SSC09-VII-10

# Simulation-Based Testing of Embedded Attitude Control Algorithms of an FPGA based Micro Satellite

Muhammad Yasir
Institute of Space Systems
Pfaffenwaldring 31, 70569 Stuttgart, Germany; +49-711-685 69625
yasir@irs.uni-stuttgart.de

Toshinori Kuwahara, Claas Ziemke, Michael Fritz, Prof. Hans-Peter Roeser
Institute of Space Systems
Pffafenwaldring 31, 70569 Stuttgart, Germany; +49-711-685 62375
roeser@irs.uni-stuttgart.de

## ABSTRACT

This paper focuses on the issues regarding the software based testing approach which mainly utilizes the software models for the satellite environment and the satellite itself. This approach facilitates the use of real on-board software in a virtual satellite environment and hence gives an excellent means of system design qualification and performance verification. Use of FPGA as an on-board computer leads to the requirement of implementing the attitude control algorithms in the hardware which emphasizes the need of its testing in the loop with the simulator. Two different type of simulation environments are used to increase the credibility of the results.

## NOMENCLATURE

| | |
|---|---|
| FLP | Flying Laptop |
| ACS | Attitude Control System |
| MGM | Magnetometer |
| SuS | Sun Sensor |
| GPS | Global Positioning System |
| STR | Star Tracker |
| FOG | Fiber Optic Gyro |
| MGT | Magnetic Torquers |
| RW | Reaction Wheels |
| FPGA | Field Programmable Gate Array |
| OBC | On-board Computer |
| I/O | Input and Output |
| $I^2C$ | Inter-Integrated Circuit |
| IBIS | Integrated Bus for Intelligent Sensors |
| CPN | Central Processing Node |
| CDV | Command Decoder and Voter |
| SHIP | Satellite Hardware Interface Protocol |
| PAL | Platform Abstraction Layer |
| PSL | Platform Support Library |

## INTRODUCTION

The micro satellite Flying laptop will be the first satellite within the “Stuttgart Small Satellite Program” in which the Institute of Space Systems at Universitaet Stuttgart is developing several small satellites. The aim of this program is to launch a small spacecraft to the moon (Lunar Mission BW1)[1]. The Flying laptop is three axis stabilized satellite having a mass of about 120 kg and it is planned to placed in a sun synchronous orbit. The mission objectives to be achieved are demonstration of new technologies as well as to perform several scientific observations.[2]

The use of FPGA based on-board computer (OBC) is one of the new technologies to be demonstrated by the FLP. The FPGA based OBC is selected on the basis of its high performance, speed and their architecture which offers massive parallelism to the area of algorithm design. Software to hardware compilers have made it possible to directly generate the logical configuration of FPGA gates which makes FPGA capable of combining the flexibility of software with the high performance of hardware.

Using an FPGA for the ACS of FLP implies that the control and navigation algorithms will be implemented as a hardware instead of running as a software. This is a different approach for designing a system which has to demonstrate programmability of a software while running as a dedicated hardware circuit. This design provides extensive parallelism and give very accurate timing by allowing precise synchronization between parallel routines. The use of highly parallel reconfigurable FPGA on one hand provides very fast execution with high performance but on the other hand

limited hardware resources put lots of constraints in using computationally exhaustive algorithms. Not only the use of filtering techniques is minimized in the designing of ACS algorithms but also the use of fixed point instead of floating point reduces the accuracy of the attitude control system

Use of software in the loop testings with the embedded ACS software is always of great importance in designing and qualification of the ACS algorithms but its importance is even further highlighted due to use of processor-less on-board computer. This paper presents the simulation-based testing of embedded ACS algorithms. Some of the issues regarding implementation of these algorithms in the hardware are also mentioned. The testing of the algorithms is carried out with the help of two different simulation environments. A brief description of these environments and the testing interfaces are also provided. Finally the results of the test simulation are presented to verify the performance of the attitude control system of FLP.

## ATTITUDE CONTROL SYSTEM OF FLP

The attitude control system for the FLP is a three axis stabilized system. The overall structure and design of the attitude control system is mainly dictated by the performance requirements[3] incurred by various scientific experiments to be carried out by the FLP and the general constraints arising from its need to work under different mission phases.

### *ACS System level architecture*

Five different types of sensors and two different kinds of actuators are selected to achieve agile and accurate control and to meet the requirements laid down in[3]. Table 1 shows the list of the ACS components.

**Table 1: List of ACS Components**

| ACS Components | Quantity |
|---|---|
| 3-Axis Magnetometer | 2 |
| Sun Sensor Unit (2 solar cells) | 8 |
| GPS | 3 |
| FOGs | 4 |
| Star Tracker Camera Head Unit | 2 |
| Magnetic Torquers | 3 |
| Reaction Wheels | 4 |

The sensors and actuators are connected to the central FPGA OBC in a star type configuration which provides parallel and independent processing of each hardware interface. The design of ACS hardware is single failure tolerant. The selected ACS hardware is also a part of technology demonstration experiments which has to be carried out by the FLP. ACS hardware is shown in the Figure 1. Further details regarding the ACS hardware are presented in[4].

### *ACS control Modes*

Six different modes are defined for the ACS to fulfill the operational requirements as shown in the Figure 2.


**Figure 1: ACS Hardware**

**Detumble Mode:** This mode will be used to reduce the angular rates after the separation of the satellite from the launcher, also in the case of accidental increase in the body rates during normal operation. The satellite will automatically switch to the detumble mode. During this mode the satellite is controlled with the B-dot law using only magnetometers and magnetic torquers.

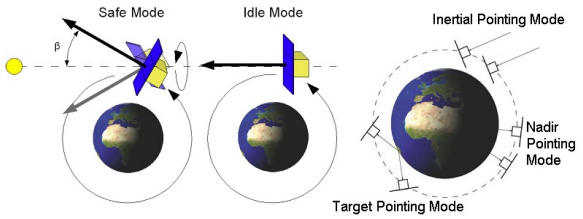



**Figure 2: ACS Operational Modes**

**Safe Mode:** The safe mode for the FLP is defined as the situation when the satellite is able to continue its normal operation with reduced functionality on the occurrence of serious anomaly in the software or the hardware of the satellite, which cannot be automatically processed and corrected on-board. This mode is used to ensure the satellites vital functions (e.g. on-board power) while in the meantime, the failure is being analyzed on ground. In this mode the satellite is spin-stabilized around one of its principle inertia axis keeping the solar panels oriented towards the sun. The satellite uses only two types of sensors (i.e. Magnetometers and sun sensors) and one type of actuators (i.e. magnetic torquer). The

selection of the sensors and actuators is based upon the reliability instead of performance; therefore only coarse attitude information is obtained in this mode.

**Idle Mode:** This is a house keeping mode and the FLP will be in the idle mode most of the time during the normal operation when it is waiting for a new command. In this mode the solar panels will be aligned to the Sun for the optimal energy supply. Also in this mode no fine attitude pointing is required. In principle all the attitude sensors are available in this mode, so the sun sensors and star trackers are used to determine the attitude and fiber optic gyros are used to determine the body rates. Reaction wheels are used as actuators in this mode.

**Inertial pointing mode:** This is one of the pointing mode which will be used to carry out scientific experiments as well as for calibration. The attitude of the satellite is fixed in inertial frame and cameras are pointed towards the fixed target relative to the fixed stars. The attitude information in this mode is acquired by using star trackers and fiber optic gyros while actuation is provided by the reaction wheels.

**Nadir pointing mode:** This is an Earth observation mode. In this mode the payload cameras are oriented towards the center of the Earth. The satellite conducts one rotation per orbit around its y-axis. The nadir coordinate system is derived from the GPS output. Star trackers and fiber optic gyros are used to determine attitude and reaction wheels are used for the actuation in this mode. For additional observation it is also possible to extend this mode to the off-nadir observations.

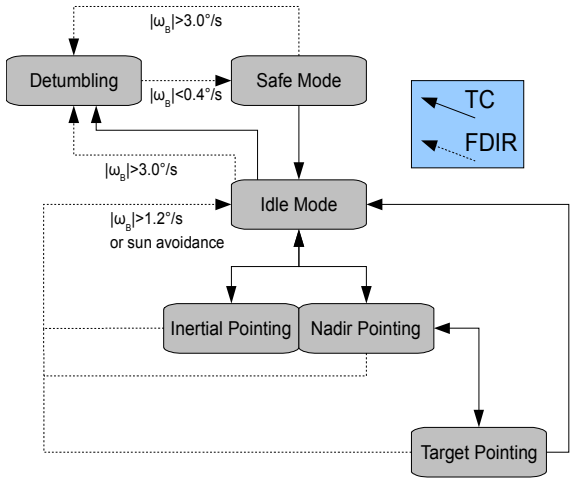



**Figure 3: Mode Switch Logic**

**Target pointing mode:** This is another earth observation mode. In this mode the payload cameras are pointed towards a coordinate on the Earth surface during a flyover. This mode will be utilized for the BRDF measurements. The absolute pointing accuracy required in this mode is equal to 150 arc seconds. This mode also utilizes GPS, star trackers and fiber optic gyros as sensors and reaction wheels as actuators.

Figure 3 shows the mode switching. Mode switching could be triggered either by telecommand (TC) or by FDIR upon any serious anomaly in the normal functioning of the satellite.

## OBC HARDWARE

The FLP is the testbed for an On-Board Computer (OBC) with a reconfigurable, redundant and self-controlling high computational ability which is based on FPGA technologies. A combinational use of different types of FPGAs are applied for the design of the OBC. The main computing element of the OBC is the multi-module redundant SRAM-FPGAs Nodes monitored by a Voter based on SEU effect tolerant FPGA. The selected SRAM-FPGA which shall be implemented into the Nodes are Virtex II-Pro, XC2VP50 of the Xilinx. This is partly because the chip was the state-of-the-art SRAM-FPGA at the time of design fix of the OBC and also partly because that the chip offers additional processor core, named PowerPC inside the chip, which can be used for normal floating point calculation with an operation system. The selected SEE tolerant FPGA for the voter of the OBC is the state-of-the-art RTA3P Flash-FPGA of the Actel. Using these FPGAs, the redundancy degree of the Node System is decided as four, for incorporating uncertainty of space radiation model and the case of failure of one of the four Nodes.

The merits of this approach are:

- Extended lifetime of the OBC in the event of a permanent error in a node hardware
- short outage times (higher availability) in the event of SEU induced errors
- redundant computations allow for voting resulting in higher reliability

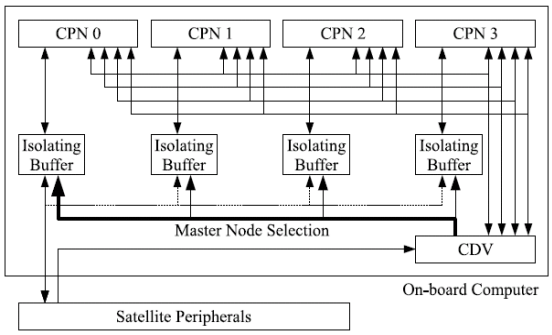



**Figure 4: Schematics of OBC Configuration**

In order to actually realize this concept, some design aspects shall be taken into account. First of all, all of the redundant nodes shall be connected to the relevant

satellite peripherals in parallel via isolating buffers so that each single CPN can execute full functions and control tasks of the OBC. Secondly, the nodes shall be physically identical. Thirdly, a failure in one node shall not be propagate through the system.

The schematics of the OBC of the Flying Laptop is illustrated Figure 4 in and the hardware configuration is illustrated in Figure 5.

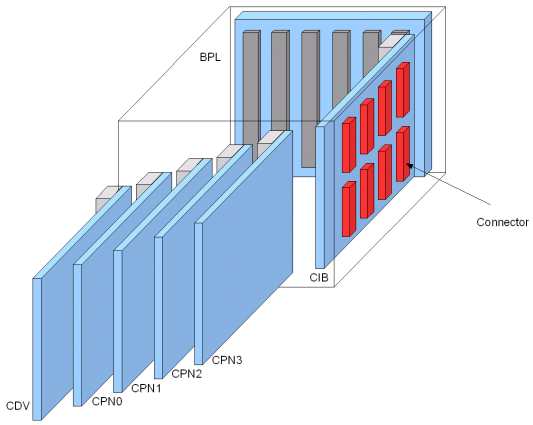



**Figure 5: OBC Hardware Configuration**

The CDV selects a master node from the four CPNs. Because this design is based on the assumption that each of the four CPNs performs the exact same tasks at the same time, all input/output signals, in total more than 200 lines, from peripheral electronics are connected to all CPNs in parallel. Due to the demands of peripheral electronics the interfaces of RS422, LVDM, M-LVDS, TTL and $I^2C$ are implemented.

## IMPLEMENTATION OF ACS ALGORITHMS

DK design Suite[5] based on Handel-C is used to map the hardware of FPGA and place and route tool from the FPGA manufacturer Xilinx are utilized for the hardware configuration. Handel-C is a programming language designed to enable the compilation of programs into synchronous hardware. Handel-C is not a hardware description language though; rather it is a programming language aimed at compiling high level algorithms directly into gate level hardware. Explicit parallelism, well defined timing, high level alternative, behavioral compilation and ANSI-C based approach are some features which makes Handel-C reliable for programming the FPGA's more efficiently.

### *ACS algorithms implementation*

ACS algorithms are developed in Matlab/Simulink environment as it was easy to analyze and modify the algorithms. In the next step these algorithms were manually ported into Handel-C. In this step the floating point design used initially in the Matlab/Simulink environment was converted into fixed point design to optimize the hardware. Lot of iterations have been done for the performance verification using DK simulation environment in this step. DK design suite then generates the logical netlists of the target hardware which is after then used to program FPGA with the help of manufacturer's *place and route* tool. FPGA evaluation boards like RC10 and RC240 manufactured by Agility[6] are used for the SIL (Software in the loop testing) to evaluate the performance and accuracy of these algorithms. The whole design and testing procedure is illustrated in the Figure 6.

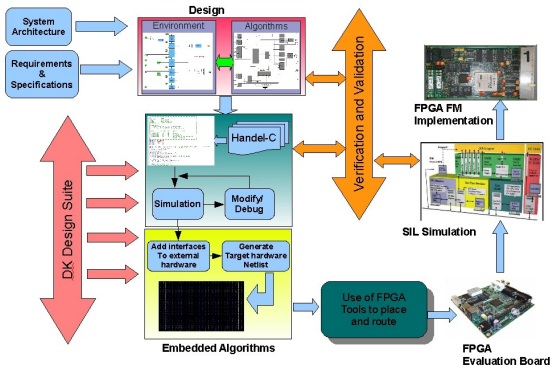



**Figure 6: ACS Algorithms Development and Testing Procedure**

The implementation of a FPGA design in Handel-C consists of three layers. The uppermost layer is the user application layer, where the high level functionality and algorithms such as ACS algorithms are implemented. This layer is independent of the FPGA hardware board. The most important requirement of this simulation and verification environment is that the application level control algorithms remain the same in flight and verification configuration so that the verified control algorithm can be directly ported into the real flight configuration.

The FPGA hardware specific functionality is included in the remaining two layers, the Platform Abstraction Layer (PAL) and the Platform Support Library (PSL). The PAL can consists of multiple Application Programming Interfaces (APIs) where each contains the functionality of specific resource. The I/O functionality of a board is encapsulated in a standard PAL API. It provides function for the control of interfaces like serial, PS2 or parallel interfaces and simplifies the use and control of different types of RAM. For the FLP ACS interfaces the serial interface functionality of the Standard-PAL API is used. Further I/O functionality that is required for the ACS is contained in the Satellite

Hardware Interface Protocol (SHIP). In the SHIP API the $I^2C$ bus and the IBIS bus are implemented. The $I^2C$ bus is used for the magnetic torquers and for the sun sensors. The IBIS bus controls the communication with the fiber optic gyros. Also included in the SHIP API are the PAL APIs for the ACS devices. They provide the functions that run, enable and disable the ACS devices and offer access to the sensor data and control values of actuators.

The PSL contains the settings for the specific FPGA board. The number of a specific type interfaces like serial interface is defined and also input and output pins are assigned to the corresponding hardware sensors and actuators.

The structure of this layered design is displayed in Figure 7.

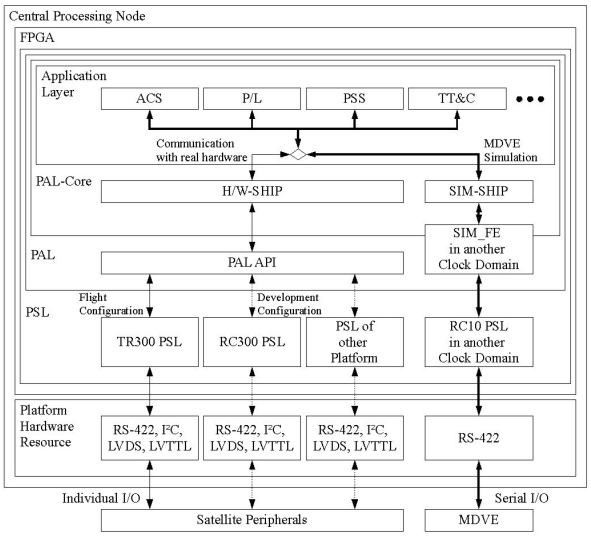



**Figure 7: Layered Structure of the algorithm implementation**

### *ACS Cycle*

To sensors and actuators that do not have an internal clock (RW, MGT, MGM, SuS, FOG), data requests or settings are sent with a predefined frequency. The timing is set inside the ACS command cycle. In ACS command cycle five cycles for the five device types are running in parallel.

For all devices a certain offset is set in order to receive all measurements synchronized to each other and compensate the delay caused by the communication time. For example the fiber optic gyros and the reaction wheels work with the same sampling frequency of 10 Hz. The communication with the reaction wheels however, runs through a serial interface with 9600 baud and uses ASCII characters for the data transmission. This takes longer than the communication with the fiber optic gyros, where the data is transmitted in binary and with a clock cycle of 2 MHz. So the communication with the reaction wheels must be started earlier than the communication with the gyros to be on time with the sync impulse.

Figure 8 shows what a configured ACS command could look like. The bars mark the communication times with the sensors.

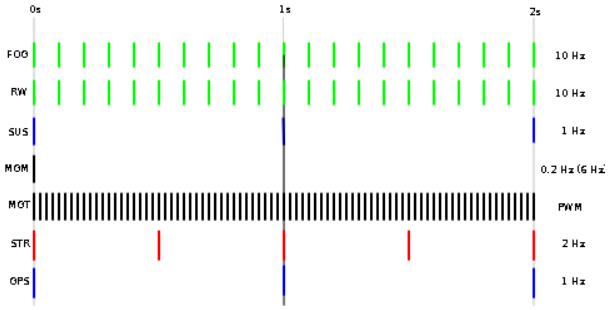



**Figure 8: ACS control cycle loop**

## MATLAB SIMULATION ENVIRONMENT.

The simulation environment was developed using Matlab/Simulink. The modular architecture of the simulation and testing environments is shown in the Figure 9

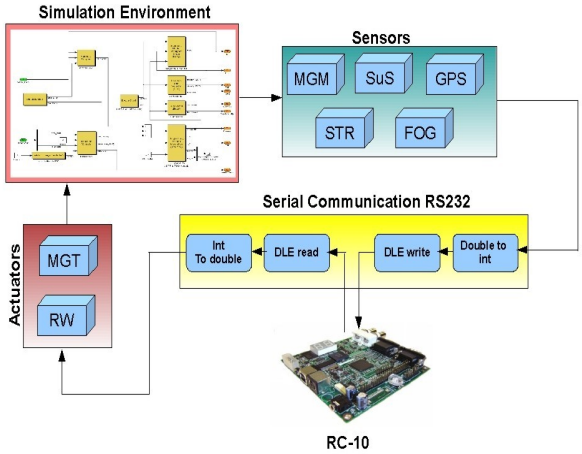



**Figure 9: Top level diagram of testing environment**

The hardware models of sensors and actuators are when combined with the simulation environment, they form the Space environment which simulated the physical orbit environment.

Several key blocks inside the simulation environment are very kindly provided by EADS Astrium to support this project, hence they are extensively used, tested and documented[7]. The blocks inside the simulation environment are processed with double precision and

they are continuous. The sensor block quantizes these values send to the serial communication block in a same format in which sensors will communicate with the OBC.

The data is then converted using DLE protocol for data package transfer over a serial interface RS232. A DLE package starts with a header byte of 0x02. The following two bytes specify the length of the data bytes and are then followed by that data. The package is terminated by a one-byte checksum and the end of transmission byte 0x03. To avoid confusion, within the package 0x02 and 0x03 must not appear and are replaced by the sequence 0x10 0x42 and 0x10 0x43 respectively. Also 0x0D and 0x10 are expanded to 0x10 0x4D and 0x10 0x10 respectively.

The ACS algorithms work as a dedicated hardware circuits on a evaluation board which calculates the required output which is again transmitted using RS232 serial interface. The output is again collected using DLE protocol of serial data package transfer. This output is sent to the actuator blocks which simulates the actuators dynamics to implement the ACS commands on the simulation environment.

## MODEL BASED DEVELOPMENT AND VERIFICATION ENVIRONMENT (MDVE)

MDVE simulation environment is a infrastructure allowing spacecraft models ranging from early pure virtual simulation via hybrid test benches up to FlatSat configuration. This largely minimizes cost for hardware engineering models, and in parallel provides risk mitigation through stepwise verification of on-board software, on-board hardware, flight procedures etc. First by utilization of simulation testbenches and later by hardware-in-the-loop configurations. A more detailed explanation of MDVE technology and its configurations can be found in Eickhoff et al.[8,9]

MDVE will be used in various project phases for a number of design, development and verification tasks in different working environments. The most important MDVE standard configurations are:

1. Development SVF (DEV-SVF)
2. Software Verification Facility (SVF)
3. System Testbed (STB)
4. Extended Real-Time Testbed (FlatSat)
5. Spacecraft simulator for the mission control center.

The core element of the MDVE is an On-board Computer Simulator (OBC-Simulator) and a Real-Time Simulator (RTS) modeling the remaining equipment units of the spacecraft plus space environment conditions. In dedicated simulation setups both of these simulators (synchronized to each other) are commanded via a control console – in most cases a Core EGSE. The functional behavior of the satellite system and all interactions of the on-board equipments are modeled on the basis of the system and equipment specifications.
A major benefit of model-based system development utilizing MDVE is the possibility for early simulated satellite mission operations, using real on-board software in the virtual satellite. By the use of the model based concept each flight hardware unit can be realized by an equivalent software model prior to the availability of the real hardware. This represents an outstanding support to system design qualification and performance verification.

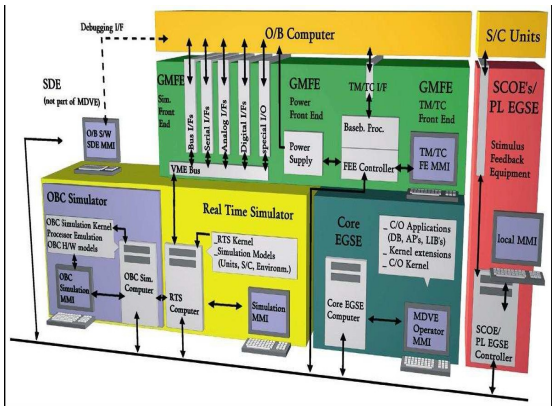



**Figure 10: Building blocks and interfaces of the MDVE construction**

## RESULTS

***Detumble mode:***

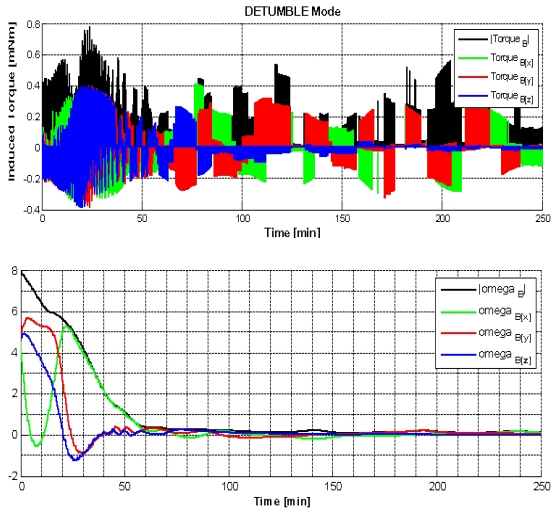



**Figure 11: Detumble mode (Matlab Simulator)**

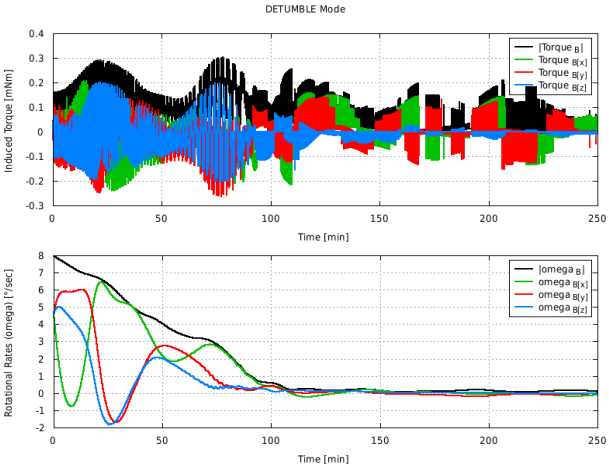
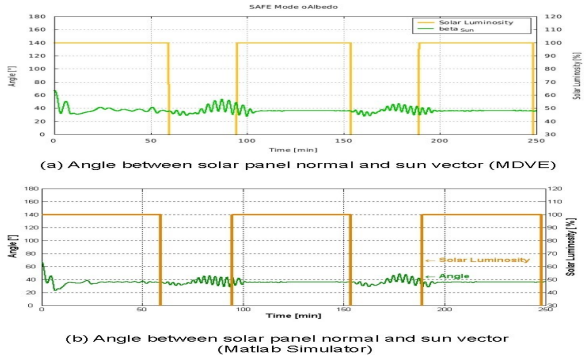



**Figure 12: Detumble mode - (MDVE)**

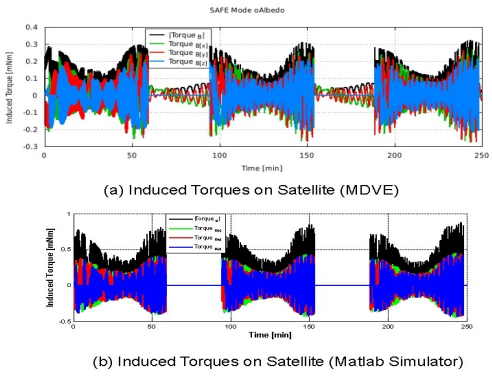
***Safe mode:***




(a) Angle between solar panel normal and sun vector (MDVE)

(b) Angle between solar panel normal and sun vector (Matlab Simulator)

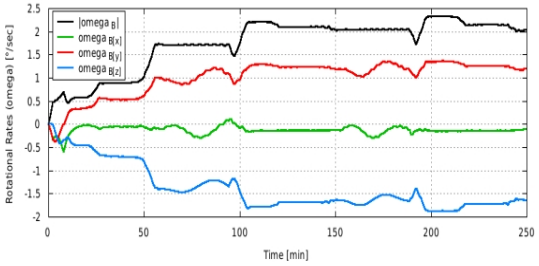
**Figure 13: Safe mode - angle between sun vector and solar panel normal**



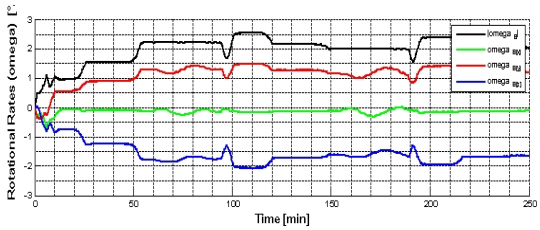

(a) Induced Torques on Satellite (MDVE)

(b) Induced Torques on Satellite (Matlab Simulator)

**Figure 14: Safe mode - induced torques on satellite body**




(a) Rotational Rates (MDVE)




(b) Rotational Rates (Matlab Simulator)

**Figure 15: Safe mode - Rotational rates**

***Idle mode***

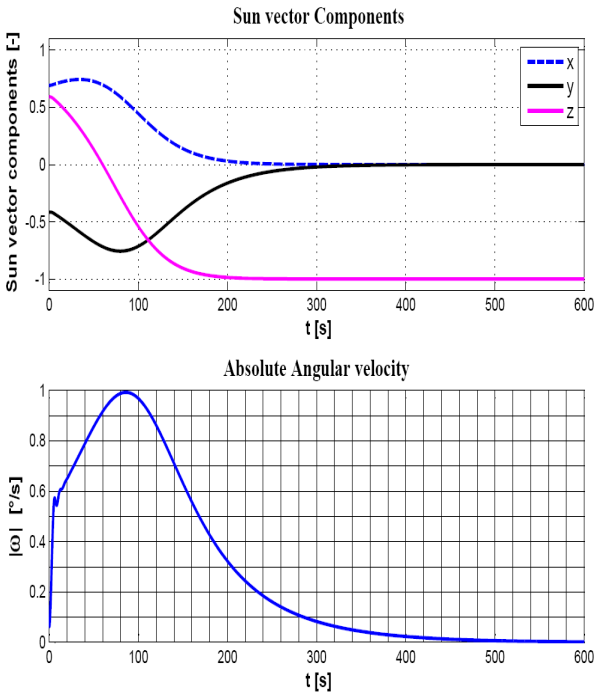



**Figure 16: Idle mode - sun vector components and angular rate (Matlab Simulator)**

***Inertial pointing mode***

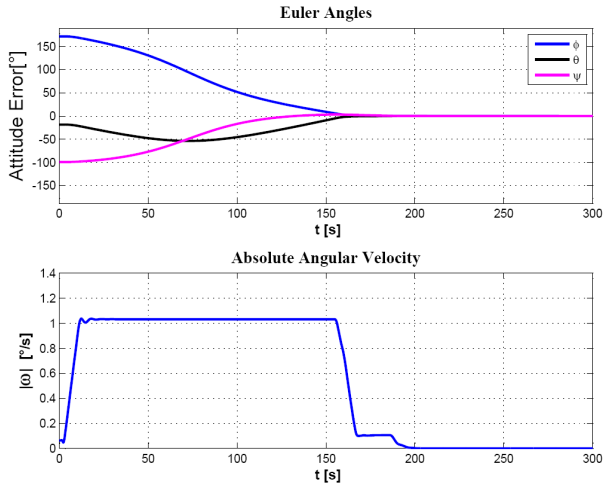



**Figure 17: Inertial pointing mode - attitude error and angular rate (Matlab Simulator)**

## CONCLUSION

In this paper the entire ACS algorithms are implemented in a FPGA-based OBC. The implementation of the interface between the simulator and OBC is described. The conducted work shows that building an ACS entirely in hardware is possible. The performance of embedded ACS algorithms is verified through the simulation in the loop tests. The use of two different kinds of simulation environments further increase the credibility of these tests. The results of MDVE are compared with the results of the Matlab Simulator. Both results are consistent and they also verify the developed simulation environment.

The established simulation environment could further be utilized for the development and verification of the on-board algorithms of the FPGA-based OBC as well as this could be further used in the Hardware-in-the-loop tests by using the satellite hardware components.

***References***


1.Laufer, R. and Röser, H.-P., "Lunar Mission BW1 – An Academic Low-cost Small Lunar Exploration Satellite", International Astronautical Federation, paper IAC-07-A3.1.A.03, Sept. 2007.

2.Schoenermark, M.v., Röser, H.-P., Huber, F., Grillmayer, G., Lengowski, M., Walz, S., Falke,A., "Earth Observation with the Flying Laptop-asmall satellite of the University of Stuttgart", St. Petersburg: 31st International Symposium on Remote Sensing of Environment, June 20-24, 2005.

3. FLP Team, "FLP - Mission and System Design Overview", FLP-RP-MMMSS-NNN-IRS (Internal Document), Apr. 2009.

4.Waidmann, M., Grillmayer, G., Wolter, V., "Use of new developments of attitude control sensors for the micro-satellite Flying Laptop", 57th IAC, Valencia, Spain, 2-6 Oct. 2006.

5."http://agilityds.com/support/download_dk.aspx", Aigility design solutions, last visited June, 2009.

6."http://www.agilityds.com/products/c_based_products/rc_computing_platform/rc240/default.aspx", Agility design solutions, last visited June, 2009.

7.Moutaux, A., Grillmayer, G., Hirth, M., Yasir, M., "ACS Matlab Model Description", Tech.Rep. FLP-TN-00000-005-IRS, (Internal Document), May 2007.

8.Eickhoff, J., Hendricks, R. and Flemming, J., "Model-based Development and Verification Environment", International Astronautical Federation, Paper IAC-03-U.3.07, Bremen, germany, Sept. 2003.

9.Eickhoff, J., Falke, A., Röser, H.-P., "model-based design and verification – State of the art from falileo constellation down to small university satellites", Acta Astronautica, 61, pp. 383-390, 2007