Increasing computing performance of ADCS subsystems in small satellites for earth observation

Johan Carvajal-Godínez, Morteza Haghayegh, Allan Granados, Jaan Viru and Jian Guo
Space Engineering Department
Faculty of Aerospace Engineering
Delft University of Technology
Outline

• Introduction
• ADCS challenges
• “Y model” approach for ADCS
• DeIFFFi ADCS modeling and simulation
• OBC Hardware selection
• ADCS software integration
• Conclusions
What does Earth Observation and Formation Flying have in common?
Challenging ADCS subsystem

- Multiple attitude modes
- High pointing accuracy
- Precise three axis control algorithms
- High resolution data types from sensors
- Onboard sensor calibration
- Fault detection and correction
- Onboard functions for autonomous operation

More complex software → improving OBC capabilities for ADCS
DelFFi Mission Statement

“The DelFFi mission shall demonstrate **autonomous formation flying** and provide enhanced scientific return within QB50 from 2016 onwards, by utilizing **two identical triple-unit Cubesats** of TU Delft which further advance the Delfi-n3Xt platform.”

**Source:** SSE TU Delft
DelFFi ADCS software development

"Y approach"
DelFFi ADCS software development

ADCS Model

ADCS Model Development

ADCS simulation and profiling

OBC platform

OBC hardware selection

Driver and service integration

converging phase

ADCS SW Integration

ADCS software architecture selection

Intergration phase

ADCS software integration

Deployment to OBC for verification

TU Delft
DeIFFi ADCS simulation model

ADCS Model

- ADCS Model Development
- ADCS simulation and profiling

- OBC hardware selection
- Driver and service integration

- ADCS software architecture selection

- ADCS software integration

- Deployment to OBC for verification

Phases:
- Converging phase
- Integration phase
- Verification phase
# DelFFi ADCS input requirements

<table>
<thead>
<tr>
<th>ADCS Modes</th>
<th>Threshold</th>
<th>Control Objective</th>
<th>Sensor</th>
<th>Actuator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detumbling</td>
<td>&gt;1°/s</td>
<td>1°/s</td>
<td>Magnetometer</td>
<td>Magnetorquer</td>
</tr>
<tr>
<td>Nadir pointing</td>
<td>&lt;1°/s</td>
<td>&lt;1°/s; &lt;10°</td>
<td>Magnetometer, Sun Sensors</td>
<td>Magnetorquer</td>
</tr>
<tr>
<td>Sun Pointing</td>
<td>Command</td>
<td>&lt;1°/s; &lt;10°</td>
<td>Magnetometer, Sun Sensors</td>
<td>Magnetorquer</td>
</tr>
<tr>
<td>Thrust Vector Control (VP)</td>
<td>Command</td>
<td>&lt;1°/s; &lt;2°</td>
<td>Magnetometer, Sun Sensors</td>
<td>Magnetorquer, Reaction Wheel</td>
</tr>
<tr>
<td>Manuevering</td>
<td>Command</td>
<td>None</td>
<td>Magnetometer</td>
<td>Magnetorquer, Reaction Wheel</td>
</tr>
<tr>
<td>Safe</td>
<td>Command</td>
<td>None</td>
<td>Magnetometer</td>
<td>None</td>
</tr>
</tbody>
</table>
ADCS software architecture (initial)
DelFFi ADCS simulation model
ADCS model profiling

**Goal:** Identifying the most demanding blocks inside the ADCS model by measuring relative CPU time utilization during simulation, for later code acceleration with digital signal processor (DSP)

**Process Steps:**
- Implement ADCS simulation model in Simulink
- Setup up the Matlab profiler to collect model performance data
- Setup up the simulation environment for ADCS model
- Run the model profiler
- Analyze ADCS model performance data
- Select most demanding model block for code acceleration with DSP
Simulation environment setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial quaternion</td>
<td>$\begin{bmatrix} 0.378 &amp; -0.378 &amp; 0.756 &amp; 0.378 \end{bmatrix}^T$</td>
<td>[-]</td>
</tr>
<tr>
<td>Orbit type</td>
<td>Circular</td>
<td></td>
</tr>
<tr>
<td>Eccentricity</td>
<td>0</td>
<td>[-]</td>
</tr>
<tr>
<td>Inclination</td>
<td>98.6</td>
<td>degree</td>
</tr>
<tr>
<td>Right ascension of the ascending node</td>
<td>-15</td>
<td>degree</td>
</tr>
<tr>
<td>Argument of perigee</td>
<td>0</td>
<td>degree</td>
</tr>
<tr>
<td>Initial mean anomaly</td>
<td>0</td>
<td>degree</td>
</tr>
<tr>
<td>Satellite inertia</td>
<td>$\begin{bmatrix} 0.017 &amp; 0 &amp; 0 \ 0 &amp; 0.055 &amp; 0 \ 0 &amp; 0 &amp; 0.055 \end{bmatrix}$</td>
<td>kg·m²</td>
</tr>
<tr>
<td>Satellite dimensions</td>
<td>$\begin{bmatrix} 0.1 &amp; 0.34 &amp; 0.1 \end{bmatrix}^T$</td>
<td>m</td>
</tr>
<tr>
<td>Center of pressure offset from center of gravity</td>
<td>$\begin{bmatrix} 0.005 &amp; 0.005 &amp; 0.005 \end{bmatrix}^T$</td>
<td>m</td>
</tr>
<tr>
<td>Drag coefficient</td>
<td>2.2</td>
<td>[-]</td>
</tr>
<tr>
<td>Maximum magnetic torquer dipole moment</td>
<td>$\begin{bmatrix} 0.4 &amp; 0.4 &amp; 0.4 \end{bmatrix}^T$</td>
<td>A·m²</td>
</tr>
<tr>
<td>Residual magnetic dipole moment</td>
<td>$\begin{bmatrix} 0.005 &amp; 0.005 &amp; 0.005 \end{bmatrix}^T$</td>
<td>A·m²</td>
</tr>
<tr>
<td>Magnetometer bias</td>
<td>500</td>
<td>nT</td>
</tr>
<tr>
<td>Magnetometer noise standard deviation</td>
<td>170</td>
<td>nT</td>
</tr>
<tr>
<td>Sun sensor bias</td>
<td>&lt; 3</td>
<td>degree</td>
</tr>
<tr>
<td>Sun sensor noise standard deviation</td>
<td>0.4</td>
<td>degree</td>
</tr>
</tbody>
</table>
ADCS model profiling results

<table>
<thead>
<tr>
<th>Block Function</th>
<th>Number of calls</th>
<th>Percentage of relative CPU time usage during simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attitude estimation (EKF)</td>
<td>110 000</td>
<td>52%</td>
</tr>
<tr>
<td>Velocity Pointing mode</td>
<td>110 000</td>
<td>14%</td>
</tr>
<tr>
<td>Other OBC functions</td>
<td>110 000</td>
<td>34%</td>
</tr>
</tbody>
</table>

Other functions include:
- Environment initialization
- ADCS Mode determination algorithm
- Other attitude modes (Safe and De-Tumbling)
- Simulation Termination
DeLIFFi OBC selection

1. ADCS Model Development
2. ADCS simulation and profiling
3. ADCS software architecture selection
4. ADCS software integration
5. Deployment to OBC for verification

OBC platform

- OBC hardware selection
- Driver and service integration

Converging phase

Integration phase

Verification phase
OBC architecture trade off

OBC Requirements:

- Code acceleration support
- Build system support from open embedded community
- I2C, SPI and UART support
- Power efficient floating point unit performance w.r.t. FPGA
- COTS available
- Open hardware and software
Hardware Test bed

Beagleboard XM:
- COTS and open hardware/software platform
- TI DM3730 SoC (ARM processor + Digital Signal Processor)
- Build system support from embedded community (Yocto project)
- Continuous integration support (Jenkins)
ADCS software build system
Results on services and drivers support

<table>
<thead>
<tr>
<th>Image</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLO</td>
<td>48K</td>
</tr>
<tr>
<td>u-boot.img</td>
<td>369K</td>
</tr>
<tr>
<td>u-boot-spl.bin</td>
<td>47K</td>
</tr>
<tr>
<td>uImage</td>
<td>4,1M</td>
</tr>
<tr>
<td>uImage-omap3-beagle-xm.dtb</td>
<td>57K</td>
</tr>
<tr>
<td>FileSystem</td>
<td>13M</td>
</tr>
<tr>
<td>Total</td>
<td>18M</td>
</tr>
</tbody>
</table>

Operating system size with all driver support is 18 MB
DelFFi ADCS software development

ADCS Model Development

ADCS simulation and profiling

Driver and service integration

OBC hardware selection

ADCS SW Integration

ADCS software architecture selection

ADCS software integration

Deployment to OBC for verification
DeIFFi ADCS architecture (final)

Non Accelerated blocks

Flow Control and FDIR
ADCS Mode Determination
ADCS modes and control
Service and Library Layer
Boot
OS Kernel
Drivers I2C, SPI, UART
ARM Processor (GPP)

Accelerated block

Estimation Algorithm (EKF)
DSP LINK BIOS
LINK Driver
DSP (ACC)
ADCS OBC

ADCS application
C code from Simulink

Yocto Project: Build System and Integration
Texas Instrument (DM3730 SoC)
ADCS OBC
Communication inside ADCS application

Optimized buffer size for EKF requirements
Integration Results

• EKF execution speedup of 5-10 times (based on baseline performance for FFT)

• ADCS software footprint size is less than 20 MB (regular footprint is 100-200 MB)

• Memory size for data exchange between ARM-DSP was reduced to 64MB (initially 128 MB)

• Fully automated operating system image generation with Yocto project + Jenkins
Conclusions

• “Y approach ” was introduced and implemented for DelFFi ADCS software architecture exploration

• Model profiling technique helped to identify and quantify computing demand for ADCS attitude estimation algorithm.

• Using a build system (Yocto project) and continuous integration tools improved software productivity problem
Further Work

• Continue the work in the porting of Simulink model to BeagleBoard XM board

• Compare ADCS performance with results from Flight Model OBC (Benchmark)

• Continue to investigate on code acceleration in space software applications with heterogeneous onboard processors
Thank You for your attention!!!

j.carvajalgodinez@tudelft.nl