method assumes that there are only 2 degrees of overlap between output MFs. Membership functions are divided into non-overlapping segments (A, B and C in Figure 6). The Moment of Area (obtained from the numerator of equation (3)) and Area (from the denominator) are pre-calculated and stored in LUTs for each segment for all alpha cut levels.

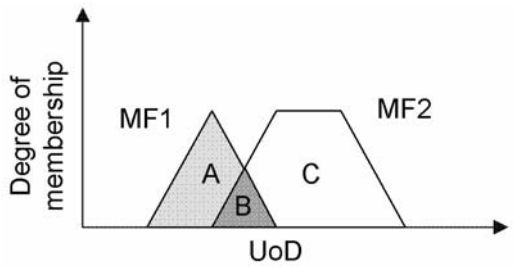

Figure 6: Segmenting Two Overlapping MFs into Non-overlapping Regions A, B and C

In this way, the aggregated truncation level from the inferencing stage can be used to look up the moment of area and area in one step. The sums for the moments of

area and area for all segments are calculated and a division is performed to obtain the final CoG. (Figure 7)

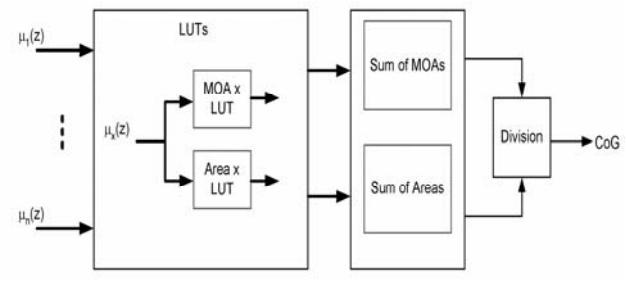

Figure 7: Defuzzification [1]

Lees' implementation is further enhanced by the use of a pipelined divider thereby greatly reducing the system pipeline latency, which in turn increases the net data throughput.

3.3 FPGA Implementation

A 36 rule, dual input single output fuzzy system was implemented as a test system as it reflected the test system employed by Lees [1]. It had 6 MFs per input variable and 6 MFs per output variable. The system made use of singletons for fuzzification, employed Mamdani inferencing and offered a choice of max or CoG defuzzification.

With max defuzzification, a pipeline delay of 5ns, corresponding to a throughput of 7.2GFLIPS was simulated. The entire fuzzy processor used just 7% of the available logic elements on the Cyclone II FPGA (EP2C35F672C6) and 5% of on chip memory. This result (7.2GFLIPS) was confirmed through hardware experimentation. Similarly, a throughput of 3.6GFLIPS was achieved with a pipeline delay of 10ns using CoG defuzzification. Again, resource utilisation was just 8% of logic elements and 14% of available memory.

It can be seen that the achieved performance is an order of magnitude greater than all of the aforementioned fuzzy processors and even greater than that originally presented by Lees [1]. Therefore, this architecture is particularly suited for applications requiring very high fuzzy processing rates.

4. Evaluation

It is prudent to make some observations regarding the commonly used measure of performance of fuzzy systems (based on Fuzzy Logic Inferences Per Second).

4.1 Performance Measurement

Presently, there is a lack of consensus on a standard method of performance measurement and this gives rise to ambiguity when comparing the performance of different fuzzy processors [6]. However, a survey of the literature revealed that many authors make use of the FLIPS measurement. The number of FLIPS is calculated by multiplying the number of inference rules by the number of complete end-to-end fuzzy evaluations per second. Therefore, the FLIPS criterion only provides an indication of the throughput of the processor and is heavily influenced by the size of the rule base.

Another method for performance assessment is to measure the system latency but this too is also dependent on the test fuzzy system. One could investigate the latencies of components in the system to obtain a more pertinent evaluation of the scalability of the architecture. Another method for measuring performance is to find the pipeline delay – this provides an indication of the throughput of the architecture that is independent of the test system.

4.2 Fuzzy Processor Performance

The experimental results are summarised and compared against other works in Table 1. It can be seen that the performance achieved is significantly greater than that obtained by other authors. It was found that there were some timing discrepancies between simulation and physical hardware tests but these were attributed to Altera's preliminary simulation timing characteristics for the Cyclone II.

Table 1. Performance Comparison

Author	Throughput	System latency
Proposed work with CoG defuzzification	3.6GFLIPS 10ns pipeline delay	292ns
Proposed work with Max defuzzification	7.2GFLIPS 5ns pipeline delay	95.2ns
Lees [1] with CoG defuzzification	25MFLIPS	980ns
Lees [1] with Max defuzzification	1.2GFLIPS 28.7ns pipeline delay	265.5ns
Dualibe [16]	5.26MFLIPS	
Salcic [17] with CoG defuzzification	50MFLIPS	
Samman [18]	Pipeline delay 100ns	
Aranguren [19]		423ns
Daijin [20]		5.14ms
Motorola 68HC12		µseconds

5. Conclusion

The fuzzy computational architecture presented in this paper is shown to produce vastly greater levels of throughput (3.6GFLIPS) than that of existing fuzzy processing systems. Based on the original work by Lees [1], it further capitalises on opportunities for pipelining, parallelisation and newer FPGA technology, including on-chip memory and lower logic element latencies. The architecture is extensible and easily adaptable to different fuzzy systems with differing numbers of inputs, outputs, rules, methods of inferencing and defuzzification. Hence, this generic architecture is suited to a wide variety of fuzzy applications which require high performance.

In particular, the combination of low system latency and high throughput makes it especially suited to UAV decision making. The need for real-time fuzzy processing, coupled with the complexity involved in multi-criteria decision making could be satisfied using a fuzzy processor employing the architecture presented here.

Future Work

It is a natural extension to this work to devise a computational architecture for type-2 fuzzy reasoning which embeds the ability to cater for uncertainty in input data. This is of particular interest in UAVs where there may be elements of uncertainty through sensors, or inferences of particular aspects of the state of the aircraft. This, and other candidate computational platforms, will be implemented and evaluated on test UAV platforms for decision making operations.

Acknowledgements

The authors wish to acknowledge the support of the Queensland University of Technology (QUT), the Australian Research Centre for Aerospace Automation (ARCAA) and the CSIRO.

References

[1] M. J. Lees and D. A. Campbell, A 1.2 GFLIPS fuzzy logic inference processor: breaking the billion rules per second barrier, *Proc. International Conference on Neural Information Processing and Intelligent Information Systems*, 1997, 696-699.

[2] T. Cox, *Civil UAV Capability Assessment* (NASA, 2004)

[3] S. S. Wegener, S. S. Schoenung, J. Totah, D. Sullivan, J. Frank, F. Enomoto, C. Frost and C. Theodore, UAV Autonomous Operations for Airborne Science Missions, *Proc. AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, Chicago, Illinois, 2004.

[4] Federal Aviation Administration, *Order 7610.4K Special Military Operations* (2004)

[5] J. C. Bezdek, M. R. Pal, J. Keller and R. Krisnapuram, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing* (Norwell, MA: Kluwer Academic Publishers, 1999)

[6] E. H. Ruspini, P. P. Bonissone and W. Pedrycz, *Handbook of Fuzzy Computation* (Bristol: Institute of Physics Pub., 1998)

[7] I. Mcmanus, *A Multidisciplinary Approach to Highly Autonomous UAV Mission Planning and Piloting for Civilian Airspace* (Brisbane: QUT, 2004)

[8] D. Fitzgerald, R. Walker and D. Campbell, A Vision Based Forced Landing Site Selection System for an Autonomous UAV, *Proc. International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2005, 397-402.

[9] R. P. Bonasso, D. Kortenkamp, D. P. Miller and M. G. Slack, Experiences with an Architecture for Intelligent, Reactive Agents, *Proc. International Joint Conference on Artificial Intelligence*, 1995.

[10] M. Lundell, J. Tang and K. Nygard, Fuzzy petri net for UAV decision making, *Proc. International Symposium on Collaborative Technologies and Systems*, 2005, 347-352.

[11] J. D. Boskovic, R. Prasanth and R. K. Mehra, A multilayer control architecture for unmanned aerial vehicles, *Proc. American Control Conference*, 2002, 1825-1830 vol.3.

[12] D. Shi, M. F. Selekwa, E. G. Collins, Jr. and C. A. Moore, Fuzzy behavior navigation for an unmanned helicopter in unknown environments, *Proc. IEEE International Conference on Systems, Man and Cybernetics*, 2005, 3897-3902 vol.4.

[13] Y. Nojima, F. Kojima and N. Kubota, Trajectory generation for human-friendly behavior of partner robot using fuzzy evaluating interactive genetic algorithm, *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2003, 306-311 vol.1.

[14] T. Dong, X. H. Liao, R. Zhang, Z. Sun and Y. D. Song, Path tracking and obstacle avoidance of UAVs - Fuzzy logic approach, *Proc. IEEE International Conference on Fuzzy Systems*, 2005, 43-48.

[15] L. A. Zadeh, Fuzzy Sets, *Inf. Control*, *8*, 1965, 338–353.

[16] C. Dualibe, P. Jespers and M. Verleysen, A 5.26 MFLIPS programmable analogue fuzzy logic controller in a standard CMOS 2.4μ technology, *Proc. IEEE International Symposium on Circuits and Systems*, 2000, 377-380 vol.5.

[17] Z. Salcic, High-speed customizable fuzzy-logic processor: architecture and implementation, *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, *31*(6), 2001, 731-737.

[18] F. A. Samman and E. Y. Syamsuddin, Programmable fuzzy logic controller circuit on CPLD chip, *Proc. Asia-Pacific Conference on Circuits and Systems*, 2002, 561-564 vol.2.

[19] G. Aranguren, M. Barron, J. L. Arroyabe and G. Garcia-Carreira, A pipe-line fuzzy controller in FPGA, *Proc. Sixth IEEE International Conference on Fuzzy Systems*, 1997, 635-640 vol.2.

[20] D. Kim, An implementation of fuzzy logic controller on the reconfigurable FPGA system, *Industrial Electronics, IEEE Transactions on*, *47*(3), 2000, 703-715.

[21] Freescale Semiconductor, *CPU12 Reference Manual* (2006)

[22] Xilinx, *Virtex 4* (2004)

[23] Altera Corporation, *Cyclone II Device Handbook* (2005)

[24] K. Morris, Stratix II - Altera unveils new 90nm architecture, *FPGA and Programmable Logic Journal*, *2*, 2004.

[25] P. Wu, *A Reconfigurable Hardware Fuzzy Logic Implementation* (Brisbane, Australia: Queensland University of Technology, 2005)