FPGA Hardware Nonlinear Control Design for Modular Nanosatellite Attitude Control System

Junquan Li
Department of Earth and Space Science and Engineering, York University Toronto, Canada
junquanl@yorku.ca

Mark Post
Space Mechatronic Systems Technology Laboratory
Department of Design, Manufacture and Engineering Management, University of Strathclyde
Glasgow, United Kingdom
mark.post@strath.ac.uk

Regina Lee
Department of Earth and Space Science and Engineering, York University
Toronto, Canada
reginal@yorku.ca

Abstract—CubeSat attitude control systems must be compact, fast, and accurate to achieve success in space missions with stringent control requirements. Nonlinear control strategies allow the creation of robust algorithms for orbit and attitude control, but can have higher processing requirements in operation. In addition, it is desirable to implement distributed components of a CubeSat control system across hardware as much as possible to decrease the load on a central computing unit and to make the system more tolerant to point failures.

Field Programmable Gate Array technology has become popular as a solution to the limited electronic space and more demanding speed requirements present in new-generation CubeSat systems. This paper will focus on demonstrating the feasibility and effectiveness of a proposed nonlinear adaptive fuzzy controller implemented on a Field Programmable Gate Array as part of a highly-integrated space hardware system that is under development.

To facilitate integration with other system-on-a-chip CubeSat hardware, the new controller is implemented as a single logic block that can be added as a module to an FPGA-based system. Unlike the majority of FPGA-based controller systems that are based on soft-core processors, the controller is implemented directly in digital logic, saving resources on the FPGA fabric and increasing speed significantly. From a stream of sensor data, the logic-based adaptive fuzzy controller performs all filtering and calculations necessary for attitude control of the CubeSat, with the addition of a serial interface used for producing debugging logs.

The difficulty of this approach is largely due to the limited functionality available in existing FPGA libraries. We overcome this by using freely-available High-Level Synthesis tools to convert proven C code into Hardware Description Language, which is then integrated on a modular level with the rest of the FPGA-based system. As full floating-point calculation requires high complexity in FPGA fabric, fixed-point arithmetic implementations are used in place of floating-point implementations as needed.

A complete design of the resulting system is provided. Purely numerical simulations for the controller are done using Simulink on a host computer, and these simulations are compared with results obtained from hardware-in-the-loop testing of the FPGA-based controller using simulated sensor inputs. A practical comparison is also made between the high-precision floating point calculation of the simulations, and the more limited precision available on the FPGA hardware.

Evaluation of the simulation and hardware test results shows that the control performance of the FPGA hardware control system is suitable for small satellite control, can meet precise pointing requirements, and that the algorithmic speed of the FPGA-based controller is significantly higher than the conventionally-programmed controller despite running at a much lower hardware clock speed.

This implementation of the proposed controller shows the practicality of directly implementing nonlinear control algorithms directly on logic hardware as an alternative to microcontrollers and soft-core processors. Direct hardware implementations using modern synthesis methods are expected to continue to increase speed, reliability, and hardware modularity of CubeSat control algorithms by moving essential components into reconfigurable hardware.

1. INTRODUCTION

Microsatellites, nanosatellites, and CubeSats are the fastest growing segment of the spacecraft market. Yusend (a Cubsat under development at York University Canada) is a 10x10x10 cm three unit nanosatellite concept.1 The current total mass is approximately 1 kg. This research is focused on the design of an FPGA-based attitude control system for such a cubesatellite. In pursuing the next generation of cubesatellites, on-board processing is very important (such as attitude determination, power electronics, on-board computer design, and communication). An FPGA provides a promising design alternative to conventional microprocessor-based designs. The FPGA allows flexibility, low cost, redundancy and
configurability, and is our focus for hardware implementation of current research in compact, efficient nonlinear control strategies. The current study on FPGA-based subsystem design heavily relies on laboratory prototypes and bench testing. A primary challenge in developing high-performance attitude control systems for nanosatellites is that of bus integration and efficient miniaturization of redundant control components. The use of reconfigurable logic allows many functions to be performed in parallel at high speed without the need for multiple discrete components or a multifunction bus. A modular FPGA-based control architecture for nanosatellite use is under development for these reasons.

Research on FPGA-based control of systems is growing quickly as FPGA applications such as industrial automation, robotic surgery, and space mechatronic systems such as the Canadarm demand more accuracy, reliability, and performance. FPGA-based controllers offer advantages such as high processing speed and high functional integration. Some FPGA-based control systems are focusing on FPGA soft-core processors but in general, soft-core processor approaches lack the speed of pure-logic implementations while gaining little over hard-core microcontroller-based implementations. In this paper, we are going to focus on complex nonlinear control hardware implementation. PID controllers have been used for satellite attitude control systems for years, and advanced adaptive control theories have also been well developed, although most of them require system dynamic models. Intelligent tools, such as neural networks, genetic algorithms or fuzzy logic systems can be applied with adaptive control theories to improve overall performance without the need for a system model. Many successful FPGA control designs for nonlinear systems have been used for motor drives or robot manipulators. PID controller FPGA-based fuzzy PID control and sliding mode control are implemented using VHDL language for FPGA devices (Xilinx or Altera FPGA boards) with design tools (Xilinx Integrated Software Environment (ISE) or Altera’s Quartus II). Xilinx and Altera are the two leading FPGA vendors with different MATLAB co-simulation tools and logic core libraries for FPGA control design.

FPGAs have been used for spacecraft and satellite applications, and a neural network control system has been used for spacecraft power systems. FPGA based on onboard computers have been applied for a 100kg three axis stabilized micro-satellite at the University of Stuttgart. In this paper, an attitude control system has been implemented as hardware in a Xilinx Zynq hybrid FPGA with ARM microcontroller. The ACS architecture has high performance with a fast processing speed. For development purposes, the ACS sensors and actuators are currently simulated, and data is logged, using the ARM microcontroller part of the Zynq, while the controller itself is implemented in the FPGA part. For integration of the FPGA into CubeSats, a Xilinx Virtex4QV radiation tolerant FPGA can be used for 1U and 3U CubeSats. We propose the use of a novel nonlinear FPGA ACS algorithm for 1U CubeSat using Xilinx FPGA hardware for future missions.

The organization of this paper proceeds as follows: First, the nonlinear system model and measurement model are presented. Then, the attitude control systems using FPGA control and Type-2 fuzzy control are addressed. Third, the simulation and hardware-in-the-loop of FPGA based ACS architectures are presented. Finally, the performance of the proposed ACS will be demonstrated in the final paper.

2. Mathematical Model of the Nanosatellite

The satellite is modeled as a rigid body with reaction wheels that provide torques about three mutually perpendicular axes that define a body-fixed frame $B$. The equations of motion are given by

$$\dot{\omega} = -\omega \times (J_ω \omega + A_1 J_ω \Omega) + A_1 \tau + d$$

$$\dot{q} = \frac{1}{2} \left[ q 1 3 \times 3 + \tilde{q} \tilde{q}^T \right] \omega + \frac{1}{2} A(q) \omega$$

$$\left[ \begin{array}{c} \psi \\ \alpha \\ \gamma \end{array} \right] = \left[ \begin{array}{ccc} 1 & \sin(\psi) \tan(\alpha) & \cos(\psi) \tan(\alpha) \\ 0 & \cos(\psi) & -\sin(\psi) \cos(\alpha) \\ 0 & \sin(\psi) \cos(\alpha) & \cos(\psi) \cos(\alpha) \end{array} \right] \omega$$

where $\omega = (ω_1, ω_2, ω_3)^T$ is the angular velocity of the spacecraft with respect to an inertial frame $I$ and expressed in the body frame $B$, $\Omega$ is the angular velocity of a reaction wheel, $J_ω \in R^{3 \times 3}$ is the inertia matrix of the spacecraft, $\dot{J} = J_ω - A_1 J_ω A_1^T$; $\tau \in R^3$ is the torque control, $A_1$ is the $3 \times 4$ or $3 \times 3$ (depending on the layout and the number of reaction wheels) matrix whose columns represent the influence of each reaction wheel on the angular acceleration of the satellite, $d \in R^3$ is the bounded external disturbance (included Solar Radiation Pressure Disturbance, Aerodynamic drag, and Gravity Gradient Torque), $x^\times \in R^{3 \times 3}$ represents the cross product operator for a vector $x = (x_1, x_2, x_3)^T$ given as

$$x^\times = \left( \begin{array}{ccc} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{array} \right)$$

and the unit quaternion $q = (q_1, q_2, q_3, q_4)^T = (q_1 q_2, q_3, q_4)^T$ represents the attitude orientation of a rigid spacecraft in the body frame $B$ with respect to the inertial frame $I$, which is defined by

$$q = (q_1, q_2, q_3, q_4)^T = e \sin(\theta/2), \quad q_4 = \cos(\theta/2)$$

where $e$ is the Euler axis, and $\theta$ is the Euler angle. $\psi$ is the roll angle about the x-axis, $\alpha$ is the pitch angle about the y-axis, $\gamma$ is the yaw angle about the z-axis. The unit quaternion $q$ satisfies the constraint

$$q^T q = 1$$

The torques generated by the reaction wheels $\tau$ are given by

$$\tau = J_ω (\dot{\Omega} + A_1^T \dot{\omega})$$

To address the attitude tracking problem, the attitude tracking error $q_e = (\bar{q}^T, q_4)^T$ is defined as the relative orientation between the body frame $B$ and the desired frame $D$ with orientation $q_d = (\bar{q}_d^T, q_4 d)^T$, and it is computed by the quaternion multiplication rule as

$$q_e = q_d q - q_d q + q^\times q_d$$

$$q_{4e} = q_{4d} q + q_d^\times q_d$$

where $q_d \in R^4$ and $q_e \in R^4$ satisfy the constraints $q_{4d}^T q_d = 1$ and $q_{4e}^T q_e = 1$, respectively. The corresponding rotation matrix is given by

$$C(q_e) = \left( q_{2e}^2 - q_{4e}^2 \right) I_{3 \times 3} + 2 q_{2e} q_{4e}^T - 2 q_{2e} q_{4e}$$
Note that $\|C\| = 1$ and $\dot{C} = -\omega^\times C$, where $\omega_e = \omega - C\omega_d$ is the relative angular velocity of $B$ with respect to $D$, and $\omega_d \in \mathbb{R}^3$ is the desired angular velocity.

In order to apply the proposed adaptive terminal fuzzy sliding mode control, the equations of motion have to be rewritten as

$$\ddot{q}_e = 2(T^{-1})^T J^{-1} [\dot{\omega} \times (J_\omega \omega + A_1 \omega \Omega) + A_1 \dot{\tau}_w + d] - 4(T^{-1})^T (\dot{T}^{-1}) \dot{\theta}_e$$

where $T = [q_{ke} I + (\dot{q}_e)^\times]^{-1}$ and $T \neq 0$.

The dynamics equation for the nonlinear controller design can be written as:

$$\ddot{q}_e = f(\dot{q}_e, \dot{q}_e) + \ddot{\tau}_w + \ddot{d}$$

A dynamic model of the satellite’s relative attitude tracking is represented by a diagonal matrix $E$.

$$f(\dot{q}_e, \dot{q}_e) = 2(T^{-1})^T [J^{-1} [\dot{\omega} \times (J_\omega \omega + A_1 \omega \Omega)] - 4(T^{-1})^T (\dot{T}^{-1}) \dot{\theta}_e + 2(T^{-1})^T J^{-1} \omega \times \omega]$$

$$\ddot{\tau}_w = E[2(T^{-1})^T J^{-1} A_1 \dot{\tau}_w]$$

$$\ddot{d} = 2(T^{-1})^T J^{-1} d$$

The dynamics equation for the nonlinear controller design can be written as:

$$\dot{X}_e = F(X_e, \dot{X}_e) + G(X_e, \dot{X}_e) \tau_c + \tau_d$$

where $X_e = q_e$, $\dot{X}_e = \dot{q}_e$, $F(X_e, \dot{X}_e) = f(\dot{q}_e, \dot{q}_e)$, $G(X_e, \dot{X}_e) = 1$, $\tau_c = \ddot{\tau}_w$, and $\tau_d = \ddot{d}$.

### 4. Attitude Control Systems

For the ACS nadir pointing and target tracking phase, three magnetic rods and one momentum wheel are used as actuators. The nonlinear adaptive fuzzy sliding controller used for this phase is given by.

In our earlier research paper, adaptive type-1 fuzzy control have been developed and tested on the 1U nanosatellite air bearing testing system. The focus of this paper is to develop a FPGA based type-2 fuzzy control to solve the computation load issues.

#### Type-1 and Type-2 Fuzzy Logic System

Zadeh introduced type-2 fuzzy sets as a generalization of ordinary fuzzy sets. In a type-1 fuzzy set, the degree of membership for each point is a normal fuzzy number can have the range of $[0, 1]$, and the number is a crisp number. A type-2 fuzzy set can have a membership function with uncertainty.

A type-1 fuzzy control system generally includes a fuzzifier, a rule base, an inference engine and a defuzzifier. For this type-1 fuzzy system, a Mamdani minimum inference engine, singleton fuzzifier and center average defuzzier are chosen. A fuzzy system in general is a collection of if-then rules that can be expressed as

$$\forall i : \text{If } x_1 \text{ is } W_1^k, \text{ And } \cdots, \text{ And } x_n \text{ is } W_n^k, \text{ Then } y \text{ is } Z^k.$$

The output of the fuzzy system (using singleton fuzzification, product inference and center average defuzzification) can be written as

$$\varrho^T \zeta = \sum_{l=1}^P \varrho^T_F(X) \prod_{i=1}^N \mu_{W_i}^l(x_i)$$

where $P$ is the total number of fuzzy rules. The $N$ Gaussian membership functions are $\mu_{W_1^l}(x_1), \ldots, \mu_{W_n^l}(x_n)$, and $\zeta$ is a fuzzy basis function defined as

$$\zeta^l(x) = \left( \frac{\prod_{i=1}^N \mu_{W_i}^l(x_i)}{\sum_{j=1}^P \prod_{i=1}^N \mu_{W_i}^j(x_i)} \right).$$

To implement the adaptive fuzzy terminal sliding mode control law, type-1 fuzzy sets over the interval of $x_i$ are defined.

Seven Gaussian membership functions are used in the type-1 fuzzy system for each variable $W_i$ ($i = 1, 2, \cdots, 7$), defined
as
\[
\begin{align*}
\mu_{W1}(x_i) & = \left(1 + \exp(5(x_i + 3 \times a))\right)^{-1} \\
\mu_{W2}(x_i) & = \exp\left(-\frac{(x_i + 2 \times a)^2}{b}\right) \\
\mu_{W3}(x_i) & = \exp\left(-\frac{(x_i + 1 \times a)^2}{b}\right) \\
\mu_{W4}(x_i) & = \exp\left(-\frac{(x_i)^2}{b}\right) \\
\mu_{W5}(x_i) & = \exp\left(-\frac{(x_i - 1 \times a)^2}{b}\right) \\
\mu_{W6}(x_i) & = \exp\left(-\frac{(x_i - 2 \times a)^2}{b}\right) \\
\mu_{W7}(x_i) & = \left(1 + \exp(5(x_i - 3 \times a))\right)^{-1}
\end{align*}
\]

where \(a\) and \(b\) are different constant numbers designed according to \(x_i\). The type-2 fuzzy sets are constructed using the same set of seven Gaussian functions, except that instead of only one function being used about one mean value \(\mu_{W1}\), two mean values \(\mu_{W1}^1\) and \(\mu_{W1}^2\), are used and two functions superimposed (added) above one another.

The details of the constructed Type-2 and Type-1 membership functions are given in Table 1.

### Table 1. Type-2 and Type-1 Fuzzy Membership Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type-2 membership (\mu_{W1}^1) and (\mu_{W1}^2)</th>
<th>Type-1 membership (\mu_{W1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mu_{W1}(x_i))</td>
<td>2.5, 1.5</td>
<td>3</td>
</tr>
<tr>
<td>(\mu_{W2}(x_i))</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(\mu_{W3}(x_i))</td>
<td>1.5, 0.5</td>
<td>1</td>
</tr>
<tr>
<td>(\mu_{W4}(x_i))</td>
<td>0.5, 0.5</td>
<td>0</td>
</tr>
<tr>
<td>(\mu_{W5}(x_i))</td>
<td>0.5, 1.5</td>
<td>1</td>
</tr>
<tr>
<td>(\mu_{W6}(x_i))</td>
<td>1, 2</td>
<td>3</td>
</tr>
<tr>
<td>(\mu_{W7}(x_i))</td>
<td>1.5, 2.5</td>
<td>3</td>
</tr>
</tbody>
</table>

**Adaptive Fuzzy Sliding Mode Control**

A sliding mode control law can be derived using the sign of the switching function and the switching-type function \(\hat{H}\) as
\[
\tau_c = G(X)^{-1} [\dot{\delta}_d - \delta - F(X) - \tau_d - \hat{H}(S) \text{sgn}(S)].
\tag{23}
\]

System functions \(F, G\), and disturbance \(\tau_d\) are usually difficult to be obtained. The indirect adaptive fuzzy sliding mode control law is given as
\[
\tau_c = \hat{G}(X | \delta_g)^{-1} [\dot{\delta}_d - \delta - \hat{F}(X | \delta_f) - \hat{H}(S | \delta_h)]
\tag{24}
\]

where \(F, G\) and \(\tau_d\) are replaced with type-1 or type-2 fuzzy system. For a type 1-fuzzy system, \(\hat{F}(X | \delta_f) = \hat{F}_f \zeta_f(X)\), \(\hat{G}(X | \delta_g) = \hat{G}_g \zeta_g(X)\) and \(\hat{H}(S | \delta_h) = \hat{H}_h \zeta_h(S)\) are used to approximate \(f(X), g(X)\), and the switching-type control law \(H \text{sgn}(S)\). For a type 2-fuzzy system, the following equations are used to approximate \(\hat{F}(X), G(X)\), and the switching-type control law \(H \text{sgn}(S)\).

\[
\dot{\delta}_f = \arg \min_{\delta_f \in \mathcal{F}} \sup_{X \in D_x} \left| \hat{F}(X | \delta_f) - F(X) \right|
\tag{28}
\]
\[
\dot{\delta}_g = \arg \min_{\delta_g \in \mathcal{G}} \sup_{X \in D_x} \left| \hat{G}(X | \delta_g) - G(X) \right|
\tag{29}
\]
\[
\dot{\delta}_h = \arg \min_{\delta_h \in \mathcal{H}} \sup_{S \in D_s} \left| \hat{H}(S | \delta_h) - H(S) \right|
\tag{30}
\]

It is assumed that there exists an optimal fuzzy logic system that can approximate the nonlinear terms \(F(X), G(X)\) and the switching-type control law \(H \text{sgn}(S)\) in Equation 24 such that

\[
\begin{align*}
F(X) - F^*(X | \delta_f) & = \varpi_F(X) \\
G(X) - G^*(X | \delta_g) & = \varpi_G(X) \\
H(S) - H^*(S | \delta_h) & = \varpi_h(S)
\end{align*}
\tag{31}
\]

where \(\varpi_F, \varpi_G\), and \(\varpi_h\) are approximation errors and bounded in the compact set \(U_\varpi\), i.e., \(\|\varpi_F\| \leq \varpi_F, \|\varpi_G\| \leq \varpi_G\) and \(\|\varpi_h\| \leq \varpi_h\). Approximation errors can be reduced by increasing the number of fuzzy rules.

**Nonlinear Controller Design**

The controller design involves the construction of a sliding surface containing tracking errors to ensure that the system is restricted to the sliding surface. It also involves the derivation of parameter adaptation laws and fuzzy logic feedback control gains that can drive the desired trajectory to the sliding surface and maintain it in the manifold. A nonlinear hyperplane based sliding mode can provide a wide variety of design alternatives with fast and finite time convergence. It is redefined as

\[
\dot{S} = \delta + \dot{\delta} + K_0 e^{\frac{p}{q}}
\tag{32}
\]

where \(K_0 R_{3 \times 3}\) is a constant, diagonal, positive-definite, control design matrix, and \(p\) and \(q\) are the positive odd integers \((p < 2q)\).

\[
\dot{S} = \dot{\delta} + \dot{\delta} + K_0 e^{\frac{2p}{q} - 1} \dot{\delta}.
\tag{33}
\]
The adaptive fuzzy terminal sliding control law is then redefined as

\[ \tau_c = \hat{G}(X | \vartheta_g)^{-1} \left[ \ddot{x}_d - E - \hat{F}(X | \vartheta_f) - \dot{h}(S | \vartheta_h) - K_1 S \right] \]  

(34)

where \( K_1 = \text{diag}\{k_{11}, k_{22}, k_{33}\} \), \( I = \text{diag}\{1, 1, 1\} \) and \( E = \dot{\vartheta}_g e^T \dot{\vartheta}_g \). This control term (Equation 34) is not well defined when the estimated matrix \( \hat{G}(X | \vartheta_g) \) is singular. The matrix is generated online via the estimation of the parameters \( \vartheta_g \). In order to implement this control law, additional precautions have to be taken to guarantee that \( \vartheta_g \) remains in the feasible region in which \( \hat{G}(X | \vartheta_g) \) is non-singular. In order to overcome this problem, the control law is modified to be

\[ \tau_{cf} = \hat{G}^T(X | \vartheta_g)\varepsilon_0 I + \hat{G}^T(X | \vartheta_g)\hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)^{-1} E \]

(35a)

\[ - \hat{G}^T(X | \vartheta_g)\hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)^{-1} \dot{F}(X | \vartheta_f) \]

(35b)

\[ - \hat{G}^T(X | \vartheta_g)\hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)^{-1} \dot{h}(S | \vartheta_h) \]

(35c)

\[ - \hat{G}^T(X | \vartheta_g)\hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)^{-1} K_1 S \]

where \( \varepsilon_0 \) is a small positive constant. The approximation of \( G^{-1}(X | \vartheta_g) \) by the regularized inverse and unavoidable reconstruction errors of the unknown functions \( F(X) \) and \( G(X) \) will occur. A more robust control term for \( \tau_{cf} \) defined as

\[ \tau_{cf} = \varepsilon_0 \hat{G}(X | \vartheta_g)\varepsilon_0 I + \hat{G}(X | \vartheta_g)\hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)^{-1} \left[ \ddot{x}_d - E - \dot{F}(X | \vartheta_f) - \dot{h}(S | \vartheta_h) - K_1 S \right] \]

(36)

and is combined with the control law Equation 35 to give

\[ \tau_c = \tau_{cf} - \tau_{cr} \]

(37)

The controller Equation 37 is the sum of two control terms: the robust control term \( \tau_{cf} \), and a modified certainty equivalent control term \( \tau_{cr} \), where

\[ \tau_{cr} = \frac{S\|S\|\left(\vartheta_f + \vartheta_G \|\tau_{cf}\| + \vartheta_h + \|\tau_0\|\right)}{g_0\|S\|^2 + \hat{\chi}} \]

(38)

where \( \hat{\chi} \) is a design time varying parameter defined below. The adaptive parameters \( \vartheta_f, \vartheta_g, \vartheta_h \) and design parameter \( \hat{\chi} \) are updated by the adaptive laws

\[ \dot{\vartheta}_f = \alpha \xi S \], \quad \dot{\vartheta}_g = \beta \xi S \tau_{cf}, \quad \dot{\vartheta}_h = \varsigma \xi S \]

(39)

\[ \dot{\hat{\chi}} = -\kappa_0 \frac{\|S\|\left(\vartheta_f + \vartheta_G \|\tau_{cf}\| + \vartheta_h + \|\tau_0\|\right)}{g_0\|S\|^2 + \hat{\chi}} \]

(40)

where \( \alpha > 0, \beta > 0, \varsigma > 0, \kappa_0 > 0, \hat{\chi}_0 > 0 \). Also, \( 2q > p \) are used to avoid singularities. We can now consider the system Equation 2, and a Type-2 fuzzy terminal sliding control law defined by Equations 35-36 with adaptive control laws given by Equations 39 and 40. This guarantees the following properties:

1. All signals that may be due to disturbances and changes in the closed-loop system are bounded.

2. The tracking error is UUB (Uniformly Ultimately Bounded), meaning that it converges to the neighbourhood of zero by appropriately choosing the design parameters.

The proof of this is obtained with \( \tau_{cr} (\hat{X}) = \tau_{cf} (\hat{X}) - \tau_{cr} (\hat{X}) \) as a control law that considers disturbances, and \( \tau_{cr} (\hat{X}) = \tau_{cf} (\hat{X}) - \tau_{cr} (\hat{X}) \) as the control law without considering disturbances. Using the mean value theorem to bound the nonlinear vector, we have

\[ \|G(\tau_{cr} - \tau_{c2})\| \leq g_m \|\hat{X} - X\| \]

(41)

where \( \hat{X} \) is the mean of \( X \) and \( g_m \) is a bounding constant number. Rewriting Equation 33, the following equation is obtained.

\[ \dot{S}(\hat{X}) \]

\[ \ddot{x}_d + E = -\ddot{x}_d + E + F(X) + G(X)(\tau_{c1}) + \tau_d \]

(42)

\[ = -\dot{x}_d + E + F(X) + G(X)(\tau_{c1} - \tau_{c2}) + G(X)(\tau_{c2}) + \tau_d \]

\[ \leq -\dot{x}_d + E + F(X) + G(X)(\tau_{c1} - \tau_{c2}) + G(X)(\tau_{c2}) + \tau_d \]

\[ + G(X)(\tau_{c2}) + \tau_d \]

Using the fact that

\[ \hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)\varepsilon_0 I + \hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)^{-1} \left[ \ddot{x}_d - E - \dot{F}(X | \vartheta_f) - \dot{h}(S | \vartheta_h) - K_1 S \right] \]

\[ = I - \varepsilon_0 \varepsilon_0 I + \hat{G}(X | \vartheta_g)\hat{G}^T(X | \vartheta_g)^{-1} \]

(43)

and

\[ \hat{G}(X | \vartheta_g)\tau_{cf} = \ddot{x}_d - E - \dot{F}(X | \vartheta_f) - \dot{h}(S | \vartheta_h) - K_1 S - \tau_0 \]

(44)

we denote the disturbance torque \( \tau_d \), denoted as \( \hat{\tau}_d \) to imply boundedness, as being bounded \( \|\hat{\tau}_d\| < \hat{\tau}_d M \) and define a constant \( k = \hat{\tau}_d M + \hat{\tau}_d \), with \( \Gamma \) as a small constant number, and Equation 42 can be written as

\[ \dot{S} \leq -\ddot{x}_d + E + F(X) + \left(G(X) - \hat{G}(X | \vartheta_g)\right)\tau_{cf} + \hat{G}(X | \vartheta_g)\tau_{cf} - G(X)\tau_{cr} + \hat{\tau}_d \]

\[ \leq -K_1 S + F(X) - \dot{F}(X | \vartheta_f) + (G(X) - \hat{G}(X | \vartheta_g))\tau_{cf} \]

\[ - G(X)\tau_{cr} - \dot{h}(S | \vartheta_h) + \hat{\tau}_d - \tau_0 \]

\[ \leq -K_1 S + \dot{F}(X | \vartheta_f) - \dot{F}(X | \vartheta_f) + \vartheta_F \]

\[ + (\hat{G}(X | \vartheta_g) - \hat{G}(X | \vartheta_g) + \vartheta_G)\tau_{cf} \]

\[ - G(X)\tau_{cr} - \dot{h}(S | \vartheta_h) + \hat{\tau}_d - \tau_0 \]

(45)
Multiplying $S^T$ to Equation 45 gives
\[ S^T \dot{S} \leq -S^T K_1 S + S^T (\hat{F}(X | \theta_T^*) - \hat{F}(X | \theta_T)) + S^T \varpi_F + S^T (\hat{G}(X | \theta_g^*) - \hat{G}(X | \theta_g) + \varpi_G) \tau_{cr} - S^T G(X) \tau_{cr} + S^T \dot{h}(S | \theta_T^*) - S^T \dot{h}(S | \theta_T) + S^T \dot{\tau}_d - S^T \tau_{cr}. \] 

Define $\Psi_{fi}, \Psi_{gij}$, and $\Psi_{hi}$ to represent the fuzzy parameter errors such that $\Psi_{fi} = \hat{\theta}_f - \theta_f$, $\Psi_{gij} = \hat{\theta}_g - \theta_g$, and $\Psi_{hi} = \hat{\theta}_h - \theta_h$. We can then rewrite Equation 46 as
\[
S^T \dot{S} \leq -S^T K_1 S + \sum_{i=1}^{\infty} \Psi_{fi} \zeta_{fi}(S) S_i + S^T \varpi + \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \Psi_{gij} \zeta_{gij}(S) S_i \tau_{fj} + \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \Psi_{hi} \zeta_{hi}(S_i) S_i + S^T \dot{\tau}_d - S^T \dot{h}(S | \theta_T) - S^T \tau_{cr} - S^T g(X) \tau_{cr}
\]

where $S^T \varpi = S^T \varpi_f + S^T \varpi_G \tau_{cr} + S^T \varpi_h$.

A Lyapunov function candidate is defined as
\[
V = \frac{1}{2} S^T S + \frac{1}{2} \sum_{i=1}^{\infty} \frac{1}{\alpha_i} \Psi_{fi}^T \Psi_{fi} + \frac{1}{2} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{\beta_{ij}} \Psi_{gij}^T \Psi_{gij} + \sum_{i=1}^{\infty} \frac{1}{\xi_i} \Psi_{hi}^T \Psi_{hi} + \frac{1}{2 \kappa_0} \chi^2 \tag{47}
\]

The time derivative of $V$ is obtained as
\[
\dot{V} = S^T \dot{S} + \sum_{i=1}^{\infty} \frac{1}{\alpha_i} \Psi_{fi}^T \dot{\Psi}_{fi} + \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{\beta_{ij}} \Psi_{gij}^T \dot{\Psi}_{gij} + \sum_{i=1}^{\infty} \frac{1}{\xi_i} \Psi_{hi}^T \dot{\Psi}_{hi} + \frac{1}{\kappa_0} \ddot{\chi} \tag{48}
\]

Substituting Equation 47 into Equation 48 and using the definition that $\dot{h}(S | \theta_T^*) = k \text{sgn}(S)$, we obtain
\[
\dot{V} \leq -S^T K_1 S + \sum_{i=1}^{\infty} \frac{1}{\alpha_i} \Psi_{fi} (\alpha_i \zeta_{fi}(S) S_i + \Psi_{fi}) + S^T \varpi + \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{\beta_{ij}} \Psi_{gij} (\beta_{ij} \zeta_{gij}(S) S_i \tau_{fj} + \Psi_{gij}) + S^T (\dot{\tau}_d - k \text{sgn}(S)) + \sum_{i=1}^{\infty} \frac{1}{\xi_i} \Psi_{hi} (\xi_i \zeta_{hi}(S_i) + \Psi_{hi}) - S^T (\tau_0) - S^T G(X) \tau_{cr} + \frac{1}{\kappa_0} \ddot{\chi} \tag{49}
\]

The time derivative of $V$ can then be written as the following.
\[
\dot{V} \leq V_0 + \dot{V}_1 + \dot{V}_2 + \dot{V}_3 \tag{50}
\]

We now rewrite $\dot{\Psi}_{fi} = \dot{\theta}_f - \theta_f$, $\dot{\Psi}_{gij} = \dot{\theta}_g - \theta_g$, and $\dot{\Psi}_{hi} = \dot{\theta}_h - \theta_h$. Since $\dot{\theta}_f$, $\dot{\theta}_g$, and $\dot{\theta}_h$ are constant numbers, $\dot{\Psi}_{fi} = -\dot{\theta}_f$, $\dot{\Psi}_{gij} = -\dot{\theta}_g$, and $\dot{\Psi}_{hi} = -\dot{\theta}_h$.

Using Equation 39 and rewriting Equation 53 we can obtain
\[
\dot{V}_2 = 0. \tag{55}
\]

Using Equation 38, the following equation can be written.
\[
S^T G(X) \tau_{cr} \geq \| S \| (\varpi_F + \varpi_G \| \tau_{cr} \| + \varpi_h + \| \tau_0 \|) - \chi \| S \| (\varpi_F + \varpi_G \| \tau_{cr} \| + \varpi_h + \| \tau_0 \|) \geq g_0 \| S \|^2 + \dot{\chi} \tag{56}
\]

and we make use of the following inequality
\[
S^T G(X) S \geq g_0 \| S \|^2 \tag{57}
\]

Combining the adaptive law Equation 40 with Equation 54, we have
\[
\dot{V}_3 \leq -S^T G(X) \tau_{cr} + \| S \| (\varpi_F + \varpi_G \| \tau_{cr} \| + \varpi_h + \| \tau_0 \|) + \frac{1}{\kappa_0} \ddot{\chi} = 0. \tag{58}
\]

From the above Equations 50, 51 and 52,
\[
\dot{V} \leq \dot{V}_0 + \dot{V}_1 \leq -\lambda_{\text{min}} (K_1) \| S \|^2 - \| S \| (\Gamma). \tag{59}
\]
Based on Equations 55, 58 and 59, Equation 50 can be written as

\[ \dot{V} \leq -\lambda_{\text{min}}(K_1) \|S\|^2 + \|S\| (q_1 d_1 - \Gamma) \leq -\lambda_{\text{min}}(K_1) \|S\|^2 \]  

(60)

and \( \lambda_{\text{min}}(K_1) \) is the minimum eigenvalue of matrix \( K_1 \). In this way, the control input and all signals involved in the control system are bounded. This provides a stable, nonlinear and model-free control system for managing rover dynamics.

5. Simulation Results

![Four Reaction Wheels Layout for Attitude Control System](image)

Figure 1. Four Reaction Wheels Layout for Attitude Control System

In this paper, the mass of the satellite is 1 kg. The altitude of the satellite is 500 km. Moment of inertia of the satellites and reaction wheel are 0.0020 and \( 10^{-5} \) kgm² respectively. The attitude motion of the 1U CubeSat with four reaction wheels is studied in this research work [Fig. 1] (\( A_i \) is a 3x4 matrix). Figures 2-5 show the relative attitude tracking and velocity responses of the satellite by type 2 AFTSMC control laws with gravity gradient torque disturbances without reaction wheels failure and faults. Figures 6-9 show the relative attitude tracking and velocity responses of the satellite using type 2 AFTSMC control laws with reaction wheels failure and faults. A severe case of failure was simulated in which not only some wheels lose partial power to randomly varying degrees, but also some wheels totally fail or are shut down purposely. Among the four wheels, wheels 3 and 4 only supply 20% and 40% respectively of the initial actuation torque at the time \( t = 10s \), after which wheel 2 has failed or is shut down. At the time \( t = 10s \), the driving torques are all changed in Figure 8. The control precision and attitude tracking are as good as in the fault-free case. Figures 10-13 show the relative attitude tracking and velocity responses of the satellite using type 1 AFTSMC control laws with reaction wheels failure and faults. The tracking results using Type 2 AFTSMC shows better results than that of using Type 1 AFTSMC.

6. FPGA based Nanosatellite Attitude Control System Simulation and HWIL Testing

In this paper, a Field Programmable Gate Array is to perform the algorithmic data processing for a 1U cubesatellite ACS. Only one similar work was done for nanosatellite attitude control.\(^{16}\) The author implemented PD and LQR controllers using Altera DSP builder on a Cyclone III FPGA. As our design focuses on efficiency and flexibility, we use a more general approach to implementing controllers on hardware. To implement the nonlinear controller, we first write the controller algorithm in C code which can be tested and validated separately, and then use High-Level Synthesis (HLS) tools to convert the C code to HDL (hardware description language). This allows low-level code to be more easily modified, and open tools to be used for synthesis. As there are several freely-available HLS tools, we have conducted a short survey and evaluation of some of the available options, given below:

PandA: PandA is an open-source (GPL-3) framework for hardware-software co-design. It is intended to include research on synthesis of hardware accelerators, parallelism extraction for embedded systems, hardware/software partitioning, performance estimation, and dynamically reconfigurable devices. Currently, it mainly covers HLS of C code.
using the Bambu compiler, which rather than using LLVM
is implemented as a plug-in to GCC. PandA is under active
development, and beta version 0.9.1 (September 2013) was
used for testing due to problems compiling version 0.9.2
(February 2014).
NISC Toolset: No-Instruction-Set-Computer (NISC) is a
computing architecture and compiler technology aimed at the
design of highly efficient custom processors and hardware ac-
celerators by applying hardware resources within a statically-
scheduled horizontal nanocoded architecture. The compiler
determines scheduling and hazard handling of operations, and
generates nanocodes for directly controlling functional u nits
of a hardware system. The NISC tools can be used to syn-
thesize Verilog RTL code and a corresponding architecture
description from C language code.
FPGAC: The FPGA C compiler is derived from the
Transmogrifier-C (TMCC) compiler developed by Dave Gal-
loway at the University of Toronto. It is a standalone compiler
that compiles a subset of the C language to netlist format
or VHDL for implementation on logic hardware. It is a
relatively small and minimal compiler implementation, and
currently does not support as many operations as larger
compiler systems, but is relatively straightforward to use and
modify as well as being easier to port.
ROCCC: The Riverside Optimizing Compiler for Config-
urable Computing by Jacquard Computing is advertised to
be a C to HDL compilation framework with focus on com-
pile time optimizations to accelerate code execution speed
in hardware. ROCCC is distributed as binaries under the
Eclipse public license and a plug-in for the Eclipse IDE. While it functions well for HDL generation as of the 2012
release and provides a good IDE framework, its licensing as
a closed-source product and limited release schedule mean
that additional functionality and OS compatibility updates
must be added by the host company. The free availability
of ROCCC makes it attractive for small projects, though.
Trident: The Trident compiler is designed for compiling
floating point algorithms written in C to produce parallel-

Figure 4. Reaction wheel Torques using Type 2 AFTSMC
do fault case

Figure 5. Reaction wheel speeds using Type 2 AFTSMC no
do fault case

Figure 6. Quaternion tracking errors using Type 2 AFTSMC
do fault case

Figure 7. Angular velocity tracking errors using Type 2
AFTSMC fault case
execution VHDL code. It builds on the earlier SeaCucumber compiler, and incorporates a four-step process. First, the LLVM compiler framework is used to produce platform-independent code from C and C++, and second an intermediate representation of logic is generated. The third step schedules and pipelines hardware operations that are then converted into VHDL in the fourth step. While it has high potential, and version 0.7.1 dates from April 2013, it is still dependent on LLVM 1.5, which is so out of date that it will not build on a recent GNU/Linux system. Consequently, some work will be required to update the code to function with a more recent version of LLVM before it is of use.

C-to-Verilog: As its name suggests, C-to-Verilog is a compiler built to convert C code into Verilog. It is a modified version of the closed source SystemRacer synthesis system by Yosi Ben Asher and was created by Nadav Rotem as the result of an academic study into HLS techniques at Haifa University. It is open source, and therefore most likely does not contain the same capabilities of SystemRacer, though this is not clarified in the documentation. C-to-Verilog uses the LLVM compiler framework to build an optimizing compiler that schedules, pipelines, and reuses code rather than performing a direct translation. The current source code is based on LLVM 2.5 and has not been updated since 2009, so as with Trident, some work is required to bring the code up to date so development is possible.

CTOV Compiler: The CTOV Compiler dates from 1995 and compiles C and C++ code to synthesizable Verilog. Some logic minimization and code optimization is done, but the compilation process is otherwise straightforward. Source code and binaries for the CTOV compiler are not currently available.

MyHDL: The MyHDL language deserves mention here, as rather than a C to HDL converter, it is a Python-based hardware description language. Python generators are used to model hardware concurrency, and the Python unit test framework can be applied to hardware designs. With some limitations, MyHDL code can be converted directly to Verilog.
The internal architecture of the FPGA based control system for the satellite ACS is shown in Fig. 14. Both type-1 and type-2 adaptive fuzzy controllers have been implemented in FPGA logic using HLS techniques. Currently, the Bambu compiler from the PandA framework is being used for conversion of controller code written in C to Verilog as it is up to date and integrates well with the Xilinx tools. To facilitate integration with other system-on-a-chip CubeSat hardware, the controller is implemented as a single logic block that can be added as a module to an FPGA-based system. A fuzzy type-1 controller block and fuzzy type-2 controller block have been synthesized to Verilog using Bambu, and both HDL blocks use the same set of logic inputs so they can be easily interchanged. The testbeds for hardware ACS development are shown in Figure 1. The controller blocks are run for simulation and validation on a Xilinx Zynq hybrid FPGA, with the nanosatellite simulation model and simulated sensor inputs running on the ARM microcontroller, and the controller implemented in the parallel FPGA hardware. In Fig. 15, hardware-in-the-loop testing is shown that will be used for the verification of the system using the Zynq hybrid FPGA. The ARM microcontroller part running a C program is used to provide command inputs and read the results from the FPGA. The controller is a logic core implemented in VHDL, which includes a system clock generator, adaptive fuzzy controller and sensor interfaces.

**FPGA based Nanosatellite Attitude Control System Hardware in the Loop Results**

Figs. 16 and 17 show the results of a 90 degree slew of the system. Fig. 18 shows the controller values used. The cubesatellite makes a 90 degree slew with a settling time of roughly 60 seconds, settling on a steady state error of 0.8 degree using adaptive fuzzy type 1 sliding mode control and adaptive fuzzy type 2 sliding mode control. Figs. 19 and 20 show the results of a 90 degree slew of the system with wheel power fault. Fig. 21 shows the controller values used. The fuzzy type 1 and type 2 adaptive sliding mode controllers deal with the fault happened during 60-180s with a settling time of roughly 1000 seconds, settling on a steady state error of 0 degree. Fuzzy type 2 controller has better fault tolerant results compared with type 1 controller.

**Figure 12.** Reaction wheel Torques using Type 1 AFTSMC fault case

**Figure 13.** Reaction wheel speeds using Type 1 AFTSMC fault case

Nonlinear satellite attitude dynamics models are considered in this paper. An FPGA based PID and nonlinear adaptive fuzzy controller are proposed for nanosatellite attitude pointing and tracking. Software implementations of fuzzy logic systems has been widely used, but the large computational load of a fuzzy logic system is not suitable for real time applications on processing-limited hardware. An adaptive fuzzy type-2 controller has better performance than type-1 controller, but it has not been used for satellite attitude control. In this research, the use of an FPGA based type-2 fuzzy logic systems has the advantages of low cost, high speed, and small size. The proposed FPGA controller will be shown to be more computationally efficient than a soft-core controller. The methodology proposed in this paper will help improve control technology for the next generation of nanosatellites.

**7. Conclusion**

The internal architecture of the FPGA based control system for the satellite ACS is shown in Fig. 14. Both type-1 and type-2 adaptive fuzzy controllers have been implemented in FPGA logic using HLS techniques. Currently, the Bambu compiler from the PandA framework is being used for conversion of controller code written in C to Verilog as it is up to date and integrates well with the Xilinx tools. To facilitate integration with other system-on-a-chip CubeSat hardware, the controller is implemented as a single logic block that can be added as a module to an FPGA-based system. A fuzzy type-1 controller block and fuzzy type-2 controller block have been synthesized to Verilog using Bambu, and both HDL blocks use the same set of logic inputs so they can be easily interchanged. The testbeds for hardware ACS development are shown in Figure 1. The controller blocks are run for simulation and validation on a Xilinx Zynq hybrid FPGA, with the nanosatellite simulation model and simulated sensor inputs running on the ARM microcontroller, and the controller implemented in the parallel FPGA hardware. In Fig. 15, hardware-in-the-loop testing is shown that will be used for the verification of the system using the Zynq hybrid FPGA. The ARM microcontroller part running a C program is used to provide command inputs and read the results from the FPGA. The controller is a logic core implemented in VHDL, which includes a system clock generator, adaptive fuzzy controller and sensor interfaces.

**FPGA based Nanosatellite Attitude Control System Hardware in the Loop Results**

Figs. 16 and 17 show the results of a 90 degree slew of the system. Fig. 18 shows the controller values used. The cubesatellite makes a 90 degree slew with a settling time of roughly 60 seconds, settling on a steady state error of 0.8 degree using adaptive fuzzy type 1 sliding mode control and adaptive fuzzy type 2 sliding mode control. Figs. 19 and 20 show the results of a 90 degree slew of the system with wheel power fault. Fig. 21 shows the controller values used. The fuzzy type 1 and type 2 adaptive sliding mode controllers deal with the fault happened during 60-180s with a settling time of roughly 1000 seconds, settling on a steady state error of 0 degree. Fuzzy type 2 controller has better fault tolerant results compared with type 1 controller.
Figure 15. FPGA based Control Design Hardware in Loop Testing

Figure 16. Nanosatellite 90 Degrees Slew without fault

Figure 17. Nanosatellite Tracking Error during 90 Degrees Slew without fault

Figure 18. Nanosatellite Control during 90 Degrees Slew without fault

Figure 19. Nanosatellite 90 Degrees Slew with wheel power fault 60-180s
Figure 20. Nanosatellite Tracking Error during 90 Degrees Slew with wheel power fault 60-180s

Figure 21. Nanosatellite Control during 90 Degrees Slew with wheel power fault 60-180s

ACKNOWLEDGMENTS

The financial support from the Natural Sciences and Engineering Research Council of Canada and Xiphos, Inc. for this project is gratefully acknowledged.

REFERENCES


**Biography**

**Junquan Li** received her Ph.D. in Aerospace Engineering from Ryerson University Canada 2012. She was a MITACS PDF fellow in Canada during 2012-2013. She is currently a Marie-Curie PDF researcher at Astronet II FP7 networks in the United Kingdom. Her research areas are attitude determination and control, nanosatellite and CubeSat engineering, formation flying, and fault tolerant control theory and applications.

**Mark Post** received his B.A.Sc. degree in electrical engineering from the University of Toronto in 2004 and his M.Sc. and Ph. D. degrees in Space Science and Engineering from York University Canada in 2008 and 2014. He currently holds the position of lecturer at the University of Strathclyde in the United Kingdom. His research includes mechatronic and embedded design of reliable and efficient robotic space systems and intelligent algorithms for autonomous control for spacecraft and planetary rovers.

**Regina Lee** received her B.A.Sc in 1994 and M.A.Sc in 1995 from the University of Toronto. She received her Ph.D. from the University of Toronto in 2000. She is currently an associate professor and chair of the department of Earth and Space Science Engineering at York University Canada. Prof. Lee’s research interests center on nanosatellite technology development, micro propulsion system design, MEMS based attitude sensor, actuator design and algorithms and FPGA subsystem development.