FPGA-based Reconfigurable On-board Computing Systems for Space Applications

A thesis accepted by the Faculty of Aerospace Engineering and Geodesy
of the Universität Stuttgart in partial fulfilment of the requirements
for the degree of Doctor of Engineering Sciences (Dr.-Ing.)

by

M. Eng. Toshinori Kuwahara

born in Kumamoto, Japan

main referee:	Prof. Dr. rer. nat. Hans-Peter Röser
co-referee:	Prof. Dr. Eng. Tetsuo Yasaka
Date of defence:	October 29, 2009

Institute of Space Systems
Universität Stuttgart
2010

Abstract

The purpose of the thesis is to conceptualize an application method of ground-based reconfigurable FPGA (Field Programmable Gate Array) technologies for space systems and to apply the method to the on-board computer of the small satellite *Flying Laptop* for the on-orbit demonstration. The *Flying Laptop* satellite is the first small satellite within the “Stuttgart small satellite program” in which several small satellites are developed by the Institute of Space Systems at the Universität Stuttgart. The main mission of the *Flying Laptop* is to demonstrate the space use of reconfigurable FPGAs for the “reconfigurable computing” on an central on-board computer aboard a spacecraft. Due to their radiation vulnerabilities reconfigurable FPGAs have not yet been employed in practical space applications with high reliability requirements. The *Flying Laptop* project aims to achieve the world’s first orbit demonstration of a purely FPGA-based central on-board computer.

Within this research firstly, application methods of reconfigurable FPGAs for space systems were investigated, which are not limited to small satellites but for general space systems. The investigation is based on thorough experimental data survey and analysis of radiation effects on existing FPGA devices. Main radiation effects of single event effects and total ionizing dose effects were extensively investigated. Based on the data obtained, a combinational use of SRAM-FPGAs (multi-chip redundant) and Flash-FPGAs (voting element) for mitigating radiation effects was conceptualized. A mathematical system reliability analysis of repairable multi-redundant systems has been conducted and reliability models of “2-out-of-m” systems were established. The analysis illustrates that a multi-redundant system based on SRAM-FPGAs together with a Flash-FPGA based voter provides a sufficiently high reliability for Low Earth Orbit (LEO) missions against radiation effects.

After the conceptualization of application methods of reconfigurable FPGAs for the space environment, it is applied to the on-board computer of the small satellite *Flying Laptop*. *Flying Laptop* is a cubic, 3-axis stabilized satellite with the edge lengths of about 600 mm × 700 mm × 800 mm and a mass of about 120 kg, which shall be launched into sun-synchronous LEO in an altitude of around 600 km. A system architecture with four SRAM-FPGA based central processing nodes and one Flash-FPGA based voter was applied for the on-board computer of the *Flying Laptop*. This on-board computer is the central computing system aboard the satellite and shall be capable of controlling all satellite peripheral electronics. First of all, the system design of the whole satellite has been conducted within the scope of the thesis in order to allow the design of the on-board computer. Based on the established system requirements, the on-board computer of the *Flying Laptop* was designed and the breadboard model and partly the engineering model of its components are developed.

The hardware logic (control algorithm) which shall be implemented into FPGAs can be designed by means of hardware description languages. However, it is no longer software engineering but hardware engineering for generating real hardware logics inside FPGAs which are executed in parallel in real-time. The implementation of the complete satellite control algorithms into

FPGAs is the world's first achievement. By means of the breadboard model assembly, the control algorithm of the *Flying Laptop* is developed in such a way that the computational capability of FPGAs can be maximized by utilizing their internal massive parallelism. The satellite main functions are designed, developed, and implemented in FPGAs by means of the hardware description languages Handel-C and VHDL. The thesis provides development methods of the control algorithms. In addition to this, a control algorithm development facility has been established for the further design activities.

Finally, the developed control algorithms are verified in a simulation and verification environment in order to prove the validities of the above described developments. First of all, an FPGA hardware-in-the-loop real-time simulation environment has been established based on the Model-based Development and Verification Environment (MDVE). MDVE was established at the Institute of Space Systems supported by EADS Astrium. The communication interface between the MDVE and FPGAs are developed, including the required hardware components and the serialization algorithms of communication lines inside an FPGA. Using this simulation and verification environment, extensive simulations have been conducted and the design of the on-board computer, as well as the system design of the whole satellite are validated. At the end, an extended investigation has been conducted on formal verification methods of the hardware-logic in order to provide the way of strict design verifications. The formal semantics of Handel-C are extended to enable simultaneous communications between multiple hardware logic elements. This investigation cultivates the possibility of the formal verification of parallel running hardware logic elements, which will be able to realize secure implementation of reliable hardware logic into FPGAs for future space applications.

This thesis establishes the basis of principle application methods of reconfigurable FPGA technologies for "reconfigurable computing" on space systems which provides innovative solutions for high computational demands of future space applications.

Zusammenfassung

Das Ziel dieser Dissertation ist der Entwurf eines Verfahrens zum Einsatz von herkömmlichen FPGA (Field Programmable Gate Array) Technologien für die Raumfahrt und die Demonstration dieses Verfahrens als Bordrechner des Kleinsatelliten *Flying Laptop*. Der *Flying Laptop* ist der erste von mehreren Kleinsatelliten, der im Rahmen des "Stuttgarter Kleinsatellitenprogrammes," am Institut für Raumfahrtsysteme der Universität Stuttgart entwickelt wird. Der Haupteinsatz des *Flying Laptop* ist die Demonstration von rekonfigurierbaren FPGAs im Weltraum zum "Reconfigurable Computing" auf einem zentralen Bordrechner eines Raumfahrzeugs. Dies wurde wegen der Strahlungsanfälligkeit dieser Bauteile bisher nicht in Raumfahrtsystemen mit hoher Zuverlässigkeitsanforderung verwendet. Der *Flying Laptop* wird als weltweit erstes Raumfahrzeug einen reinen FPGA-basierten Bordrechner verwenden.

Im Rahmen dieser Arbeit wurden zuerst die Anwendungsverfahren der rekonfigurierbaren FPGAs für Raumfahrtsysteme erforscht, die nicht nur für Kleinsatelliten, sondern auch für allgemeine Raumfahrtsysteme gelten. Die Forschung basiert auf einer vollständigen Recherche über experimentelle Daten und Analysen der Strahlungsanfälligkeit vorhandener FPGA-Bauteile. Die Hauptauswirkungen von "Single Event Effects" und "Total Ionizing Dose" wurden ausführlich untersucht. Ausgehend von den erhaltenen Daten wurde ein kombinierter Einsatz der SRAM-FPGAs (multi-chip redundant) und Flash-FPGAs (voting element) entworfen, um die Strahlungsanfälligkeit zu reduzieren. Eine mathematische Systemzuverlässigkeitsanalyse der wiederherstellbaren Multi-Redundantsystemen wurde durchgeführt und das Zuverlässigkeitsmodell von "2-out-of-m" Systemen wurde aufgestellt. Diese Analysen stellen dar, dass ein Multi-Redundantsystem mit SRAM-FPGAs zusammen mit einem Flash-FPGA basierten Voter trotz Strahlungsanfälligkeit in einer niedrigen Erdumlaufbahn eine hinreichende Zuverlässigkeit gewährleistet.

Dieses Implementierungsverfahren der rekonfigurierbaren FPGAs für die Weltraumumgebung wurde nach dem Entwurf für den Bordrechner des Kleinsatelliten *Flying Laptop* angewendet. Der *Flying Laptop* ist ein würfelförmiger dreiachsenstabilisierter Satellit mit den Kantenlängen von etwa 600 mm × 700 mm × 800 mm und einer Masse von etwa 120 kg, der in einer sonnensynchronen niedrigen Erdumlaufbahn mit einer Höhe von etwa 600 km eingesetzt werden soll. Eine Systemarchitektur mit vier SRAM-FPGA basierten "Central Processing Nodes" und einem Flash-FPGA basierten Voter wurde für den Bordrechner des *Flying Laptop* festgelegt. Dieser Bordrechner ist das zentrale Rechensystem auf dem Satelliten, das alle Satellitenkomponenten steuern soll. Als Erstes wurde im Rahmen der Dissertation das Systemdesign des ganzen Satellitensystems durchgeführt, um das Design des Bordrechners zu ermöglichen. Basierend auf den festgestellten Systemanforderungen wurde der Bordrechner des *Flying Laptop* ausgelegt. Außerdem wurden ein Breadboard-Modell des Bordrechners und teilweise die Engineering-Modelle seiner Komponenten entwickelt.

Die Hardwarelogik (Steuerungsalgorithmen), die in den FPGAs implementiert werden sollen, können mit Hardwarebeschreibungssprachen entwickelt werden. Es handelt sich hierbei also nicht mehr um Softwareentwicklung, sondern um Hardwareentwicklung. Dabei wird die echte Hardwarelogik in den FPGAs produziert, die parallel in Echtzeit abgearbeitet wird. Die Implementierung der kompletten Steuerungsalgorithmen eines Satelliten in FPGAs ist weltweit einzigartig. Mittels des Breadboard-Modells wurden die Steuerungsalgorithmen des *Flying Laptop* so entwickelt, dass die Rechenleistungen der FPGAs unter Ausnutzung massiver Parallelität maximiert werden. Die Hauptfunktionen des Satelliten wurden mit den Hardwarebeschreibungssprachen Handel-C und VHDL ausgelegt, entwickelt, und in die FPGAs implementiert. Die Dissertation stellt die Entwicklungsstrategie für die Steuerungsalgorithmen bereit. Zusätzlich wurde die Infrastruktur für weitere Designaktivitäten aufgebaut.

Schließlich wurden die Steuerungsalgorithmen in einer Simulations- und Verifikationsumgebung geprüft. Dafür wurde zunächst eine “Hardware-in-the-Loop”-Simulationsumgebung aufgebaut, die auf einer Modell-basierten Entwicklungs- und Verifikationsumgebung (Model-based Development and Verifikation Environment (MDVE)) basiert. Die MDVE wurde mit Unterstützung von EADS Astrium am Institut für Raumfahrtsysteme zur Verfügung gestellt. Die Kommunikationsschnittstelle zwischen MDVE und FPGAs wurde entwickelt, einschließlich der benötigten Hardwarekomponenten und des Algorithmus, welcher die Kommunikationskanäle innerhalb einem FPGA in serielle Reihenfolge bringt. Mittels dieser Simulations- und Verifikationsumgebung wurden ausführliche Simulationen durchgeführt. Daraus resultierend wurde die Gültigkeit des Bordrechnerdesigns sowie das Systemdesigns für den gesamten Satelliten validiert. Schlussendlich wurde eine erweiterte Untersuchung über formale Verifikationsverfahren der Steuerungsalgorithmen durchgeführt, um eine strikte Designverifikation zu ermöglichen. Die formale Semantik von Handel-C wurde erweitert, um die gleichzeitige Kommunikation zwischen mehreren Hardwarelogikelementen zu ermöglichen. Diese Untersuchung stellt eine formale Verifikation der parallel laufenden Hardwarelogikelemente in Aussicht. Dadurch würde die verlässliche Implementierung von betriebssicherer Hardwarelogik in die FPGAs für zukünftige Raumfahrtanwendungen ermöglicht.

Diese Dissertation legt die Basis für ein Anwendungsverfahren von rekonfigurierbaren FPGA-Technologien auf Raumfahrzeugen. Dieses Verfahren stellt eine innovative Lösung für erhöhte Rechenanforderungen von zukünftigen Raumfahrtanwendungen dar.

Contents

Abstract i

Zusammenfassung iii

List of Figures x

List of Tables xiii

Acronyms xiv

Nomenclature xvii

0. Introduction 1
- 0.1. Purpose of research 1
- 0.2. State of the art 2
  - 0.2.1. FPGA and reconfigurable computing 2
  - 0.2.2. Reconfigurable FPGAs in space applications 2
  - 0.2.3. Evolution of small satellites 3
- 0.3. FPGA technology for small satellites 3
- 0.4. Scope of research 4

I. Reconfigurable FPGA technologies for space applications 5

1. Field Programmable Gate Array 6
- 1.1. Types of Field Programmable Gate Arrays 6
  - 1.1.1. State of the art FPGA product 7
- 1.2. FPGA technologies 7
  - 1.2.1. Mechanism of programmable elements 7
  - 1.2.2. Application fields of FPGAs 10
  - 1.2.3. FPGA technologies in space applications 10
- 1.3. Radiation effects on FPGAs 11
  - 1.3.1. Radiation environment in space 11
  - 1.3.2. Total ionizing dose effect 11
  - 1.3.3. Single event effects 11
- 1.4. Radiation effects analysis on SRAM- and Flash-FPGAs 13
  - 1.4.1. TID effects on SRAM-FPGAs 13
  - 1.4.2. TID effects on Flash-FPGAs 13
  - 1.4.3. SEEs on SRAM-FPGAs 14
  - 1.4.4. SEEs on Flash-FPGAs 18

1.5. Summary 18

2. Application method of reconfigurable FPGAs for space systems 19
2.1. Combinational use of different types of FPGAs 19
2.2. System reliability analysis 20
2.2.1. Basic probability theory 21
2.2.2. Reliability modeling of repairable single node 21
2.2.3. 2-out-of-3 systems modeling 23
2.2.4. 2-out-of-m systems with identical components 25
2.3. Reliability against TID effects 26
2.3.1. Reliability of voter against TID effects 27
2.3.2. Reliability of node system against TID effects 28
2.4. Node system design 29
2.4.1. Trade-off on degree of redundancy 29
2.4.2. Node system reliability after latest repair 30
2.4.3. Effect of periodic preventive renewals 31
2.5. Summary 31

II. Demonstration with the small satellite Flying Laptop 33

3. Flying Laptop satellite 34
3.1. Flying Laptop project 35
3.2. Mission description 36
3.3. Scientific payload instruments 37
3.3.1. Camera systems 37
3.3.2. Scientific communication systems 39
3.4. Attitude control system 39
3.4.1. Attitude pointing modes 40

4. System design of the Flying Laptop 42
4.1. Electrical architecture design 44
4.1.1. Electrical interconnections block diagram 44
4.1.2. Input/Output diagram 44
4.1.3. Mock-up model development 45
4.1.4. Harness cabling plan 46
4.2. System electrical architecture 47
4.3. Operational design 49
4.3.1. Operational design concept 49
4.3.2. Operational mode design 50
4.3.3. Commanding concept 50
4.3.4. Data handling 51
4.3.5. Communication structure 52
4.3.6. Contingency operation 52
4.4. Attitude control 53
4.4.1. Attitude control modes 53
4.4.2. Attitude control system state-machine 53

4.5. Power balancing design 54
4.5.1. Launch and early orbit phase scenario 55
4.5.2. Analysis of failure scenarios 57
4.5.3. Analysis of a maximum power consumption scenario 58
4.6. Summary 60

5. Design of OBC hardware 61
5.1. Requirements of OBC 61
5.2. Applied OBC design 62
5.3. System reliability 63
5.4. Functional requirements allocation 64
5.4.1. FDIR strategy 65
5.4.2. Node system configuration and interface replication 65
5.5. Design implementation of CPN 66
5.5.1. Node structure 66
5.5.2. Interface isolation 67
5.5.3. System clock 67
5.5.4. Engineering model of CPN 68
5.6. Design implementation of CDV 69
5.6.1. Flash-FPGA RTProASIC 69
5.6.2. Internal structure and interfaces 70
5.6.3. Main decoder/voter FPGA 70
5.6.4. Data handling support FPGA 74
5.6.5. Mass memory 74
5.7. Design implementation of backplane 75
5.8. Summary 76

6. Design of OBC control algorithm 77
6.1. Mechanism of SRAM-FPGA 77
6.1.1. Configurable logic block 78
6.1.2. Peripheral hardware resources 79
6.2. Mechanism of Flash-FPGA 80
6.3. Control algorithm design methods 81
6.3.1. Hardware description languages 81
6.3.2. Development environment of CPN 81
6.3.3. Development environment of CDV 81
6.4. CPN Control algorithm architecture 82
6.4.1. Vertical layered structure 82
6.4.2. Sequential and parallel processes 84
6.4.3. Decomposition into parallel processes 86
6.4.4. Multi-agent programming 87
6.4.5. FDIR structure 89
6.4.6. Decision making based on information filtering 91
6.4.7. Control algorithm architecture of Flying Laptop 93
6.4.8. State-machine design 95
6.5. CDV Control algorithm 96
6.5.1. Single event upset mitigation methodology 96

6.5.2. Single event transient filtering 97
6.6. Control algorithm development environment 99
6.7. Summary 100

III. Simulation and verification 101

7. Establishment of an FPGA hardware-in-the-loop simulation environment 102
7.1. Model based development and verification environment 102
7.1.1. State of the art 102
7.1.2. MDVE configurations 104
7.2. Development of a hardware-in-the-loop environment 104
7.2.1. Implementation concept 104
7.2.2. Extended layered structure of control algorithm 105
7.3. Development of a simulator front-end 105
7.3.1. Communication link budget 106
7.3.2. Simulator transfer frame and DLE protocol 106
7.3.3. Data transfer strategies 107
7.3.4. Clock domains 107
7.3.5. Structure of simulator front-end 108
7.4. Timing analysis 109
7.5. Developed hardware-in-the-loop simulation environment 110
7.6. Control algorithm development facility 112

8. FPGA Hardware-in-the-loop simulation results 113
8.1. Attitude control algorithm simulation 113
8.1.1. Detumbling mode simulation 113
8.1.2. Safe mode simulation 115
8.2. Launch and early orbit phase simulation 118
8.3. Power budget simulation 118
8.4. Satellite operation simulation 120
8.5. Summary 123

9. Formal verification of Handel-C program 124
9.1. Formal verification of concurrent systems 124
9.2. Scope of Handel-C semantics 125
9.2.1. Handel-C abstract syntax 126
9.3. Extended operational semantics 127
9.3.1. Operational phase 127
9.3.2. Phase transitions 127
9.4. Extended hardware compiler semantics 130
9.4.1. Hardware compilation semantics 131
9.5. Formal verification of logics in multiple clock domains 132
9.5.1. Introduction of observation clock domain 133
9.5.2. Derivation of synchronization function 133
9.6. Formal verification with UTP 134
9.7. Discussion on using signal constructs 135

9.8. Conclusion 136

IV. Conclusions and appendices 137

10. Conclusions and outlook 138

Appendices 141

A. Failure frequency of SRAM-FPGAs 142

B. Basic probability calculus 143
- B.1. Definition of distribution function and density function 143
- B.2. Distribution of the sum of two random variables 144
- B.3. Basic Laplace transformation 145
- B.4. Markov model of a repairable unit 146
- B.5. Basic Boolean functions 147

C. Flying Laptop satellite subsystem design 148
- C.1. Telemetry, Tracking & Command subsystem design 148
  - C.1.1. Link equation 148
  - C.1.2. Link budget of Flying Laptop 151
  - C.1.3. State-machine of the TT&C subsystem of Flying Laptop 151
- C.2. Power supply system deisgn 153

D. Attitude control algorithms 156
- D.1. Detumbling mode 156
- D.2. Safe mode 156
- D.3. Inertia matrices 157

E. Hardware logic design with Handel-C 159
- E.1. Types and operators 159
- E.2. Parallelism 161
- E.3. Conditional logic 162
- E.4. Arrays, RAMs and ROMs 163
- E.5. Initialization 163
- E.6. Arithmetic operation 163
- E.7. Communication between logics 164
- E.8. Pipelining 165
- E.9. Timing 165
- E.10. Prialt 166

F. Notation reference for formal verification of Handel-C code 167

Bibliography 168

List of Figures

1.1. Relation between FPGA devices and ASIC 7
1.2. CMOS SRAM cell 8
1.3. Elements of a Flash memory cell 8
1.4. Floating gate structure in Flash-FPGA 9
1.5. Interconnect elements of Antifuse-FPGA 9

2.1. Proposed on-board computer concept 20
2.2. Up and down states of a repairable unit 22
2.3. Fault tree of 2-out-of-3 system 23
2.4. 2-out-of-m systems unavailability 29
2.5. 2-out-of-m systems MTBF 29

3.1. *Flying Laptop* satellite, artist impression 34
3.2. Schematics of MICS 37
3.3. Schematics of TICS 38
3.4. Attitude control system hardware components 40
3.5. Target pointing mode 41
3.6. Nadir pointing mode 41
3.7. Inertial pointing mode 41

4.1. *Flying Laptop* satellite mechanical configuration 42
4.2. Electrical interconnection block diagram 45
4.3. Input/Output diagram (magnetic torquer unit) 45
4.4. Mock-up model of *Flying Laptop* satellite 46
4.5. Harness cabling plan 47
4.6. *Flying Laptop* system electrical architecture 48
4.7. Orbit envelop and reference orbits 49
4.8. *Flying Laptop* satellite system state-machine 51
4.9. Attitude control modes of ACS 53
4.10. State-machine of ACS 54
4.11. Solar panel deployment configuration 55
4.12. Power balance simulation (Configuration A) 56
4.13. Attitude simulation (Configuration A) 56
4.14. Power balance simulation (Configuration B_R and C) 57
4.15. Maximum power consumption scenario 59
4.16. Rotational rate and sun vector angle 59

5.1. OBC internal configuration 62
5.2. OBC mechanical structure 62
5.3. OBC internal architecture 65

5.4. Block diagram of CPN . . . 66
5.5. Signalling rates vs. distance of the isolation buffer devices . . . 67
5.6. System clock distribution . . . 67
5.7. Engineering model of CPN . . . 68
5.8. CDV internal structure and peripheral interfaces . . . 70
5.9. Hash signal interface between CDV and CPNs . . . 72
5.10. Mechanism of node synchronization . . . 73
5.11. Clock signal interface between CDV and CPNs . . . 73
5.12. Flash memory (EM) . . . 75
5.13. Flash memory (BBM) . . . 75
5.14. Backplane (EM) . . . 75
5.15. Termination resistors . . . 75
5.16. Breadboard model of the on-board computer . . . 76

6.1. Mechanism of SRAM-FPGA . . . 78
6.2. Configuration logic block . . . 78
6.3. Structure of a slice . . . 79
6.4. Mechanism of Flash-FPGA . . . 80
6.5. LUT configurations . . . 80
6.6. Layered control algorithm structure . . . 83
6.7. Decomposition of control algorithms . . . 86
6.8. Compositional multi-agent System . . . 88
6.9. Fault tree of a BRDF measurement . . . 89
6.10. Information filtering for decision making . . . 91
6.11. Architecture of parallel asynchronous reactive system . . . 92
6.12. Control algorithm architecture of the *Flying Laptop* satellite . . . 94
6.13. Simulink and Stateflow model . . . 95
6.14. Radiation effects on Flash-FPGAs . . . 96
6.15. Mechanisms of SET . . . 96
6.16. TMR design method for SEE mitigation . . . 97
6.17. Implemented TMR design into Flash-FPGA . . . 97
6.18. SET filtering mechanism . . . 98
6.19. Possible design trade-off . . . 98
6.20. Control algorithm development environment . . . 99

7.1. Model based development and verification environment . . . 103
7.2. Extended layered structure of control algorithm . . . 105
7.3. Simulator transfer frame . . . 107
7.4. Internal architecture of simulator front-end . . . 108
7.5. Measured timing of magnetometer communication . . . 109
7.6. Statistics of measured communication timing . . . 110
7.7. FPGA hardware-in-the-loop simulation environment . . . 111
7.8. Control algorithm development environment and procedures . . . 112

8.1. Rotational rate in detumbling mode . . . 114
8.2. Induced torque in detumbling mode . . . 114
8.3. Sun vector angle against solar panel normal in safe mode . . . 115
8.4. Rotational rate in safe mode . . . 116

8.5. Induced torque in safe mode 116
8.6. LEOP simulation 117
8.7. Power balance simulation 119
8.8. Normal operation scenario simulation 122

9.1. Multiple clock domains 133

B.1. Non-negative random variable 144
B.2. Markov graph of a single repairable unit 146

C.1. State-machine of TT&C subsystem of *Flying Laptop* 151
C.2. Solar panel configuration 153
C.3. Flight model of the middle solar panel 154
C.4. Configuration of solar panels and PCDU 154
C.5. Li-Ion battery cell 155

E.1. Hardware logic of variable allocation 159
E.2. Combinational logic in LUT 160
E.3. Conditional operator implementation 160
E.4. Sequential processes 161
E.5. Synchronization block of parallel logics 161
E.6. Variable swap implementation 162
E.7. Hardware implementation of “if” condition 162
E.8. Hardware implementation of “do-while” condition 163
E.9. Pipelining of design 165

List of Tables

1.1. FPGA product families from leading manufacturers 7
1.2. TID effects on SRAM-FPGA 13
1.3. Limit of TID effects on Flash-FPGA 13
1.4. Parameters of Xilinx SRAM FPGAs 15
1.5. Weibull parameter of Virtex-II Pro XC2VP50 16
1.6. Weibull parameter of Virtex-4QV XQR4VFX60 16

2.1. 2-out-of-m system $MTBF_{2ofm}$ for selected SRAM-FPGAs with $MTTR_{NU} = 5\text{s}$ 30

3.1. *Flying Laptop* mission parameters 35
3.2. Characteristics of scientific payload camera instruments 38
3.3. Characteristics of scientific payload communication instruments 39
3.4. Quantity of attitude control system components 39

4.1. *Flying Laptop* system components 43
4.2. Parameters of reference orbits 50
4.3. Mission operation sample scenario (BRDF measurement) 51
4.4. Communication channels of *Flying Laptop* 52
4.5. Attitude control system components on/off table 53
4.6. Solar panel parameters 54
4.7. High-speed communication simulation scenario 58

5.1. Characteristics of RTProASIC FPGAs 69
5.2. Housekeeping and control signals of CPN 71

7.1. SimFE uplink budget 106
7.2. SimFE downlink budget 106
7.3. Control characters of DLE protocol 107

8.1. Satellite operation scenario 121

9.1. Extended transition conditions 129

A.1. Failure frequency of Virtex-II Pro XC2VP50 142
A.2. Failure frequency of Virtex-4QV XQR4VFX60 142

C.1. System noise sources 150
C.2. Link budget of omnidirectional communications 152
C.3. Link budget of directional communications 152
C.4. Battery cell characteristics 155

Acronyms

ACS	Attitude Control System
ADC	Analog-to-Digital Converter
AIM	Adaptive Instrument Module
ALU	Arithmetic LUT
API	Application Program Interface
ASIC	Application Specific Integrated Circuit
BBM	Breadboard Model
BOL	Beginning Of Life
BPL	BackPLane
BPSK	Binary Phase-Shift Keying
BRAM	Block RAM
BRDF	Bi-directional Reflectance Distribution Function
CA	Control Algorithm
CC	Configuration Controller
CCL	Central Control Logic
CDV	Command Decoder and Voter
CIB	Connector Interface Board
CLB	Configurable Logic Block
CLK	Clock
(C)MOS	Complementary Metal-Oxide-Semiconductor
COTS	Commercial Off-The-Shelf
CPLD	Complex Programmable Logic Device
CPN	Central Processing Node
CREME96	Cosmic Ray Effects on Micro Electronics 96
CSP	Communicating Sequential Processes
(D)AI	(Distributed) Artificial Intelligence
DCM	Digital Clock Manager
DDRRAM	Double Data Rate SDRAM
(D)FF	(Data) Flip-Flop
DHS	Data Handling Support unit
DIP	Digital Image Processing
DLE	Data Link Escape
DOD	Department of Defense
DSP	Digital Signal Processing
EDIF	Electronic Design Interchange Format
(EEP)ROM	(Electrically Erasable Programmable) Read-Only Memory
EGSE	Electrical Ground Support Equipment
EM	Engineering Model
EMC	Electromagnetic Compatibility

EOL	End Of Life
EPC	Electronic Power Conditioner
FDIR	Failure Detection, Isolation, and Recovery
FDR	Failures/Divergence Refinement
FLP	Flying Laptop
FM	Flight Model
FOG	Fiber Optic Gyro
FORS	FPGA-based OBC for Reconfigurable Computing on Space Systems
FPGA	Field Programmable Gate Array
FSK	Frequency-Shift Keying
FT	Fault Tree
GG	Guard Gate
GMFE	Generic Modular Front-End
GPS	Global Positioning system
HDL	Hardware Description Language
HDLC	High-level Data Link Control
HG	High Gain
I^2C	Inter-Integrated Circuit
IBIS	Integrated Bus for Intelligent Sensors
I/O	Input/Output
IOB	Input/Outpot Block
IRS	*Institut für Raumfahrtsysteme* (Institute of Space Systems)
ISRO	Indian Space Research Organization
JTAG	Joint Test Action Group
LCM	Least Common Multiple
LEO	Low Earth Orbit
LEOP	Launch and Early Orbit Phase
LET	Linear Energy Transfer
LG	Low Gain
LSB	Least Significant Bit
LTDN	Local Time of Descending Node
LUT	Look Up Table
LVDS	Low-Voltage Differential Signaling
(LV)TTL	(Low-Voltage) Transistor-Transistor Logic
MDV	Main Decoder/Voter
MDVE	Model-based Development and Verification Environment
MGM	MaGnetoMeter
MGT	MaGnetic Torquer
MICS	Muti-spectral Imaging Camera System
MSB	Most Significant Bit
MTBF	Mean Time Between Failures
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
N/A	Not Applicable
OBC	On-board Computer
OFDM	Orthogonal Frequency-Division Multiplexing
OTP	One-Time Programmable

(O)QPSK	(Offset) Quadrature Phase-Shift Keying
PAL	Platform Abstraction Layer
PARS	Parallel Asynchronous Reactive System
PAMCAM	Panoramic Camera
PC	Personal Computer
PCDU	Power Control and Distribution Unit
PFM	Proto-Flight Model
P/L	Payload
PLD	Programmable Logic Device
PSL	Platform Support Layer
PSLV	Polar Satellite Launch vehicle
PSS	Power Supply System
RC	Reconfigurable Computing
RTB	Real-time TestBed
RTC	Real-Time Clock
RTS	Real-Time Simulator
RW	Reaction Wheel
SBC	Satellite Bus Controller
SCOE	Special Check-Out Equipment
SCOS	Satellite Control and Operation System
SEE	Single Event Effect
SEFI	Single Event Functional Interrupt
SEL	Single Event Latch-up
SET	Single Event Transient
SEU	Single Event Upset
SHIP	Satellite Hardware Interface Protocol
SimFE	Simulator Front-End
SOC	State Of Charge
SPENVIS	SPace ENVironment Information System
SRAM	Static Random Access Memory
STR	Star Tracker
SuS	Sun Sensor
SVF	Software Verification Facility
S&M	Structure & Mechanisms
TC	TeleCommand
TICS	Thermal Infrared Camera system
TID	Total Ionizing Does
TM	TeleMetry
TMR	Triple Module Redundancy
TT&C	Telemetry Tracking & Command
TWT(A)	Traveling Wave Tube (Amplifier)
USO	Ultra Stable Oscillator
UTP	Unifying Theories of Programming
VHDL	Very-high-speed integrated circuit Hardware Description Language

Nomenclature

$\mathbf{A}$	[-]	alignment matrix
A	[-]	availability
A_r	[m^2]	effective receiver antenna aperture area
$\mathbf{B}$	[T]	Earth magnetic field
B	[Hz]	band width
c	[m/s]	speed of light ($\approx 3 \times 10^8$ m/s)
C	[W]	received power
C_i	[-]	component i
D	[s]	down time
D_r	[m]	receiver antenna aperture diameter
e	[°]	pointing error
$E(\alpha)$	[-]	expectation of random quantity α
E_b	[W·s]	energy-per-bit
$EIRP$	[W]	effective isotropic radiated power
f	[Hz]	carrier signal frequency
f_α	[-]	probability density function of α
F_α	[-]	probability distribution function of α
F_{noise}	[-]	noise figure
G	[-]	antenna gain
$\mathbf{I}$	[$\mathrm{kg \cdot m}^2$]	moment of inertia
k	[J/K]	Boltzmann constant ($= 1.380 \times 10^{-23}$ J/K)
L	[s]	life time
L_a	[-]	transmission path loss
$L_{a,rain}$	[-]	rain attenuation loss
L_l	[-]	line loss
L_s	[-]	space loss
L_w	[MeV, $\mathrm{MeV \cdot cm}^2$/mg]	effective linear energy transfer value
$L_{w,0}$	[MeV, $\mathrm{MeV \cdot cm}^2$/mg]	linear energy transfer threshold parameter
$\mathbf{m}$	[$\mathrm{A \cdot m}^2$]	magnetic dipole moment
$MTBF$	[s]	mean time between failures ($= MTTF + MTTR$)
$MTTF$	[s]	mean time to failure
$MTTR$	[s]	mean time to repair
n	[-]	total number of items in population
N	[W]	noise power
N_0	[W/Hz]	noise-density
p_i	[-]	criticality probability of component i
P	[-]	probability
P_t	[W]	transmitter power
R	[-]	reliability

R_d	[bps]	data rate
$\mathbf{s}$	[-]	unit sun vector
s	[-]	exponent of Weibull distribution function
S	[m]	communication path length
t	[s]	time
$\mathbf{T}$	[N·m]	torque
T	[s]	periodic renewal interval
T_{ant}	[K]	antenna noise temperature
T_r	[K]	receiver noise temperature
T_s	[K]	system noise temperature
TID	[krad]	total ionizing dose
u		random variable
U	[-]	unavailability
W	[s]	work time
W_f	[W/m^2]	power flux density
W_w	[-]	width parameter of Weibull distribution function
X	[-]	Boolean indicator variable
z_i		random variable sample
Z		random variable
β	[°]	elevation angle
ζ	[-]	population mean
η	[-]	antenna efficiency
θ	[°]	antenna half-power beam width angle
λ	[Hz]	failure rate
λ_w	[m]	wave length
μ	[Hz]	repair rate
ν	[Hz]	mean failure frequency of a repairable unit
σ	[-]	population standard deviation
σ_{sat}	[-]	limiting cross-section of Weibull distribution function
φ	[-]	Boolean function
ω	[°/s]	angular velocity

0. Introduction

0.1. Purpose of research

The purpose of the thesis is to investigate and conceptualize application methods of reconfigurable FPGA technologies for space systems, and to apply them to the actual On-board Computer (OBC) of the demonstration small satellite *Flying Laptop*, the first satellite within the Stuttgart small satellite program, in which several small satellites are developed by the Institute of Space Systems (Institut für Raumfahrtsysteme: IRS) of the Universität Stuttgart. Recently, applying ground-based reconfigurable FPGA technologies for space systems is of great interest, which promises an innovative leap of the capability of their computing systems. Together with the capability of reconfigurable computing, the extremely high throughputs and flexibilities of FPGAs due to their internal parallel processing mechanisms are the ideal solutions for future space applications, e.g., for space robotics and/or intelligent space systems. The purpose of the thesis can be observed from the following two aspects:

- Application of ground-based reconfigurable FPGAs for space systems.
- Demonstration of an FPGA-based OBC with the small satellite *Flying Laptop*, which is the central and the only OBC aboard the spacecraft.

On the one hand, the small satellite *Flying Laptop* provides the best platform for a quick and low-cost demonstration, on the other hand, it will unveil the attractive features of reconfigurable FPGAs for small satellite applications. Indeed, reconfigurable FPGAs have revealed various attractive features for small satellites compared to traditional microprocessors, namely in terms of their higher computational capability, dense interface implementation capability, less power consumption, low cost, and small hardware size. This will give an answer to the recent high demands on computational capability of small satellites.

The first challenge to be solved is the radiation susceptibility of reconfigurable FPGAs. Due to their susceptibility against radiation, the above concept has never been demonstrated in the world before. In this sense, the thesis is the first literature on this topic. The main hurdles to the accomplishment of the purpose of the thesis are:

- Establishment of radiation mitigation methods for reconfigurable FPGAs based on a comprehensive research and analysis on radiation effects.
- Integrated system design of the small satellite *Flying Laptop* in order to allow implementation of the FPGA-based central OBC.
- Design of the hardware logic inside the FPGAs, which shall replace the software functions of traditional satellites and shall conduct the whole control functions of the *Flying Laptop*.
- Verification of the functionality of the developed hardware logic, which ensures a valid implementation of algorithms and the satellite system design.

The thesis aims to apply a combinational use of different types of reconfigurable FPGA technologies (SRAM-based and Flash-based FPGAs) for the radiation mitigation with a sufficient redundancy concept. It is also the world's first trial to implement whole satellite control functions into FPGAs. According to the requirement of the *Flying Laptop* satellite, the Low Earth Orbits (LEO) are selected for the target orbit of the demonstration.

0.2. State of the art

0.2.1. FPGA and reconfigurable computing

FPGA stands for Field Programmable Gate Array. FPGAs are programmable semiconductor logic devices. Users can realize desired logical functions by designing the logic in programming languages. This can be done even after the manufacturing process for several types of FPGAs such as SRAM-based FPGA (hereinafter denoted as SRAM-FPGA). This outstanding feature triggered the initiation of Reconfigurable Computing (RC). The main application fields are Digital Signal Processing (DSP) and Digital Image Processing (DIP). The practical history of RC, the use of programmable logic to accelerate computation, started in the late 1980s with the widespread commercial availability of FPGAs [1]. RC is an emerging field, in which many different hardware algorithms could execute, in turn, on a single device, just as many different software algorithms can run on a conventional processor. Due to the rapid increase in density of FPGAs following the improvement of semiconductor memory devices which continues to track Moore's Law, reconfigurable computers advance technologically at a faster rate than microprocessors [1]. According to this background, space qualification of reconfigurable computers will lead to great enhancements of space systems' performance, and therefore, is recently of great interest.

The speed advantage of FPGAs derives from the fact that the programmable hardware is customized to a particular algorithm. It is estimated routinely 10-100 times the equivalent software algorithm on microprocessors [1]. In addition to this, due to the substantially lower clock frequencies than microprocessors, software to FPGA migration also results in a reduction of power consumption. Furthermore, compared with Application-Specific Integrated Circuits (ASICs) and Mask-Programmable Gate Arrays (MPGAs), FPGAs have several advantages for their users, including: quick time to market; commercial availability; no non-recurring engineering costs for fabrication; pre-tested silicon for use by the designer; and reprogrammability by users, allowing designers to upgrade or change logic through in-system programming.

0.2.2. Reconfigurable FPGAs in space applications

The involvement of FPGAs in space applications has been, however, very limited to radiation tolerant Antifuse-FPGA's, which can be configured only once. In later years, the SRAM-FPGA was used in some applications in noncritical peripheral components such as experimental instruments. For example, the Adaptive Instrument Module (AIM) of the Australian small satellite *Fedsat* has been developed to evaluate radiation effects on SRAM-FPGAs in orbit [2]. Several technical reports describe conceptual designs and implementations of the AIM ([3], [4], and [5]), however, there is no final report about AIM's flight experience available. Generally, even if military experiments on SRAM-FPGAs exist, no detailed report is publicly available.

To the knowledge of author, so far no satellite main on-board computing system has been launched, which is purely based on reconfigurable FPGAs. In this sense, it is a great interest to realize RC for space applications, which is the ultimate goal of this thesis as the world's first attempt. The detailed implementation concept and radiation effect mitigation methods based on a combinational use of different types of FPGAs are described in Part I.

0.2.3. Evolution of small satellites

After the dawn of the space development initiated by the satellite Sputnik, the size of satellites in early ages became gradually bigger according to the increase of mission complexities. After the birth of satellites with microcontroller, the trend of mass of recent space systems has spread out into two opposite ways: large satellites and small satellites. Within the course of this history, small satellites are recently of increasing interest all over the world for their attractive applications [6]. An advantage of small satellites is the fast and cost-effective development possibility, which makes them a suitable platform for rapid new technology evaluations and demonstrations. Indeed, the capability of payload instruments is becoming dramatically better and the attitude and orbit control systems are becoming complexer based on growing complex mission requirements of small satellites. Based on this background, against the limitation of available size, mass and power aboard small satellites, scientists are encountering demands on high computational capabilities on small satellites. This is the driving force of developing an innovative solution of on-board computing systems.

On the contrary to the above mentioned main stream of space development, due to the initiation of Cube Sat projects, small satellites are developed at many academic institutions even for educational purposes [7], [8], and [9]. Most of them are very simple which downlink their housekeeping data, sensor data, and sometimes image data [10]. Some researchers are even investigating the development of Femto-satellites as a goal of miniaturization [6].

0.3. FPGA technology for small satellites

Small satellites have cultivated their markets and are nowadays playing very important roles in the fields of Earth observation, such as disaster monitoring, and technology demonstrations, which makes the satellites very complex and highly integrated [11]. Also, the amount of generated data by latest sensors is enormous due to their high spectral, spatial, temporal, and radiometric resolutions. The future need of OBCs for small satellites is to process the high resolution payload data besides the complex attitude and orbit control algorithms. Given limited resources of small satellites, a smart solution for this is to use a single highly integrated OBC as a central on-board computing system. Recent FPGA technologies are the ideal candidate with the potential to fulfill these requirements. The unification of different computing functions aboard satellites onto a single computer system results in a clear-cut system architecture and provides a high degree of fault tolerance with minimal resources. In this way, the design of satellites becomes simpler resulting in a higher reliability, a higher agility, and a higher integrity, which makes the integration and verification easy to a considerable extent. According to this background, there is a great interest on demonstrating the application of FPGA technologies for innovative small satellite on-board computing systems.

0.4. Scope of research

The thesis is arranged in following parts:

- Part I: General application methods of reconfigurable FPGA technologies for space systems with a radiation mitigation method based on a combinational use of different types of reconfigurable FPGAs (Chapters 1, 2)
- Part II: Application of the developed method in Part I to the small satellite *Flying Laptop*, and development of the OBC and hardware logics inside FPGAs (Chapters 3–6)
- Part III: Development of a simulation and verification environment and the verification results of the developed OBC hardware logic, as well as investigations on formal verification methods of hardware logic algorithms (Chapters 7–9)
- Part IV: Conclusions and Appendices

Chapter 1 starts from the historical background of FPGA technologies, then summarizes the radiation effects on different types of FPGA devices based on experimental data. Chapter 2 describes the investigation on application methods of reconfigurable FPGAs for space environment and conceptualizes a combinational use of different types of FPGAs. Chapter 3 provides an introduction to the *Flying Laptop* project, including its scientific instruments and attitude determination and control components. In Chapter 4 the system design of the *Flying Laptop* is summarized reflecting the aspects of applying FPGAs as the central computing system. Chapters 5 and 6 describe the development of hardware components and control logic of the OBC, respectively. Following the project phases of the *Flying Laptop*, the breadboard model of the OBC and engineering models of some components are developed within the scope of this thesis. Chapter 6 provides information on how to implement whole satellite control functions into a single FPGA chip in a sufficient depth. Chapter 7 describes the development of an FPGA hardware-in-the-loop simulation and verification environment. Chapter 8 illustrates the simulation and verification results. Finally, Chapter 9 summarizes investigation on formal verification possibilities of the designed control algorithm. At the end, Chapter 10 provides conclusions and an outlook for future works. The calculated failure frequencies of the selected FPGA devices by the radiation model CREME96 are summarized in Appendix A. Appendix B provides theory and basic background for basic probability calculus. Appendix C summarizes additional information on system design of the *Flying Laptop* satellite. Information on attitude control algorithms is provided in Appendix D. Details of hardware logic design by the hardware description language Handel-C is summarized in Appendix E. Finally, Appendix F provides notation references for formal verification of the Handel-C programs.

Part I.

Reconfigurable FPGA technologies for space applications

In this part, general application methods of terrestrial reconfigurable FPGA technologies for space environment are described. In Chapter 1, mechanisms of different types of state-of-the-art FPGA technologies and radiation effects on them are summarized. Experimentally obtained characteristics against radiation effects are surveyed for SRAM- and Flash-based FPGA devices. Based on the information, failure frequencies of FPGA devices are calculated by means of the radiation effect model CREME96. In Chapter 2, the developed radiation mitigation method based on a combinational use of different types of reconfigurable FPGAs is described. Reliability modeling of the conceptualized repairable “2-out-of-m” redundant systems with a voting mechanism has been conducted and their reliabilities and operational concept are analyzed. Investigations on radiation effects provided in Chapter 1 and the extensive system reliability analysis in Chapter 2 prove the validity of proposed application methods of FPGAs for space systems.

1. Field Programmable Gate Array

An FPGA is a programmable semiconductor logic device. The user can realize desired logical functions by designing the logic in programming languages. This can be done even after the manufacturing process. Furthermore, non-antifuse type FPGA can be even reconfigured many times. In this chapter, the history of FPGA technologies and their mechanisms and radiation effects are summarized. This chapter will provide a sufficient technical background for further discussions on application of FPGA technologies for space systems.

1.1. Types of Field Programmable Gate Arrays

There are mainly three different types of FPGAs: SRAM, Flash, and Antifuse-based FPGAs (hereafter denoted just as SRAM-FPGA, Flash-FPGA, and Antifuse-FPGA). The first FPGA ever was based on the SRAM technology by Xilinx [12]. It has characteristics similar to a gate array (only programmable by the manufacturer) and has a higher freedom of design compared to traditional Programmable Logic Devices (PLDs), therefore it was named as Field Programmable Gate Array. SRAM-FPGAs are field programmable and reconfigurable. The logic and wiring information can be loaded into SRAM-based memory elements within the chip by the user. This FPGA technology is volatile and the program shall be stored in external nonvolatile memory storage. The program shall be loaded at the beginning of use.

The contact points in the circuit of Antifuse-FPGAs are, on the contrary, made of antifuses and the desired logic circuit can be realized by burning them off. This type of FPGA is one-time programmable (OTP), and once it is programmed, the device can be driven with lower circuit resistance. This is non-volatile. The circuit resistance and area size of Antifuse-FPGA can be minimized compared to the SRAM-FPGA.

Flash-FPGAs are reconfigurable and nonvolatile. However, because of their complexity in manufacturing, it was difficult to make them highly integrated and cheep. They were not used so much in fields which need large amount of logic and high-speed operation. In Figure 1.1 the relation between these logic devices are illustrated.

In spite of the capability of FPGAs in making development of logic device dramatically faster compared to ASICs, it was still difficult to realize efficient design because of the lower degree of integration and lower capability of design tools in early years. However, as time goes by, the degree of integration has increased thanks to the state-of-the-art semiconductor processing technology and efficient design became possible thanks to the advancement of design tools. At the same time, the prices of FPGA devices have become reasonable so that general developers can afford them. Semiconductor memory is an essential part of modern information processors, and like all silicon technology it has been more or less growing in density and performance in accordance with Moore's law [13]. In this sense, FPGAs hold enormous potential to become future central logic devices, which can offer attractive design solutions to space applications.

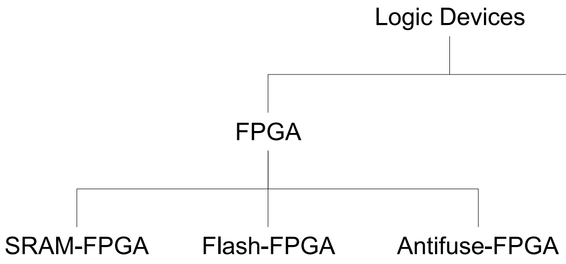


Figure 1.1.: Relation between FPGA devices and ASIC

1.1.1. State of the art FPGA product

The state-of-the-art FPGA products are listed in Table 1.1 [14]. Xilinx Inc. [12] is the world's largest SRAM-FPGA supplier followed by Altera Corporation [15] (hereafter denoted just as Xilinx and Altera). Both of them have been providing different types and size of SRAM-FPGA devices as listed in the table. Actel Corporation [16] is the world's leading manufacturer in the fields of Flash and Antifuse-FPGAs, offering different types of devices listed in the table. Because of the flexibility, high integrity, and larger logic size of SRAM-FPGAs, they are the most attractive candidates to implement reconfigurable complex large scale satellite control algorithms. Also the new radiation tolerant version of the ProASIC family (RTProASIC) offers attractive performances for space applications [17], [18].

Table 1.1.: FPGA product families from leading manufacturers ([12], [15], and [16])

FPGA	Xilinx	Altera	Actel
SRAM-FPGA	· Virtex · Spartan	· Stratix · Arria GX · Cyclone	N/A
Flash-FPGA	N/A	N/A	· IGLOO · ProASIC · RTProASIC
Antifuse-FPGA	N/A	N/A	· Axcelerator · SX-A, eX, MX · RTAX, RTSX

1.2. FPGA technologies

1.2.1. Mechanism of programmable elements

An FPGA device consists of electrically programmable interconnections and logic modules. The programmability is accomplished by turning a switch on/off to control the connection of two lines [19]. The logic is actually implemented in the programmable logic components and interconnects. Programmable logic components can realize not only basic logic gates such as AND, OR, XOR, NOT, but also complex combinational logic functions. Reprogramming of SRAM- and Flash-FPGAs, as well as programming of Antifuse-FPGAs can be done by the user. Detailed mechanisms of the three different FPGAs are summarized below.

SRAM-FPGA

An SRAM is a semiconductor memory, which does not need to be periodically refreshed, unlike DRAM. SRAM uses bistable latching circuitry, or flip-flop to store each bit information. SRAM can store bit information while the power is supplied but it is still volatile in the conventional sense that data is eventually lost when the power is turned off. Each bit information in an SRAM is stored on four transistors that form two cross-coupled inverters as illustrated in Figure 1.2. This cell has two stable states which are used to denote 0 and 1. SRAM-FPGAs can be highly integrated into large scale logic and is the most flexible. The bit information in the cross-coupled inverters may be converted by radiation effects occurring inside the circuitry. SRAM-FPGAs are the most radiation vulnerable among the three [20], [21].

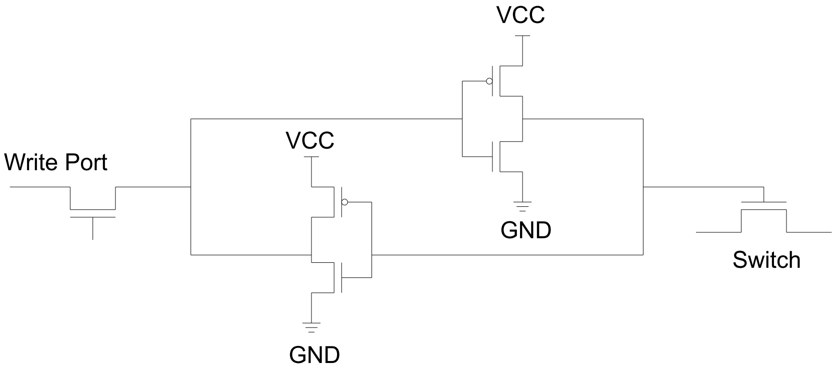


Figure 1.2.: CMOS SRAM cell

Flash-FPGA

Flash-FPGA combines two major traits of the SRAM and the Antifuse-FPGA: reconfigurability and being nonvolatile. Flash-FPGAs are based on Flash memory technologies based on EEPROM technologies. Hence, it is possible to erase the entire chip or a subarray within the chip at one time. Flash memory makes use of a charge stored on a floating-gate to accomplish nonvolatile data storage. Figure 1.3 provides a cross-section sketch of a floating-gate transistor and its circuit symbol representation. The floating-gate electrode usually consists of a polysilicon layer formed within the gate insulator of a field-effect transistor between the normal gate electrode (the control gate) and the channel. The amount of charge on the floating gate will determine whether the transistor will conduct or not.

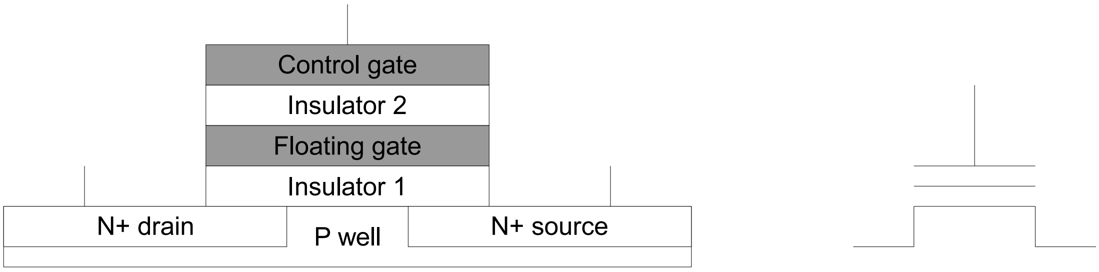


Figure 1.3.: Elements of a Flash memory cell

As shown in Figure 1.4, the switch element consists of two floating gate NMOS-transistors: A switch transistor turns on or off the data path, and a program/sense transistor programs the floating gate voltage and senses the current during threshold voltage measurement. These two transistors share the same control gate and the same floating-gate. The floating switch is “programmed” to a low threshold state to turn the switch on, and “erased” to high threshold state to turn it off [22].

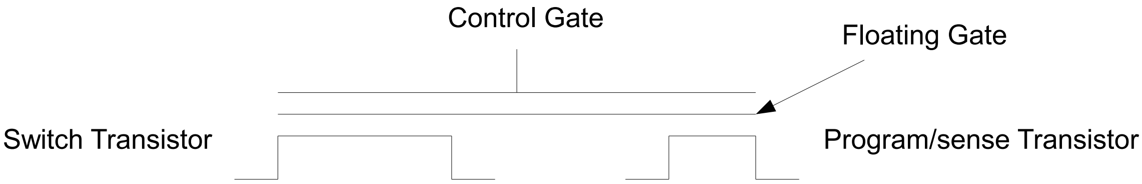


Figure 1.4.: Floating gate structure in Flash-FPGA

The fact that the floating gate is completely surrounded by insulators allows it to retain charge for a long period of time independent of whether the circuit power supply voltage is present. Furthermore, the act of reading the data can be performed without loss of the information [13]. The amount of electrical charge in the floating gate can be affected by radiation effects, which can lead to conversion of the information stored.

Antifuse-FPGA

A fuse is a type of electrical over-current protection device. In case of over-current, it breaks the circuit permanently and protects other components on the same circuit from damage. An antifuse is an electrical device that performs the opposite function to a fuse. An antifuse has a high resistance at the beginning and is designed to create an electrically conductive circuit path permanently, typically at an excessive voltage. An antifuse can be realized using a thin barrier of non-conducting amorphous silicon between two metal conductors. When a sufficiently high voltage is applied across the amorphous silicon, it is turned into a polycrystalline silicon-metal alloy with a low resistance.

In Figure 1.5 the structure of the HiRel SX-A family of Actel is illustrated, which consists of metal interconnect layers with antifuse routing interconnect resources between them [23]. Radiation tolerant versions of Actel’s Antifuse-FPGAs are equipped with triple module redundant flip-flops in order to mitigate single event upsets (see Section 1.3) caused by radiation effects. Furthermore, a mitigation method of single event transient has been also established [24].

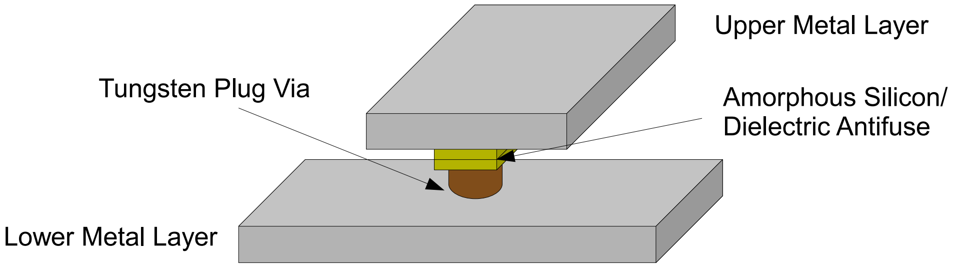


Figure 1.5.: Interconnect elements of Antifuse-FPGA (Actel SX-A) [23]

1.2.2. Application fields of FPGAs

FPGAs are nowadays the densest and most advanced programmable logic devices. It enables designers to implement large digital designs in a device at any time at any location. The advantages of FPGAs are their high density, high computational performance, fast turnaround and low cost per design implementation. Application fields of FPGAs include ASIC prototyping, DIP, DSP such as FFT, cryptography, computer vision, and so forth. FPGA's can be especially applied for any area that makes use of its massive parallelism. In the fields of terrestrial applications, FPGAs are currently widely used in almost all semiconductor application fields, while space application of reconfigurable FPGAs is still not yet common due to radiation effects.

1.2.3. FPGA technologies in space applications

Due to the extremely attractive features of FPGA technology for aerospace applications, the space community has actively evaluated radiation effects for every new FPGA coming out of the production line [19]. NASA's Office of Logic Design is the leading organization of this activities. It provides radiation test data for FPGA products by leading manufactures such as Xilinx, Actel, Altera, etc.

The involvement of FPGAs in space applications have been, however, very limited to radiation tolerant Antifuse-FPGAs, which can be configured only once, due to the radiation vulnerability of other reconfigurable FPGAs. In the history, the Antifuse-FPGA started to be accepted by aerospace designers after the JPL (Jet Propulsion Laboratory) published a report recommending the use of several Antifuse-based FPGAs for space applications in 1992 [25]. In later years, the use of SRAM-FPGAs in space was investigated by, e.g., AIM of the Australian small satellite Fedsat ([3], [4], and [5]), and possibly on military satellites, after their single event latch-up problems were solved. However, to the knowledge of author, there is no information available in public. In the last few years, the Flash-FPGA joined the action after its introduction and subsequent elimination of its single event latch-up susceptibility. The first radiation tolerant Flash-FPGA by Actel came to the world in a fairly recent past, but their space qualification has not yet done. Based on this background, because of its tolerance against radiation effects, Antifuse technology is the dominant one for applying FPGAs in harsh radiation environment.

On the contrary, there has been a tremendous interest on reconfigurable FPGAs for the potential realization of a “reconfigurable satellite” in the future [26]. Even through, these Antifuse-FPGA's are playing significant roles it is also a fact, in the present situation, that they just replace the function of previous ASICs and are not contributing to realizing reconfigurable systems in space. The FPGA technology to be used for those satellite's on-board computing systems shall be reconfigurable even after the satellite was launched into orbit. To allow flexible implementation and the capability of reconfiguration for space systems, innovative technology has to be established for employing reconfigurable FPGAs for space applications. The attractive fields in space applications are especially image processing and digital signal processing which can be used for, for example, computer vision for payload camera data processing and decision making aboard a spacecraft, which are very attractive to future intelligent space explorations.

1.3. Radiation effects on FPGAs

In this section, radiation effects on FPGA devices in terms of Total Ionizing Dose (TID) effects and Single Event Effects (SEEs) are summarized at first. Because antifuse switches are nonvolatile and tolerant against single event and total ionizing dose effects [26], detailed investigation for SRAM- and Flash-FPGAs are conducted in separate sections.

1.3.1. Radiation environment in space

The use of commercial off-the-shelf (COTS) semiconductors of all kinds has become normal on Earth, mostly with no respect to radiation at all. Radiation becomes a problem when going into the space environment. The radiation effects on semiconductors are not to be underestimated and have to be considered a high risk when designing on-board computing elements of space systems.

The radiation in space consists of different particles, mainly electrons, protons, helium nuclei, and heavy ions originating from solar particle events, novas, and supernovas also known as cosmic rays. As electrons and protons come closer to Earth they are influenced by the Earth's magnetic field, resulting in different zones of charged particles around the planet, which are called Van Allen radiation belts. Radiation effects can be roughly divided into two groups: TID effects and SEEs.

1.3.2. Total ionizing dose effect

TID defines the total sum of radiation hitting the target component during in-flight mission time and is one main criteria for designing electric circuits with MOS parts. Heavy ions originating mostly from the cosmic rays and partly from solar events, as well as secondary X-Rays or Bremsstrahlung generated by charged particles penetrating the skin of a spacecraft are contributing a significant part to the TID. The unit "rad" specifies the radiation dose or deposited energy. Rad is defined as $100\,\text{rad} = 1\,\text{Gray}$ (Gy), whereas Gy is the SI unit defined by the amount of 1 Joule of energy deposited per kilogram. Particles entering the semiconductor create electron-hole pairs in the silicon dioxide layer. Even though some self-healing processes take place, the effect is essentially permanent. This leads to an performance degradation and finally into the failing of MOS devices.

1.3.3. Single event effects

Another criteria for designing MOS circuits is the amount of Linear Energy Transfer (LET), defined as energy deposited per traversing length per material specific density ($\text{MeV}\cdot\text{cm}^2/\text{mg}$). The higher the energy of one charged particle traversing through a MOS part is, the more energy the particle deposits in it. Depending on the amount of deposited energy, different effects occur, which are generally named as Single Event Effects (SEEs). The most important effects for FPGA devices are the Single Event Transient (SET), the Single Event Upset (SEU), and the Single Event Latch-up (SEL). Each of these events will be explained in detail below.

Single event transients

The Single Event Transients are general effects, called soft error, which happen when a charged particle hits a conductive part in an electrical device and induces voltage. The charged particle's transfered energy can be routed through the device and result in a state change of, e.g., a flip-flop and thereby result in a SEU. This error affects mainly SRAM devices due to their proneness to SEU. Another effect of a SET is the interaction with the internal clock signal of computer systems, by widening the pulse signal by falling on the trailing edge and narrowing it down by falling on the front edge of the signal. In this way SETs influence the processing speed of the clock signal dependent system [26]. Furthermore, it is well known, that errors originating from SETs are strongly dependent on the internal clock frequencies [27]. As charged particles can hit the electronic system anytime and anywhere no countermeasures except for full radiation shielding is possible. A SET can be critical when complex computer systems with independent but synchronized calculating nodes are used. The synchronizing clock signal routed through the system can be altered all the way between the sending node and the receiving nodes resulting in possible difficulties for the exact synchronization.

Single event upset

Another soft error is the SEU, which shall be considered as normal phenomena on SRAM devices when exposed to radiation. The energy transferred by charged particles results in state changes of flip-flops (1 changes to 0 and vice versa). Mitigation methods like Triple Module Redundancy (TMR) and partial re-construction offer a high protection, minimizing the risk of further data damage. However some single SEUs can be classified as Single Event Functional Interrupts (SEFIs), which trigger unusual far reaching consequences that disable large portions of the device function and can lead to device malfunction.

As Flash devices are not based on flip-flop technology but on EEPROM technology the configuration elements of them are immune to SEU. Nevertheless, Flash based FPGAs still contain SRAM parts and these are vulnerable against SEUs. For the application of Flash-FPGAs in space environment, radiation mitigation methods shall be established. Chapter 6 describes the detail of these mitigation methods.

Single event latch-up

Far more dangerous than soft errors are hard errors like a Single Event Latch-up, originating from a sufficient energy from a charged particle induced in a parasitic thyristor of a n-p-n-p device. This leads to an excessive supply power and consequently to a permanent loss of device functionality. If the power is not controlled by an external source the device burns out making the SEL one of the most concerned SEEs for CMOS devices. In other words, SEL can be prevented from by monitoring power consumption of the chip and separating from the power line quickly enough in case a SEL occurred. SELs can happen to SRAM devices as well as Flash devices and are mainly dependent on the specific hardware architecture of the device. SEL for all types of FPGAs can be nowadays mitigated by design at manufacturing and devices with the mitigation design are classified as radiation tolerant devices. For Flash-FPGAs, probability of occurrence of a SEL is estimated to be high during the reprogramming phase due to the high voltage needed for the process.

1.4. Radiation effects analysis on SRAM- and Flash-FPGAs

In this section, the results of information survey of TID effects and SEEs on SRAM- and Flash-FPGAs are summarized. The data obtained here makes the basis of discussion in Chapter 2. In the following chapters, due to the limitation on available reliable information, SRAM-FPGAs of Xilinx and Flash-FPGAs of Actel are selected for further analysis and discussion.

1.4.1. TID effects on SRAM-FPGAs

TID effects on Virtex SRAM-FPGA families of Xilinx have been experimentally measured and actively tracked by the interested community for about a decade every time a new product came to the market [21], [28], and [29]. A summary of the experimentally measured data by Fabula and George are summarized in Table 1.2. As illustrated in this table, the TID tolerance of Xilinx Virtex family is becoming gradually better recently achieving more than 300 krad. Though some investigations on TID mitigation method can be seen in [30], the state-of-the-art SRAM-FPGAs are immune enough against TID effects for, e.g., LEO missions.

Table 1.2.: TID effects on SRAM-FPGA [31], [32]

Product	Process Technoogy	Metal layers	TID [krad]
Virtex	220 nm CMOS	6	100
Virtex-II	150 nm CMOS	7	200
Virtex-II Pro	130 nm CMOS	8	250 - 300
Virtex-4	90 nm CMOS	10	300
Virtex-5	65 nm CMOS	11	< 500

1.4.2. TID effects on Flash-FPGAs

The most attractive Flash-FPGA for space application today is the RTProASIC family of Actel, which is the brand-new radiation tolerant Flash-FPGAs today [17]. In this section, RTProASIC and ProASIC3 families of Actel are analyzed [33]. Investigation on radiation response of floating-gate of EEPROM memory itself has been done since 1980's [34]. The recent report on this topic can be seen in [22]. The test results for ProASIC3 can be seen in [35] and that of RTProASIC in [27]. These data are summarized in Table 1.3. In the table, the amount of TID for safe programming and erase, as well as that for normal operation are summarized. The former value is smaller than the latter. It can be seen that compared to TID performance of SRAM-FPGAs above, Flash-FPGAs are remarkably vulnerable against TID.

Table 1.3.: Limit of TID effects on Flash-FPGA [17]

Product	Safe programming and erase [krad]	Normal operation [krad]
ProASIC3	-	20
RTProASIC	15	25

1.4.3. SEEs on SRAM-FPGAs

After the invention of SRAM-FPGA technology, Xilinx has been so far offering different types of SRAM-FPGAs. In terms of the size of devices and performance, the Virtex family of Xilinx is very attractive for space applications. The first series of this was Virtex, followed by Virtex-II, Virtex-II Pro, Virtex-4, and Virtex-5. The Virtex-5 family is the state-of-the-art SRAM-FPGA today. Xilinx recognized the meaning of space application of SRAM-FPGA and paid great efforts for developing radiation tolerant ones in terms of SEL. Xilinx started with the Virtex-II series to offer radiation tolerant versions, which led to recent successful development of the radiation tolerant Virtex-4. Today, Xilinx is offering four different radiation tolerant SRAM-FPGA chips. However, they are still vulnerable against other types of SEEs, thus these shall be mitigated for space applications.

Investigations on SEE started already for the first Virtex family, which were called as XQVR series. For example, the Los Alamos National Laboratory presented excellent reports on this topic [36], [37], and [20]. Since that time, the community has been tracking the SEE on Xilinx Virtex family FPGAs.

Detailed experimental results on SEEs on Virtex-II devices are conducted by the Xilinx SEE consortium members in 2004 [38]. Koga and George have reported detailed comparison of Xilinx Virtex-II FPGA SEE Sensitivities to Protons and Heavy Ions [39]. Yui and Swift have reported SEU susceptibility of the Virtex-II [40]. Investigations on the mitigation method of SEE and SEU of Virtex-II have been conducted and reported in [41], [42]. Foster and O'Neil have simulated Proton Upsets in Virtex-II using Monte Carlo simulations [43]. Edmonds and Irom have conducted investigations on extension of a proton SEU cross section model to include neutron effects [44].

Compared to the Virtex-II, less radiation effects investigation on Virtex-II Pro have been conducted. Virtex-II Pro is an extended version of the Xilinx-II. Virtex-II Pro is equipped with microprocessors (PowerPC Core) inside, which enables users to run software-based codes in FPGA. According to [45], Virtex-II Pro has similar or improved radiation performance, but not sufficient data is available for Virtex-II Pro. Test results of SEEs on Virtex-II Pro can be seen in [32]. NASA Goddard Space Flight Center has conducted proton tests on the PowerPC elements inside the Virtex-II Pro [46]. Xilinx has originally planned to develop radiation tolerant versions of the Virtex-II Pro family but before being successful, it was decided to develop new types of radiation tolerant versions for Virtex-4 families.

For Virtex-4 families, even though SEU characterization tests have ever been performed, detailed parameters are not available [47], [48]. Partly because there exist radiation tolerant versions of Virtex-4, called Virtex-4QV, detailed radiation effect tests have been conducted for these devices. JPL has provided an extensive report on SEU on Virtex-4QV [49].

Virtex-5 is very new in the market and so far few test results have been reported. Quinn and Morgan have conducted static proton and heavy ion testing on them [50], and George and Koga have conducted investigations on neutron soft errors in Virtex-5 together with Virtex-4 [45]. Investigation on Virtex-5 is not yet finished. Xilinx is now developing radiation tolerant versions of Virtex-5, which will last until 2010. Past studies on mitigation methods of SEE on SRAM-FPGAs can be seen in [51], [52].

Consequently, based on the above information survey, it is revealed that there are sufficient information for further discussions only for Virtex-II and Virtex-4QV. However, as mentioned above, the Virtex-II Pro family offers additional functionality to the Virtex-II, which is very attractive to space applications, and also because the radiation performance is comparably similar to that of Virtex-II, one of the Virtex-II Pro devicee XC2VP50 is selected for the further investigation (the difference between the two devices are regarded as safety margin.) together with the comparable device XQR4VFX60 of Vierex-4QV family. In Table 1.4, the parameters of both selected devices are listed. Though, there are several differences between the two devices due to generation difference, they are well comparable in size.

Table 1.4.: Parameters of Xilinx SRAM FPGAs [53], [54]

Parameter	Virtex-II Pro (XC2VP50)	Virtex-4 QV (XQR4VFX60)
RocketIO Transceiver Blocks	0 or 16	N/A
PowerPC Processor Blocks	2	2
LogicCells	53 136	56 880
Slices	23 616	25 280
Max. Distributed RAM [Kb]	738	395
Digital Signal Processor	232 (18x18 Bit Multiplier Blocks)	128 (XtremeDSP Slices)
BRAM max bit	4 176 000	4 276 224
BRAM 18kb Blocks	232	232
DCMs	8	12
Maximum User I/O Pads	852	567
configuration bits	19 021 344	14 500 000

Weibull distribution function

For the radiation characterization of SEE-proneness of FPGAs so-called "cross-section over LET" models are used. The cross-section can thereby refer to one specific element of the FPGA, for instance one bit of the configuration memory or it can refer to the whole chip. For making predictions from obtained measurements the Weibull distribution has become the most commonly utilized tool, especially for heavy ion radiations. For this distribution several parameters are used which are described below. The general form of the Weibull-curve is:

$$F(L_w) = \sigma_{sat} \cdot (1 - e^{-A}) \text{ with } A = \left(\frac{(L_w - L_{w,0})}{W_w} \right)^s , \tag{1.1}$$

where:

- σ_{sat}: limiting or plateau cross-section, also called "limit"
- $L_{w,0}$: LET threshold parameter, also called "onset"
- L_w: functions input value, in this case the effective LET value
- W_w: dimensionless width parameter
- s: dimensionless exponent called as power

By being provided with these parameters, Weibull distribution function can be uniquely specified. For the selected components here, the Weibull parameters of Proton and heavy ions effects are extracted from above mentioned reports as listed in Table 1.5 for the Virtex-II Pro and in Table 1.6 for the Virtex-4QV. Due to the difference in the structures, different failure modes have been identified for the two devices. Total failure modes are:

- CFG: A SEU in the configuration memory
- BRAM: A SEU in the Block RAM
- POR: SEFI in the Power-on-Reset circuitry
- SMAP: A SEFI in the SelectMAP circuitry
- JCFG: A SEFI in the JTAG Configuration Access Port circuitry
- F/F: User flip-flops ("ones" means state of 1, and "zeros" means state of 0)
- GSIG: Global Signal

Some of them are only applicable to the latter device. These values in the two tables can be used for further investigation on total Single Event Effects on both SRAM-FPGAs.

Table 1.5.: Weibull parameter of Virtex-II Pro XC2VP50 [38]

Parameter	CFG	BRAM	POR	SMAP	JCFG
Proton					
$L_{w,0}$ [MeV]	3.0	3.0	7.0	6.5	6.0
W_w [-]	12	12	12	12	12
s [-]	0.5	0.6	1.0	0.5	0.5
σ_{sat} [cm^2/10^{-12}]	0.038	0.041	0.374	0.572	0.286
Heavy Ions					
$L_{w,0}$ [MeV·cm^2/mg]	1	1	1.5	1.5	1.5
W_w [-]	33	17	22	17	17
s [-]	0.8	0.9	1.2	1	1
σ_{sat} [cm^2/10^{-6}]	0.0437	0.0419	2.50	1.72	2.51E-7

Table 1.6.: Weibull parameter of Virtex-4QV XQR4VFX60 [49]

Parameter	CFG	BRAM	F/F(1)	F/F(0)	POR	SMAP	GSIG
Proton							
$L_{w,0}$ [MeV]	4	1	2.5	5	5.8	6.5	7.9
W_w [-]	80	20	20	20	40	10	55
s [-]	0.586	1.546	1.546	1.546	0.76	0.568	0.545
σ_{sat} [cm^2/10^{-12}]	0.0473	0.0450	0.150	0.0450	0.2200	0.0170	0.5500
Heavy Ions							
$L_{w,0}$ [MeV·cm^2/mg]	0.5	0.2	0.5	1.5	0.2	0.2	0.2
W_w [-]	400	70	400	400	150	1200	400
s [-]	0.985	0.724	0.923	0.923	1.169	1.169	0.935
σ_{sat} [cm^2/10^{-8}]	26.3	3.50	75.0	61.0	62.7	55.2	50.3

Radiation effect model: CREME96

Based on the above knowledge on radiation characteristics of FPGAs one can calculate actual possible failure rates of given devices, under the assumption that one has a reliable space radiation environment model and also knows the detailed mechanism of radiation effects inside the FPGA devices. Indeed, the Naval Research Laboratory of the USA has conducted research on this topic and has developed a radiation effects model called CREME, which stands for Cosmic Ray Effects on Micro-Electronics. CREME is a suite of programs for creating numerical models of the ionizing radiation environment in near-Earth orbits and for evaluating radiation effects on spacecraft. Since its first release in 1981, CREME has literally become an industry standard, mandated for DOD systems in MILSTD-1809 and is also a frequent contractual requirement in both NASA and commercial programs [55]. The newest version CREME96 is now accessible at [56]. On the other hand, European Space Agency (ESA) is also providing space environment information system called SPENVIS [57]. However, because SPENVIS uses so-called “critical charge” option model of CREME86, in which the cross-section is treated as a simple step function in LET, it does not give accurate results for space applications [56]. According to this background, CREME96 is used for further investigation.

For the calculation, following parameters have been used:

- Orbit: 900 km circular orbit with a inclination of 99.03°
- evaluating trapped proton fluxes near solar maximum conditions
- stormy magnetic weather conditions
- all elements from an atomic number from 1 to 92 are included
- inside Earth’s magnetosphere

Here, a relatively high orbit in LEO has been chosen due to the higher particle flux which is proportional with the distance from Earth’s surface for LEO. According to these parameters and characteristics of FPGAs identified in above tables, calculations for following conditions can be done by CREME96 based on its internal environment models: Solar ray maximum (cosmic ray minimum), Solar ray minimum (cosmic ray maximum), Worst week, Worst day, and peak 5-minute-average fluxes. The calculated results of failure frequencies of both Virtex-II Pro and Virtex-4QV for input data identified in above tables are summarized in Appendix A in Table A.1, and Table A.2. In the tables, “PUP” and “HUP” represent proton induced and heavy ion induced events, respectively.

According to these results, it can be seen that the configuration memories and internal Block RAMs of each device are the most vulnerable part against radiation compared to other factors with many orders of magnitude. Furthermore, the failure frequency caused by the heavy ion induced events are much higher than that by the proton induced events and therefore, the latter can be negligible. In case of the Virtex-II Pro device (XC2VP50), the dominating failure frequencies are 7.46×10^{-5} of CFG (HUP) and 2.10×10^{-5} of BRAM (HUP). These failure modes with higher failure rates need to be mitigated in order to realize space application of reconfigurable SRAM-FPGAs. The mitigation method, which is proposed by this thesis is described in detail in Chapter 2.

1.4.4. SEEs on Flash-FPGAs

Unlike SRAM-FPGAs, Flash memory cells in Flash-FPGAs do not experience SEU, due to their CMOS-less implementation. The most critical SEE on Flash-FPGAs are instead single event transients. Investigations have been done on mitigation methods of SET in Flash-FPGAs [58]. Radiation test results show that with consideration to the mitigation techniques for SETs the RTProASIC family is able to perform a stable operation in space environment. The implementation of mitigation methods is described in detail in Chapter 6.

1.5. Summary

Firstly, this chapter summarized the historical background of the FPGA technologies and described the mechanisms of different types of state-of-the-art FPGA devices. Detailed radiation effects on the SRAM-, Flash-, and Antifuse-FPGAs are investigated. Especially, single event effects and total ionizing dose effects on reconfigurable FPGAs (SRAM- and Flash-FPGAs) are summarized in detail, based on an extensive experimental data survey. The SRAM-FPGAs Virtex-II Pro and Virtex-4QV of Xilinx and the Flash-FPGA RTProASIC and ProASIC3 of Actel are selected for further investigations due to their attractive performances and available data. For the single event effects investigation of SRAM-FPGAs, Weibull parameters are obtained from available technical notes. Based on this information, failure frequencies of FPGA devices are calculated by means of the industry-standard radiation effect model CREME96 (see Appendix A). The strong tolerance of SRAM-FPGAs against TID effects allows the assumption that the radiation effects due to SEEs can be repaired by reconfigurations. On the contrary, Flash-FPGAs revealed their tolerance against SEEs with adequate mitigation methods, while they are relatively vulnerable against TID effects compared to SRAM-FPGAs.

As described in this chapter, reconfigurable FPGAs are radiation vulnerable, which is the challenge for the realization of the space application of reconfigurable FPGAs. The important point illustrated in this chapter is that the main radiation effects in SRAM- and Flash-FPGAs are different, i. e., SRAM-FPGAs are vulnerable against SEE but tolerant against TID, while Flash-FPGAs are tolerant against most of the SEEs but relatively vulnerable against TID (still sufficiently tolerant enough for LEO as described in Chapter 2). This fact indicates the possibility of realizing SRAM-FPGA-based OBCs with continuous repairs against SEEs, together with Flash-FPGAs as the voting element. This idea of using a combination of SRAM-FPGAs and Flash-FPGAs for radiation mitigation is the underlying motivation of this thesis to conceptualize an application method of ground-based reconfigurable FPGAs for space systems.

2. Application method of reconfigurable FPGAs for space systems

As described in Chapter 1, reconfigurable FPGAs are not yet widely used in fields of space applications due to the severe radiation conditions. In order to overcome this challenge, this study suggests a combinational use of different types of FPGAs together with multi-module redundancy. The idea is that the most flexible and highly integrated reconfigurable SRAM-FPGAs can be duplicated to achieve a high reliability against temporal failures caused by SEEs, while another SEE tolerant FPGA monitors them as the voter. The reliability of proposed system concept in this chapter against radiation effects is high enough so that the system can survive in the space environment. Even though, Antifuse-FPGAs are more radiation tolerant, investigation of utilizing Flash-FPGAs as the voter is of interest, in terms of demonstration of COTS components in space environment. Indeed, data obtained in Chapter 1 suggests sufficient TID budgets of Flash-FPGAs for several years of operation. Moreover, the cost of Flash-FPGAs are lower than that of Antifuse-FPGAs and also their reconfigurability can accelerate flexible ground development activities. Because the logic size of the available Flash-FPGA devices are still too small to implement whole satellite control algorithms, the SRAM-FPGAs are indispensable for the realization of the reconfigurable on-board computers for space systems.

2.1. Combinational use of different types of FPGAs

In Chapter 1 it is extensively described that different types of FPGAs experience different types of radiation effects with different failure frequency. This fact implies that there is a possibility that a combinational use of FPGAs can mitigate radiation effects as a single combined system. Principally, it is possible in engineering technique, that one uses replicated redundant components for temporally-failable components in order to obtain higher reliability. Because the failure source of SRAM-FPGAs can be regarded only as SEEs, which has higher failure frequency but repairable, use of SRAM-FPGAs as repairable replicated redundant system is meaningful. On the contrary, Flash-FPGAs are tolerant against SEE, and therefore can be used as the voter of this redundant SRAM-FPGA unit. In this way, use of these two different types of FPGAs can compensate radiation effects with each other and as a whole, the system can achieve a higher reliability. This system concept is illustrated in Figure 2.1. The redundant SRAM-FPGA system is denoted as “Node System”, meaning each unit of SRAM-FPGA as one of the redundant computing nodes.

In order to achieve a higher reliability, one can use 2-out-of-m systems. In these systems, two components out of m identical components shall work correctly in order to ensure the system reliability. If each component can be repairable, then the detected failure can be repaired within a short time period. With m higher than three, even more than one failures are allowed to occur at the same time, or second failure can occur while the first failure is being repaired.

In this chapter, reliability of these systems are modeled starting from triple module redundant system ($m = 3$).

Following discussion is based on the assumption that the SRAM-FPGA can be reconfigured by the voter and failed FPGAs by SEEs are repairable as good as new, which well reflects the real performance of SRAM-FPGAs. In addition to this, an assumption is made that SET and SEU in data flip-flops on Flash-FPGAs can be mitigated by internal mitigation methods, which is discussed in detail in Chapter 6.

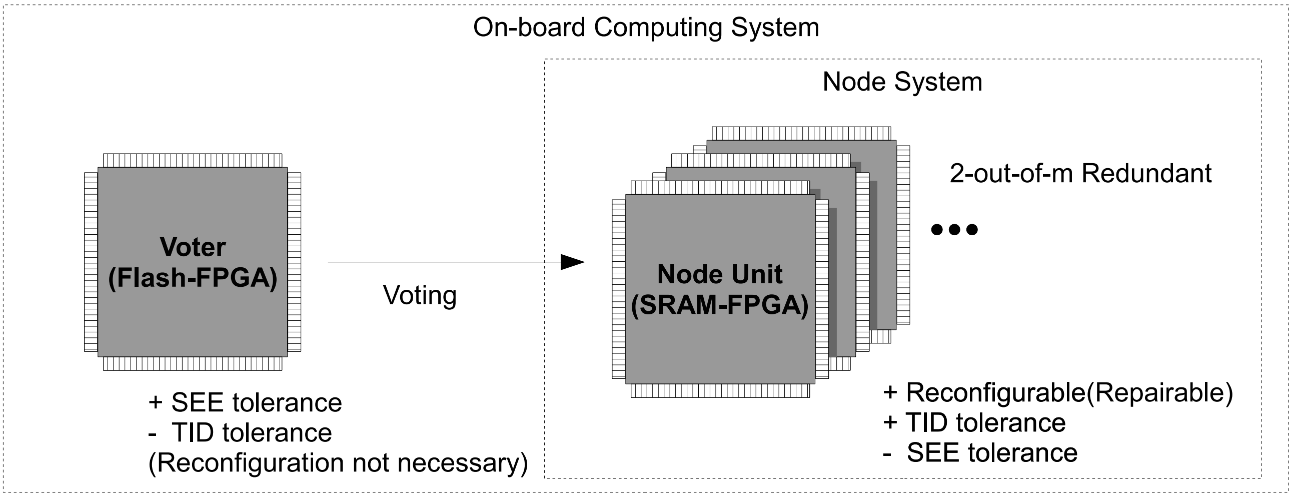


Figure 2.1.: Proposed on-board computer concept

2.2. System reliability analysis

In this section, reliabilities of systems which consist of a “Node System” and a voter unit against radiation effects are analyzed. Letting “NS” indicate the Node System, and “V” the voter, the reliability of the system can be generally described as

$$R_{sys} = R_V \cdot R_{NS} \ , \tag{2.1}$$

where R_{sys}, R_V, and R_{NS} are the reliabilities of the entire system, the voter and the node system. In this sense, the reliability of the voter and the node system can be analyzed separately. Furthermore, because the short term SEE effects can be repaired, SEEs and TID effects can be handled independently. Therefore, Equation (2.1) can be written as:

$$R_{sys} = R_{V,SEE} \cdot R_{V,TID} \cdot R_{NS,SEE} \cdot R_{NS,TID} \ . \tag{2.2}$$

Based on the data summarized in Chapter 1, the SEEs on configuration memories of Flash-FPGAs can be negligible. Because the TID effects on both SRAM- and Flash-FPGAs are static and the reliability against them can be easily calculated (see Section 2.3), the main focus of the following discussions are on the SEEs on SRAM-FPGAs and its reliability modeling with the presence of repair (reconfiguration). Therefore, the most important term is the $R_{NS,SEE}$ and this shall be thoroughly analyzed.

2.2.1. Basic probability theory

The reliability $R(t)$ at t of a single unit without any maintenance or redundancy can be modeled simply as a survivor function of the unit's life time L such as

$$R(t) = P(L > t) = \bar{F}_L(t) \equiv 1 - F_L(t) = P\{X(t) = 0\} = 1 - E\{X(t)\} \ , \tag{2.3}$$

where $X(t)$ is the Boolean indicator of failure at t and $F_L(t)$ is the probability distribution function of unit's life time, as one can see, e.g., in [59]. The mean life time can be described as:

$$E(L) = \int\limits_0^\infty R(t)dt \ . \tag{2.4}$$

In the exponential case, which is generally assumed to apply to electronics hardware after burn-in as described in [59], $F_L(t)$ and $R(t)$ can be described as

$$F_L(t) = 1 - e^{-\lambda t} \ , \tag{2.5}$$

$$R(t) = e^{-\lambda t} \ , \tag{2.6}$$

by introducing the failure rate λ. Therefore, the expected value of the life time can be then described as

$$E(L) = \int\limits_0^\infty e^{-\lambda t}dt = \frac{1}{\lambda} \ . \tag{2.7}$$

From Equation (B.2),

$$f_L(t) \equiv \frac{d}{dt}F_L(t) = \frac{d}{dt}(1 - e^{-\lambda t}) = \lambda \cdot e^{-\lambda t} \tag{2.8}$$

is the probability density function of life time L.

2.2.2. Reliability modeling of repairable single node

For the reliability modeling of a repairable unit, time-dependent probability of success or failure is of interest. Considering Figure 2.2, the point availability $A(t)$ of the repairable unit at time t can be described as a function with recursion [59]:

$$A(t) = \int\limits_0^t [f_L \otimes f_D(t')]A(t - t')dt' + \bar{F}_L(t) \ , \tag{2.9}$$

where f_D is the probability density function of down time D. The last term covers the case without a failure prior to t. Here,

$$f_L \otimes f_D(t') = f_{L+D}(t') \tag{2.10}$$

is the probability density function of the time to the first restart point t'. The Equation (2.9) is a linear integral equation that can be solved via the Laplace transformation; see Appendix B.3.

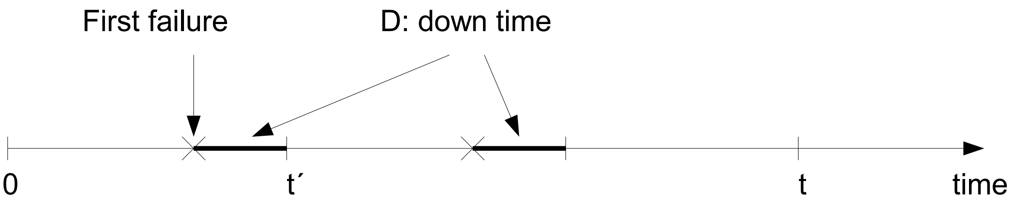


Figure 2.2.: Up and down states of a repairable unit [59]

From Equations (B.16) and (B.13), as well as the convolution theorem (Equation (B.10)) and the integration rule (Equation (B.15)), Laplace transformation for this yields

$$A^*(s) = f_L^*(s) f_D^*(s) A^*(s) + \frac{1 - f_L^*(s)}{s} \ , \tag{2.11}$$

and the solution in the s-domain is:

$$A^*(s) = \frac{1 - f_L^*(s)}{s[1 - f_L^*(s) f_D^*(s)]} \ . \tag{2.12}$$

Solution for exponential case:

In exponential case, from

$$f_L(t) = \lambda \cdot e^{-\lambda t} \Leftrightarrow f_L^*(s) = \frac{\lambda}{s + \lambda} \tag{2.13}$$

and

$$f_D(t) = \mu \cdot e^{-\mu t} \Leftrightarrow f_D^*(s) = \frac{\mu}{s + \mu} \ , \tag{2.14}$$

one finds

$$A^*(s) = \frac{s + \mu}{s(s + \lambda + \mu)} = \frac{1}{s + \lambda + \mu} + \frac{\mu}{s + \lambda + \mu} \ , \tag{2.15}$$

where μ is the repair rate. This can be back-transformed to

$$\begin{aligned} A(t) &= e^{-(\lambda+\mu)t} + \frac{\mu}{\lambda + \mu}\{1 - e^{-(\lambda+\mu)t}\} \\ &= \frac{\mu}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} e^{-(\lambda+\mu)t} \ . \end{aligned} \tag{2.16}$$

The relation between the availability and unavailability $U(t)$ can be described as:

$$A(t) + U(t) = 1 \ . \tag{2.17}$$

In the case of exponential probability functions, using mean time to failure ($MTTF$) and mean time to repair ($MTTR$),

$$\lambda \equiv \frac{1}{MTTF} \ , \tag{2.18}$$

and

$$\mu \equiv \frac{1}{MTTR} \ . \tag{2.19}$$

Applying this to Equation (2.16) with $t \to \infty$ and using mean time between failure ($MTBF$),

$$A(\infty) = \frac{\frac{1}{MTTR}}{\frac{1}{MTTF} + \frac{1}{MTTR}} = \frac{MTTF}{MTTF + MTTR} = \frac{MTTF}{MTBF} \tag{2.20}$$

and

$$U(\infty) \equiv 1 - A(\infty) = \frac{MTTR}{MTTF + MTTR} = \frac{MTTR}{MTBF} \ , \tag{2.21}$$

where

$$MTTF + MTTR = MTBF \ . \tag{2.22}$$

Using $MTBF$, one can define the mean failure frequency as

$$\nu \equiv \frac{1}{MTBF} \ . \tag{2.23}$$

Finally, from above equations one find

$$\nu = A\lambda = U\mu \ . \tag{2.24}$$

2.2.3. 2-out-of-3 systems modeling

Fault tree of 2-out-of-3 systems

A fault tree is a picture of the Boolean function which describes the way of determining the binary system state based on the n binary components states. The fault tree of 2 out of 3 system is illustrated in Figure 2.3.

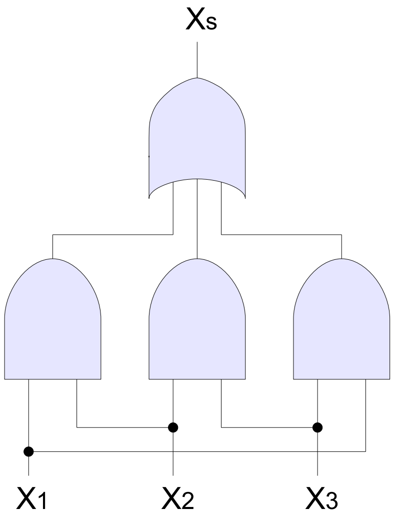


Figure 2.3.: Fault tree of 2-out-of-3 system

The Boolean indicator variables $X_1, \ldots, X_n$ and X_s of the corresponding Boolean function to this fault tree

$$\varphi : B^n \to B^1 \Leftrightarrow X_s = \varphi(X_1, \cdots, X_n) \tag{2.25}$$

can be defined as:

$$X_i = \begin{cases} 0, & \text{if component } i, C_i, \text{ is good} \\ 1, & \text{if } C_i \text{ is bad} \end{cases} \tag{2.26}$$

and

$$X_s = \begin{cases} 0, & \text{if the system } S \text{ is good} \\ 1, & \text{if } S \text{ is bad} \end{cases} , \tag{2.27}$$

where B^n, $B^1 \equiv \{0, 1\}$ is the binary n-space.

Boolean function of 2-out-of-3 systems

Following the discussion in Appendix B.5, the disjunctive normal form of a Boolean function of 2-out-of-3 systems can be described as:

$$\begin{aligned} X_{2of3} &= X_1X_2 \vee X_1X_3 \vee X_2X_3 \\ &= X_1X_2X_3 + X_1X_2\bar{X}_3 + X_1\bar{X}_2X_3 + \bar{X}_1X_2X_3 \ . \end{aligned} \tag{2.28}$$

Unavailability of 2-out-of-3 systems

From Equation (2.3), i.e., $U_i = E(X_i)$ Equation (2.28) holds for X replaced by U and $\bar{X}$ by A. Hence

$$U_{2of3} = U_1U_2U_3 + U_1U_2A_3 + U_1A_2U_3 + A_1U_2U_3 \ , \tag{2.29}$$

because $A = 1 - U$, this can be also written as

$$\begin{aligned} U_{2of3} &= U_1U_2U_3 + U_1U_2(1 - U_3) + U_1(1 - U_2)U_3 + (1 - U_1)U_2U_3 \\ &= U_1U_2 + U_1U_3 + U_2U_3 - 2U_1U_2U_3 \ . \end{aligned} \tag{2.30}$$

Mean failure frequency of 2-out-of-3 systems

For $n \in \{1, 2, 3\}$: system mean failure frequency, ν_{nof3}, is the weighted sum of the 3 components' mean failure frequencies ν_1, ν_2, ν_3. The weights are the criticality probability of the components p_i, i.e., the probability that at the time of changes of X_i from 0 to 1 or vice versa, $X_{nof3} = X_i$:

$$\nu_{nof3} = \sum_{i=1}^{3} p_{n,i}\nu_i; \ p_{n,i} \equiv P(X_{nof3} = X_i); \ n = 1, 2, 3 \ . \tag{2.31}$$

In case of $n = 2$, from Equation (2.30)

$$\begin{aligned} X_{2of3} &= X_iX_j + X_iX_k + X_jX_k - 2X_iX_jX_k \\ &= X_i(X_j + X_k - 2X_jX_k) + X_jX_k \ , \end{aligned} \tag{2.32}$$

and it can be seen that following events are equivalent:

$$\begin{aligned} (X_{2of3} = X_i) &\Leftrightarrow (X_j + X_k - 2X_jX_k = 1) \wedge (X_jX_k = 0) \\ &\Leftrightarrow (X_j \neq X_k) \ . \end{aligned} \tag{2.33}$$

Hence

$$p_{2,i} = P(X_j \neq X_k) = E(X_j) + E(X_k) - 2E(X_jX_k), \tag{2.34}$$

and specifically, for statistically independent X_j, X_k:

$$p_{2,i} = U_j + U_k - 2U_jU_k \ . \tag{2.35}$$

Inserting in Equation (2.31) and from Equation (2.24)

$$\begin{aligned} \nu_{2of3} &= (U_2 + U_3 - 2U_2U_3)U_1\mu_1 + (U_1 + U_3 - 2U_1U_3)U_2\mu_2 \\ &\quad + (U_1 + U_2 - 2U_1U_2)U_3\mu_3 \\ &= U_1U_2(\mu_1 + \mu_2) + U_1U_3(\mu_1 + \mu_3) + U_2U_3(\mu_2 + \mu_3) \\ &\quad - 2U_1U_2U_3(\mu_1 + \mu_2 + \mu_3) \ . \end{aligned} \tag{2.36}$$

For identical components, Equation (2.30) yields

$$U_{2of3} = 3U^2 - 2U^3 \ , \tag{2.37}$$

and Equation (2.36)

$$\nu_{2of3} = 6U^2\mu - 6U^3\mu = 6U^2(1-U)\mu \ . \tag{2.38}$$

2.2.4. 2-out-of-m systems with identical components

The above discussion can be extended to general 2-out-of-m systems. The Boolean function of 2-out-of-m systems can be described as:

$$\begin{aligned} X_{2ofm} &= X_1X_2\cdots X_{m-1} \vee X_1X_2\cdots X_{m-2}X_m \vee \cdots \vee X_2X_3\cdots X_m \\ &= \bigwedge_{i}^{m} X_i \vee \bigvee_{i}^{m} \left\{ \left(\bigwedge_{k,k\neq i}^{m} X_k \right) \wedge \bar{X}_i \right\} \ . \end{aligned} \tag{2.39}$$

Similar to the derivation of Equation (2.37), the corresponding unavailability becomes

$$\begin{aligned} U_{2ofm} &= U^m + mU^{m-1}A \\ &= U^m + mU^{m-1}(1-U) \\ &= mU^{m-1} + (1-m)U^m \ . \end{aligned} \tag{2.40}$$

According to the theorem on frequency of unidirectional changes of state presented in [59], if the Boolean function $\varphi(X_1, \cdots, X_n)$ is given in the multi-linear pseudo-polynomial, the mean failure frequency can be described for statistical independent $X_1, \cdots, X_n$ as:

$$\nu_\varphi = \sum_{i=1}^{m} \left[c_i \left(\prod_{j=1}^{n_i} \tilde{p}_{l_{i,j}} \right) \sum_{j=1}^{n_i} \tilde{\mu}_{l_{i,j}} \right] \ , \tag{2.41}$$

where (for $k = l_{i,j}$)

$$\tilde{p}_k = \begin{cases} p_k & \text{for } \tilde{X}_k = X_k \\ \bar{p}_k \equiv 1 - p_k & \text{for } \tilde{X}_k = \bar{X}_k \end{cases} , \tag{2.42}$$

and

$$\tilde{\mu}_k = \begin{cases} \mu_k & \text{for } \tilde{X}_k = X_k \\ -\lambda_k & \text{for } \tilde{X}_k = \bar{X}_k \end{cases} . \tag{2.43}$$

Finally, from Equations (2.24), (2.39), and (2.41), the mean failure frequency of 2-out-of-m systems with identical components can be described as

$$\begin{aligned} \nu_{2ofm} &= mU^m\mu + mU^{m-1}(1-U)\{(m-1)\mu - \lambda\} \\ &= mU^m\mu + m(m-1)U^{m-1}(1-U)\mu - mU^{m-1}A\lambda \\ &= m(m-1)U^{m-1}(1-U)\mu \ . \end{aligned} \tag{2.44}$$

These equations can be applied for $m \in \{2, 3, 4 \cdots\}$.

2.3. Reliability against TID effects

The effects of TID is cumulative and can not be repaired by means of reconfiguration. The failure rate against the TID effects are related with the amount of dose and does not obey constant failure rate over time unlike the exponential case described above. The probability density function of the failure rate against TID effects can be well modeled by applying normal (Gaussian) function. Accordingly, the probability density function of the life time can be described in terms of the amount of dose z as:

$$f_L(z) = \frac{1}{\sigma\sqrt{2\pi}}\, exp\left(-\frac{1}{2}\left(\frac{z-\zeta}{\sigma}\right)\right) \ , -\infty < z < \infty \ , \tag{2.45}$$

where

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - \zeta)^2} \tag{2.46}$$

is the population standard deviation and

$$\zeta = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{2.47}$$

is the population mean of the discrete random variable samples z_i, and n is the total number of items in the population [60]. One can transform this normal distribution function into the standard normal distribution function by applying following conversions:

$$u = \frac{z-\zeta}{\sigma} \tag{2.48}$$

$$f_L(z) = \frac{f_L(u)}{\sigma} \ , \tag{2.49}$$

resulting in:

$$f_L(u) = \frac{1}{\sqrt{2\pi}}\, exp\left(-\frac{1}{2}u^2\right) \;, -\infty < u < \infty \;. \tag{2.50}$$

From Equation (B.2), the corresponding probability distribution function can be described as:

$$F_L(u) = \int\limits_{-\infty}^{u} \frac{1}{\sqrt{2\pi}}\, exp\left(-\frac{1}{2}u^2\right) du \;. \tag{2.51}$$

The integral of this equation can not be expressed in terms of standard functions, however by using a special function erf, it can be written as:

$$F_L(u) = \frac{1}{2}\left(1 + erf\left(\frac{u}{\sqrt{2}}\right)\right) \;. \tag{2.52}$$

where $erf(x)$ is the error function:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int\limits_{0}^{x} e^{-t^2} dt \;. \tag{2.53}$$

From Equations (2.3), (2.48), and (2.49), the reliability of a system under TID effects can be finally described as

$$R_{TID}(z) = 1 - F_L(z) = 1 - \int\limits_{-\infty}^{z} f_L(z)dz = 1 - \int\limits_{-\infty}^{u} f_L(u)du = 1 - F_L(u) \;. \tag{2.54}$$

As described above, one can calculate the reliability of a specific device against TID effects if experimental data on its TID performance are available.

2.3.1. Reliability of voter against TID effects

The TID performance of FPGAs are independent of the size of the gate. Unlike SEEs, TID can be mitigated by shielding the components with, e.g., aluminum plates. By using thick aluminum plates within the allowable mass budget, the TID effects can be mitigated. The data on TID performance of the sample Flash-FPGA device RTProASIC is available in [35]. By taking the amount of reduction of the maximum frequency of the device due to TID effects as the measure, the reduction of 10 % is defined as a device failure through this analysis. According to the analysis results in [61], the approximate population mean and population standard deviation can be obtained as:

$$\sigma = 2.2249 \tag{2.55}$$

$$\zeta = 19.70 \tag{2.56}$$

From a past study, typical TID of LEO is estimated as up to around 1.5 krad/year for devices shielded with 3 mm thick aluminum plates [62]. Accordingly, from Equation (2.54) the reliability of the voter with RTProASIC against TID effects in 2 years is:

$$R_{V,TID}(t = 2years) = 0.99999 \approx 1 \; . \tag{2.57}$$

The value of the reliability is directly related with the curve of the error function, and becomes, e.g., $R_{V,TID}(t = 10years) = 0.98268$ and $R_{V,TID}(t = 15years) = 0.10410$. This indicates that the safe operation is assured for at least 10 years of mission life time, however, the system will fail most probably in the next some more years.

2.3.2. Reliability of node system against TID effects

The reliability of a single SRAM-FPGA node against TID effects can be calculated in the same way as above section. Because the node system is redundant, the total reliability of the node system against TID effects shall be analyzed on a case by case basis.

2-out-of-3 system

In case of 2-out-of-3 system, the three nodes shall be always available for the voting. In this sense, failure of one node out of the three caused by TID effects leads to the system failure. Accordingly, letting “NU” indicate the Node Unit, the reliability of the node system against TID effects can be described as:

$$R_{2of3,NS,TID} = {R_{NU,TID}}^3 \; , \tag{2.58}$$

where $R_{NU,TID}$ is the reliability of a node unit against the TID effects.

3-out-of-4 system

Similar to the case above, the reliability of 3-out-of-4 system against TID effects are:

$$R_{3of4,NS,TID} = {R_{NU,TID}}^4 \; . \tag{2.59}$$

Example with Xilinx Virtex-II Pro

From the Table 1.2 the TID limit of Virtex-II Pro devices can be estimated as 250 krad. By considering the analysis results in Section 2.3.1, the reliability of a single Virtex-II Pro node for the mission duration of 2 years can be easily estimated to be

$$R_{NU,TID}(t = 2years) \approx 1 \; . \tag{2.60}$$

Consequently, even though the reliabilities of the node system in each above listed cases differ, TID effects on Virtex-II Pro node system can be regarded as negligible, so that

$$R_{NS,TID}(t = 2years) \approx 1 \; . \tag{2.61}$$

2.4. Node system design

2.4.1. Trade-off on degree of redundancy

Based on the reliability modeling conducted in the above section, the unavailability of 2-out-of-m systems U_{2ofm} against the unavailability of identical units U_{NU} can be illustrated in Figure 2.4. In this figure, the U_{2ofm} with $m \in \{2, 3, 4, 5\}$ are illustrated to see the characteristics.

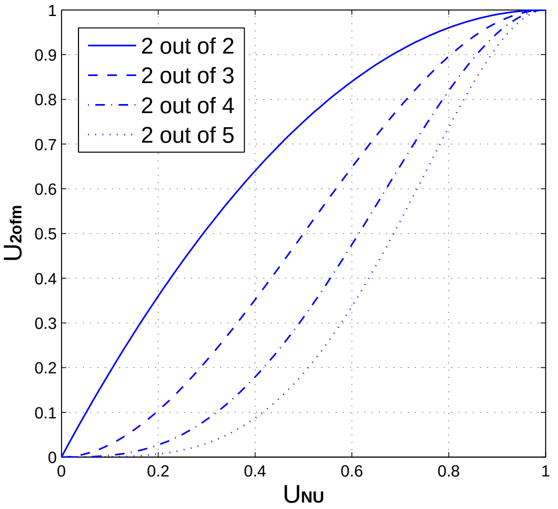


Figure 2.4.: 2-out-of-m systems unavailability

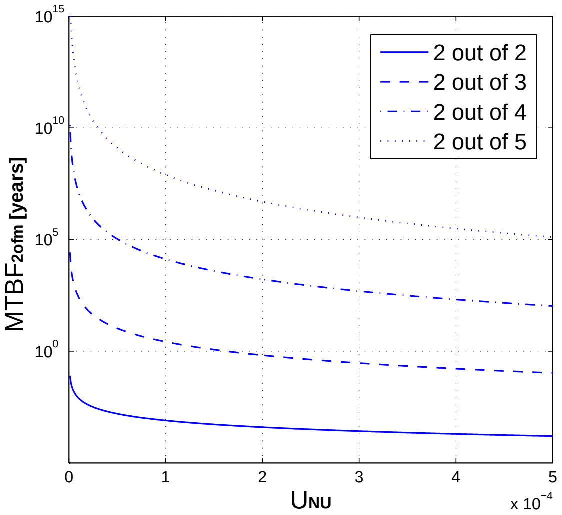


Figure 2.5.: 2-out-of-m systems MTBF

Given a unit unavailability, the higher the degree of redundancy (m) the lower the system unavailability, i.e., the higher the system availability. In other words, given a system unavailability, the higher the degree of redundancy, the higher the unit unavailability can be. This implies that in case the system reliability is fixed as a requirement, it can be achieved by more reliable components with a lower degree of redundancy, or by less reliable components with a higher degree of redundancy. It shall be noticed that the merit of using one component more becomes smaller if the m becomes larger. Because a higher value of m means a higher system complexity and a higher system development cost, trade-off shall be done so that the system fulfills the requirements with minimum number of redundancy. It is also worth noting that a higher redundancy plays important role in the region of lower system unavailability.

Another important aspect of the degree of redundancy is its effect on the mean time between failure. As described in Equations (2.23) and (2.44), one needs to specify the unit repair rate to obtain the relation between $MTBF_{2ofm}$ and U_{NU}. Because a shorter $MTTR_{NU}$ makes the node system more available, the design goal it to keep it as short as possible. Taking a realistic limit value of $MTTR_{NU} = 5\,\text{s}$ for SRAM-FPGAs, therefore $\mu_{NU} = 0.2$, the $MTBF_{2ofm}$ with $m \in \{2, 3, 4, 5\}$ against the unit unavailability can be calculated as illustrated in Figure 2.5 with y axis shown in logarithm. This $MTTR_{NU}$ includes reconfiguration of the FPGA and resynchronization with other nodes.

As Equation (2.44) indicates, the effect of U_{NU} on $MTBF_{2ofm}$ is exponential, and therefore in the region where the U_{NU} is small, the effect of the degree of redundancy is significant. Because there is a practical engineering limit on $MTTR_{NU}$, the effect of $MTTF_{NU}$ defines the U_{NU} by

Equations (2.16) and (2.17). Here, U is originally a function of t, however, the second term of Equation (2.16) is negligible for the selected μ_{NU}, assuming that unit failure rate λ_{NU} is much smaller than μ_{NU}. Therefore, the unit unavailability U_{NU} can be regarded as static. Notice that this is an approximation of safe side.

Based on the failure frequency analysis in Chapter 1 and obtained Tables A.1 and Table A.2, and by assuming a statistical independence between failure modes, the failure rate of the two selected devices Virtex-II Pro and Virtex-4QV can be summarized as listed in Table 2.1. In the table, the data of normal radiation condition and the worst radiation condition are summarized. Notice that only the failure rate of configuration memory and Block RAM are taken into account due to their higher failure rate than the others with the magnitude of 2 to 16. The corresponding unit unavailability and MTBF of 2-out-of-m systems with $m \in \{3, 4\}$ are also summarized in the table.

In the normal case, the node system mean time between failures of both devices with the redundancy degree of 3 are less than 2 months. This means that a system with this concept fails once a two months even in the normal radiation condition, which is not acceptable for space systems with years of mission life time. Compared to this, if the redundancy degree is set as 4 the node system MTBF for both devices exceed one century well beyond the practical mission life time. This deference comes from the fact, that in case of $m = 4$, one more component can fail, before the first failed component is repaired. This analysis suggests that the node system shall be designed with four or more redundant SRAM-FPGAs for a safe mission operation in the normal radiation condition. Adding one more component than necessary for the majority voting, can also offer the possibility to absorb a permanent failure of one node unit downgrading to 2-out-of-3 system.

In the worst radiation condition, however, the unit unavailability becomes about 1 for both of the devices indicating that the node system is not available at all. This worst radiation condition shall last only 5 minutes. In order to prevent malfunction of the satellite, the node system shall be completely turned off beforehand for the expected duration, letting the satellite system move into the safe mode. This analysis results summarize to what extent reconfigurable FPGAs can be used in the space environment.

2.4.2. Node system reliability after latest repair

With the assumption that the node system are repairable as good as new, following discussion holds. If the system unavailability (Equation (2.40)) and system failure frequency (Equation (2.44)) are available based on the unit MTTF and MTTR, using Equation (2.17) and

Table 2.1.: 2-out-of-m system $MTBF_{2of m}$ for selected SRAM-FPGAs with $MTTR_{NU} = 5$s

Parameters	Virtex-II Pro (XC2VP50)		Virtex-4QV (XQR4VFX60)	
Radiation Condition	Normal	Worst	Normal	Worst
λ_{NU}	9.56E-5	3.59E+2	8.31E-5	2.96E+2
U_{NU}	4.78E-4	≈ 1	4.15E-4	≈ 1
$MTBF_{2of3}$ [years]	0.116	≈ 0	0.153	≈ 0
$MTBF_{2of4}$ [years]	121	≈ 0	184	≈ 0

Equation (2.24) one can calculate the failure rate of the 2-out-of-m system $\lambda_{2ofm,NS}$:

$$\begin{aligned}\lambda_{2ofm,NS} &= \frac{1}{1-U_{2ofm,NS}} \cdot \nu_{2ofm,NS} \\ &= \frac{\{m(m-1){U_{NU}}^{m-1}(1-U_{NU})\}\mu_{NU}}{1-m{U_{NU}}^{m-1}-(1-m){U_{NU}}^{m}} \;. \end{aligned} \tag{2.62}$$

Finally, one can calculate the system reliability at t after the last complete node system repair at t' as:

$$R_{2ofm,NS}(t) = e^{-\lambda_{2ofm,NS}(t-t')} \;. \tag{2.63}$$

2.4.3. Effect of periodic preventive renewals

It is possible to reconfigure SRAM-FPGA units periodically during the normal operation. Following the discussion in [59] the point unavailability of periodic renewals at distance T can be written as:

$$U(kT+t) = F_L(t);\; 0 < t \leq T. \tag{2.64}$$

Since a statistical independence of successive unit lives is assumed, using probability distribution functions of work time F_W:

$$F_W(kT+t) = 1-[1-F_L(T)]^k[1-F_L(t)];\; 0 < t \leq T. \tag{2.65}$$

After differentiation using $\bar{F} \equiv 1-F$, the probability density function f_W can be obtained as:

$$f_W(kT+t) = [\bar{F}_L(T)]^k f_L(t);\; 0 < t \leq T. \tag{2.66}$$

In the exponential case with $f_L(t) = \lambda e^{-\lambda t}$, $\bar{F}_L(t) = e^{-\lambda t}$,

$$f_W(kT+t) = \lambda \cdot e^{-(kT+t)\lambda} \Leftrightarrow f_W(\tau) = \lambda \cdot e^{-\lambda\tau}. \tag{2.67}$$

This is obviously the same as Equation (2.8), therefore, periodic preventive renewals has no meaning. In the real case of SRAM-FPGA, it is usually not possible to detect every single SEE. In this sense, each node has always a potential risk, that it contains invisible SEE failure inside the configuration. Because of this, it is still helpful to introduce periodic preventive renewals.

2.5. Summary

The effects of TID on SRAM-FPGAs are negligible compared to those by SEEs. It is a good assumption that failures caused by SEEs on SRAM-FPGAs can be repairable as good as new. Therefore, as far as failures caused by SEEs can be detected, isolated and recovered (reconfigured), a node system based on SRAM-FPGAs can survive under space radiation environment and fulfill their functional requirements.

As one can see from the reliability analysis of repairable systems, the mean time to repair shall be short enough relative to the mean time to failure in order to keep the reliability (availability) sufficiently high. For this purpose, periodical check of failures shall be conducted with a sufficiently high frequency. In order to recover the failed unit for the voting again, the unit shall be reconfigured, synchronized to other node units, and initialized in terms of system variables. These processes shall be conducted within the required time frame. Because the unit unavailability is related with the system unavailability, the degree of redundancy influences the requirements of each unit. Higher reliability requirements can be fulfilled by employing a higher degree of redundancy. Consequently, the number of redundancy and mean time to repair of units are the main system trade-off variables, in order to mitigate radiation effects by SEEs on a node system based on given SRAM-FPGA devices.

The difficult point of the radiation effects analysis for SRAM-FPGAs is that in case of worst case scenario, the SRAM-FPGAs can not survive no longer than several seconds. However, obviously, these conditions happen only from time to time, and in case of nominal radiation environment, the radiation effect is lower than that of worst case ("Peak-5min" condition of CREME96 model) with four to seven orders of magnitude (see Appendix A). Due to this fact, the system shall be designed in such a way that it can survive worst case conditions moving into a special mode, shutting down the node system, and also shall be able to recover after the events.

Flash-FPGAs considered in this chapter revealed the feasibility of applying them to the voter under the assumption that the effects of SET and SEU on data flip-flops can be mitigated. Consequently, the important strategy for acquiring a high system reliability is to keep the voting logic as compact as possible, and make the part as reliable as possible by applying internal multi-module redundancy. Failed node units can be repaired, as far as this voting element survives.

Part II.

Demonstration with the small satellite Flying Laptop

This part describes the design and development of the proposed FPGA-based on-board computer architecture according to the conceptualized application and radiation mitigation methods in Part I. Within the scope of the thesis, the breadboard model and partially engineering model of the OBC hardware, as well as the control algorithm (hardware logic inside FPGAs) are developed. For the realization of the OBC, the system design of the target demonstration small satellite *Flying Laptop* is also conducted within the scope of the thesis. Firstly, Chapter 3 provides introduction of the small satellite *Flying Laptop* including the mission facts, scientific payload instruments, and required attitude determination and control components. The system design of the *Flying Laptop* satellite is summarized in Chapter 4 with an emphasis on the electrical architecture design. Chapter 5 summarizes the development of hardware components of the OBC of the *Flying Laptop*. Finally, Chapter 6 describes the development of the control algorithm of the OBC, which is implemented into the hardware logic inside the FPGAs. The attempt to implement all satellite control algorithms into FPGAs is new and has never demonstrated so far. This part provides complete information of the implementation method of the FPGA-based OBC. The developed breadboard model of the OBC serves as the platform for the further development and optimization of the control algorithm.

3. Flying Laptop satellite

The small satellite *Flying Laptop* is the target satellite on which the FPGA-based on-board computer, developed within the scope of this thesis, shall be demonstrated. In order to demonstrate the maximum high computational capability of the OBC, the satellite is equipped with very challenging scientific instruments and communication systems [63].

The small satellite *Flying Laptop* is the first satellite within the "Stuttgart Small Satellite Program" [64], in which the Institute of Space Systems (Institut für Raumfahrtsysteme: IRS) at the Universität Stuttgart is developing several small satellites aiming to launch a small spacecraft to the moon (*Lunar Mission BW1* [65], [66], and [67]) as the final goal. All these satellites are developed, built and operated by the IRS. The mission objectives of the *Flying Laptop* which is a cubic, 3-axis stabilized satellite with the edge lengths of about 600 mm × 700 mm × 800 mm and with a mass of about 120 kg are new technology demonstrations and scientific Earth observations. The space qualification of the reconfigurable FPGA-based on-board computer system is the one of the most important technology demonstrations. This OBC is not a test component but the main computing system of the satellite. All components are driven by this central OBC. The launch is planned by the Polar Satellite Launch Vehicle (PSLV) of Indian Space Research Organisation (ISRO) as a biggyback payload into a polar, sun-synchronous, low Earth orbit around 600 km. The mission lifetime shall be longer than 2 years. The mission parameters of the *Flying Laptop* are listed in Table 3.1.

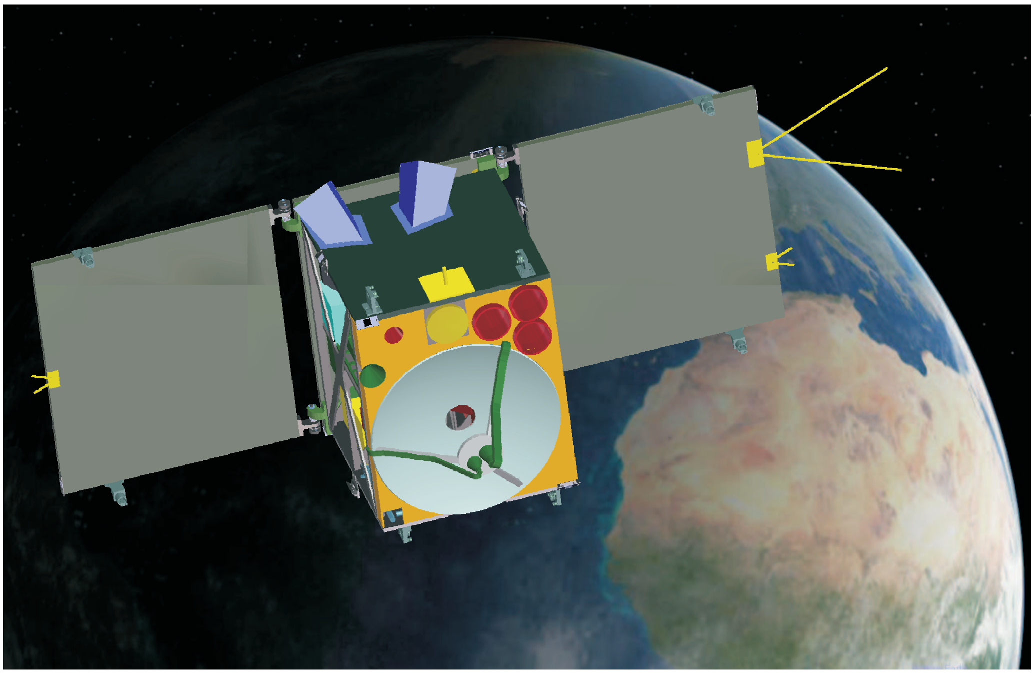


Figure 3.1.: *Flying Laptop* satellite, artist impression

This small satellite program was initiated in 2003 in order to establish a basis of space development at the university with the cooperation by regional industrial partners and academic institutions. To support the long-term development of spacecraft, an own clean room, a ground station, a thermal-vacuum chamber, an optical integration room were also installed. The "Space-Center Baden-Württemberg (Raumfahrtzentrum Baden-Württemberg)" is under construction at the time of writing, which further strengthens and supports the activities. A part of the technology demonstration mission of *Flying Laptop* is planned for the following spacecraft in the Stuttgart small satellite program.

Table 3.1.: *Flying Laptop* mission parameters

Parameters	Value
Launch Vehicle	PSLV (ISRO)
Orbit Type	Circuler, polar, sun-synchronous
Orbit Altitude	500 – 900 km
LTDN	09:30 - 11:00 h
Launcher Envelope	600 mm × 700 mm × 850 mm
Weight	< 120 kg
Mission Life Time	> 2 years

3.1. Flying Laptop project

The *Flying Laptop* project is a small satellite development project with several doctoral students being its main developers. Each doctoral student takes a responsibility for a specific aspect of the satellite development activities mostly based on a subsystem-basis. The contributions of this thesis to the project is developing the FPGA-based on-board computer as well as conducting the system design of the whole satellite system. This is necessary because all aspect of the satellite system, including electrical components/interfaces, power budget, operational concept/modes, data handling, link budget, and required size of hardware logic inside FPGAs etc. influence the functionality of the OBC. This section summarizes the contributions of each doctoral student.

Georg Grillmayer, who was responsible to the attitude control system, has established the requirements, defined the specifications, and developed the engineering model of the attitude control system. He developed mathematical attitude control laws for six attitude control modes and implemented them into the Mathworks MATLAB model [68], [69]. He also conducted orbit analysis [70], [71]. He has selected/developed sensors and actuators. He has implemented driver algorithms of those components into an FPGA development board and tested their functionalities with the EM assemblies. The translation of the MATLAB code into the hardware description language is done by both Grillmayer and Yasir Muhammad [72]. Grillmayer was responsible to the GENIUS and NEO detection experiments and organized the cooperations.

Sebastian Walz, who was responsible to the payload systems, has established requirements and defined specifications of the EM of the camera systems including the cassegrain mirror system. He has performed feasibility study on the Bidirectional Reflectance Distribution Function (BRDF) measurements and designed three camera systems in green, red, NIR and thermal infrared spectral regions as well as panchromatic in visible range [73]. He also conducted investigations on manufacturing the mirror by coated CFRP structure. EMs of these camera systems are under development at the time of writing.

Michael Lengowski, who was responsible to the structure and mechanisms and thermal control subsystems, has played a central role in designing every single structural component of the satellite. He introduced modular architecture to the satellite main structure classifying into core, payload, and service modules [74], [75]. Especially, the optical bench, on which all camera systems and star trackers are mounted, is designed and manufactured in such a way that the pointing accuracy and stabilities of the camera system fulfill the requirements. He also developed the electrical solar panel deployment mechanism and verified its functionality with laboratory models. Finally, he conducted thermal design of the satellite.

Albert Falke has installed a model-based development and verification environment at the IRS with the support of EADS Astrium GmbH (Chapter 7) [76], [77]. Based on the simulation and verification environment, he implemented software component models of the *Flying Laptop* satellite according to the hardware specifications. He also established the architecture of the simulation and verification facility including satellite and project databases and intellectual property management systems. He also installed a satellite operation facility based on the SCOS-2000 as well as a test case scripting engine based on the MOIS [78], [79], and [80].

3.2. Mission description

The *Flying Laptop* is the testbed for an on-board computer with a reconfigurable, redundant and self-controlling high computational capability, based on Field Programmable Gate Arrays. This technology shall meet the recent needs on applying new high density reconfigurable FPGA technologies for space use, which enables innovative approaches to architecture of OBCs, especially for demanding small satellite applications. As payload scientific instruments, *Flying Laptop* is equipped with three camera systems and two communication systems. Besides several technology demonstrations, *Flying Laptop* shall perform following scientific Earth observations:

- Bidirectional Reflectance Distribution Function (BRDF) measurement
- Multi-spectral Earth observation
- Precipitation measurement
- Atmospheric trace gas detection

The main interest of Earth observations by the camera systems is the BRDF measurements [81], [74]. BRDF is a mathematical function that quantifies the reflectance of a target as a function of the independent variables characterizing viewing and illumination angles, and of a set of variables determining the geometric and optical properties of the observed scene. For a scene with given properties, the reflectance observed varies with the angles of observation and illumination [81]. The BRDF helps to improve not only the classification accuracy of space images analyzing the vegetation of the Earth surface, but also accuracy of calculating albedo effects, calibrating instruments and stitching together space images from multiple overflights. The research on BRDF takes many forms. It can be measured using ground based measurement facilities, airborne scanner, or observation from spacecraft. In order to observe the target point from different observation angles, the *Flying Laptop* shall perform a target pointing mode, in which the body-mounted camera instruments are pointed toward the specific target position on the ground, through the fly-over path. The attitude control system of the *Flying Laptop* shall ensure that the same ground pixel remains in the image sensors and can be observed from different directions.

The two communication systems are used for precipitation (regional rainfall rate) measurements using dual-frequencies in Ka- and Ku-Band based on the attenuation difference in both communication signals. Also, the frequency tunable Ka-Band transmitter can detect the existence of atmospheric trace gases. In addition to this, the Ka-band system shall be able to perform high-speed communications up to 500 Mbps twice per day with ground stations. Because these communication instruments are also fixed on the satellite body, the attitude control system shall perform target pointing mode to conduct these experiments. Detailed mission objectives and their descriptions can be seen in [82].

3.3. Scientific payload instruments

The payload system is divided into two groups: camera systems and communication systems. The camera systems consist of three instruments: Multi-spectral Imaging Camera System (MICS), Thermal-infrared Imaging Camera System (TICS), and Panoramic Camera (PAMCAM), while the communication systems consist of Ka- and Ku-Band instruments.

3.3.1. Camera systems

The three types of payload camera systems are developed by Walz and his team within the project. MICS consists of three identical CCD array cameras with different interference filters in green, red and near infrared spectral regions with a bandwidth of 50 nm at half maximum [83]. The schematic of a MICS camera is illustrated in Figure 3.2.

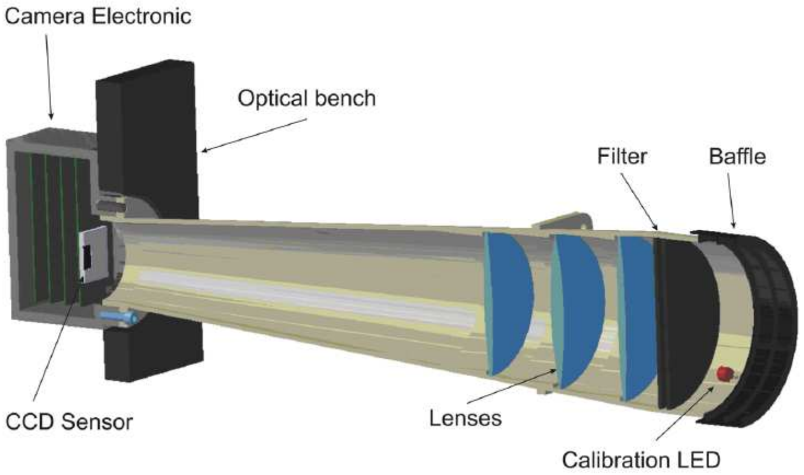


Figure 3.2.: Schematics of MICS [76]

The requirements of the MICS system is derived from the BRDF measurements. A medium geometrical resolution is pursued with the focus on radiometric accuracy. The filters are mounted on the top of the camera heads. As the sensor, Kodak KAI-1003M with 1024 × 1024 pixels is selected [84] with a 12-bit A/D converter. The on-board mass memory requirement is defined from the needs of BRDF measurement using MICS. During one BRDF measurement, each camera of MICS takes 125 images. The required size of mass memory is defined as at least 600 MB from this requirement. For the scientific evaluation of the data, LEDs are implemented inside the camera unit for in-flight calibration.

The TICS instrument will demonstrate the use of micro-bolometer detectors for infrared imaging with a relatively high geometrical resolution [85]. The goal is to achieve a ground sample distance of 100 m. Therefore, a cassegrain mirror system is used as optic. The sensor of TICS is a uncooled micro-bolometer Ulis UL 01011 with 320 × 240 pixels [86]. The primary mirror will work as antenna for the Ka-band system with the feed horn located at the hole of the secondary mirror. A schematic of the TICS camera is illustrated in Figure 3.3.

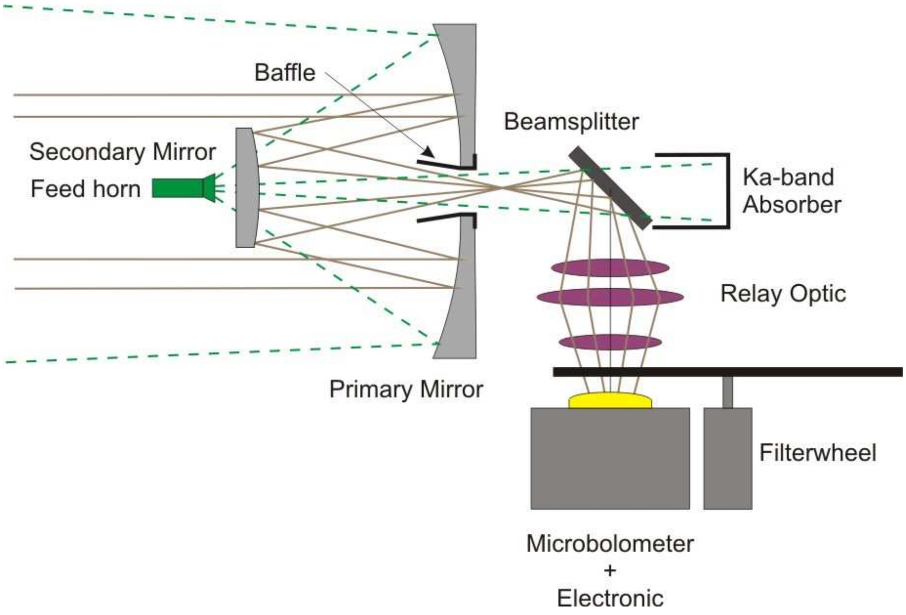


Figure 3.3.: Schematics of TICS [86]

The PAMCAM is used for public relation and live video transmissions. The high geometrical resolution will give a good overview of the observed scene, and the video mode will allow color live video with a frame rate of 5 Hz during a fly-over. The sensor is a commercial FillFactory IBIS5A-1300 CMOS sensor with 1280 × 1024 pixels with a Bayer color filter array. The characteristics of the camera systems are summarized in Table 3.2. The values are based on a reference orbit with an altitude of 600 km.

Table 3.2.: Characteristics of scientific payload camera instruments

Instruments	Spectral band	Wavelength [nm]	GSD [m]	Swath width [km]	System
MICS	Green	530 - 580	25	25	Optical
	Red	620 - 670	25	25	Optical
	Near infrared	820 - 870	25	25	Optical
TICS	TIR	8000 - 12000	100	32	Cassegrain
Pamcam	Panchromatic	400 - 700	200	250	Optical

3.3.2. Scientific communication systems

The characteristics of the scientific communication systems are listed in Table 3.3. The signals in Ka and Ku-Bands are used for scientific Earth observations. The Ka-Band system is capable of bi-directional communication with maximum down-link baud rate of 500 Mbps and up-link with up to 100 Mbps. This feature enables high-speed bi-directional communication on the platform of a small satellite. This communication system ensures down-link of all acquired image data within a single ground contact. The Ku-Band is only for the scientific experiment and no data transfer through Ku-band is foreseen.

Table 3.3.: Characteristics of scientific payload communication instruments

Band	Channel	Frequency [GHz]	Baud rate [Mbps]
Ka-Band	High Power Down-link (200 W)	20.2 - 21.2	500
	Low Power Down-link (17 W)	20.2 - 21.2	-
	Up-link	29.5 - 30.0	100
Ku-Band	Down-link	10.0	-

All of these instruments described above shall be driven by the OBC, and therefore the electrical interface design shall be performed throughly. Especially, image processing of the camera systems and signal processing for the high-speed communication systems are challenging missions. Detailed implementation of OBC is described in Chapter 5 and Chapter 6.

3.4. Attitude control system

The Attitude Control System (ACS) is developed by Grillmayer and his team within the project. The requirement of the ACS is to provide the satellite with desired attitude control capabilities. Based on the requirements coming from scientific payload instruments, ACS components are selected as listed in Table 3.4 [87].

Table 3.4.: Quantity of attitude control system components

ACS Instruments	Quantity
Reaction Wheels	4
Magnetic torquers	3
Magnetometers	2
GPS unit (GPS receivers)	1 (3)
Star Tracker (camera head units)	1 (2)
Fiber Optic Gyros	4
Sun sensor unit (sun sensors)	2 (16)

ACS hardware components are shown in Figure 3.4. The ZARM-Technik AMR-magnetometer is a microcontroller based 3-axis magnetometer with digital output. The C-FORS fiber optic gyros from Litef are arranged in a tetrahedral configuration, to allow for a single failure. The micro-Advanced Stellar Compass from the Technical University of Denmark consists of two camera head units connected to a data processing unit. The star tracker provides an attitude knowledge

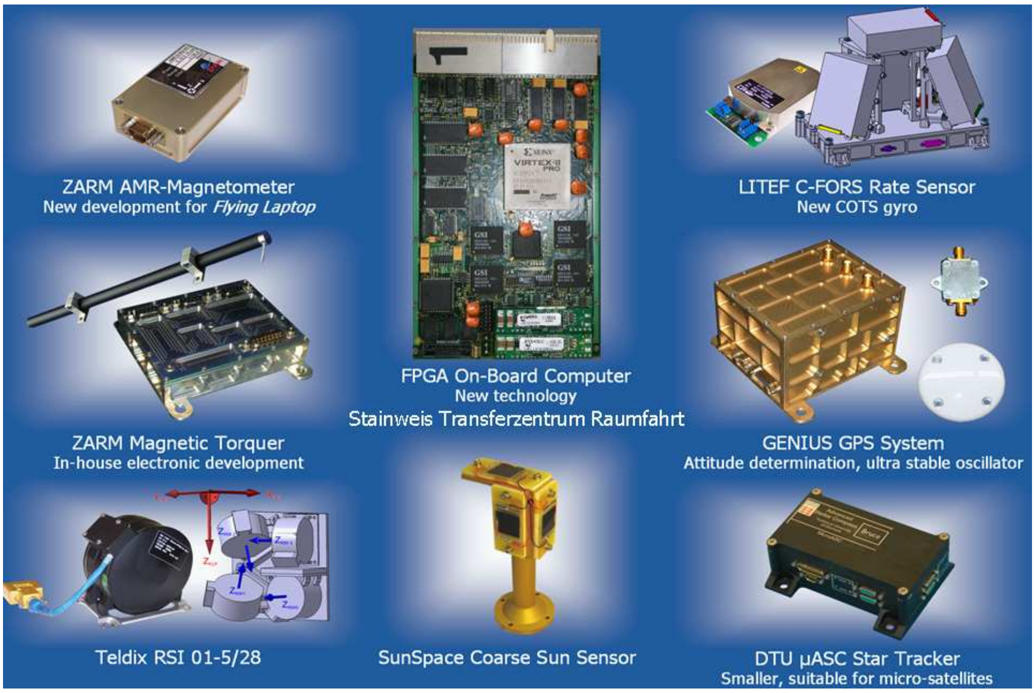

ZARM AMR-Magnetometer
New development for *Flying Laptop*
FPGA On-Board Computer
New technology
Stainweis Transferzentrum Raumfahrt
LITEF C-FORS Rate Sensor
New COTS gyro
ZARM Magnetic Torquer
In-house electronic development
GENIUS GPS System
Attitude determination, ultra stable oscillator
Teldix RSI 01-5/28
SunSpace Coarse Sun Sensor
DTU µASC Star Tracker
Smaller, suitable for micro-satellites

Figure 3.4.: Attitude control system hardware components

of 7”. The GENIUS (GPS Enhanced Navigation Instrument for the Universität Stuttgart micro-satellite) system, which is based on three COTS Phoenix GPS receivers, is developed as an experiment for accurate attitude determination using GPS [88]. The three antennas are placed at three corners of the middle solar panel in an L-like arrangement. Furthermore the receivers are synchronized via an ultra-stable 10 MHz oscillator. Each two of sixteen sun sensors based on solar cells are mounted at eight different positions as a redundant pair. The sensor voltages are digitized and sent to the OBC. Four reaction wheels from Teldix are running in a hot redundant tetrahedral configuration. Each wheel has an angular momentum capacity of 0.12 Nms and a reaction torque of 5 mNm. Three ZARM-Technik magnetic torquers with a linear dipole moment of 6 Am^2 are connected to a power electronics box. The whole system is single redundant. A detailed description of ACS components can be seen in [89]. All of these sensors and actuators are connected to the OBC.

3.4.1. Attitude pointing modes

The *Flying Laptop* satellite shall be able to perform different types of attitude pointing modes. The pointing modes of the *Flying Laptop* can be classified into three modes: inertial, nadir, and target pointing modes [90].

Target pointing mode

In the target pointing mode the payload instruments are pointed toward a fixed point on the Earth’s surface. This attitude control mode is utilized for BRDF measurements and high-speed communication with a ground station. This mode enables an observation of the target point from a wide range of angles. For the BRDF measurements, attitude pointing accuracy shall be better than 150" with a pointing knowledge of better than 7". The maximum slew rate for this mode is about 1°/s. For BRDF measurements, the angle of observation is defined as +/- 60° off nadir and +/- 30° roll as illustrated in Figure 3.5.

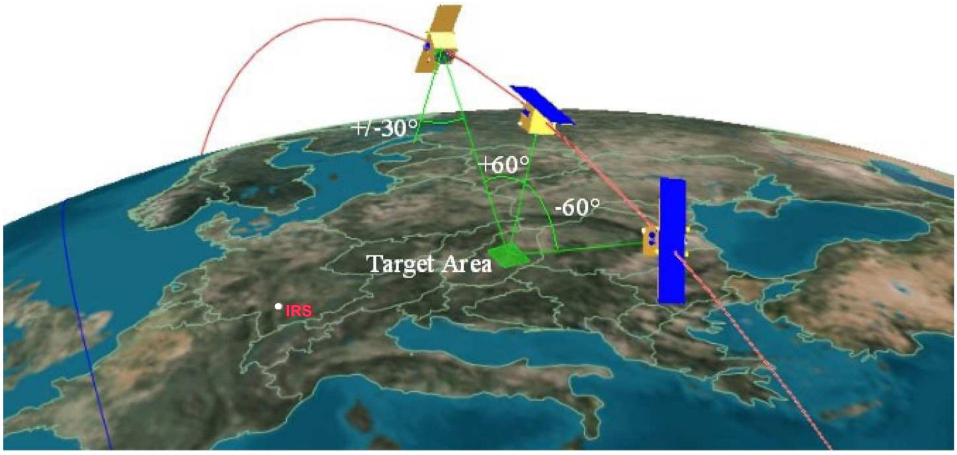


Figure 3.5.: Target pointing mode

Nadir pointing mode

In this mode the satellite body is aligned in the direction of the Earth, i.e., the +Z vector of the satellite's body coordinate system (see Chapter 4) is pointed toward the Earth center (Figure 3.6), thus the angular rate remains constant. This mode is mainly utilized for the multi-spectral Earth observations. Though the *Flying Laptop* is not equipped with an Earth sensor, data from GPS and star trackers enables transformation of coordinate systems.

Inertial pointing mode

In the inertial pointing mode, the satellite attitude is fixed relative to the inertial frame, and payload camera systems can be oriented to a given attitude relative to stars (Figure 3.7). The star trackers are used for experimental Near Earth Objects (NEOs) detection in this pointing mode.

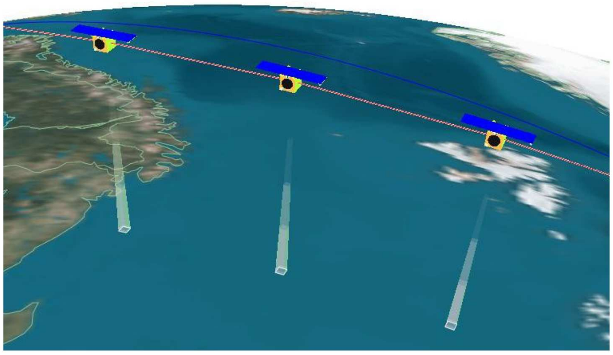


Figure 3.6.: Nadir pointing mode

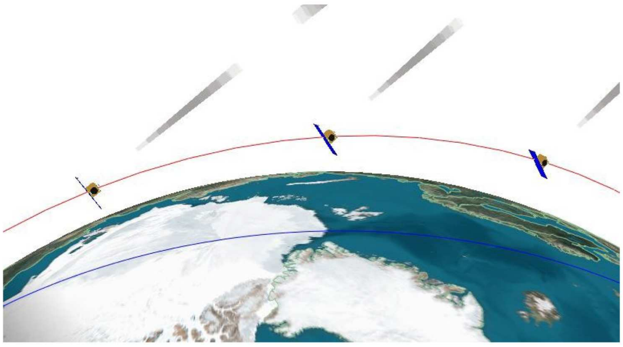


Figure 3.7.: Inertial pointing mode

4. System design of the Flying Laptop

The small satellite *Flying Laptop* is a challenging satellite with a variety of instruments, which shall demonstrate the capability of small satellites. The most important mission objective of the *Flying Laptop* satellite is the demonstration of the application of reconfigurable FPGAs in the space environment. The FPGA based OBC shall demonstrate its high computational capability with the flexible and parallel processing features. As the size and mass of the satellite is limited, the on-board computing system of the *Flying Laptop* shall be a centralized system connected with all of the peripheral components with a point-to-point star configuration. The OBC shall be capable of controlling the payload instruments as well as performing attitude determination and control, power management, thermal management, and telecommunications. In order to make this concept possible, the satellite system shall be throughly designed.

Within the scope of this thesis, the system design of the *Flying Laptop* has been conducted. An effective system design as well as an effective project management method for small satellite projects are proposed by author and applied to the *Flying Laptop* project [91]. Based on the requirements of scientific instruments, experiments, and attitude control performance, the satellite system was designed/developed. During this system design activity, operational, attitude control, mechanical and thermal, and electrical aspects are iteratively designed. As the consequence of the system design activities, satellite on-board components are identified as listed in Table 4.1. The mechanical configuration of the satellite is illustrated in Figure 4.1 together with the body coordinate system.

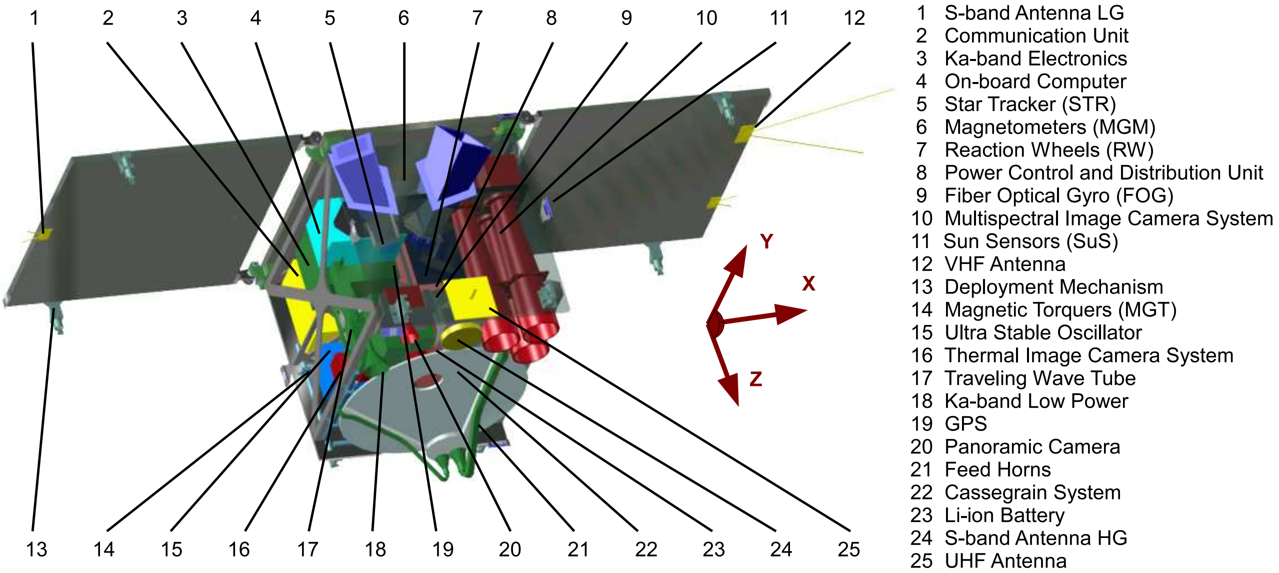


Figure 4.1.: *Flying Laptop* satellite mechanical configuration

A system design activity is multidimensional. The difficult challenge in the electrical architecture design of the *Flying Laptop* is that all peripheral electronics shall be connected with the

Table 4.1.: *Flying Laptop* system components

Sub-system	Electrical Components	Qty.	Interface	Qty.	Max. baud rate
Payload	MICS Green, Red, NIR	1 each	I^{2}C	1	100K
			LVTTL	1	16K
			LVDS	2	80M
	TICS	1	LVTTL	2	9.6K
			LVDS	5	14.3M
	Panoramic Camera System	1	LVDM	2	200M
	Ka-band TX	1	LVDM	2	250M
	Ka-Band RX	1	LVDS	1	20M
	Ku-band Transmitter	1	LVDS	1	1M
Attitude Control System	Reaction Wheels	4	RS-422	1	9.6K
	Magnetic Torqure Unit	2	I^{2}C	1	100K
	Magnetometers	2	RS-422	1	57.6K
	GPS Unit	1	RS-422	3	57.6K
			LVTTL	9	100K
	Star Tracker Unit	1	RS-422	3	115.2K
	Fiber Optic Gyro Unit	1	RS-422	4	2M
	Sun Sensor Unit	2	I^{2}C	1	100K
Telemetry, Tracking & Command	S-Band TX	1	LVDS	2	2.2M
	S-band RX	2	LVDS	2	250M
	UHF-Band TX	1	LVDS	2	38.4K
	UHF-band RK	2	LVDS	2	250M
	VHF-Band RX	1	LVDS	2	9.6K
Power Supply System	PCDU	1	I^{2}C	3	100K
			LVTTL	3	1K
	Battery Unit	1	-	-	-
	Solar Panels	3	-	-	-
Thermal Control System	Housekeeping Unit	1	I^{2}C	2	100K
	Heaters	8	-	-	-
Structure & Mechanisms	Solar Panel Depl. Mech.	2	-	-	-
Command & Data Handling	USO	1	TTL	3	10M
	On-board Computer	1	-	-	-

single OBC, while the OBC shall be as compact and light as possible to be integrated into the small satellite. Another aspect of the electrical interface design is that the actual mechanical connection shall be realizable, i.e., the spatial margin shall be large enough for connectors to be mated/unmated keeping the physical loads remain under certain limits. Furthermore, the *Flying Laptop* project aims to use as many COTS components as possible to reduce the development cost with applying a single-failure-tolerance concept to critical components. Also it introduces Proto-Flight Model (PFM) philosophy combined with partial Engineering Models (EMs) and Breadboard Models (BBMs). The purpose of this chapter is to provide a sufficient background for the initiation of the OBC design. Due to the limited space and to keep the discussion not diverged, the focus of this chapter is put on the electrical and operational designs. The system electrical architecture and the operational concept, including commanding, data handling, and contingency operation are established. Communication system and power supply system are designed and power balancing design of the satellite is conducted. The detailed implementation of subsystems are summarized in Appendix C.

4.1. Electrical architecture design

The quantity of identified electrical component units, as well as type, quantity and maximum baud rate of their electrical interfaces between the OBC are summarized in Table 4.1. The OBC shall be equipped with all of these interfaces so that it can communicate with and control them. As listed in the table, the required interface types are TTL, LVTTL, I^2C, RS-422, LVDS, and LVDM. Though LVDM belongs to LVDS, it is differentiated from it in terms of the higher communication speed grade. Taking the redundancy and reliability issues into account, all electrical interfaces between components have been designed. In the scope of this electrical interface design, the mechanical design of the actual harness system is also taken into account. Due to the small size of the satellite body, the integration of all components into the satellite can raise serious problems. In order to prevent those kind of problems, the harness cabling system is designed with a help of a real-size mock-up model. Indeed, the harness system is an important element in the overall system design for spacecraft mechanisms. The mass ratio of harness relative to the satellite mass can rise up to 10 % and the well-designed harness cabling reduces the mass ratio of harnesses, resulting in considerable weight saving, and makes a great profit for system level trade-off.

The electrical interface design of the *Flying Laptop* introduces following design tools:

- Electrical interconnections block diagram
- Input/Output diagram
- Mock-up model
- Harness cabling plan

Each item of these is described below in detail.

4.1.1. Electrical interconnections block diagram

According to the functional analysis [91], all identified functions are allocated to components. Based on the functional requirements, the interconnects between these components were defined. This design has been done by means of an electrical interconnection block diagram (Figure 4.2). This diagram summarizes the existence of electrical interconnections between all electrical components of the satellite system and provides the overview of the harness architecture. This diagram also illustrates the type of interconnections such as signal lines, power lines, analogue signal lines, and pulse signal lines and enables identification of each single interface.

4.1.2. Input/Output diagram

An I/O diagram defines electrical input and output interfaces of an electrical satellite equipment which is identified in the electrical interconnections block diagram. In this way, every single electrical interface between components including power lines can be designed and graphically represented. I/O diagrams also allocate input and output lines to actual physical element (pins) of assigned connectors. An exemplary I/O diagram of magnetic torquer driver electronic board is illustrated in Figure 4.3. This diagram defines, for example, the “pin 3” on the “connector 1” of the electronic board is connected with the “pin 1” on the “connector 3” of the Power Control and Distribution Unit (PCDU).

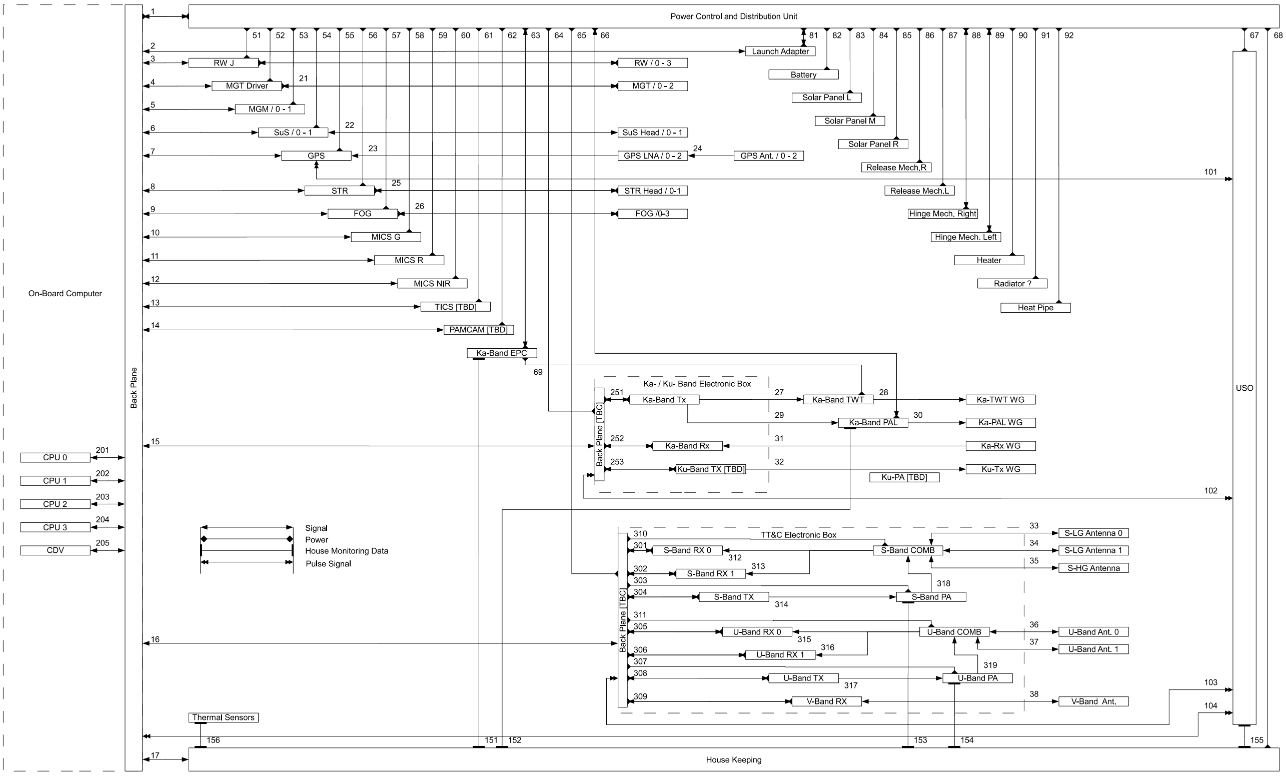


Figure 4.2.: Electrical interconnection block diagram

4.1.3. Mock-up model development

One of the most important features in designing the harness system of the *Flying Laptop* is the development of its mock-up model. Because of the model philosophy of the *Flying Laptop*, no

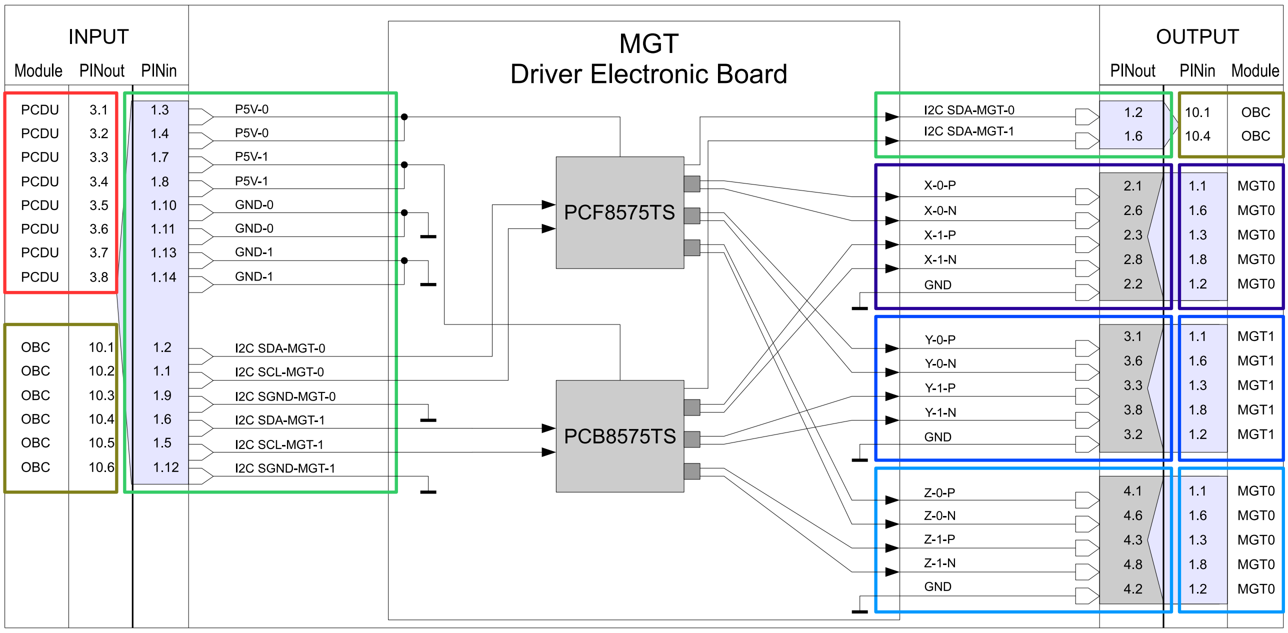


Figure 4.3.: Input/Output diagram (magnetic torquer unit)

complete engineering model of the whole satellite is built and therefore, it is quite meaningful that a mock-up model in original size is developed. This is a significant merit of developing a small satellite [92]. The mock-up model of the *Flying Laptop* satellite is shown in Figure 4.4.

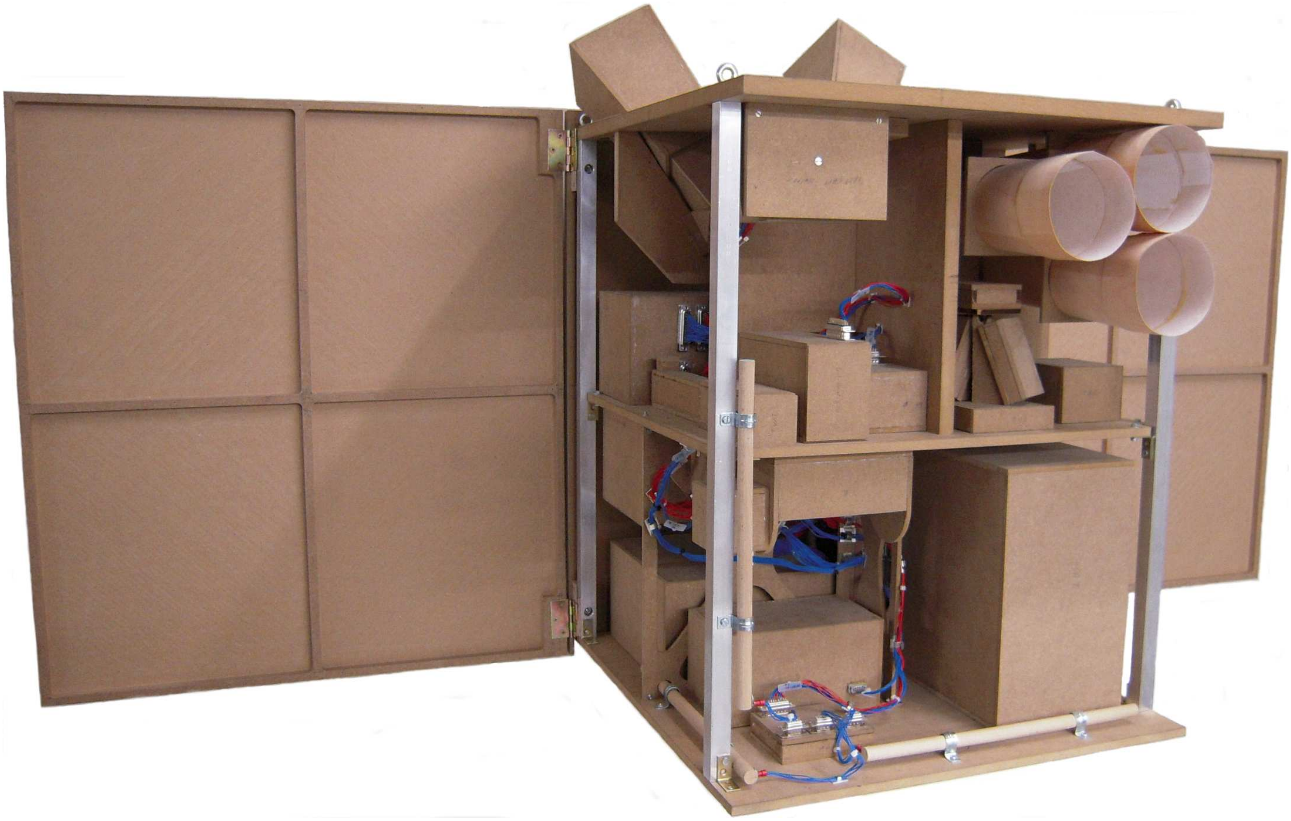


Figure 4.4.: Mock-up model of *Flying Laptop* satellite

This mock-up model enables to build the harness layout plan in consideration of the 3-dimensional effects and the integration feasibility. In addition, the model helps to propose the modification of the satellite's internal structures so that they can be adapted to harness cabling relevant aspects, such as holes and position of the cable brackets. The amenities of the mock-up model are:

- cables can be easily grouped into harness units, after once they are laid on the model,
- the accessibility of the harness, such as insertion, extraction and tightening of the screws can be evaluated,
- the capability and flexibility of mounting/demounting components can be assessed, and
- the correct cable length can be measured and the mass of harness system can be estimated accurately.

In addition to these, satellite integration procedures can be planned by using this model.

4.1.4. Harness cabling plan

The connectors with pin allocation in the I/O diagram are actually physically connected with the help of mock-up model. During this activity, the cabling path of every single cable/cable

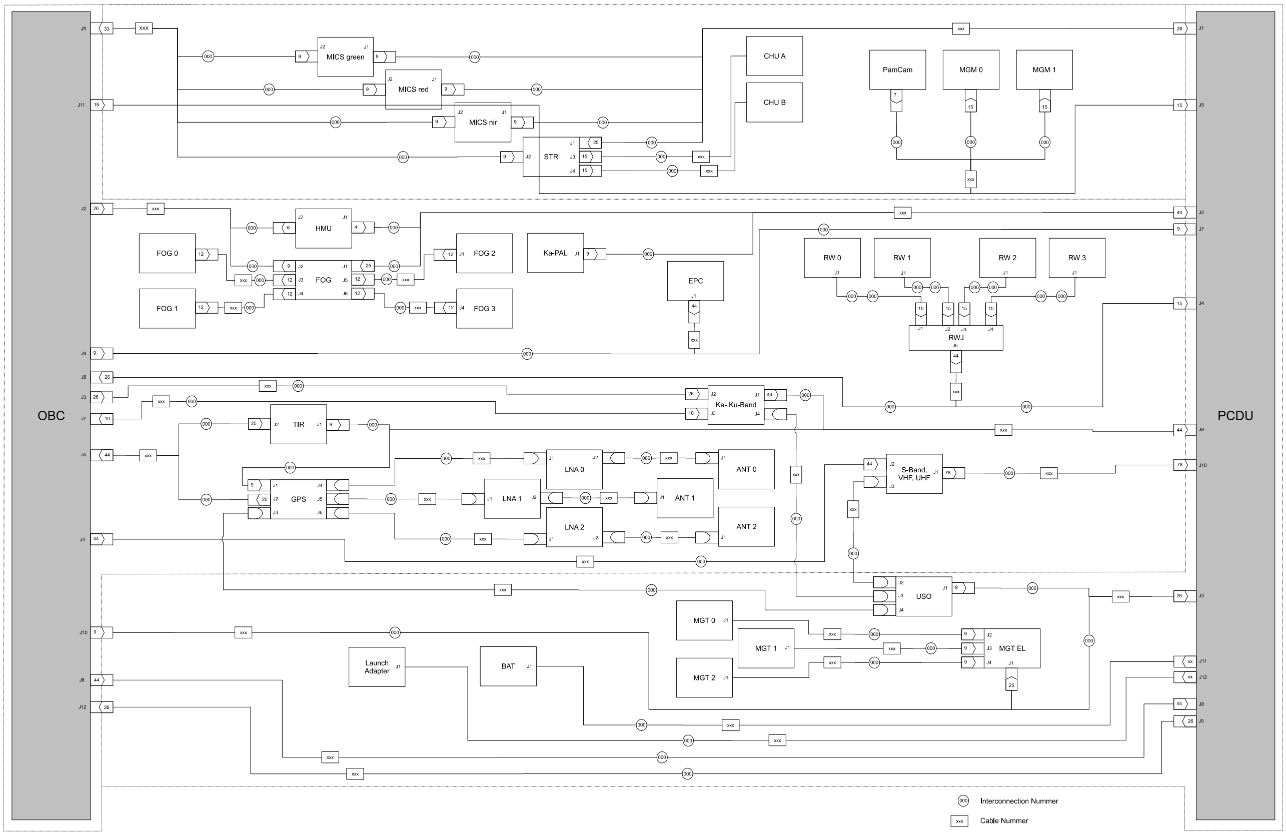


Figure 4.5.: Harness cabling plan

unit is optimized in terms of length, integration capability, wiring radius, physical loads, and consideration on electromagnetic compatibility (EMC) effects. After defining the cable paths, those cables which are on the same or partially same path are united into one harness unit, taking the integration and assembly capability into account. Consequently, a cabling plan can be obtained as illustrated in Figure 4.5.

4.2. System electrical architecture

The designed system electrical architecture is illustrated in Figure 4.6. The detailed design of the OBC is described in Chapter 5. High-speed communication interfaces are implemented as differential signals, such as RS-422, LVDS, and LVDM. The power supply lines are also illustrated in the figure. To reduce the development cost as much as possible, COTS components are used with redundancies. Sun sensor units and magnetic torquer units are organized in cross coupling configurations with redundant low speed I^2C communication interfaces. I^2C interfaces of PCDU are also triple redundant. Most of the high-reliability components are not in redundant as the result of cost-to-reliability trade-off. Otherwise, other components are designed based on a single-failure-tolerant design concept. The detailed electrical interface description of the OBC of the *Flying Laptop* is summarized in [93]

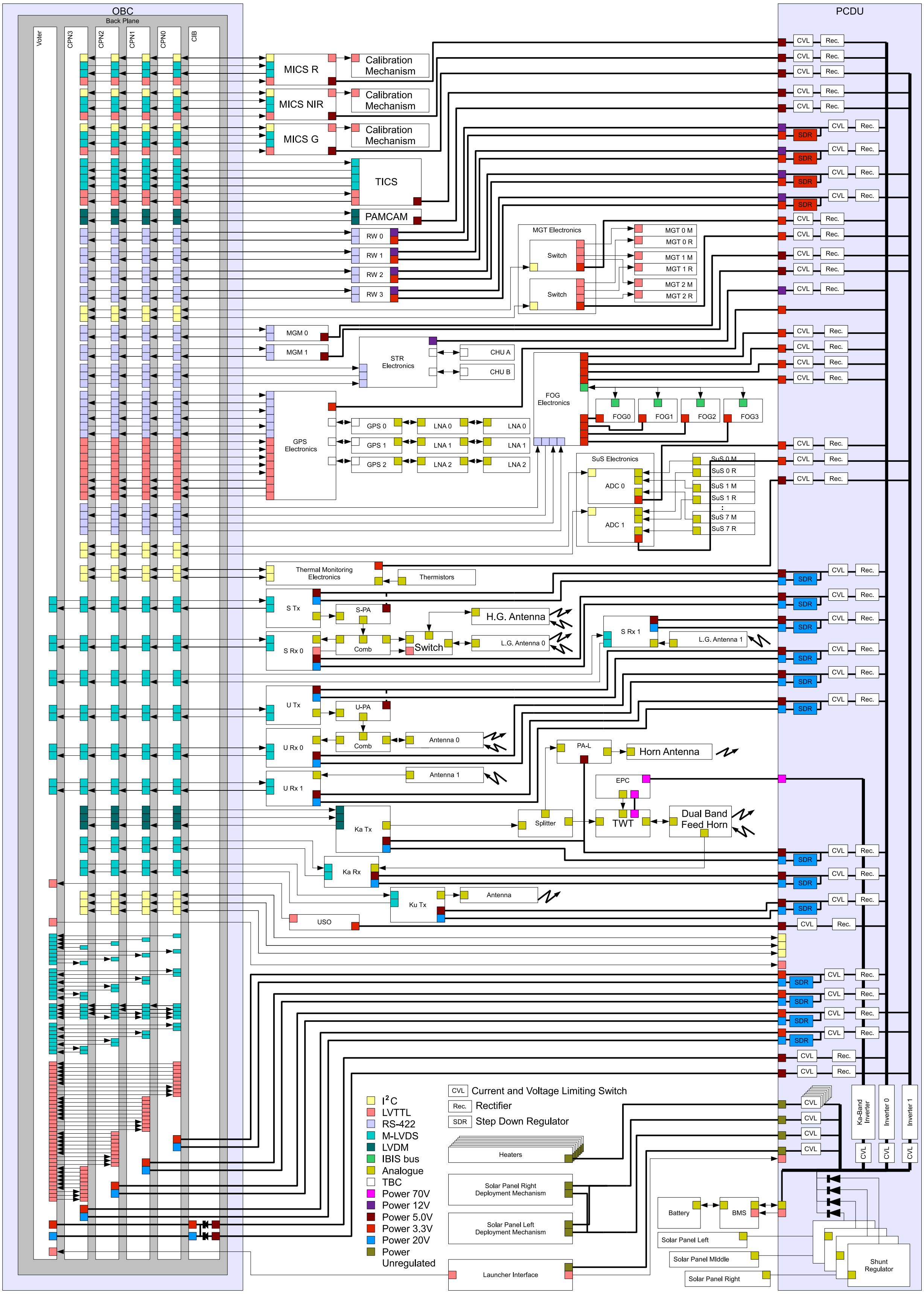


Figure 4.6.: *Flying Laptop* system electrical architecture

4.3. Operational design

In this section, the operational design of the *Flying Laptop* satellite is briefly summarized in terms of the operational mode, commanding concept, data handling, communication, and contingency operations. The launch of the *Flying Laptop* is planned by the PSLV of the ISRO as a piggy-back launch. Because of this, the satellite system shall be designed in such a way that the satellite can fulfill all mission requirements within a certain orbit envelop. This is illustrated in Figure 4.7 [70]. Seven reference orbits are defined for orbit simulations and system characterization (Table 4.2). It is determined by past research that the lower orbits No. 4 & 5 are the thermal hot cases and the higher orbits No. 6 & 7 are the cold cases [83].

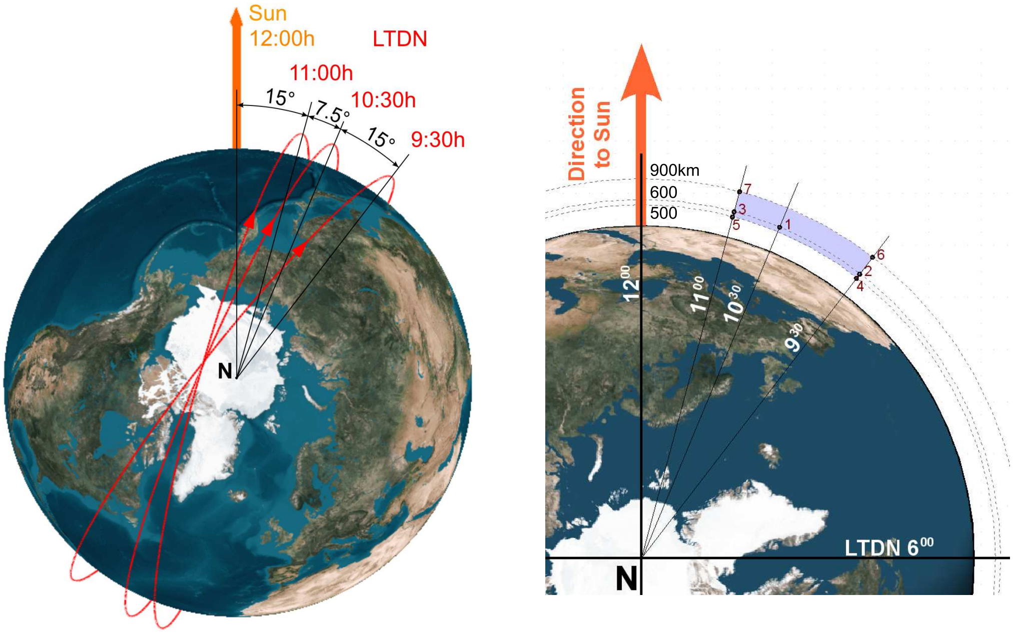


Figure 4.7.: Orbit envelop and reference orbits

4.3.1. Operational design concept

In order to achieve *Flying Laptop*'s wide-ranging mission objectives, the operation of the satellite needs to be supported by a discrete time-to-time planning rather than a continuous and periodical operational planning, partly because most of the missions are single-shot missions. For example, a BRDF measurement of a specific target at a specific time shall be planned by analyzing the availability of the star trackers according to the satellite position, target position, and sun incident angle. The operational plan shall be flexible and adaptable in the late mission phase even after the launch. In order to achieve this operational concept, the commanding function shall support any kind of combination of satellite operations. In this way, the mission operation plan can be continuously updated in consequence of individual mission/experiment planning. The other aspect of the difficulty in the mission planning of the *Flying Laptop* is that its on-board computer is based on the FPGA technology, which means there is no traditional processor aboard. The functionalities which are usually implemented into software on other satellites need to be replaced with a complex hardware logic realized in the FPGA chips. The

Table 4.2.: Parameters of reference orbits

Orbit Number	Altitude [km]	LTDN [hh:mm]	Inclination [°]	Period [min]
1	600	10:30	97.79	96.69
2	600	09:30	97.79	96.69
3	600	11:00	97.79	96.69
4	500	09:30	97.40	94.63
5	500	11:00	97.40	94.63
6	900	09:30	99.03	102.99
7	900	11:00	99.03	102.99

operational plan shall be designed so that, on the one hand, the strong capability of parallel processing of the FPGA can be maximized and, on the other hand, the operability of the satellite system by telecommanding can ensure trouble-free conduction of every single mission.

4.3.2. Operational mode design

Generally, satellite peripherals can be grouped into several subsystems. That means the satellite system can be decomposed into several subsystems and those subsystems can be further decomposed into components. In this way, a hierarchical relationship between satellite system, subsystems, and components can be established as listed in Table 4.1. In order to utilize the greatest benefit of the parallel processing features of FPGAs, each subsystem of *Flying Laptop* is controlled completely in parallel to each other. This concept is enabled by the allocation of a state-machine to each of the subsystems. A state-machine is a diagram which illustrates the relationship between several modes and the transition conditions between them. Consequently, a system mode can be described as a combination of subsystem modes. From the technical point of view, these application algorithms can be implemented as completely parallel processes in terms of hardware logic design within FPGAs. These parallel processes are controlled by individual commands. Detailed implementation method of hardware logic inside FPGAs is described in Chapter 6.

The designed system level state-machine is illustrated in Figure 4.8. After the orbit insertion, satellite system goes into safe mode through initial sequence mode. System can be switched to the idle mode from the safe mode only by telecommands. All scientific experiments are assigned to the active operation mode and can be initiated from idle mode. In safe mode it shall be ensured that the battery can be charged up to full by using reliable and minimum attitude determination and control components. The state-machine of the ACS is illustrated in Figure 4.10 and that of TT&C (Telemetry, Tracking & Command subsystem) in Figure C.1.

4.3.3. Commanding concept

The commanding concept of the *Flying Laptop* differs based on the operational phase. At the beginning of the mission life time, the satellite is operated only by telecommands without any on-board autonomy. A sample scenario is described in Table 4.3. This is therefore performed by sending telecommands one by one. Each transition number represents the corresponding telecommand. The commands are time-tagged and executed at the defined on-board real time.

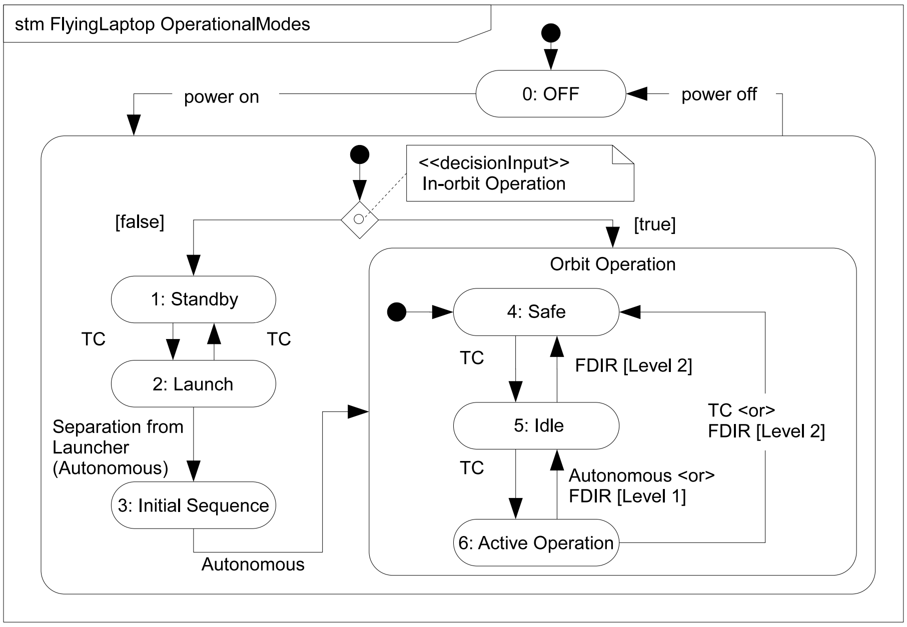


Figure 4.8.: *Flying Laptop* satellite system state-machine

At the later operational phases, the autonomy level of the telecommands can be raised. In this case, the satellite uses databases with sets of commands, which are executed by a single telecommand. For the sample scenario described above, a single telecommand "BRDF measurement" with appropriate parameters, such as target positions and camera settings, is up-linked.

Table 4.3.: Mission operation sample scenario (BRDF measurement)

Transition Number	System/Subsystem	State/State Transitions
0	System	Idle mode
1	System	Active operation mode
2	P/L (cameras)	Operation mode (Idle mode)
3	ACS	Nadir pointing mode
4	ACS	Target pointing mode
5	P/L cameras	Operation mode
6	ACS	Nadir pointing mode
7	P/L cameras	Idle mode
8	System	Idle mode
9	P/L (cameras)	Off mode (Off mode)
10	ACS	Idle mode

4.3.4. Data handling

The data to be handled on-board the satellite is composed of housekeeping and scientific data. Housekeeping data is continuously collected over time and constitutes a negligible part of the total data volume, while scientific data amounts to notable volume depending on the payload sensors, their usages, and the downlink concept. The design driving application for the on-board storage capacity is the BRDF experiment as described in Section 3.3.1.

For other possible Earth observation applications such as continuous Earth surface scanning requires much more capacity. For this application MICS generates data at 1.6 MB/s, TICS at 0.05 MB/s and PAMCAM at 0.06 MB/s, which add up to about 1.7 MB/s. Assuming a continuous scanning in nadir pointing except in eclipse, the amount of uncompressed raw data adds up to about 6.5 GB per orbit. The storage of such a data volume is expensive in terms of mass, spatial requirement, power consumption, and price. The actual storage size of the *Flying Laptop* is defined as 4 GB. Possible intelligent on-board data handling was investigated in [94].

4.3.5. Communication structure

The communication channels of the *Flying Laptop* are listed in Table 4.4 with their baud rate and operational concept. The primary communication channel of the satellite is the S-band. The LG and HG stand for Low Gain and High Gain, respectively. The LG down-link channel uses omni-directional antenna and can be used in case the satellite does not perform pointing attitude control toward the ground station. The patch antenna for the HG down-link channel is mounted on the same surface as the camera systems and can be used only when the satellite performs pointing attitude control. Unless the system is in safe mode, the satellite is able to perform HG communications. The UHF and VHF channels are intended for the cooperation with other universities and amateur band communities and used as back-up channels of the S-band. The Ka-band up and down-link is an experimental channel and can transmit up to 500 Mbps in down-link. Detailed link budget calculation, system characterization, and state-machine of the TT&C subsystem of the *Flying Laptop* are described in Appendix C.1.

Table 4.4.: Communication channels of *Flying Laptop*

Channel	Baud Rate [bps]	Modulation	Operational Concept	Min. Data Volume [Mbyte/day]
VHF Up	19.2 K	BPSK/FSK	On demand	30.87
UHF Up	4.8 K	FSK	Periodical/On Demand	7.718
UHF Down	38 K	(O)QPSK	Periodical/On Demand	61.10
S Up	19.2 K	FSK	Primary	30.87
S Down LG	150 K	OQPSK	Primary (nondirectional)	241.2
S Down HG	2.2 M	OQPSK	Primary (directional)	1900.8
Ka Up	10 M	QPSK	Max. 2 times/day	7320.0
Ka Down	< 500 M	OFDM	Max. 2 times/day	73200.0

4.3.6. Contingency operation

The contingency operation of the *Flying Laptop* is classified into two cases: contingency level 1 and 2. In case of level 1 contingency, the satellite system goes into idle mode. For example, in case of insufficient battery charge for a planned operation, the satellite goes into idle mode. The level 2 contingency represents severe contingency cases in which, for example, a primary sensors for idle mode are broken and the idle attitude control becomes impossible. In this case, the satellite system goes into safe mode. During the safe mode, the causes of the contingency are analyzed and identified on the ground basis before commanded back to idle mode.

4.4. Attitude control

4.4.1. Attitude control modes

Due to the mission objectives, the *Flying Laptop* shall perform three pointing modes: inertial pointing mode (3), nadir pointing mode (4), and target pointing mode (5), as illustrated in Figure 4.9. In addition to these, ACS performs three more control modes to support system level mission operations, which are detumbling mode (0), safe mode (1), and idle mode (2).

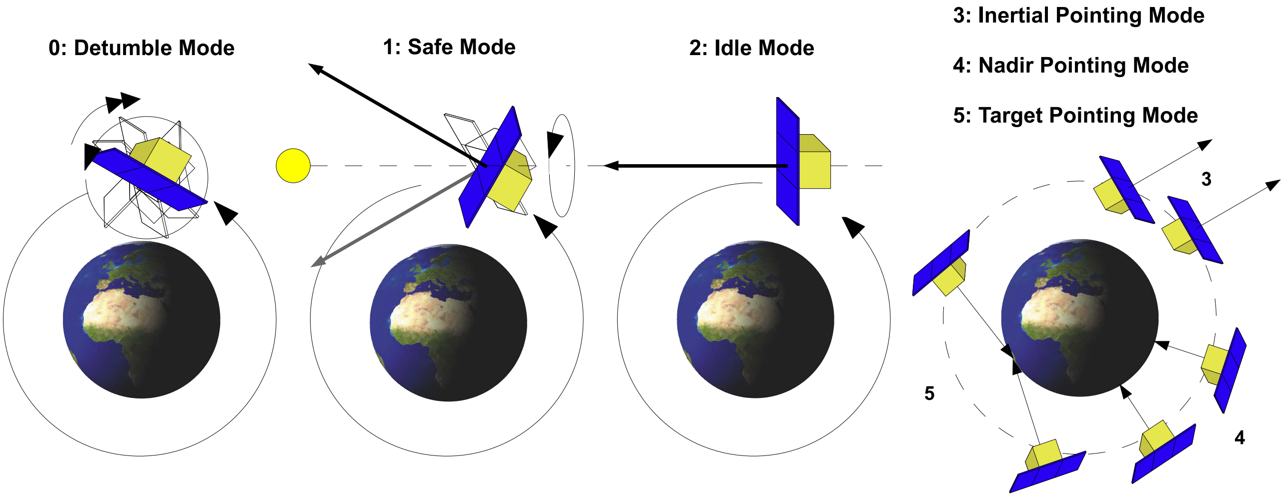


Figure 4.9.: Attitude control modes of ACS

The scientific Earth observations such as BRDF experiment and precipitation measurements, as well as high-speed communication require the target pointing mode to orient its instruments to the ground target. This mode enables payload camera systems to take desired high resolution images with sufficient integration times. Detailed description of pointing modes can be seen in Section 3.4.1. The combinations of the sensors and actuators used are different in each attitude control mode as is described in Table 4.5.

Table 4.5.: Attitude control system components on/off table (×: on)

Mode	Nr.	RW	MGT	MGM	GPS	STR	FOG	SuS
Detumbling	0		×	×				
Safe	1		×	×				×
Idle	2	×	×	×		×	×	×
Inertial P.	3	×	×	×		×	×	×
Nadir P.	4	×	×	×	×	×	×	×
Target P.	5	×	×	×	×	×	×	×

4.4.2. Attitude control system state-machine

The state-machine of the ACS subsystem is illustrated in Figure 4.10. In system level idle mode, the ACS is also in idle attitude control mode. Every time the system goes into safe mode, the attitude control system goes through detumbling mode so that the rotational rate remains under a certain limit, preventing the payload camera systems from seeing the sun directly. Pointing modes are for the scientific experiments in the system level active operation mode. Among these modes, especially detumbling mode and safe mode are critically important to ensure a high survivability of the satellite and therefore shall be designed carefully.

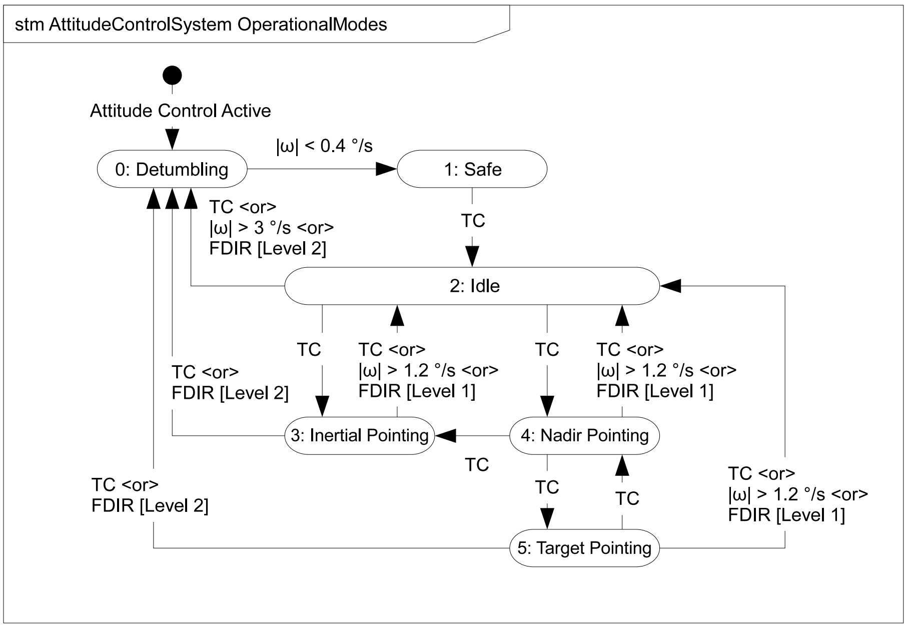


Figure 4.10.: State-machine of ACS

4.5. Power balancing design

Based on the above described operational concept, attitude control mode, and power supply system design (Appendix C.2), the power budget of the satellite is calculated for different scenarios. Mathematical models of the power supply system, solar panels, battery, and PCDU are integrated with the attitude control algorithm model inside Mathworks MATLAB/Simulink. The specification of solar panels are obtained from [95], and those of battery cells and battery assembly are from [96] and [97], respectively. Detailed modeling technique of solar panels are described in detail in [98]. Summary of characteristics of batteries can be seen in [99].

Table 4.6.: Solar panel parameters

Solar Panels	Cell Area [m^2]	Winter Solstice Max. Power [W]	Winter Solstice Max. Voltage [V]	Summer Solstice Max. Power [W]	Summer Solstice Max. Voltage [V]
Middle	0.241	55.6	27.0	53.0	27.5
Right (+X)	0.317	86.7	23.4	82.2	23.9
Left (-X)	0.317	86.7	23.4	82.2	23.9

The characteristics of the solar panels are listed in Table 4.6. The target orbit for simulations below is the reference orbit “1” in summer solstice in order to simulate the worst case scenarios.

Because the simulation model does not contain thermal model of the satellite, the temperature of the solar panels remain constant. Operational temperature of the middle solar panel is identified as 139.0 °C and those of side solar panels as 85.4 °C.

During the normal operation, the state of charge (SOC) of the battery is kept above 80 %. However, during the Launch and Early Orbit Phase (LEOP), it is required by the launcher organization to keep the SOC considerably lower not to risk the main payload satellite. Because the actual required value of SOC is not available at the time of writing, the initial SOC is set as 50 %. For the following simulations, three configuration of solar panel deployment are considered as illustrated in Figure 4.11. The moment of inertia of the satellite in three configurations are summarized in Appendix D.3.

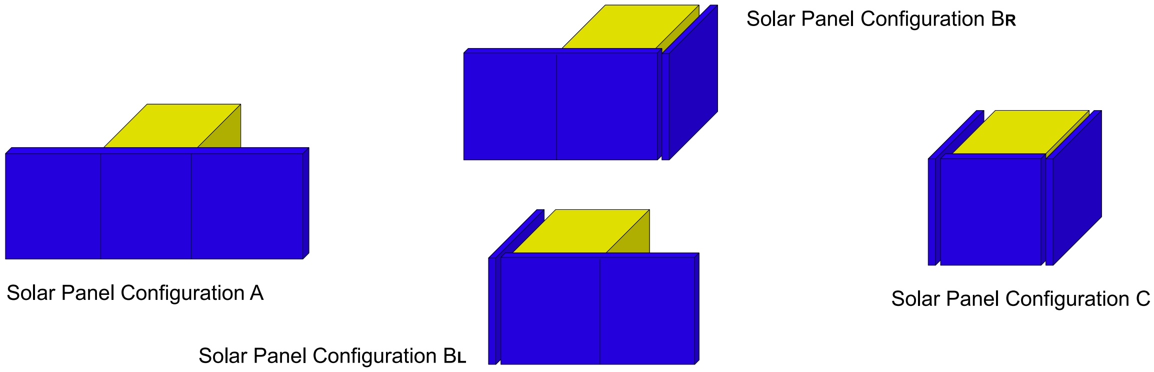

Solar Panel Configuration BR
Solar Panel Configuration A
Solar Panel Configuration C
Solar Panel Configuration BL

Figure 4.11.: Solar panel deployment configuration

4.5.1. Launch and early orbit phase scenario

After the separation from the launcher, the satellite performs detumbling mode to decrease the rotational rate, and then moves into safe mode. The solar panels are also deployed at the moment the satellite goes into the safe mode. The attitude control laws of the detumbling and safe modes and the moment of inertia of the *Flying Laptop* applied for the simulations are summarized in Appendix D. Figure 4.12 illustrates the state of charge of the battery together with the existence of the sun, as well as the amounts of total power generation of three solar panels, their individual power generation, and satellite total power consumption. It can be seen that the battery is successfully charged up to 100 % in this scenario. Solar panel can be also deployed by sending command at the initial ground contact, however, because the SOC at the beginning is decreasing, it can lead to a mission loss. Decision was made to deploy the side solar panels automatically. Figure 4.13 indicates that the solar panels are originally pointed toward different direction before they are deployed at around 800 s. After the stabilization of the rotational motion of the satellite body, the total power generation amounts to about 200 W. The SOC of the battery becomes 100 % in about 5 orbits after the initiation of safe mode.

Figure 4.13 illustrates the satellite's rotational rate, angle between the solar panel normal and the sun vector, and two elements of the rotational rate parallel and orthogonal to the reference principal axis, which is the one with the largest moment of inertia (see Appendix D). The total rotational rate is reduced to 0.4 °/s before the satellite initiates the safe mode. It can be seen that the rotational rate of about 2 °/s is achieved in the safe mode, which succeeds to keep the

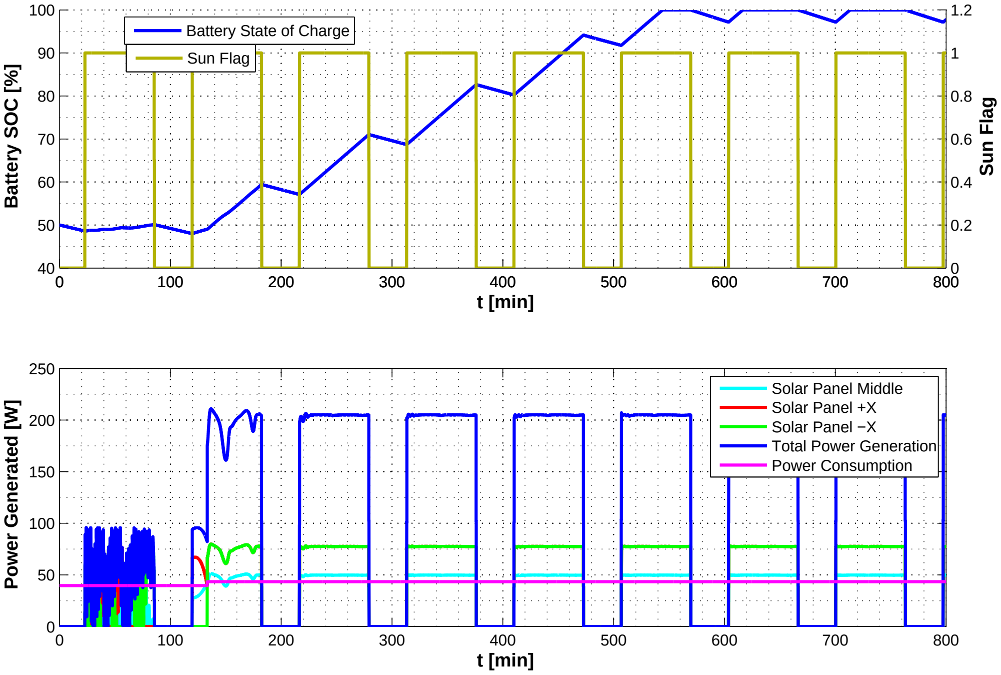


Figure 4.12.: Battery state of charge with sun flag and total power consumption and generation (Configuration A)

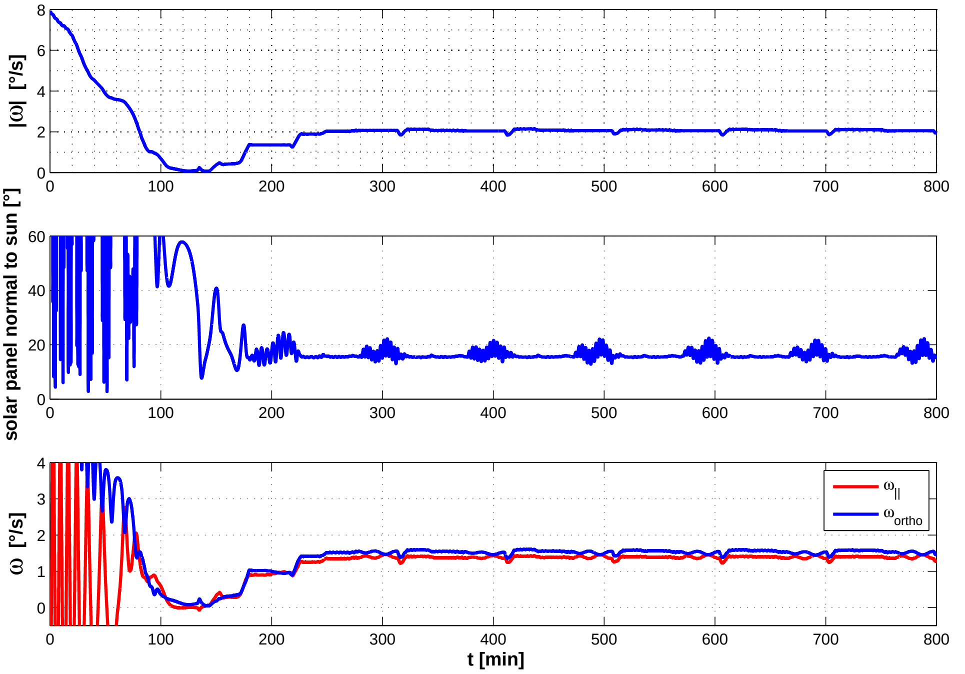


Figure 4.13.: Rotational rate, sun vector angle, and component of rotational rate parallel and orthogonal to the reference principle axis (Configuration A)

satellite attitude stable even during the eclipse phase. The solar panels are coarsely pointed toward the sun with an offset angle of about 18 °.

4.5.2. Analysis of failure scenarios

The failure of deployment of solar panels can lead to a mission loss. Based on the single failure tolerant design concept, the satellite shall be able to survive in case one of the two solar panels failed to be deployed. An analysis has been done on this as illustrated below. The above figure of Figure 4.14 illustrates the power budget simulation with the same condition except that one of the two side solar panels (-X) remains closed. The below figure of Figure 4.14 illustrates the simulation results in case both of the solar panels are not deployed.

As illustrated in Figure 4.14, the SOC of the battery can be charged up to 100 % even if one solar panel is closed. It can be also seen that the failed solar panel is in sun light, because of the misalignment between the reference axis and the solar panel normal. It is worth noting that in case of total failure of one side solar panel (e.g. electrical open circuit), the power budget becomes even worse than illustrated in this figure.

Failure of deployment of both side solar panels are considered as multiple failures, therefore, this case is not a driving requirement for the system design. However, it is still important to take a look at what happens to the satellite. As illustrated in Figure 4.14, the SOC increases slightly. Because the middle solar panel is the smallest solar panel (Table 4.6), the amount of

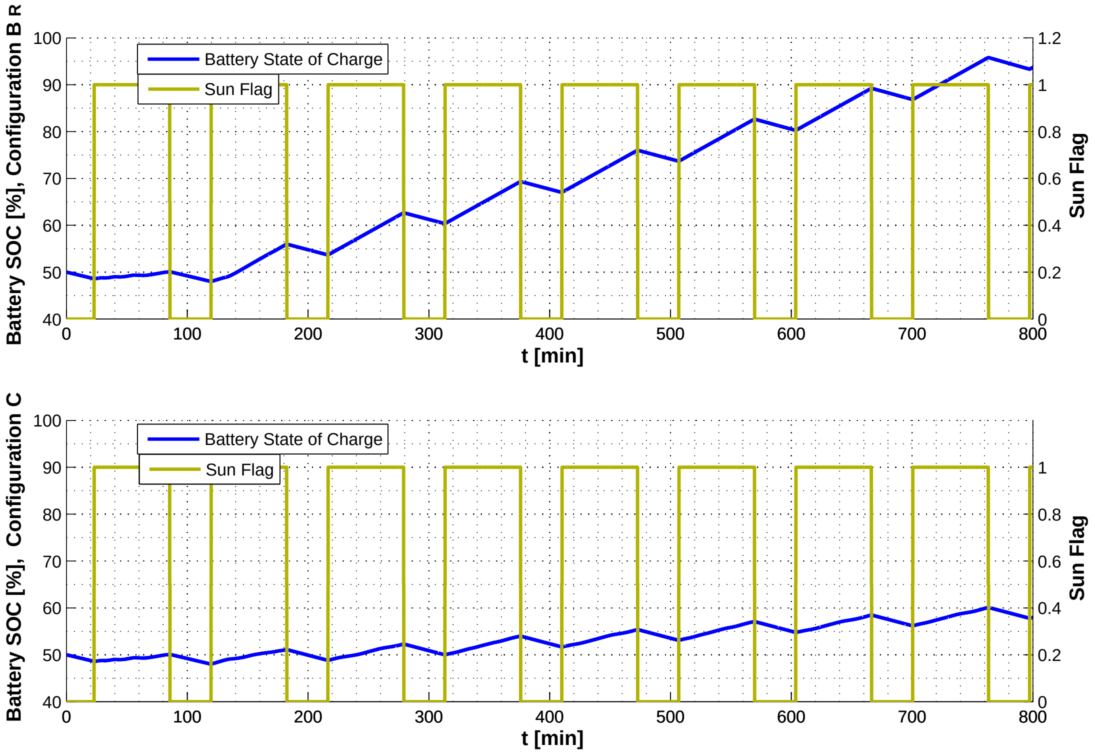


Figure 4.14.: Battery state of charge and total power consumption and generation in configuration B_R (above) and C (below)

power generation is significantly small. One can orient one of the side solar panels toward the sun except for the middle solar panel, but for this purpose two different safe modes shall be prepared and be switched with according to the status of the solar panel deployment. This complexity can also lead to a severe failure aboard the satellite. Decision was made to use only one common safe mode for all of above three cases, mostly because the third case is not a design driving requirement. Therefore, the designed safe mode is optimized for the solar panel configuration A. In the worst case with the solar panel configuration C, a new configuration file shall be uplinked for the OBC in order to completely change the safe mode strategy.

4.5.3. Analysis of a maximum power consumption scenario

The feasibility of high-speed communication with the Traveling Wave Tube Amplifier (TWTA) shall be demonstrated by the power analysis. This communication is one of the most important technology demonstration missions of the satellite and also will be used for periodical downlink of the large amount of scientific data. The TWTA requires electrical power of about 200 W. This operation is conducted in the target pointing mode in order to point the Ka-band antenna toward the communicating ground station. The amount of total power consumption of the satellite system sums up to about 310 W during this operation. As defined in requirements, this high-speed communication shall be able to be conducted twice per day. In the following simulation, this communication has conducted twice in consequent two ground contacts. The initial SOC is set to 100 %. The simulation scenario is summarized in Table 4.7. The simulation results are illustrates in Figure 4.15 and Figure 4.16. The initial rotational rate is set to 3 °/s around the Y body-axis.

Table 4.7.: High-speed communication simulation scenario

Sequence [min]	System mode	ACS mode	TT&C mode
0	Safe	Detumbling (0)	-
35	Safe	Safe (1)	-
220	Idle	Idle (2)	-
235	Active operation	Nadir pointing (4)	-
240	Active operation	Target pointing (5)	High-speed communication
255	Idle	Idle (2)	-
320	Active operation	Nadir pointing (4)	-
340	Active operation	Target pointing (5)	High-speed communication
355	Safe	Detumbling (0)	-
400	Safe	Safe (1)	-

Figure 4.15 illustrates the battery SOC, existence of the sun, and the attitude control system mode, together with the summary of system power consumption and generations. As described in Figure 4.10, in order to move into target pointing mode, ACS needs to be commanded through idle mode and nadir pointing mode, starting from the safe mode. The mode number of the ACS is also listed in Table 4.7. The first high-speed communication takes place at 240 minutes, and the second at 340 minutes, respectively. Each communication lasts 15 minutes. After the first communication, the system goes back into idle mode, while the second communication is followed by system safe mode (ACS goes into the detumbling mode).

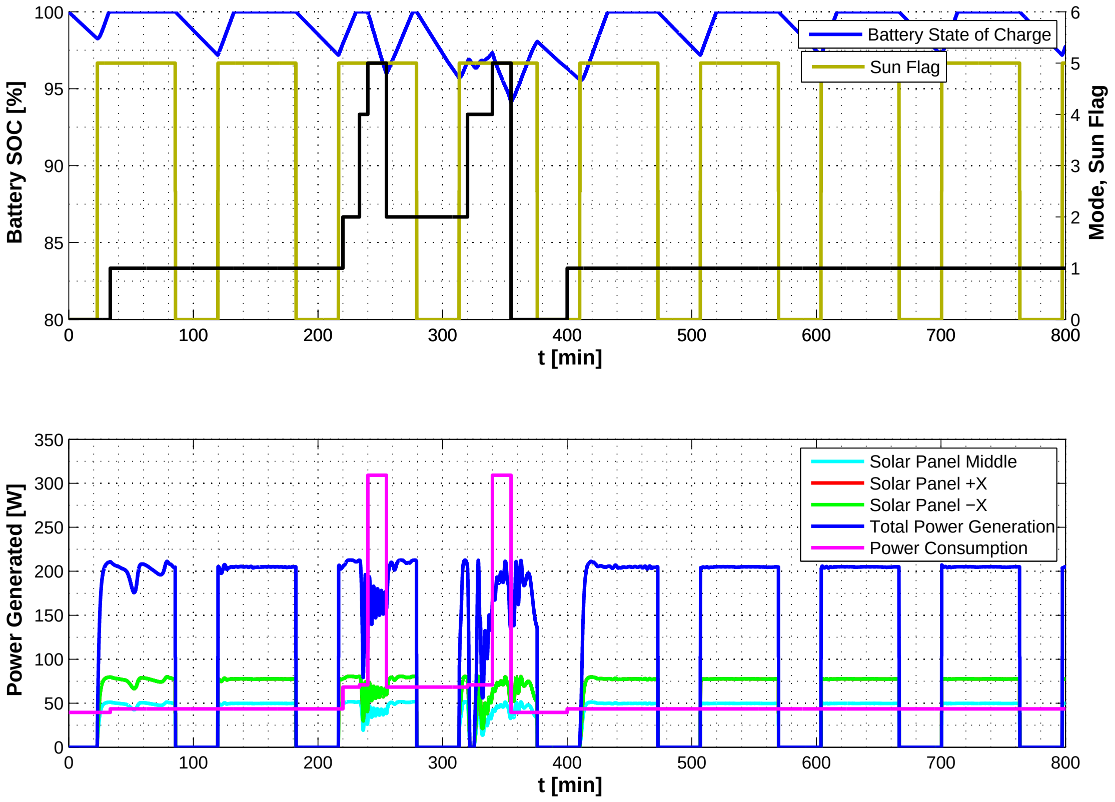


Figure 4.15.: Battery state of charge, ACS mode, and total power consumption and generation

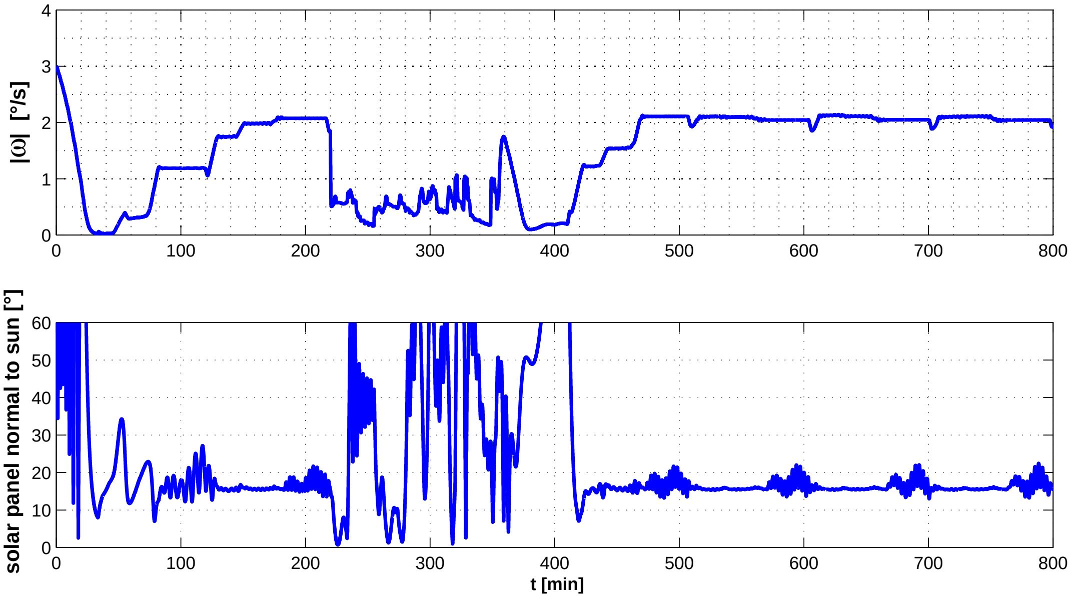


Figure 4.16.: Rotational rate and sun vector angle

From Figure 4.15, it can be seen that the SOC of the battery decreases down to about 93 % after the second communication, and again increases up to 100 % in the next orbit. It is also illustrated that the power consumption rises up to the maximum value of about 310 W during the high-speed communications. Figure 4.16 illustrates the absolute rotational rate of the satellite body and the angle between the sun vector and the solar panel normal. It can be seen that the initial rotational rate was successfully detumbled at first, before moving into safe mode. Also the rotational rate remains lower than about 1 °/s during the active operation. In the safe mode, the solar panels are successfully pointed toward the sun with an offset of about 18 °. It can be seen that the angle between the solar panel normal and the sun vector changes dramatically during the active operation. In this simulation scenario the satellite goes into safe mode (ACS once goes into detumbling mode and then into safe mode) just after the second communication in order to simulate mulfunction of attitude control components as the worst case. As illustrated in the figure, the satellite establishes its stable attitude by the end of the next orbital phase. This performance of the satellite promises its safety power management. Consequently, this simulation results ensures that the Ka-band high-speed communication can be conducted at least twice per day as required.

4.6. Summary

This chapter described the system design of the *Flying Laptop* mainly from the aspects of electrical architecture and operational design. According to the mission requirements and constraints, the electrical architecture of the satellite system was designed. Functional analysis defined required satellite components. The electrical interfaces between satellite peripherals and the on-board computer are designed by means of electrical interconnections block diagram, I/O diagram, the real-size mock-up model of the satellite system, and the harness cabling plan. This activity provides design requirements of the OBC.

The operational design of the *Flying Laptop* satellite was summarized in terms of operational mode design, commanding design, data handling design, communication structure design, and contingency operation design. The validities of these operational designs in terms of power balancing are verified by the mathematical model based on Mathworks MATLAB/Simulink model. Simulations have illustrated that the established LEOP operation scenario can ensure secure establishment of the initial condition of the satellite. Simulations indicated that the satellite can survive and conduct missions even if one side solar panel is failed to be deployed. Finally, it is also validated, that the Ka-band high-speed communication can be safely conducted twice per day. The contingency operation concept ensures secure recovery of the system.

The FPGA-based on-board computer of the *Flying Laptop* shall fulfill the requirements identified in this chapter. Furthermore, the hardware logic inside FPGAs shall support the operational concept developed in this chapter, including the operational modes, commanding structure, and communication algorithms, together with the attitude control algorithms.

5. Design of OBC hardware

In this chapter, the proposed concept of FPGA-based computer system is applied to the demonstration small satellite *Flying Laptop*. This system concept is named as FPGA-based On-board Computer for Reconfigurale Computing on Space Systems: FORS. As described in Chapter 4, the design of the OBC is strongly related with the system design of the whole satellite system. The computing resource of the *Flying Laptop* satellite are provided by a single unified OBC that performs the control functions of the traditional satellite bus controller (SBC) and all tasks of the payload data handling and processing system. In general, when designing an OBC based on COTS components, the design strategy has to be based on the assumption that, because of the SEE, failures within the system must not be considered as an exception but, in principle, as normal operation. Such a design principle requires fault tolerance features to be added to the system [100].

5.1. Requirements of OBC

From the system design activity described in Chapter 4, the requirements of the OBC are defined as follows. The OBC of the *Flying Laptop* shall

- be equipped with reconfigurable SRAM-FPGAs for the demonstration of reconfigurable computing on spacecraft,
- be a single centralized OBC which performs all kind of general functions of SBC,
- be connected with all peripheral instruments,
- be capable of conducting all required subsystem functions such as image processing, attitude determination and control, telecommunication, and housekeeping,
- be capable of numerical computation based on fixed point calculations with an absolute pointing accuracy of 150 ",
- be capable of high-speed data transfer for Ka-Band communications up to 500 Mbps,
- be compatible to the mechanical structure of Eurocard 3U specification,
- be equipped with compactPCI connectors with the quantities of input/output pins of each board up to 220,
- be equipped with mass memories equal to or larger than 4 GB in which the scientific and housekeeping data, as well as uplinked commands are stored,
- have total power consumption of up to 20 W,
- be reconfigurable via telecommands,
- have a design life time of 2 years,
- be based on COTS components as much as possible,

- be based on a single failure redundant design concept, and
- offer capability of floating-point calculation based on internal PowerPC Cores.

5.2. Applied OBC design

According to the discussion in Chapter 2, the selected concept of the FORS for *Flying Laptop* satellite is based on a combinational use of reconfigurable SRAM-FPGAs and Flash-FPGAs. The internal configuration and mechanical structure of the OBC is illustrated in Figure 5.1 and Figure 5.2.

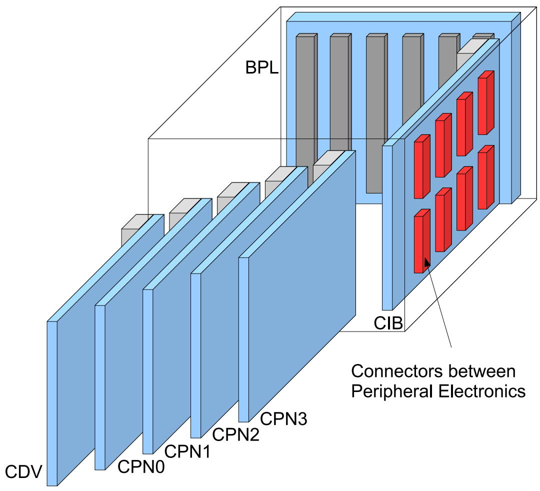


Figure 5.1.: OBC internal configuration

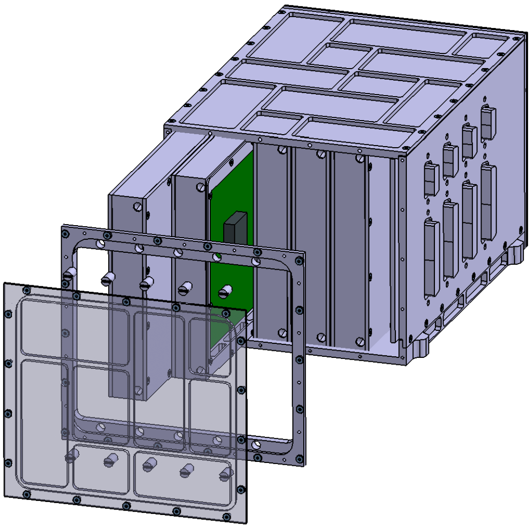


Figure 5.2.: OBC mechanical structure

The OBC of the *Flying Laptop* consists of four Central Processing Nodes (CPNs) and one Command Decoder and Voter (CDV). They are connected with each other through a backplane (BPL) and with other peripheral electronics through a Connector Interface Board (CIB) as illustrated in above figures. Each CPN is based on a reconfigurable SRAM-FPGA: Virtex-II Pro XC2VP50 of Xilinx. The CDV is based on two Flash-FPGAs: RTProASIC RT3PE3000L of Actel. The SRAM-FPGA is selected partly because the chip was the state-of-the-art SRAM-FPGA at the time of its design fix and also partly because the chip offers additional processor core named PowerPC inside. Therefore, the CPN can also be loaded with a soft-core of a microprocessor so that usual floating-point calculation can be performed on demands. This capability enables a “Rent-A-Sat Mode” for the satellite, in which the satellite is temporally lent to third parties for orbit verification of their satellite software such as attitude determination and control algorithm. The applied SEE-tolerant FPGA for the CDV is the state-of-the-art radiation tolerant Flash-FPGA. Both of the applied FPGAs, XC2VP50 and RT3PE3000L, are the very devices whose SEE and TID effects are investigated in Part I, and therefore, the exact results of reliability analysis hold for the OBC of the *Flying Laptop*. The development of the OBC follows a model philosophy of breadboard model, engineering model, and then flight model.

The redundancy degree of the CPN is decided as four. This can also incorporate the uncertainty of space radiation model and the case of permanent failure of one of the four CPNs. Furthermore, 4 nodes allows self voting by themselves by communicating with each other if required. Generally, $3n + 1$ system can detect n simultaneous failures. The merits of this approach are:

- the extended lifetime of the OBC in the event of a permanent failure of a CPN,
- the short outage time (higher availability) in the event of SEU induced failures, and
- the redundant internal voting capability on demand.

The design is based on the assumption that each CPN performs the exact same tasks at the same time. This is why all input/output signals between peripheral components, in total more than 200 lines, shall be connected to CPNs in parallel, so that each single CPN can execute full functions and control tasks. CPNs shall be also physically identical. The hardware structure of the OBC plays an important role against TID. The width of the aluminum structure is 3 mm.

5.3. System reliability

Following the discussions in Part I, the system reliability can be simply described as Equation (2.2). Because the reliability of the node system against SEEs is repairable, the term $R_{NS,SEE}$ is dependent on the time after the last (complete) node system start-up at t', which can be obtained by Equation (2.63). Consequently, the system reliability at t with the redundancy degree of $m = 4$ can be described as

$$\begin{aligned} R_{sys,NS_{3of4}}(t) &= R_V(t) \cdot R_{2of4,NS}(t) \\ &= R_{V,SEE}(t) \cdot R_{V,TID}(t) \cdot R_{2of4,NS,SEE}(t-t') \cdot R_{3of4,NS,TID}(t) \ , \end{aligned} \tag{5.1}$$

where $R_{3of4,NS,TID}(t)$ can be obtained by Equation (2.59) with knowing the amount of total ionizing dose in time t. In case one node unit failed permanently due to any kind of reasons, the node system works as a 2-out-of-3 system. The reliability of this system can be described as

$$R_{sys,NS_{2of3}}(t) = R_{V,SEE}(t) \cdot R_{V,TID}(t) \cdot R_{2of3,NS,SEE}(t-t') \cdot R_{2of3,NS,TID}(t) \ , \tag{5.2}$$

where $R_{2of3,NS,TID}(t)$ can be obtained by Equation (2.58) and the t remains as the total mission time after the launch.

According to the analysis on the reliabilities of the selected FPGA devices against TID effects, the terms $R_{V,TID}(t)$, $R_{3of4,NS,TID}(t)$, and $R_{2of3,NS,TID}(t)$ for $t \leqq 2$ years can be regarded as ≈ 1. Therefore, under the assumption that the SEE effects on Flash-FPGAs can be mitigated, the system reliability mainly depends on the time interval $t - t'$. Suppose that $t - t' = 30$ days, for the given unit failure frequency (or given $MTTF_{NU}$) identified in Section 1.4.3 (in total $9.56 \cdot 10^{-5}$ assuming a statistical independence) and a $MTTR_{NU} = 5$ s, the total system reliability in $t = 2$ years for $m = 4$ and for $m = 3$ can be calculated by above equations as

$$R_{sys,NS_{3of4}}(t - t' = 30days) = 0.99932 \ , \tag{5.3}$$

and

$$R_{sys,NS_{2of3}}(t - t' = 30days) = 0.49181 \ . \tag{5.4}$$

5.4. Functional requirements allocation

The requirements of OBC shall be fulfilled by the combinational functionality of its components. The functional requirements of CPN and CDV are derived from the above mentioned top level requirements as follows:

Functional Requirements of the CPN

CPNs shall

- conduct all satellite control functions,
- be reconfigurable both from stored memory and in streaming mode from the CDV,
- be quad-redundant and be capable of absorbing a permanent failure of one CPN,
- be able to communicate with each other,
- report checksums to CDV,
- be equipped with a safety circuit in order to detect SEL and isolate from the power supply,
- be isolated from the communication bus via isolation buffers controlled by the CDV,
- be synchronized with the other CPNs within 5 s after reconfiguration, and
- be able to transport data with a baud rate of up to 500 Mbps to the Ka-band system.

Functional requirements of the CDV

CDV shall

- be single event effects tolerant,
- be equipped with Flash mass memories equal to or larger than 4 GB,
- be connected with primary communication components to decode up-link commands,
- monitor the checksum from CPNs and vote the master CPN out of the four,
- forward the SRAM-FPGA's new configuration data to the CPNs in both normal transfer mode and streaming mode,
- be capable of enforcing reset/recovery of each CPNs,
- be capable of shutting down each CPN,
- be capable of transporting date to each CPNs with a baud rate of up to 500 Mbps,
- conduct power cycling of a CPN,
- execute high level commands which are not processed by CPNs,
- retain on-board real time clock information and deliver it to CPNs,
- distribute reference clock signals to CPNs, and
- not be reconfigured on orbit.

All control and data processing functions of the satellite, which are not directly executed by the CDV, are implemented into the SRAM-FPGA on the CPNs. Main satellite controlling functions of the CPNs are attitude control algorithm computation, real-time image processing, generation of scientific telemetry data, TM/TC processing and execution of housekeeping routines.

5.4.1. FDIR strategy

Due to the SEU sensitivity of the SRAM-FPGAs, arbitrary failures of the CPN shall be expected. The Failure Detection, Isolation, and Recovery (FDIR) strategy of the OBC operation is based on:

- failure detection by comparing the checksums generated by the replicated redundant CPNs
- failure isolation by voting the checksums and selecting a master CPN
- failure recovery by enforcing the CPN delivering a wrong checksum to perform reset/restart procedures and subsequently re-synchronize with the other nodes

Within this straight forward FDIR strategy, failure handling mainly relies on the implementation of the robust restart capability.

5.4.2. Node system configuration and interface replication

Figure 5.3 illustrates the internal functional architecture of the OBC. The CDV selects one CPN as the master node by switching the isolation buffers. Only the master node is able to drive the output lines to the devices of the satellite. The redundant nodes in general are receiving the same digital inputs and executing the same functions, thus producing the same results.

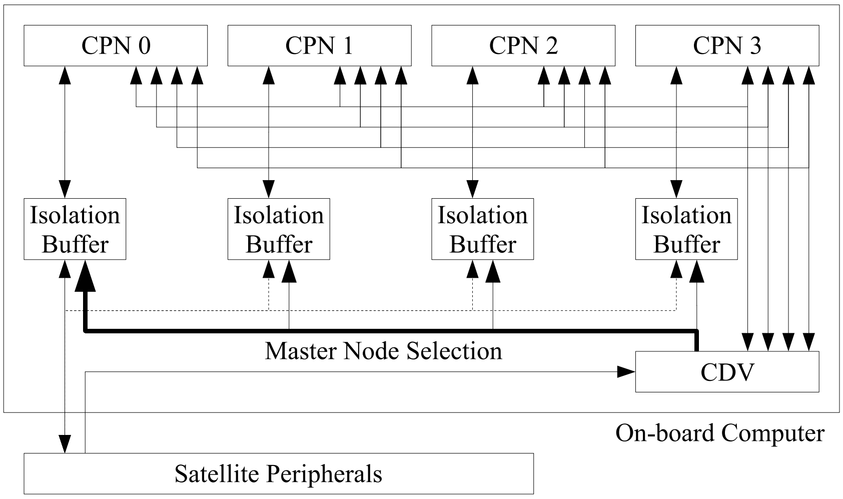


Figure 5.3.: OBC internal architecture

The faulty node delivering a wrong or no result are enforced to perform the reset/restart function. If the current master fails to produce the correct checksum, the CDV will select another node as the master which is giving correct outputs. To avoid accumulation of latent errors in the master node, the CDV may cyclically select a new master enforcing the old master to execute the reset/restart function. Following CPN states shall be determined by the CDV.

- Inactive (power off, isolated from the shared buses)
- Master (selected to drive the output lines of the shared I/O buses)
- Monitor under recovery (performing the restart/recovery function)
- Monitor synchronized (runs synchronously, provides correct checksums, can be selected as the master)

5.5. Design implementation of CPN

5.5.1. Node structure

The block diagram of the developed engineering model of the CPN is illustrated in Figure 5.4. The CPN was developed under a contract by the Steinbeis Transferzentrum Raumfahrt (TZR) [101]. The design of the CPN is specified in [100]. The secure reconfiguration of the Virtex-II Pro FPGA is performed by a dedicated configuration controller (CC) implemented in a separate FPGA with radiation hard configuration on EEPROM. Multiple versions of the configuration files for the Virtex-II Pro FPGA can be stored in three separate NOR-Flash memories. The CDV has access to them via the CC either to load new versions of configuration files or to restore corrupted ones.

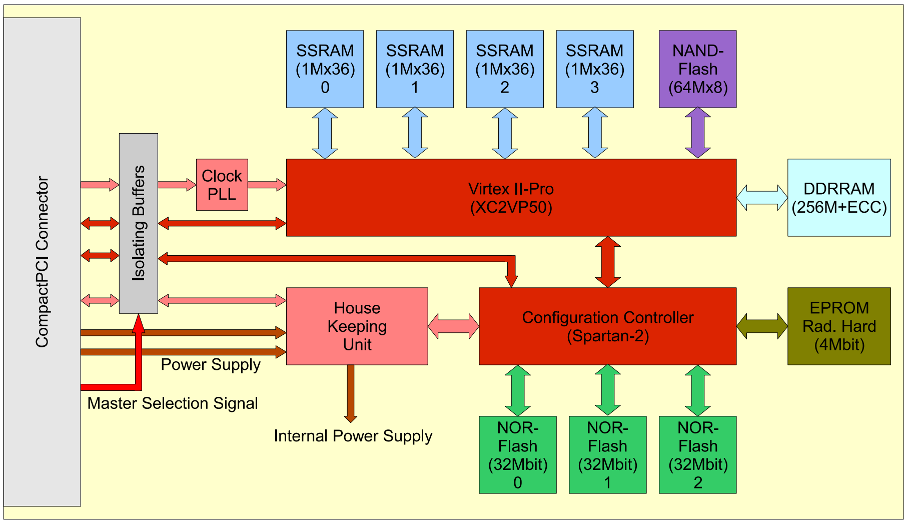


Figure 5.4.: Block diagram of CPN [100]

Storing different versions of configuration data for the Virtex-II Pro FPGA in different Flash memory devices allows secure in orbit reconfiguration of the satellite functions. Loading the Flash memories under control of the CDV in a streaming mode ensures that CPNs can be reconfigured even if all configuration data in the Flash memory has been corrupted.

In addition to the robust restart capability of the nodes, the availability and reliability of the OBC highly depends on the feature to securely isolate a faulty node so that it can not obstruct the correct operation of the rest of the system. All I/O connections are implemented by isolation buffers that completely disconnect all interface signals from the shared I/O and communication buses if the CDV switches off the power supply of a node. In addition, the enable control signal of the isolation buffer devices is used to control the output drivers of the CPNs such that only the selected master node is able to drive the output lines.

The Virtex-II Pro FPGA has accesses to four parallel operating banks of fast SSRAM (Synchronous Static Random Access Memory) for intermediate results and to two banks of DDR SDRAM (Synchronous Dynamic Random Access Memory) to be used for application data or optionally as program memory of the PowerPC core. Additional NAND Flash memory can be used for permanent storage of application data or optionally of the PowerPC code. PowerPC Cores can be used for intensive numerical calculations such as, e.g., Kalman filtering.

5.5.2. Interface isolation

The possible data rates of the isolating communication drivers can be obtained from the diagram in Figure 5.5. Input buffers are always enabled, while output buffers are disabled by default. Only the output drivers of the master node are enabled by the CDV.

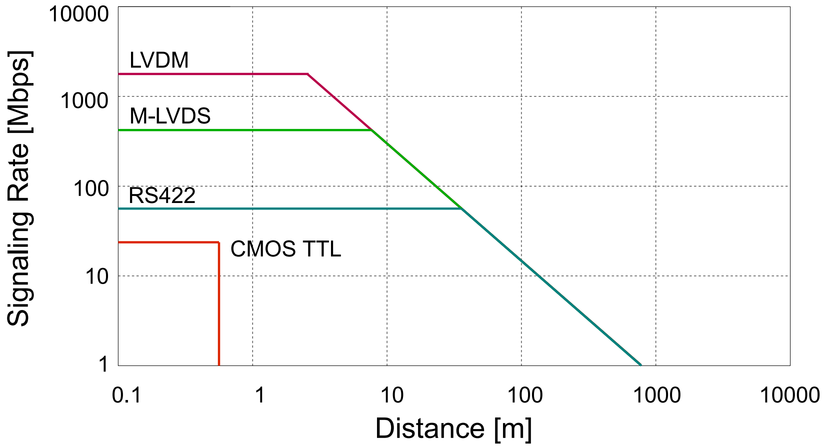


Figure 5.5.: Signalling rates vs. distance of the isolation buffer devices [100]

The differential high-speed serial connections (M-LVDS, LVDM, RS-422) and the I^{2}C bus require appropriate termination of the I/O signals. Depending on the topology of the connection the termination resistors must be placed nearby the receiver and/or transmitter connected by the longest path of the topology. To have four identical CPNs, termination cannot be implemented on the boards, thus the termination resistors need to be placed on the BPL.

The high-speed communication interface between the Ka-band transmitter is realized as two 250 Mbps LVDM channels, in order to reduce the required clock frequency inside FPGAs, though the interface standard allows higher baud rates. Implementation techniques of high-speed asynchronous interfaces with low clock frequencies are provided by Xilinx in [102].

5.5.3. System clock

To allow an efficient and tight synchronization a reference clock of 10 MHz is generated by the CDV and distributed to CPNs via point-to-point LVDS signals routed on the BPL.

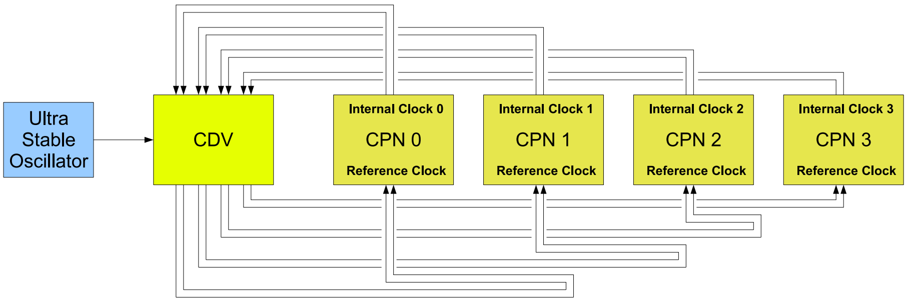
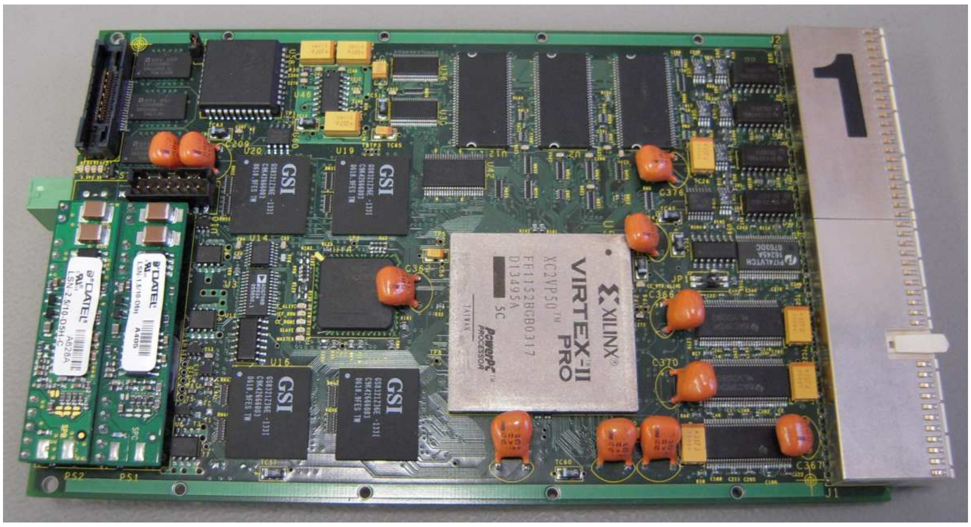


Figure 5.6.: System clock distribution

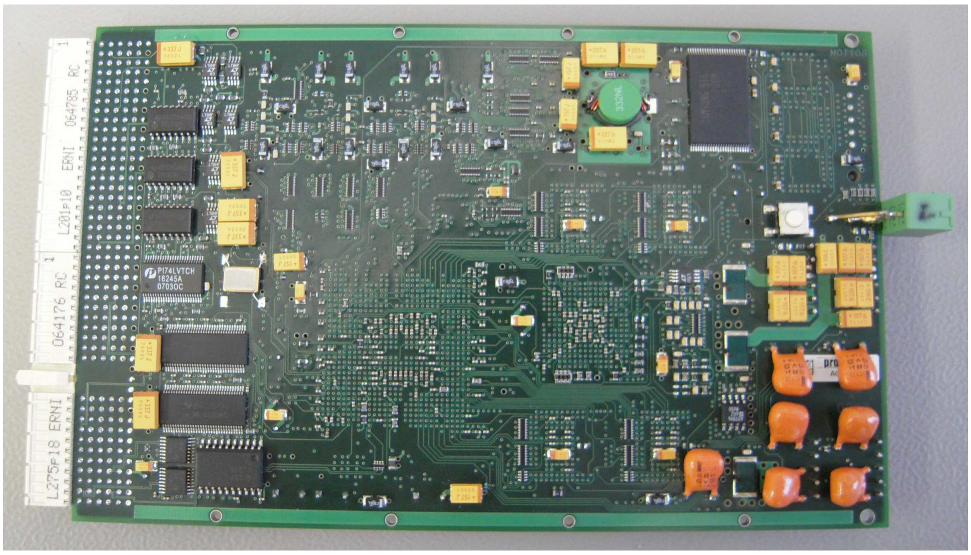
The routing characteristics of the signal (length, impedance, delay) must be identical for each connection as illustrated in Figure 5.6. An internal zero delay clock multiplier generates clock signals for the FPGAs. For the Virtex-II Pro, the reference clock is multiplied by 8 to deliver an 80 MHz clock. To support stand alone operation of the node an internal 10 MHz clock oscillator is implemented on CPNs. The output of this internal clock is connected to the CDV.

5.5.4. Engineering model of CPN

The engineering model of the CPN is developed based on the above mentioned functionalities. The developed EM of the CPN can be seen in Figure 5.7. The detailed internal architecture of the CPN can be seen in [100].



(a) Front side



(b) Back side

Figure 5.7.: Engineering model of CPN

5.6. Design implementation of CDV

The CDV is the component which requires the highest reliability among the on-board components. The solution to this requirement is to keep the high reliability logic as compact as possible and apply internal multi-module redundancy to achieve a higher reliability. Indeed, the size of the available Flash-FPGA RTProASIC is very limited to be implemented with all functionalities of the CDV in a TMR configuration, therefore, trade-off shall be done defining priorities of functionalities and applied radiation mitigation methods. Applied radiation mitigation methods are described in Chapter 6.

According to this background, besides the Flash-FPGA which is responsible to command decoding and voting, one more Flash-FPGA for on-board data handling is also implemented on the CDV. The FPGA for the command decoding and voting is named as Main Decoder/Voter unit (MDV) and the additional Flash-FPGA is named as Data Handling Support unit (DHS). For both FPGAs, the same Flash FPGA RTProASIC is selected.

Because the functionalities of the CDV is very critical, on-board reconfiguration of the hardware configuration of the CDV is not planned. Therefore, the control algorithms of the CDV shall have been developed before the launch. The final FM can be theoretically configured many times, it shall be kept minimum in order to prevent extra degradations before flight. Due to this, the most of the functionalities shall be throughly developed based mainly on breadboard and engineering models. The FM shall be reconfigurable on ground with external reconfiguration instruments.

5.6.1. Flash-FPGA RTProASIC

RTProASIC3 devices are pin-compatible and timing-compatible with the commercial equivalent ProASIC3EL devices in Fine-Pitch Ball Grid Array (FG) packages. For example, to prototype a space-flight design intended for RT3PE600L, the A3PE600L can be used. Since ProASIC3 FPGAs are reconfigurable, only a very small number of prototyping devices are required. Characteristics of available two types of RTProASIC chips are summarized in Table 5.1. Due to the demands on larger logic size, the RT3PE3000L is selected for both of MDV and DHS.

Table 5.1.: Characteristics of RTProASIC FPGAs [17]

Device	RT3PE600L	RT3PE3000L
System Gates	600,000	3,000,000
Logic Tiles	13,824	75,264
Core RAM Blocks	24	112
Core RAM Bits (k=1,024)	108 k	504 k
FlashROM Bits (k=1,024)	1 k	1 k
Routed	18	18
PLLs	6	6
I/O Banks	8	8
User I/Os	270	620
I/O Registers	810	1,860

5.6.2. Internal structure and interfaces

The designed internal structure and the interfaces between peripheral electronics of the CDV including CPNs are illustrated in Figure 5.8. The main components of the CDV is the MDV and DHS Flash-FPGAs, Flash mass memory, Real Time Clocks (RTCs), and housekeeping unit. Each hardware components and its functions are described in detail below.

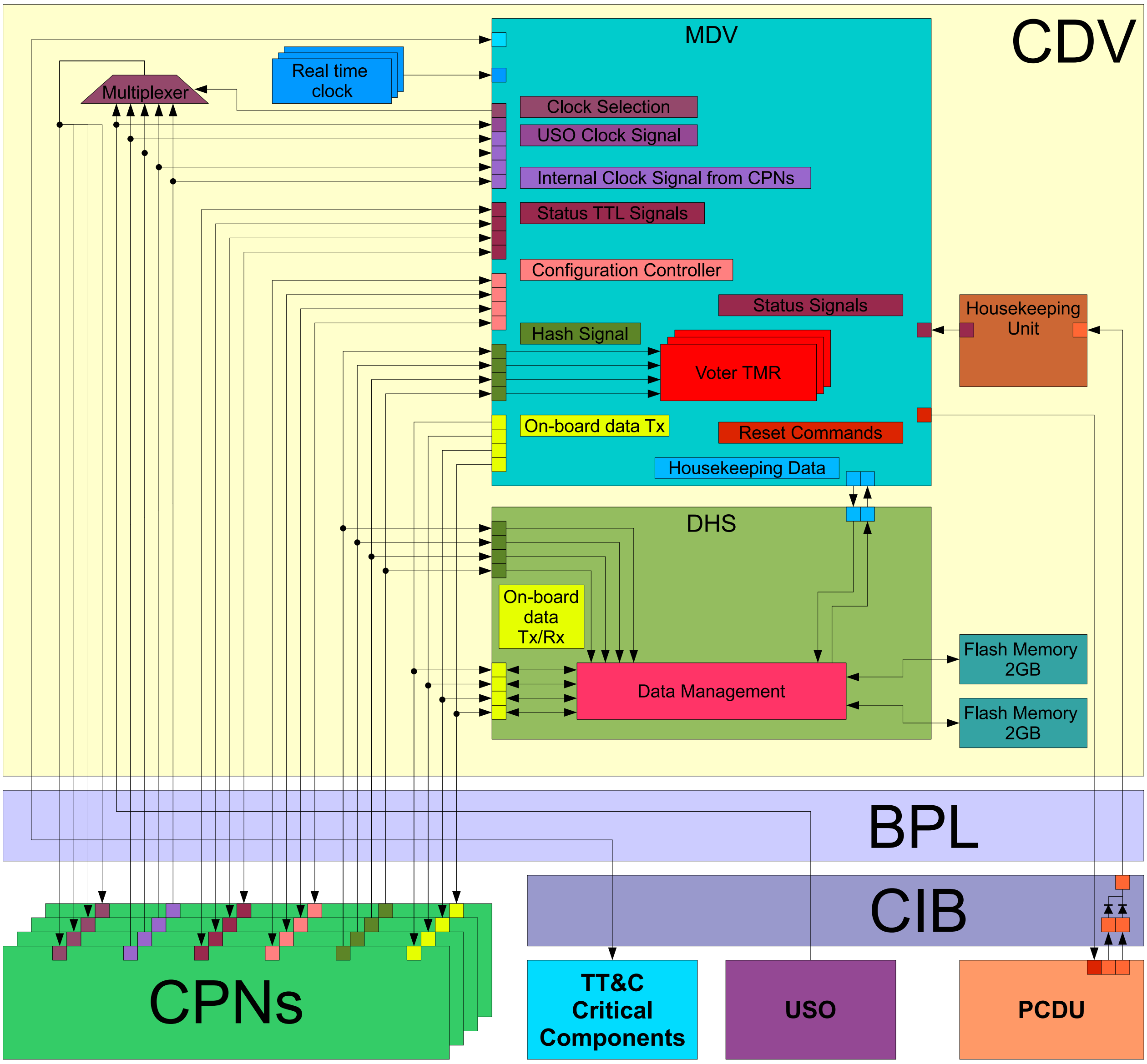


Figure 5.8.: CDV internal structure and peripheral interfaces

5.6.3. Main decoder/voter FPGA

The MDV is responsible for the safe operation of the CPNs performing their voting and command decoding. The logic inside the MDV shall be compact to ensure highly reliable voting. The RTCs and housekeeping unit is connected with the MDV. The functions of MDV are:

- command decoding
- telemetry downlink in contingency
- voting of CPNs by monitoring the checksum and housekeeping data
- clock signal selection and distribution
- real time clock management
- configuration management of CPNs
- self housekeeping monitoring

Command decoding

All uplink channels of the *Flying Laptop* except for the Ka-band are connected with the MDV as well as the four CPNs. Demodulated uplink bit streams can be decoded at the MDV. *Flying Laptop* introduces two levels of telecommands, high level and low level. High level commands includes CDV operation commands, e.g., “CPN reset”, are executed by the MDV. Low level commands, e.g., “take an image”, are executed by CPNs directly.

Telemetry downlink in contingency

Downlink channels of the UHF and S-Bands are also connected with the MDV, as well as the four CPNs. MDV is capable of sending primitive housekeeping data to ground stations activated by high level commands. This is necessary in case of severe radiation conditions caused by coronal mass ejections, because the CPNs can not be operated safely. MDV is also capable of sending reset commands to the PCDU directly, shutting down all electric components which are not needed in safe mode. In this way, MDV can reset the satellite status without CPNs.

Voting mechanism

The voting of the master node is closely related with housekeeping data monitoring. The MDV has nine TTL interfaces between each CPN in order to monitor its status and also to control the operation of CPNs. The information exchanged and control signals are listed in Table 5.2.

Table 5.2.: Housekeeping and control signals of CPN

Data Line	Type	Description
V_ALERT	HK Data	Voltage Alert: Supply voltages lower than threshold
C_ALERT	HK Data	Current Alert: Supply Current higher than threshold
T_ALERT	HK Data	Temperature Alert: Temperature higher than threshold
CC_ERROR	HK Data	Reconfiguration of CPN failed
READY	HK Data	Reconfiguration of CPN succeeded
SYNCED	HK Data	Synchronized with other CPNs
POWER_ON	Control	Power-on CPN
OBC_RESET	Control	Reset (Reconfigure) CPN
MASTER	Control	Select as the Master Node

Each CPN reports six different housekeeping data. The first three ALERT signals are for monitoring supply voltage, supply current, and temperature of the CPN. If the voltage drops below the threshold a hard reset of the CPN is performed and reported to the MDV. If the current exceeds maximum threshold, it is reported to the MDV and the CPN once becomes isolated from the power source and then powered on again. The CC_ERROR reports a failure of reconfiguration and the READY reports a success of reconfiguration of the CPN. A CPN performs synchronization with other CPNs and reports SYNCED after successfully synchronized. MDV can control whether the CPN shall be powered on, reset, and selected as the master or not through control signals.

For the voting of the CPN, each CPN also reports hash signals to the MDV. These lines are only one directional from CPNs to the CDV, however, a hash signal from a CPN is connected with all other CPNs as illustrated in Figure 5.9. After the synchronization, all CPNs perform the same task at the very same clock cycle and then start to send hash signal (information for voting) to the CDV. CDV can simply compare the contents of the hash packages from CPNs and if an anomaly exists, it isolates the failing CPN. CDV selects one master node out of the synchronized and consistent CPNs. This communication is conducted based on the High-Level Data Link Control (HDLC) protocol [103].

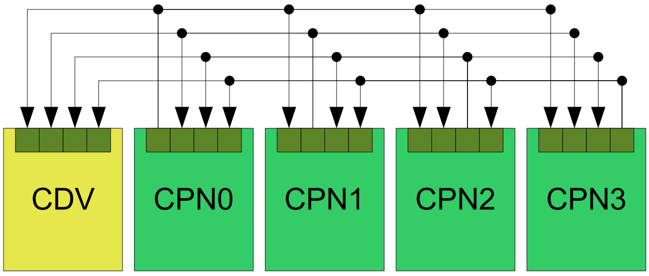


Figure 5.9.: Hash signal interface between CDV and CPNs

Node synchronization

The synchronization of the four nodes are performed through “Sync” packets exchange between each other via hash channels based on the HDLC protocol as illustrated in Figure 5.10. Each CPN is capable of shifting its operation phase and performs self-synchronization automatically by receiving the “Sync” packets from other three CPNs and resetting the internal clock counter. If a CPN is powered on or reset, it starts a synchronization procedure just after the initialization, sending a “Sync” packet every 100 ms. It also counts up its internal clock counter to measure the interval of 100 ms. If a CPN receives a “Sync” packet, it sets the value of the internal counter with an initialization value. This procedure ensures that the clock counters inside the receiving and sending CPNs are synchronized. In case of system power on or reset as illustrated in Figure 5.10 (a), the clock counters of CPN B, C, and D, which run slightly later than A, are synchronized with that of the CPN A. Similarly, in case one CPN is reset, the CPN synchronizes itself to the other CPNs as illustrated in Figure 5.10 (b). Once more than one CPNs are already synchronized, they behave just like a single CPN. In this way, synchronization can be done even in case more than one CPNs are randomly reset. The trick in this mechanism is that the first “Sync” packet is suppressed and is not really sent to others (Figure 5.10). This mechanism ensures that joining CPNs synchronize themselves to already synchronized CPNs.

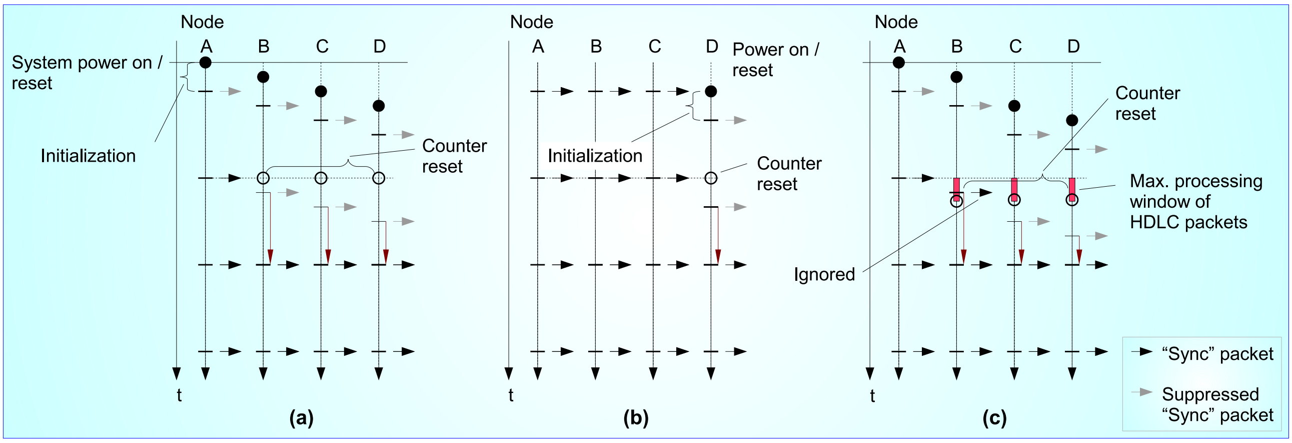


Figure 5.10.: Mechanism of node synchronization

The exceptional case is that more than one CPNs start synchronization procedures almost simultaneously with a deviation of around 100 ns. In this case, more than one "Sync" packets are sent because of the finite processing period of an HDLC packet. If this happens, the second packet is just ignored as illustrated in Figure 5.10 (c). This feature also ensures that already synchronized CPNs are not influenced by newly joining CPNs.

Clock signal selection and distribution

CDV performs clock signal selection and distribution. CDV receives one Ultra Stable Oscillator (USO) clock signal "USOCLK" and four clock signals from the internal oscillator "INTCLK" on the CPNs. CDV selects one of them and distribute it to the four CPNs so that all CPNs receive the same reference clock signal "REFCLK". The schematics of this operation is illustrated in Figure 5.11. After the orbit insertion, the USO is always powered-on. This component is already space qualified. However, in case of absence of "USOCLK", one of the "INTCLK" is selected and distributed. The oscillators on CPNs are designed in such a way that the "INTCLK" is always synchronized with the "REFCLK." This mechanism ensures smooth switching between the reference clock signals without any phase mismatch at the CDV.

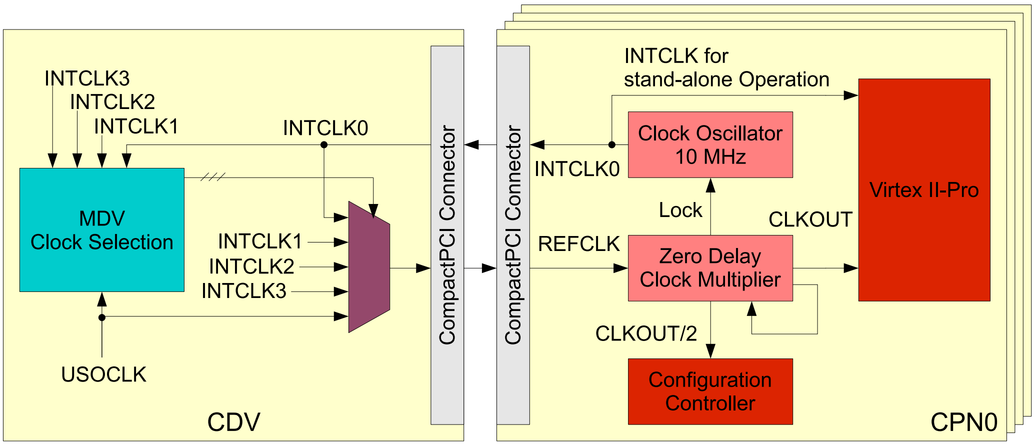


Figure 5.11.: Clock signal interface between CDV and CPNs

Real time clock monitoring and distribution

The CDV is also equipped with triple redundant, on-board Real Time Clocks (RTCs). This RTCs is used to store and propagate the clock from a given point in orbit. Real time information can be obtained from the on-board GPS receivers (triple redundant) and command uplink. Every time GPS is powered on, or experiencing ground contact, the value is reset. This RTC is not mission critical.

Configuration management of CPNs

All CPNs are connected with the MDV through their own special bidirectional channels for configuration control of the main SRAM-FPGA. MDV shall conduct

- programming of configuration Flash memories 0, 1, and 2 on CPNs,
- reconfiguration of the SRAM-FPGA in a streaming mode,
- selection of a Flash memory from which the SRAM-FPGA shall be reconfigured, and
- erasure of configuration data in Flash memories.

The three NOR-Flash memories on CPNs can be used as triple module redundancy, and also single separate configuration memories. MDV can forward received new configuration files toward these Flash memories or directly reconfigure the SRAM-FPGA on CPNs in a streaming mode. MDV can also select one Flash memory from which the SRAM-FPGA shall be reconfigured. MDV can also erase the data in the Flash memories. All these operations are conducted using predefined communication packages between CDV and CPNs.

5.6.4. Data handling support FPGA

DHS is responsible to telemetry data and memory management. The telemetry data of the *Flying Laptop* is classified into housekeeping data and scientific data. Between the MDV and DHS FPGAs bidirectional dedicated data transfer interface is implemented and the housekeeping data gathered at the MDV is once transported into the DHS and saved into the Flash mass memory. The main data transport channels are “On-board Data Tx/Rx.” The Hash signal channels are also connected with DHS and can be also used for data transfer as back-up channels. DHS performs the data handling received from the four CPN, the memory management and the data traffic between the master CPN and the mass memory. Thus this FPGA contains a file management capability for managing this data stream. Moreover, it saves the housekeeping data of the last three orbits into the mass memory.

5.6.5. Mass memory

The mass memory is selected according to the investigation of radiation effects on different types of Flash memories by Micron, Hynix and Samsung [104]. The 4 Gbit Flash memory by Samsung revealed its excellent performance against TID effects. Detailed experimental results can be seen in [105]. The high performance of the Samsung NAND-Flash memory against Toshiba NOR and Spansion NOR, STMicroelectronics NAND-Flash memories is also reported by the NASA in [106]. Detailed analysis of TID effects on Flash memories can be seen in [107].

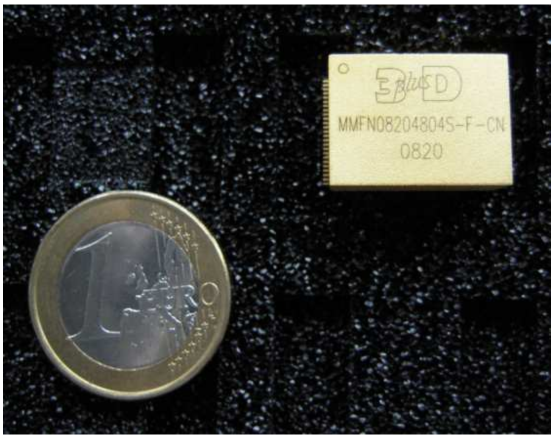


Figure 5.12.: Flash memory (EM)

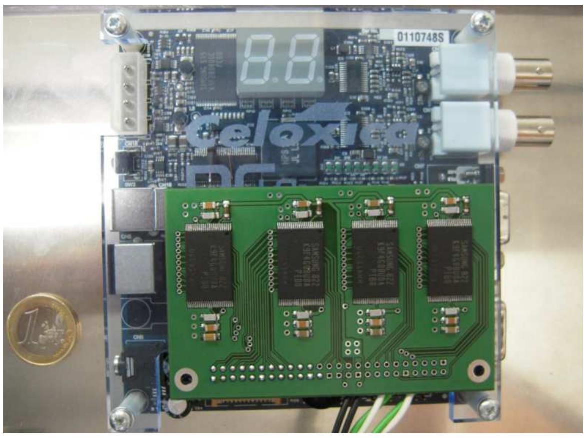


Figure 5.13.: Flash memory (BBM)

The selected Flash memory is the 16 Gbit (2 GB) NAND-Flash memory by 3D PLUS (Figure 5.12) [108]. This memory is internally based on four of the above mentioned 4 Gbit Samsung Flash memories [109]. The CDV is equipped with two of these Flash memories. For the development purpose of control algorithm, functionally equal engineering model with the same electrical interface is developed based on the original Samsung memory (Figure 5.13). One module is used for scientific data and the other is used for housekeeping data.

5.7. Design implementation of backplane

The limitation in pin numbers comes from the maximum quantity of the pins of CompactPCI connectors mounted on the Eurocards. By having four CPNs and one CDV, as well as one CIB, the total quantity of pins routed through the BPL exceeds 1800 (Figure 5.14). This integrated dense electrical interconnects allows the over-all satellite design concept. Interface lines of LVDM are designed to allow 250 Mbps communications. Because the CPNs are connected with the satellite peripheral electronics in a bus configuration, termination resistors are mounted on the back side of the BPL in order to realize identical manufacturing of the CPNs (Figure 5.15).

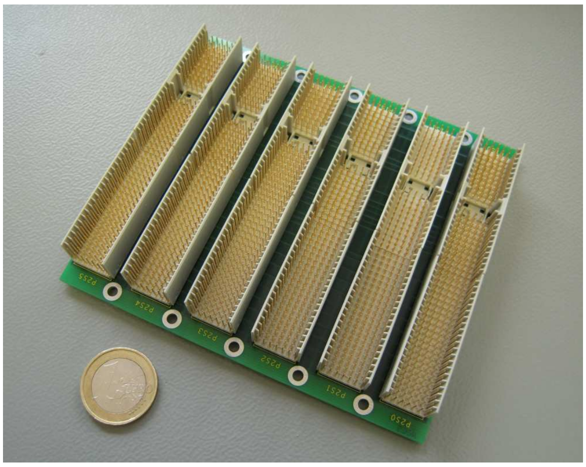


Figure 5.14.: Backplane (EM)

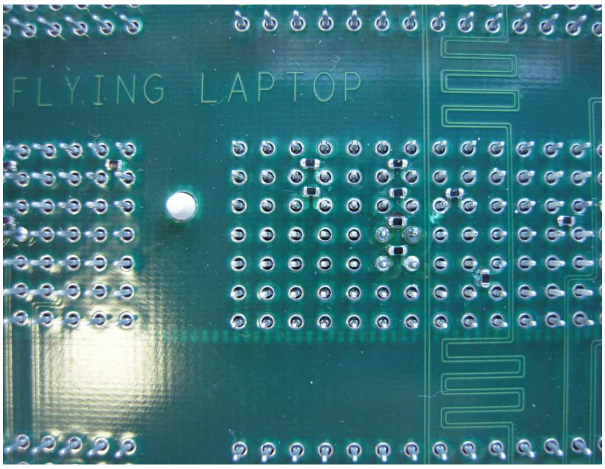


Figure 5.15.: Termination resistors

5.8. Summary

As the consequence of the above described hardware development activities, the breadboard model of the OBC is developed and assembled as illustrated in Figure 5.16. This BBM assembly demonstrates the electrical functionality and is the fundamental basis of the control algorithm development of the OBC of the *Flying Laptop*, which is described in detail in Chapter 6.

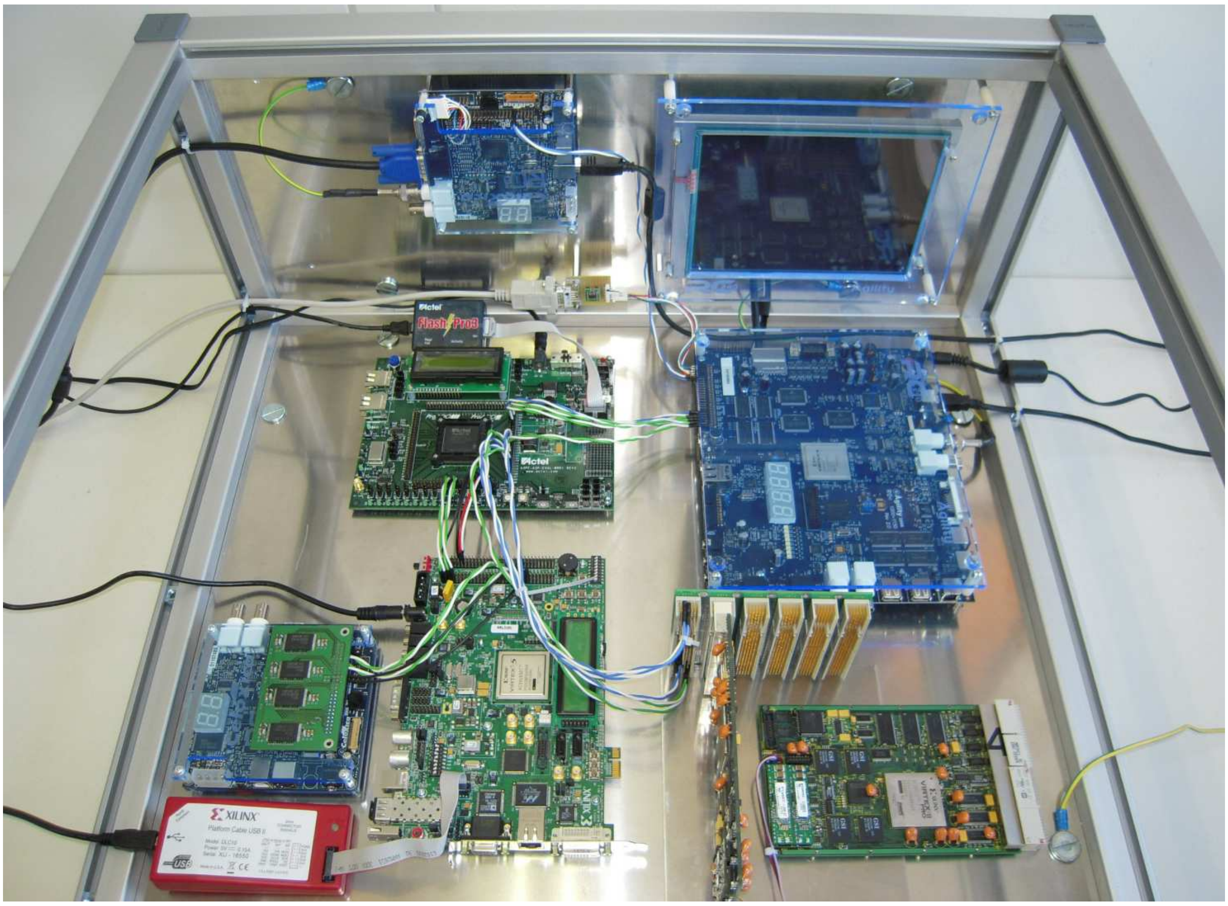


Figure 5.16.: Breadboard model of the on-board computer

According to the BBM-EM-FM model philosophy of the OBC development, most of the components applied for the BBM are electrically equivalent to the FM. For the agile development purposes, several FPGA development boards such as RC10 and RC240 of Agility Design Solutions are integrated in the model. The Flash-FPGAs on the CDV are replaced with a commercial ProASIC3 with a support of one Virtex-5 chip. This combination shall reduce the development cost as well as accelerate the control algorithm development in Handel-C followed by porting through VHDL into Flash-FPGAs. All components of the BBM can be gradually upgraded with EM and FM, according to the progress of hardware development. Indeed, EMs of the CPN are integrated in this environment to verify their functionalities. The architecture of the control algorithms are designed in a portable manner (see Chapter 6) so that the developed and verified control algorithms by BBM can be further implemented into EM and FM. The design activities based on this BBM shall produce detailed requirements and implementation techniques for EM and FM. This model is also used for the verification of developed control algorithms with the simulation and verification environment described in Chapter 7.

6. Design of OBC control algorithm

The hardware logic inside FPGAs are different from the traditional software which is executed in a processor. In the scope of this study the hardware logic designed for FPGAs are denoted as Control Algorithm (CA), indicating the difference from software. Control algorithms can be designed by software tools and can be updated even after the launch. The significant difference in designing CA is that it is no longer software design but hardware design. Therefore, the designer of CA requires complete knowledge about the behavior of hardware components inside the target FPGA as well as types, sizes, timing, and performances of available hardware resources. Especially, the size of designed hardware logic shall be optimized to be as small as possible, so that it can fit inside the target FPGAs. Moreover, due to the massive parallelism of the FPGA architectures, the CA shall be implemented in a parallel manner. In other words, in order to maximize the capability of FPGA devices, the CA shall be as parallel as possible, otherwise lower clock frequency of FPGAs becomes demerit compared to processors.

In the beginning of this chapter, internal hardware resources of SRAM- and Flash-FPGAs are briefly summarized. After that the developed design methods and the development environment of CA for the *Flying Laptop* are described.

6.1. Mechanism of SRAM-FPGA

In this chapter the internal hardware resources of the Virtex-II Pro SRAM-FPGA by Xilinx is briefly summarized with an emphasis on programmable logics. Even though, there are several differences in size and performance between the different families, general mechanism of the Virtex series are quite similar. The mechanism of SRAM-FPGA products of other companies can be regarded as comparable.

The CA of the SRAM-FPGA is stored in external memory devices such as Flash-memory or EEPROM. After powered on, the program stored there are loaded into the FPGA. The files which are called in this process are called bit stream data. This information is loaded into configuration memory in the FPGA, which consists of 1 bit column SRAM memory cell. This loaded information becomes the source of FPGA configuration. The internal structure of the FPGA can be classified into followings:

- Configurable Logic Block (CLB)
- Multipliers (Special Purpose Logic)
- Block RAM (BRAM)
- Digital Clock Manager (DCM) and clock network
- I/O Block (IOB)
- Programmable Interconnects

The internal block diagram of an FPGA is illustrated in Figure 6.1.

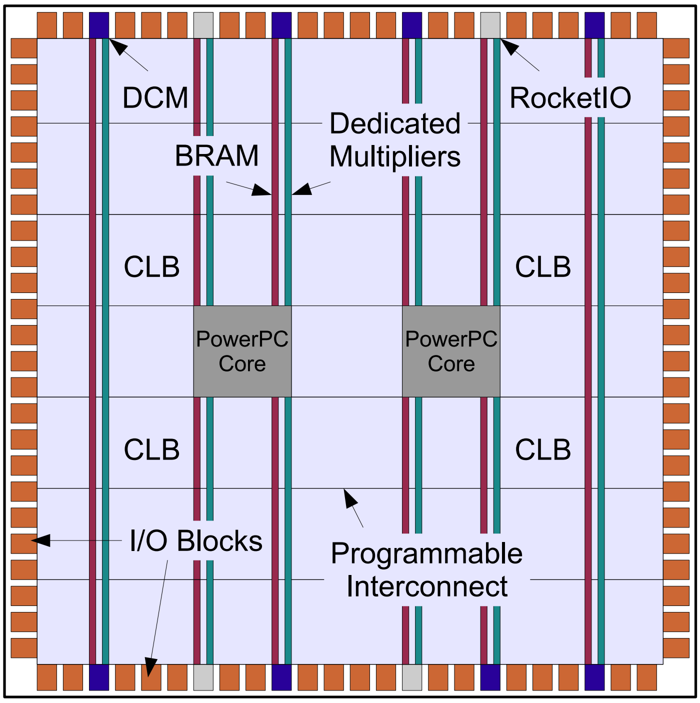


Figure 6.1.: Mechanism of SRAM-FPGA [53]

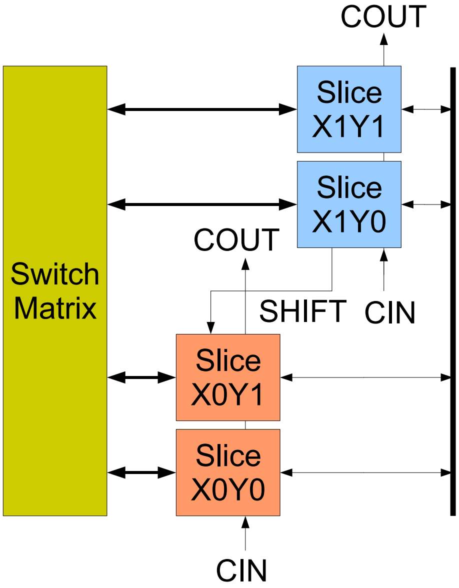


Figure 6.2.: Configuration logic block [53]

6.1.1. Configurable logic block

The configurable logic is the central programmable element of an FPGA. Most of the user logic are implemented into this element. CLBs are organized in an array and used to build combinational and synchronous logic designs. A CLB element of Virtex-II Pro comprises 4 slices which are split into two columns of two slices (Figure 6.2).

Function generators

Each slice consists of two 4-input function generators, carry logic, arithmetic logic gates, multiplexers and two storage elements [53] (Figure 6.3). The function generators are implemented as 4-input Look-Up Tables (LUTs), which can also be used as distributed SelectRAM+ memory or 16-bit variable-tap shift register element. LUTs are able to implement any arbitrary boolean function of four inputs. The output signals of LUTs can be routed either to the data input (D) of the storage element, or to the MUXF5, or to the carry-logic multiplexer, or to the XOR gates, or can exit the slice from X or Y output. Each slice contains one MUXF5 and one of either MUXF6, MUXF7, or MUXF8 multiplexer. These multiplexers enable combinational use of function generators across the slices.

Storage element

Storage elements can be configured either as flip-flops (FFs) or as level sensitive latches. Tow storage elements inside a slice have common clock (CK), clock enable (CE), and set/reset (SR) input signals. FFs are essential for step-wise logic operation in sequential clock cycles.

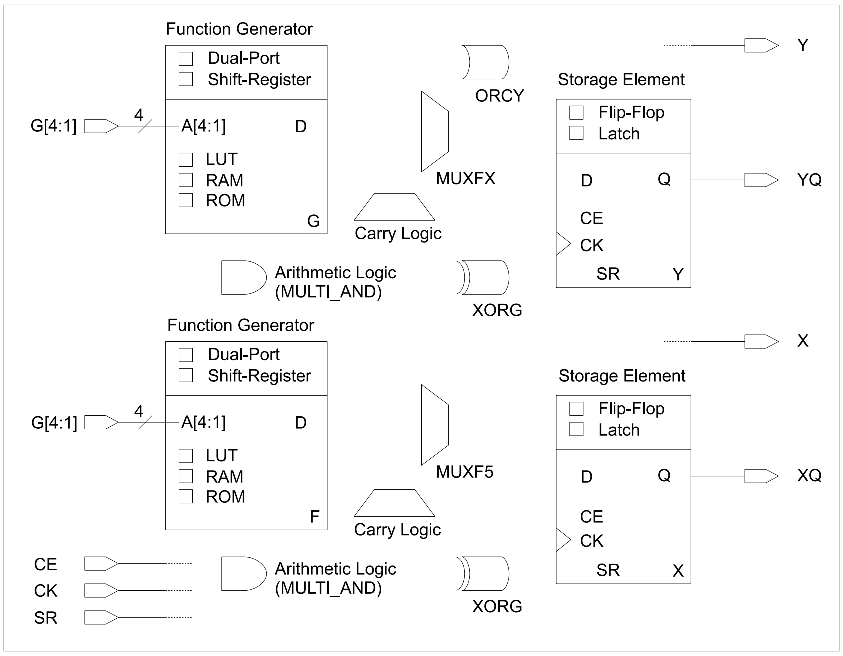


Figure 6.3.: Structure of a slice

6.1.2. Peripheral hardware resources

Besides CLBs, Virtex-II Pro is equipped with following hardware resources as illustrated in Figure 6.1. Detailed information can be found in [53].

Multipliers: Virtex-II Pro multipliers, also called as special purpose logic, are optimized for high-speed operations with a lower power consumption compared to an 18-bit × 18-bit multiplier in slices. Complex signal processing such as arithmetic calculations can be performed.

Block RAM: Virtex-II Pro devices incorporate large amount of Block RAM resources. Each BRAM is an 18 Kb true dual-port RAM with two independently clocked and independently controlled synchronous ports that access a common storage area. BRAM also supports various configurations, including single- and dual-port RAM and various data/address aspect ratios.

Digital clock manager: For the internal clock management, Virtex-II Pro is equipped with Digital Clock Managers (DCMs). DCMs can generate new clocks with clock multiplication/division and phase shifting. DCMs allow to have multiple clock domains inside an FPGA.

I/O Block: I/O blocks are in groups and provide external interfaces. Each IOB can be used as input and/or output for single-ended I/Os. Two IOB can be used as a differential pair.

Programmable Interconnects: Most of the signals inside an FPGA are routed by means of programmable interconnects through the device. These global routing resources are located in horizontal and vertical routing channels between each switch matrix.

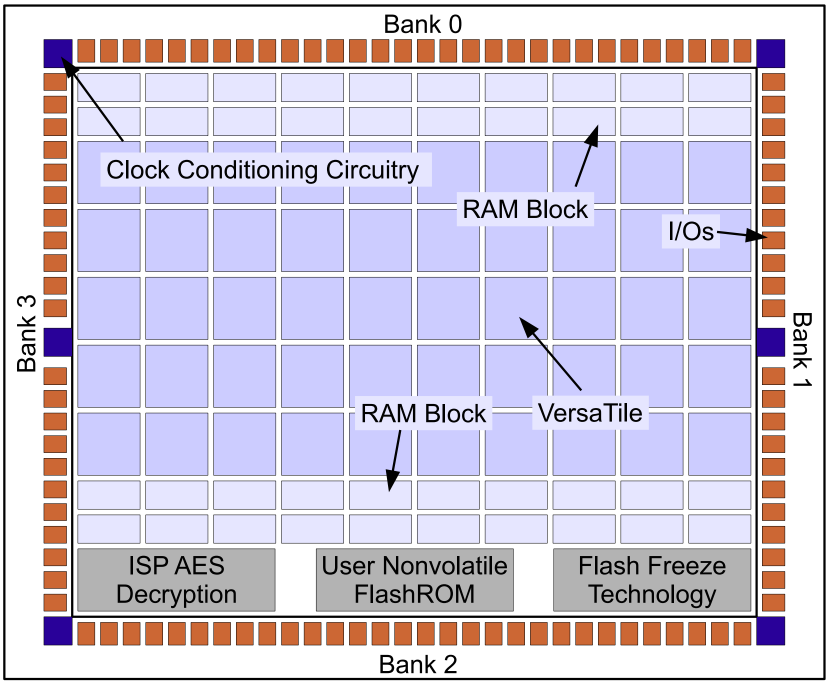


Figure 6.4.: Mechanism of Flash-FPGA [17]

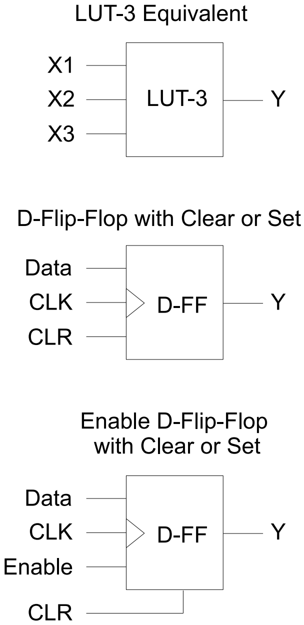

LUT-3 Equivalent
X1
X2
X3
LUT-3
Y
D-Flip-Flop with Clear or Set
Data
CLK
CLR
D-FF
Y
Enable D-Flip-Flop with Clear or Set
Data
CLK
Enable
CLR
D-FF
Y

Figure 6.5.: LUT configurations

6.2. Mechanism of Flash-FPGA

The internal structure of the RTProASIC chips are illustrated in Figure 6.4 [17]. The structure of RTProASIC Flash-FPGA is simpler than that of the Virtex-II Pro SRAM-FPGA. The main programmable logic part is the VersaTiles. This VersaTile supports:

- All 3-input logic functions - LUT-3 equivalent
- Latch with clear or set
- D-flip-flop with clear or set
- Enable D-flip-flop with clear or set.

VersaTiles generate user defined functionalities similar to the CLBc in Xilinx FPGA. The Flash-FPGA also incorporates RAM Blocks based on SRAMs, however, based on the requirements of high reliability, it is not used for reliable logic in the scope of this research. In addition to these, the chip also offers 1 kbit of on-chip, user accessible nonvolatile FlashROM. This can be used in diverse system applications, for example:

- System calibration settings
- Secure key storage for secure communications algorithms
- Date stamping

Furthermore, RTProASIC devices offer Flash*Freeze technology, which enables the devices to be instantaneously shut off dynamic power consumption while retaining all SRAM and register information. Consequently, most of the design effort of Flash-FPGA in the scope of this research is done for VersaTile.

6.3. Control algorithm design methods

6.3.1. Hardware description languages

The selected hardware description languages (HDLs) for the development of control algorithms of *Flying Laptop* are VHDL and Handel-C. The VHDL is a low level HDL and stands for Very-high-speed integrated circuit HDL. VHDL has constructs to handle the parallelism inherent in concurrent hardware designs and is able to design complex, tailor-made hardware logics. Designers can specify every single gate or flip-flop built and manipulate the propagation delays of signals throughout the system. VHDL is a dataflow language whose processes are parallel by default.

The second language Handel-C is a high level HDL, which is originally developed by the Oxford University Computing Laboratory [110], and is a product of Mentor Graphics at the time of writing. Handel-C is a programming language for compiling programs into hardware images in SRAM-FPGAs. It is a rich subset of ANSI C, with non-standard extensions to control hardware instantiation and parallelism. Even though, floating point data types were omitted, floating point arithmetic can be supported through external libraries. Opposed to existing other simple C-translators for FPGAs, Handel-C targets hardware directly, and further provides hardware optimizing features. A big advantage, compared to the C-translators, is that variables and constants can be given a certain width, as small as one bit. Also, Handel-C provides bit manipulation operators and the possibility of parallel processing of single statements or whole modules. This can not be realized with other approaches based on a sequential language.

6.3.2. Development environment of CPN

The programming of CPNs is based on the DK design suits of Mentor Graphics, which is the programming environment of Handel-C. In contrast to VHDL, the focus of Handel-C is on fast prototyping and design optimization at the algorithmic level. Low-level problems are hidden completely and all gate-level decisions and optimizations are done by the compiler. It also provides simulation capabilities of the circuit behavior at the algorithmic level, based on the Handel-C semantics. Designers can track the values of variables throughout the simulations, which is very attractive to satellite control algorithm implementation.

Handel-C has not ever been applied for on-board computers of spacecraft, therefore this thesis also focuses on investigating the capability and feasibility of applying Handel-C for space systems to realize complete satellite control functions in FPGA chips. Though the programming with Handel-C is very much similar to programming, designers require hardware logic level knowledge. The mechanism of hardware mapping by Handel-C is separately investigated and summarized in Appendix E.

6.3.3. Development environment of CDV

For the development of hardware logic inside Flash-FPGAs, both of the Handel-C and VHDL are applied. Even though, the Handel-C is attractive, it does not directly support Flash-FPGAs nor Antifuse-FPGAs. However, Handel-C also supports VHDL output and this can be further used as input files into Actel's VHDL programming environment Libero IDE.

6.4. CPN Control algorithm architecture

The strictly limiting factor in implementing traditional satellite control functions into FPGAs is the amount of hardware resources available inside an FPGA chip. The designed control algorithm shall as compact as possible in terms of generated hardware to fit into the targeting device. In order to make the idea come true, there are several important criteria which influence the design of the control algorithm architecture.

The first criterion is the portability. The developed CA shall be portable into different platforms, for example not only into the flight model hardware, but also into the engineering model or any kind of verification model, so that the verified control algorithm can be directly ported into the real flight configuration. This is indispensable to achieve a higher verifiablility of the developed functionality of the CA prior to the actual implementation into the FM. This criteria accelerates a modular design of CAs.

The second criterion is the parallelism. The computational performance and efficiency of FPGA devices are depending on the degree of internal parallelization of the algorithm. Due to their inherent low clock frequencies relative to modern processors, implementing a purely sequential process can not exploit the potential capability of FPGA technologies. To achieve high computational throughputs, significant parallelization shall be introduced. This criterion invokes innovative parallelization techniques unlike usual satellite software in sequential programs.

The control algorithm architecture of the *Flying Laptop* is designed to fulfill above requirements. First of all, in order to realize a high portability, abstraction layers are introduced. Secondly, parallelization techniques of satellite control functions are developed.

6.4.1. Vertical layered structure

The first decomposition of the control algorithm is done in the vertical direction in terms of the level of abstraction. An abstraction layer is originally a concept of software architecture. It is a way of hiding implementation details of a particular functionality. This helps to make applications independent from platforms by well defined interfaces between the layers. In order to communicate with the outer world (read and write from/to devices) at the application level, the program uses low level functions [111].

Since there is no operating system or firmware on an FPGA, the designed layers of the control algorithm are, from lower level layer to the higher:

- PSL - Platform Support Library,
- PAL API - Platform Abstraction Layer Application Programming Interface,
- PAL-Core - Platform Abstraction Layer Core, and
- Application.

This is also illustrated in Figure 6.6. This four level approach conceals the low level functions and platform dependent interfaces from application programmers and leads to a great portability of the application algorithms. The functionality of each layer is described below.

Platform support library: The platform support library provides accesses to the platform dependent external hardware resources, such as I/O devices and on-board memories. The PSL

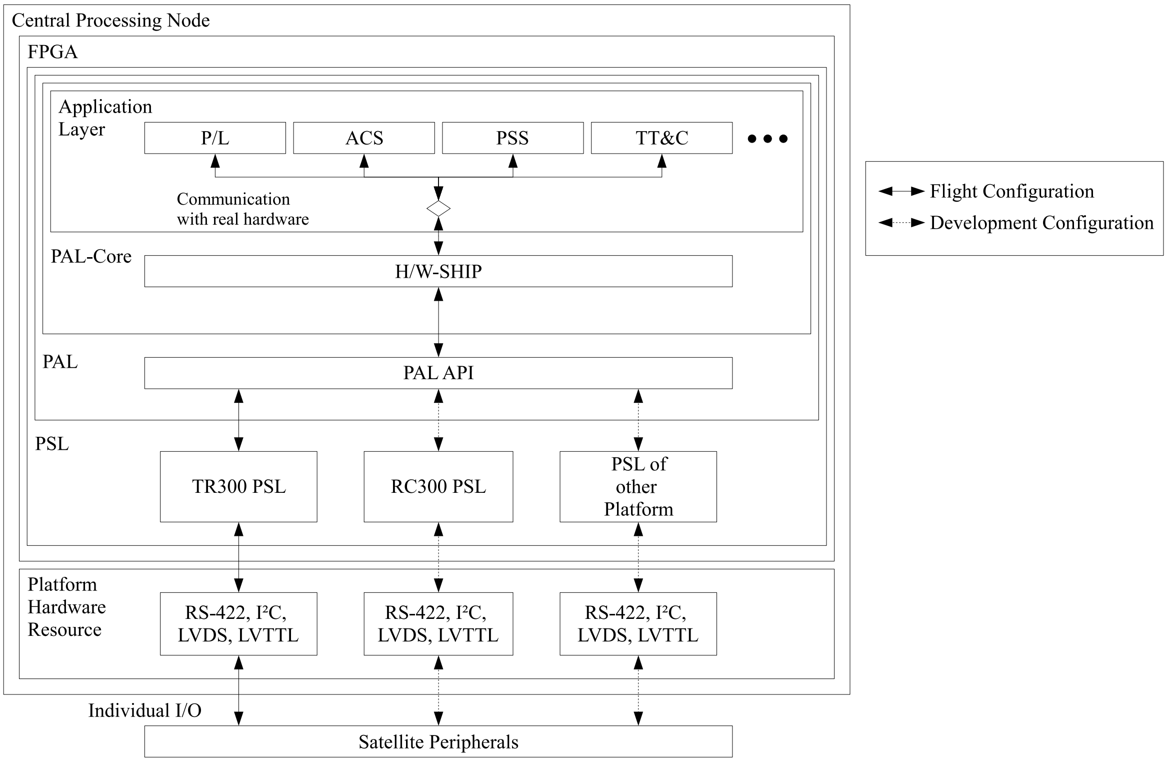


Figure 6.6.: Layered control algorithm structure

is comparable to the board support package provided by embedded operating system vendors. Once a PSL has been implemented, a PAL can be developed to provide accesses to the PSL using the PAL standard API calls.

Platform abstraction layer API: The Platform Abstraction Layer is divided into the PAL API and the PAL Core. The PAL API provides generic and easy-to-use access functions to the PSL and thus shields the programmer from low-level and hardware specific programming. PAL-API represents the resources of the board through a set of functions according to the specific functionality of the resources.

Platform abstraction layer core: The PAL-Core, also called as Satellite Hardware Interface Protocol (SHIP), is an implementation specific layer, in which every satellite component is represented by relevant variables and functions, enabling application programmers to access the peripherals without knowing their exact specifications. For example, periodical sensor data collection is implemented in this layer. For example, the details of the SHIP for the attitude control system is specified in [112].

Application layer: The application layer is the highest layer where all user specific computing, such as satellite control algorithm, attitude control algorithm, telecommunication algorithms, FDIR routines, and every mission specific processing are implemented. The layer is fully platform independent.

6.4.2. Sequential and parallel processes

Traditional satellite control algorithms are sequential processes with partial time-divided parallel execution. The target CA of the *Flying Laptop* shall perform ACS algorithms (periodical), payload instrument operations (continuous), telecommunications (periodically continuous), and housekeeping (periodical). Due to the limited maximum clock frequency of FPGA up to several hundreds of MHz, implementing these processes in a traditional sequential way is not feasible, and therefore, parallelism of the FPGA shall be sufficiently exploited. This section describes the implementation methods of both sequential and parallel processes in an FPGA.

Sequential processes in FPGA

Unlike software, each piece of CA is implemented as a part of huge state-machines inside a hardware logic. Indeed, DK design suit translates CA into complex state-machines. A process in Handel-C can call sub-processes in a form of macro expressions. It is still possible that a sequential process calls two parallel processes in parallel. In order to come back to the original process, both of the sub-processes shall terminate. In the following example, the “sequentialProcess3” will be executed only after both sub-processes (1&2) were terminated.

```
do { // parent sequential process
    par{
        sequentialProcess1;
        sequentialProcess2;
    }
    sequentialProcess3;
} while(1);
```

The demerit of this structure is that the shorter sub-process freezes until the longer terminated. Furthermore, if one of the two processes goes into a dead-lock, the whole process freezes. Consequently, functions such as interrupt and watch-dog timer shall be implemented.

Interrupt implementation: The only way of implementing interrupt functions into Handel-C code, is to let the parent process periodically monitor the existence of interrupt events, for example timeout report from a watch-dog timer. Otherwise, there is no possibility to cut in the middle of a sequential process. For this purpose, one can use “prialt” construct (see Appendix E.10) to monitor the existence of events without loosing extra time for it in the following method:

```
prialt {
    chWD ? a: WDStatement; break;
    default: break;
}
statement1:
```

If the chWD is going to be written by a watch-dog timer, the channel communication is activated followed by the WDStatement, and then statement1. If there is no watch dog event, the default branch is activated and statement1 is executed immediately in the very same clock cycle.

Watch-dog timer implementation: One can implement a watch-dog timer as described below. Note that this while loop goes through the whole process once in a clock cycle. If the counter reaches to the "upperLimit," then it tries to perform channel communication with the main process. At the same time, it resets the value of the counter to 0. Until the communication is actually conducted, the watch-dog timer waits for the channel communication. If the value of the counter is not yet equal to that of "upperLimit," then it resets the value of the counter to 0 in case it receives a counter reset request from another process, otherwise it increments the counter by one. By applying these techniques, CAs can be implemented as sequential processes with partial parallelism on demand.

```
do { // watch-dog timer
    if (counter==upperLimit) {
        par {
            chWD ! 1; // wait until chWD? is ready
            counter = 0; // reset
        }
    } else {
        prialt {
            chReset ? b: counter = 0; break;
            default: counter++; break;
        }
    }
} while(1);
```

Parallel processes in FPGA

Firstly, one can produce parallel processes by declaring multiple main's. In Handel-C one can use more than one main functions, which are implemented as independent parallel processes. One can even run these main's in different clock domains. The second way to declare parallel processes is to use "par{}" syntax. The difference between these two methods is that declaring multiple main's can create independent processes, while "par{}" can declare either sequential processes which come back to the parent process, or while-loop processes which will never come back to the parent process. This is illustrated in the following example:

```
seq { // parent process       loop1() {              loop2() {
    par{                          do{                    do{
        loop1();                      statement1;            statement2;
        loop2();                      ch ! a;                ch ? b;
    }                             }while (1);                delay;
    statement3;                }                          }while (1);
}                                                    }
```

In this example, the statement3 will never be executed. This latter method is the only way to declare (interacting) independent parallel processes in the scope of single main process. Indeed, these "do-while" loops are the basic processing units inside an FPGA which provide continuous functions. Consequently, the whole scheme of FPGA CA can be regarded as a combination of multiple "do-while" (- equivalent) loops, which communicate with each other.

6.4.3. Decomposition into parallel processes

By applying the above method, one can decompose a main function into a combination of repetitive (loops) and non-repetitive (sequential) processes. Because loops are the fundamental units, one can illustrate this decomposition in terms of loops. The following decomposition code generates the left hand side figure of Figure 6.7.

```
main(){
    initialization();
    par{
        A(); B(); C(); D(); E(); F(); G();
    }
}
```

Here the “initialization()” is the initialization function for environment variables, and processes in “par{}” are loops. In the figure, the exemplary communication lines between processes are also illustrated. As is clear from this figure, the more processes there are, the more complex the entire architecture becomes, which makes the verification, especially partial verification of the control algorithm extremely difficult.

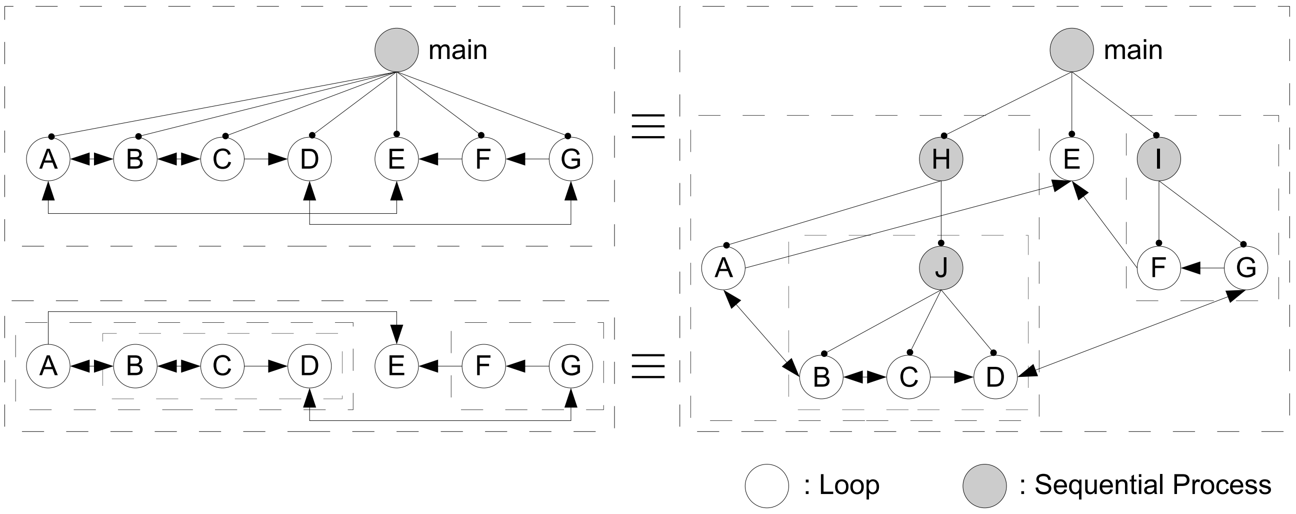


Figure 6.7.: Decomposition of control algorithms

On the contrary, the following codes generate the processes illustrated in the right hand side of Figure 6.7, which is actually the same architecture as that of above example.

```
main(){              H(){              I(){              J(){
    initMain();          initH();          initI();          initJ();
    par{                 par{              par{              par{
        H();                 A();              F();              B();
        E();                 J();              G();              C();
        I();             }                 }                     D();
    }                }                 }                     }
}                                                        }
```

The main, H, I, and J are sequential processes and the functions with“init” are initialization functions specially designed for the process group. Processes vanish after they called loop processes.

The merit of latter method is, firstly, that the loop processes are classified into several groups, which helps to get the picture of the whole complex control algorithms. Secondly, the initialization functions, which can be specially designed for each group encapsulate variables and constructs such as signals and channels from others. One can clearly define the interfaces and see the scope of groups. Thirdly, by making the group according to their functionalities, one can easily identify functional grains and develop them as modular and verify each of them separately from the others. In the following section, this idea is further extended by introducing a multi-agent programming method.

6.4.4. Multi-agent programming

In this chapter the decomposition methodology of the CA architecture according to the functional units based on multi-agent approach is described. Multi-agent systems are very closely related with Distributed Artificial Intelligence (DAI). The history of DAI starts from mid 1970s and it is an established and promising research and application field which brings together and draws on results, concepts, and ideas from many disciplines, including artificial intelligence (AI), computer science, sociology, economics, organization and management science, and philosophy [113]. In [113] it is described as:

“DAI is the study, construction, and application of multiagent systems, that is systems in which several interacting intelligent agents pursue some set of goals or perform some set of tasks.”

“An agent is a computational entity such as a software program or a robot that can be viewed as perceiving and acting upon its environment and that is autonomous in that its behavior at least partially depends on its own experience.”

Now it is worth considering an example of agents. A simplest example selected here is a thermostat, which keeps the room temperature higher than a limit temperature. The thermostat has a temperature sensor for measuring room temperature, and a heater to heat the room. Based on the information from the temperature sensor, the thermostat makes decisions, either “temperature lower than limit temperature” or “temperature higher than limit temperature.” The heater simply heats the room when it receives the decision, that is:

$$\begin{aligned} \text{temperature lower than the limit temperature} &\rightarrow \text{heater on} \\ \text{temperature higher than the limit temperature} &\rightarrow \text{heater off} \end{aligned}$$

Clearly, this thermostat can be regarded as an agent based on the above definition. As a part of the satellite controlling functions exactly the same function as the above example, which maintains the satellite’s temperature instead of the room temperature, needs to be implemented. This kind of function can be implemented as an agent inside the FPGA.

Introduction of compositional multi-agent system

For the purpose of implementing CA of *Flying Laptop* into an FPGA, the concept of multi-agent system is extended to be compositional, in which an agent can be a composition of multiple lower level agents. This idea is illustrated in Figure 6.8. The architecture in the figure is corresponding to that of Figure 6.7. All circles in the figure represent each agent.

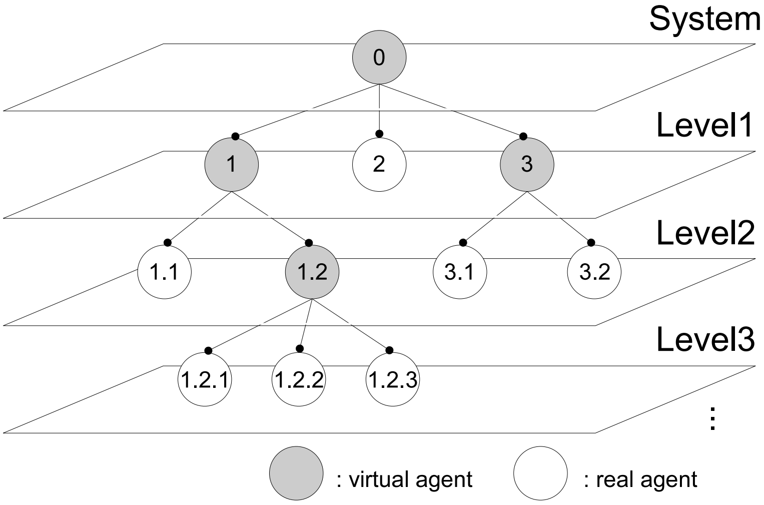


Figure 6.8.: Compositional multi-agent System

In this figure the system consists of three agents 1, 2, and 3, and the agent 1 further consists of agent 1.1 and 1.2, and so forth. Because the substance of the system does not exists, (it is actually a composition of three agents) the circle is marked as "virtual" agent. By comparing this figure with the Figure 6.7, one can recognize the analogy between them. Indeed, by regarding the functional units of loop processes as agents, the whole CA can be understood as a compositional multi-agent system. For example, the temperature sensor and the heater in the above example of an agent, can be also regarded as lower level agents of the "thermostat agent."

Decomposition of satellite system

As described above sections, the decomposition of the CA shall be done in terms of functional units. A satellite system can be decomposed into multi-agents as followings:

- level 1: subsystems and additional high level functions such as mission scheduler
- level 2: units of components such as magnetometers and camera instruments, as well as subsystem control functions such as attitude control algorithms, telecommunication algorithms
- level 3: unit level control algorithms such as a magnetometer

All these agents run in parallel and communicate with each other. The another merit of using multi-agent approach is that a failure in process can be easily allocated with the corresponding functional unit, which makes the failure management clearer. However, because more than one processes are running in parallel, it is difficult to manage an consistent FDIR activities. FDIR concept shall be always taken into account in the architecture design.

6.4.5. FDIR structure

FDIR stands for Fault Detection, Isolation and Recovery. The control algorithm shall detect all failure sources and types, which have been identified through system design activities, and isolate and recover them. The failure sources are usually identified by FMECA (Failure Mode, Effects, and Criticality Analysis) activities. Because agents can face to failures, we need a concept which can organize these events, and corresponding reactions in a total manner [114].

Fault tree

A Fault Tree (FT) is a picture of the Boolean function describing the way in which the n binary components states jointly determine the binary system state. For the detail of Boolean functions, see Appendix B.5. An FT is an excellent tool to connect system level fault with component level faults, also providing the reasoning at the same time. In Figure 6.9, a part of the FT of BRDF measurement is illustrated. As illustrated in the figure, the system level fault is reasoned by logical connection with lower level faults. It is worth noting that this implementation into FPGAs costs only simple logical connection with most of the part in wires. In Figure 6.9, from the definition of indicator variables, $X_s = 1$ indicates that the BRDF is not ready.

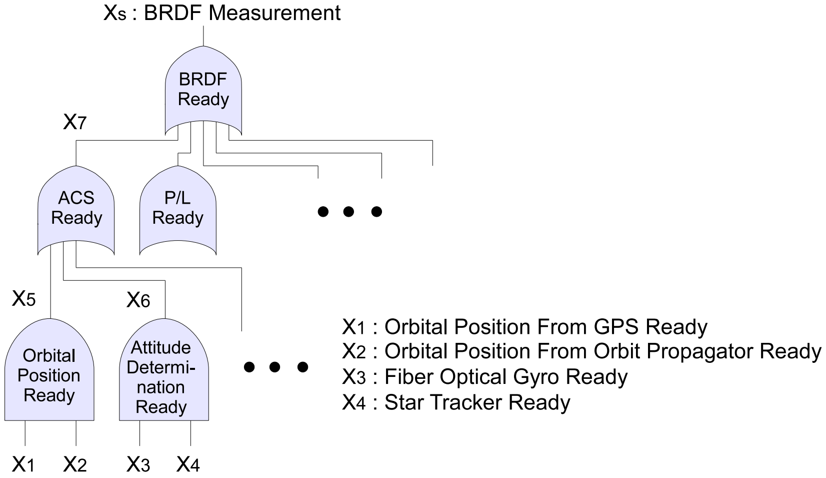

Xs : BRDF Measurement
BRDF Ready
X7
ACS Ready
P/L Ready
X5
X6
Orbital Position Ready
Attitude Determi-nation Ready
X1 X2 X3 X4
X1 : Orbital Position From GPS Ready
X2 : Orbital Position From Orbit Propagator Ready
X3 : Fiber Optical Gyro Ready
X4 : Star Tracker Ready

Figure 6.9.: Fault tree of a BRDF measurement

Fault tree boolean function

FTs can be represented by Boolean functions as a part of the FDIR function. The purpose of these Boolean functions is to built bridges between the system level faults and component level faults in an FPGA hardware logic.

Generally, the status information of a target component can be described by a state vector $\mathbf{V}$, which is a n-bit binary variable. For example, $\mathbf{V}_{GPS}$ contains all status information about the GPS component in a binary form. By introducing Boolean function φ, the status of an one level higher binary indicator variable can be reasoned by the combinational logic based on the status of the lower level components, that is:

$$X_s = \varphi(X_1, \ldots, X_n) \ . \tag{6.1}$$

The state vector, which is a binary variable, can be also regarded as an adjacency matrix of Boolean indicators, such that:

$$\mathbf{V} \equiv \underline{X} \equiv (X_1, \cdots, X_n) \ . \tag{6.2}$$

Introducing one more Boolean function ψ, which resolves state vector into boolean indicator

$$X = \psi(\mathbf{V}) \ , \tag{6.3}$$

the FT in Figure 6.9 can be simply described as

$$X_1 = X_{Orb.Pos.Gps\_Ready} = \psi(\mathbf{V}_{GPS}) \ , \tag{6.4}$$

$$X_5 = X_{Orb.Pos\_Ready} = \varphi(X_{Orb.Pos.Gps\_Ready}, X_{Orb.Pos.Orb.Prop\_Ready}) \ , \tag{6.5}$$

$$X_7 = X_{ACS\_Ready} = \varphi(X_{Orb.Pos\_Ready}, X_{Att.Det\_Ready}, \ldots) \ . \tag{6.6}$$

Finally,

$$X_s = X_{BRDF\_Ready} = \varphi(X_{ACS\_Ready}, X_{P/L\_Ready}, \ldots) \ . \tag{6.7}$$

Assuming that both X_{ACS_Ready} and X_{P/L_Ready} are 1-bit binary variable, Equation (6.7) can be implemented in Handel-C codes as:

```
X_BRDF = X_ACS || X_P/L || ... ;
```

State vector operation

Suppose there are three heaters 0, 1, and 2, we can generate a state vector (which contains state information of all heaters) from state vectors of each heater by concatenation operations, such that:

$$\mathbf{V}_{heaters} = \mathbf{V}_{heater_0} @ \mathbf{V}_{heater_1} @ \mathbf{V}_{heater_2} \ , \tag{6.8}$$

where @ is the concatenation function (see Appendix E.1). If all heaters are intact (that is $\mathbf{V}_{heater_i} = 0$), the generated state vector is $\mathbf{V}_{heaters} = 0b000$, where "0b" indicates binary variable.

Suppose these three heaters are a 1-out-of-3 redundant system, applying the bit selection operator (see Appendix E.1) the indicator variable of the system can be described as:

```
X_HeaterSystem = (V_Heaters[0]==1)&&(V_Heaters[1]==1)&&(V_Heaters[2]==1);
               = V_Heaters[0] && V_Heaters[1] && V_Heaters[2];
```

Clearly, if all of the heaters are defect, the value of "X_HeaterSystem" becomes "1".

Consideration on layered FDIR functions

Because the entire processes are parallel processes, one shall decide where to place the FDIR function, either top or distributed. If one put a central FDIR function at the top of the whole processes, it looks the same as left hand side of the Figure 6.7. Firstly, one shall deal with complex communications between the FDIR function and all other parallel processes. Secondly, it is hard to keep the modularity of the entire control algorithm, and therefore, it is hard to conduct partial verification.

The solution to this problem is to implement the FDIR function in a layered manner, letting the system level (highest) FDIR function possess the highest authority. This can be achieved implementing an FDIR function to functional grains identified in the right figure of Figure 6.7. For example, a magnetometer units in a 1-out-of-2 configuration is operable as far as one of them is ready. The fault in one sensor is reported as a part of the state vector, however it does not affect system's functionality, because the system can fulfill all required tasks as far as one of two magnetometers is ready. In this case, the internal status of the unit can be encapsulated inside the unit level FDIR function. The system level FDIR function may make a decision to turn the faulty magnetometer off by sending a command. For the realization of this suggested layered FDIR concept, the information exchange between parallel processes shall take place as smooth as possible. The next section describes about the solution to this topic.

6.4.6. Decision making based on information filtering

Logical hierarchy

An agent in a compositional multi-agent system consists of internal parallel processes. Although these processes are running parallel, there are usually logical hierarchy between them. For example, an organizer defines what action the agent shall perform. In addition to this, an agent may receive command from other agents, and also may have FDIR function which generates action requests according to the faults happened. This is illustrated in Figure 6.10.

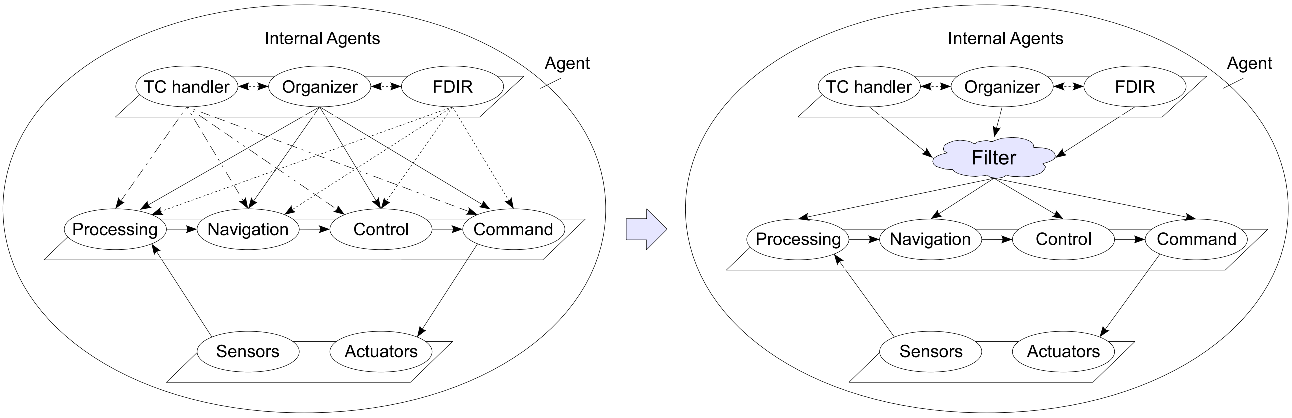


Figure 6.10.: Information filtering for decision making

One can try to implement these top-level processes in a combined single loop process. However, the logic becomes large and long making the reaction speed very slow, because one need to deal with a large number of asynchronous and occasionally simultaneous communications.

Parallel asynchronous reactive system

In this section, the developed new programming method of control algorithms in a form of PARS (Parallel Asynchronous Reactive System) is described. This idea is originated from the subsumption architecture, which was at first introduced by Rodney Brooks in his publication in 1986 [115]. After that it is influencing autonomous robotics and real-time AI. A subsumption architecture is based on bottom-up approach, and can decompose complicated (intelligent) behaviors into simple behavior modules organized in layers. Each layer implements a particular objectives with higher layers having higher degree of abstraction. Layers are made up of asynchronous modules that communicate with each other. Brooks also elaborates the idea in [116] and [117].

The common points of PARS and the subsumption architecture is that both of them target asynchronous hierarchically layered systems which produces decisions in order to fulfill system level objectives. On the other hand, PARS is a simple decision making mechanism based on information filtering which is specially suitable to logical bit-operation in FPGAs.

The task of PARS is to produce a consequent decision out of n inputs. Here we consider the case with four inputs: Low Priority Telecommands, Internal Information Basis, FDIR, and High Priority TC. The low priority TC represents stored commands on-board with time tags. Because these commands are occasionally not executable based on the status of environment, it is evaluated before execution. Figure 6.11 illustrates internal structure of the PARS.

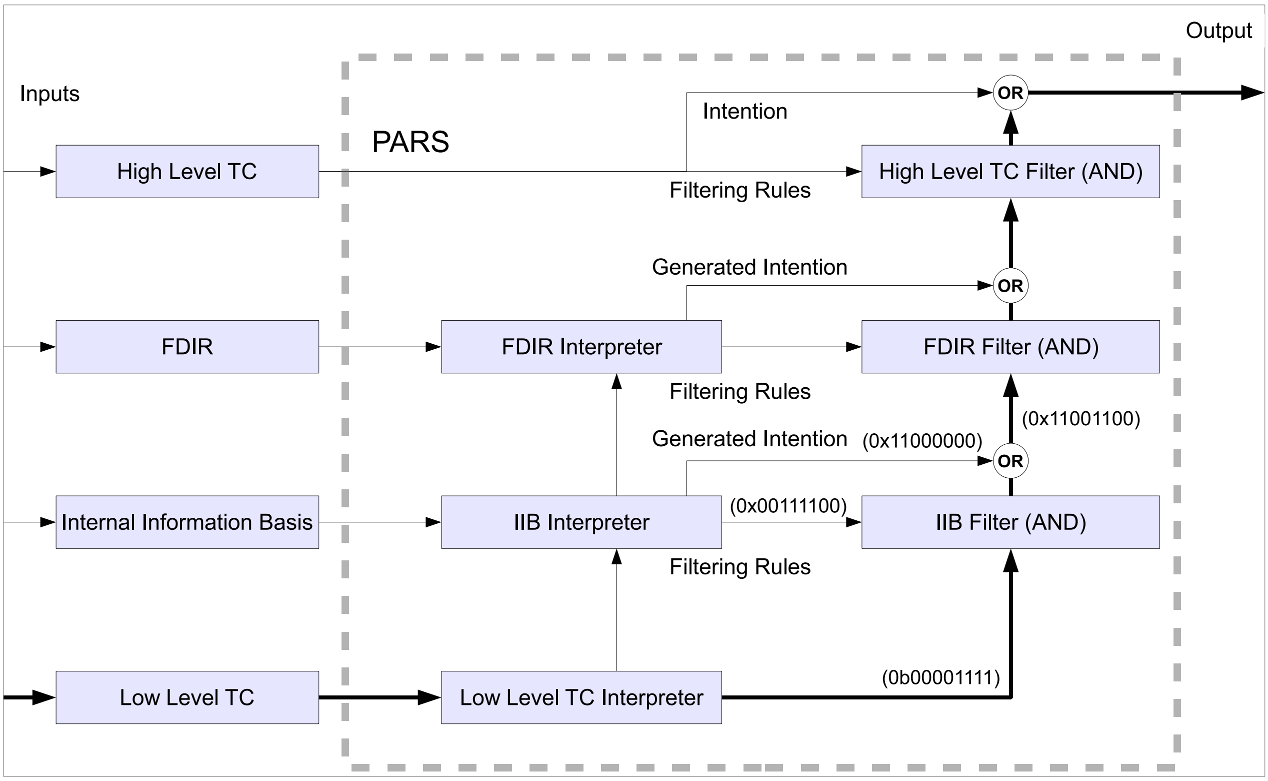


Figure 6.11.: Architecture of parallel asynchronous reactive system

Let I_{low} be a set of intention of lower level input, I_{high} be a set of intentions of higher level input, and F_{high} be a set of filtering rules of higher level input, then

$$I_{low} \times F_{high} \times I_{high} \rightarrow I_{high} \ , \tag{6.9}$$

where $I_{low} \times F_{high} \times I_{high}$ is implemented as a sequential combination of **AND** and **OR**. For example, suppose we send a Low Level Command (LLC) intending turning components on/off. This information is implemented in a bit table such that $i_{LLC} = 0b00001111$, where each bit corresponding a electrical component which shall be turned on (1) or turned off (0). Now the Internal Information Basis (IIB) interpreter sets the IIB filter as $f_{IIB} = 0b00111100$ and generates intention as $i_{IIB} = 0b11000000$. Then we get final intention as:

$$\begin{aligned} f_{IIB} \ \& \ i_{LLC} &= \text{0b00001100} \\ i_{IIB} &= i_{IIB} \mid (f_{IIB} \ \& \ i_{LLC}) = \text{0b11001100} \ , \end{aligned} \tag{6.10}$$

where & and | are bitwise **AND** and **OR** operators, respectively. This process takes only few clock cycles to make a decision.

6.4.7. Control algorithm architecture of Flying Laptop

The designed control algorithm architecture of the *Flying Laptop* is illustrated in Figure 6.12. The system consists of system level agents representing subsystems and several other functions – the colored components inside the application level control algorithms. Some of these system level agents are further decomposed into sub-agents applying the developed multi-agent programming method described in Section 6.4.4. Subsystems consist of a Central Control Logic (CCL), an Inter-agent I/O, a pair of TC and TM, a Fault Detection, and Functions. The CCLs are the central logic of subsystem level agents. They receive TC from the Up-link Manager and send TM to the data manager, which further forwards the data either to the Inter-lane Communication or to the Down-link Manager. The subsystem agents also communicate with each other through the Inter-agent I/O. The communication is based on predefined state vectors and a subsystem agent broadcasts its state vector information every time the internal state is changed, e.g., its mode transited from idle mode to safe mode. The other subsystems have knowledge about what the state vector actually means and are capable of making decisions if required. For example, if the ACS fails to achieve the required pointing accuracy and sends this information through the communication channel, the P/L decides to terminate its image acquisition activity and return to idle mode. The Fault Detection is responsible to observing internal variables and health status of the corresponding hardware components. If an anomaly happens, e.g., a sensor does not responds for the request or inconsistency between sensor data happens, it reports the fact to the CCL. Finally, all subsystem-relevant functions, such as attitude determination and control and/or image data processing are implemented in the Functions. In this way, CCLs can observe all circumstances around them and generate decisions and execute appropriate functions in order to fulfill their required tasks. This decision making is based on the developed PARS mechanism as described in Section 6.4.6. In order to react to changes on the surrounding environment in a sufficiently short period, all incoming information are processed by interpreters at first and internal filtering conditions are configured.

As indicated in Figure 6.12, CCLs also communicate with the System FDIR agent in order to achieve system level consistency. The subsystem level status and event of FDIR functions are implemented in indicator variables as described in Section 6.4.5, which are actually n-bit

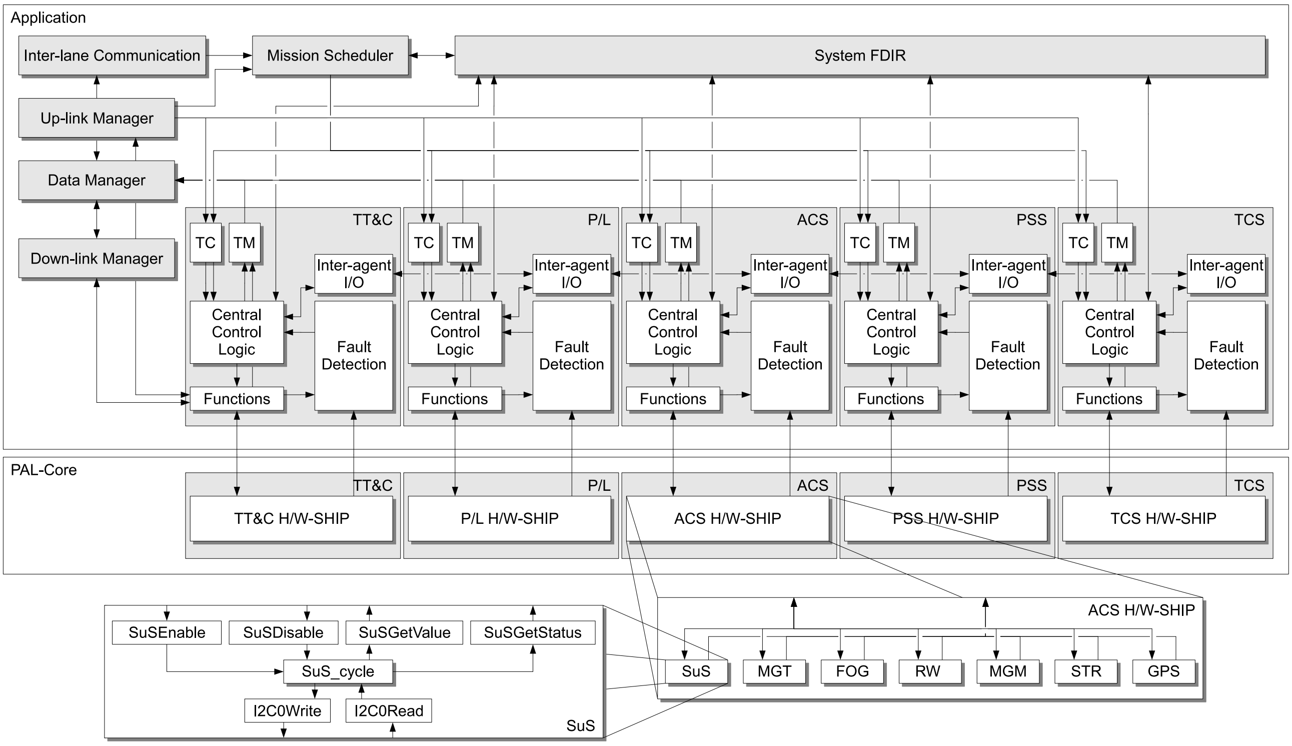


Figure 6.12.: Control algorithm architecture of the *Flying Laptop* satellite

binary variables. CCL is responsible to deal with subsystem level failures performing FDIR functions. System FDIR gathers reported system level fault events and their status from CCLs and performs system level FDIR functions, which has a higher priority than the decisions of CCLs. If required, System FDIR generates commands to CCLs.

The TCs uplinked from the IRS ground station are received by the TT&C subsystem agent and then forwarded to the Up-link Manager. Similarly, the TM delivered to the Down-link Manager can be downlinked by the TT&C subsystem agent through its communication hardware components. If the received TC shall be executed as fast as possible, the Up-link Manager forwards it to the corresponding subsystems. However, if the TC is a time-tagged one, it is forwarded to the Mission Scheduler, which is capable of managing on-board real time clock and tagged TCs. If the RTC reaches to the pre-defined time, the TC is sent to the corresponding agent. Mission Scheduler also communicates with the System FDIR so that it does not forward unauthorized TCs in case of system fault. The Inter-lane Communication is responsible to synchronization with the other CPNs, as well as exchanging data with the other CPNs and the CDV. Scientific data and housekeeping data gathered at the Data Manager can be transported to and from the mass memory on the CDV through this agent.

In Figure 6.12, the PAL-Core layer is also illustrated. A PAL-SHIP consists of control algorithms of component units. Fault Detection and Functions agents of subsystem level agents communicate with the corresponding units. A unit is further decomposed into several loop and sequential functions. The loop functions run eternally after their initialization and deal with routine tasks which are required to control real hardware components. As the strategy of organizing activities of all subsystem agents and to ensure the consistent behavior of the satellite system, their functionalities are organized in state-machines as described in the next section.

6.4.8. State-machine design

For the design of state-machines of subsystems, the following two aspects play an important role: mode design and transition logic design. The former ensures that there are enough modes provided with which the system, subsystems, and components can fulfill their functional requirements. The latter defines possible transition paths between all modes and specifies triggering events and conditions for all of those transition paths. This transition logic shall be designed in such a way that the consistency between all subsystems can always be ensured. Generally, a system with i subsystems has $i \cdot (i-1)$ interfaces between them, and a subsystem with j modes has $j \cdot (j-1)$ theoretically possible transition paths between the modes. This large number of combinational states shall always be consistent. For the subsystem level state-machine design of the *Flying Laptop*, MathWorks Simulink-/Stateflow Toolbox is utilized.

Figure 6.13 illustrates the Simulink model which includes four subsystem Stateflow blocks together with the internal structure of the attitude control system Stateflow block. These Stateflow blocks communicate with each other and exchange current mode information. In order to let a state transition in a Stateflow block trigger the consequential state transitions in other Stateflow blocks the triggering Stateflow block generates event signals, and delivers them to the other blocks. Some of the triggered blocks transit to another mode and also generate event signals. In the end, all blocks come to a consistent mode combination as the result of iterative interaction between each other. As is illustrated in this figure, the Stateflow Toolbox allows to define the possible transition path as a single directional arrow between two states, and also one can define the transition condition from one state to another allocating them to the specific transition path. In addition to this, each Stateflow block can contain internal variables and pre-defined logic-bit-tables so that allowed transition mask tables can be implemented. It also supports execution of variable manipulation at three different phases: entering phase into a state, maintaining phase in a state and moving-out phase from a state. These functions provide a convenient state-machine logic development environment.

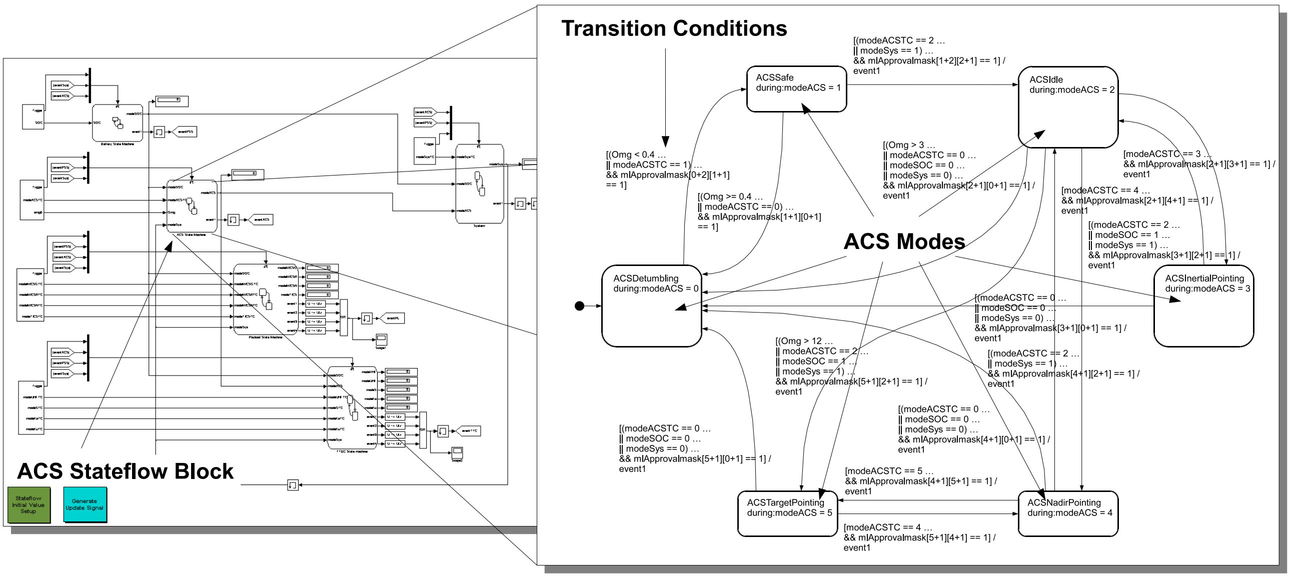

Transition Conditions
ACS Modes
ACS Stateflow Block
Stateflow Initial Value Setup
Generate Update Signal
ACSSafe during:modeACS = 1
ACSIdle during:modeACS = 2
ACSDetumbling during:modeACS = 0
ACSInertialPointing during:modeACS = 3
ACSTargetPointing during:modeACS = 5
ACSNadirPointing during:modeACS = 4
[(modeACSTC == 2 … || modeSys == 1) … && mlApprovalmask[1+2][2+1] == 1] / event1
[(Omg < 0.4 … || modeACSTC == 1) … && mlApprovalmask[0+2][1+1] == 1]
[(Omg >= 0.4 … || modeACSTC == 0) … && mlApprovalmask[1+1][0+1] == 1]
[(Omg > 3 … || modeACSTC == 0 … || modeSOC == 0 … || modeSys == 0) … && mlApprovalmask[2+1][0+1] == 1] / event1
[modeACSTC == 3 … && mlApprovalmask[2+1][3+1] == 1] / event1
[modeACSTC == 4 … && mlApprovalmask[2+1][4+1] == 1] / event1
[(modeACSTC == 2 … || modeSOC == 1 … || modeSys == 1) … && mlApprovalmask[3+1][2+1] == 1] / event1
[(modeACSTC == 0 … || modeSOC == 0 … || modeSys == 0) … && mlApprovalmask[3+1][0+1] == 1] / event1
[(Omg > 12 … || modeACSTC == 2 … || modeSOC == 1 … || modeSys == 1) … && mlApprovalmask[5+1][2+1] == 1] / event1
[(modeACSTC == 2 … || modeSys == 1) … && mlApprovalmask[4+1][2+1] == 1] / event1
[(modeACSTC == 0 … || modeSOC == 0 … || modeSys == 0) … && mlApprovalmask[4+1][0+1] == 1] / event1
[(modeACSTC == 0 … || modeSOC == 0 … || modeSys == 0) … && mlApprovalmask[5+1][0+1] == 1] / event1
[modeACSTC == 5 … && mlApprovalmask[4+1][5+1] == 1] / event1
[modeACSTC == 4 … && mlApprovalmask[5+1][4+1] == 1] / event1

Figure 6.13.: Simulink and Stateflow model

6.5. CDV Control algorithm

The most important objectives of designing the CA of the CDV is the mitigation of SEEs (see Section 1.3). The relevant radiation effects are SEEs in the Data Flip-Flops (DFFs) and SETs.

6.5.1. Single event upset mitigation methodology

Mechanism of single event upsets

The hardware logic elements of Flash-FPGAs can be used as configuration elements and DFFs. Unlike volatile memory-based FPGAs, the configuration memory (FPGA core) of a non-volatile FPGA cannot be upset and therefore will suffer no changes of functionality. The combinational logic is only sensitive to SETs while the sequential logic (DFF) could have SEUs and SETs as illustrated in Figure 6.14. Since the configuration memory is not sensitive to SEU, no refreshment (scrubbing) of the configuration memory content is necessary.

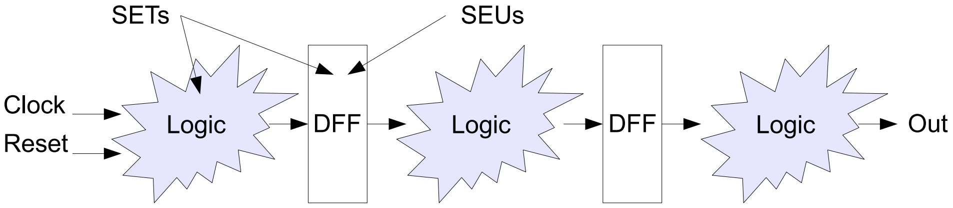


Figure 6.14.: Radiation effects on Flash-FPGAs

Due to this background, following radiation effect mitigation strategies can be applied.

- Mitigation of SEUs of the DFFs.
- Filtering of SETs in the combinational logic at the inputs of the DFFs.

A heavy ion that hits a DFF can induce an SEU but if it hits a combinational logic cell it starts an SET, and that can induce SEUs if it occurs on a clock edge of the DFF (Figure 6.15). The duration of the SET is related with the amount of the energy of the heavy ion. The higher the clock frequency is, the higher the probability that the DFF captures faulty values due to SETs.

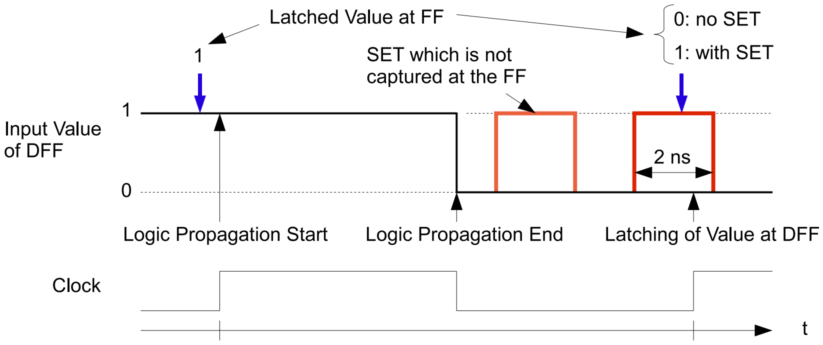


Figure 6.15.: Mechanisms of SET

Mitigation methods of SEUs

In order to ensure a safe operation of the satellite, SEU effects inside Flash-FPGAs shall be surely detected, isolated, and recovered. The mitigation method applied for the *Flying Laptop* is the Triple Module Redundancy (TMR). Each combinational logic and following DFFs are TMR'ed and the saved results in DFFs are voted before starting the next logical operation. This is illustrated in Figure 6.16.

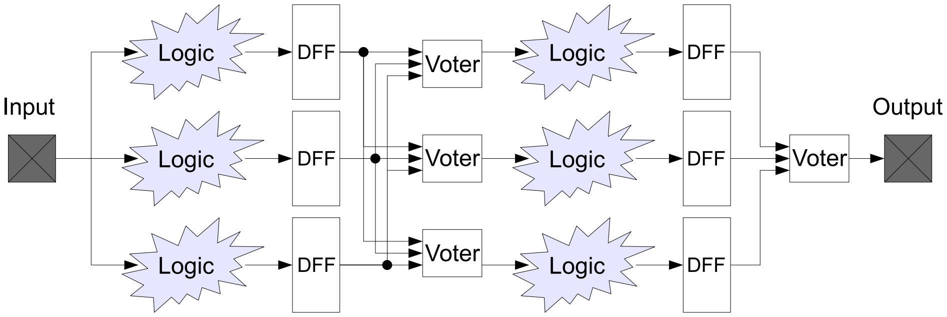


Figure 6.16.: TMR design method for SEE mitigation

Having the TMR'ed configuration logic and DFFs, a SET can affect only single DFF of them, and therefore, the output information of all voters are always correct. In this way the combinational logic in the next stage can operate with correct information delivered from the previous logic. Using TMR, SET effects happened between two sequential DFFs, SET effects trapped by a DFF, SEU effects inside a DFF can be mitigated. An example of TMR'ed design actually implemented in the CDV of the *Flying Laptop* by means of the Libero IDE is illustrated in Figure 6.17. It is worth noting that this is the actual implementation of the 2-out-of-3 system illustrated in Figure 2.3.

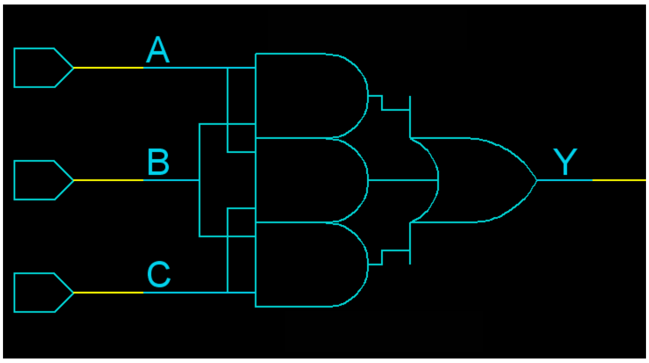
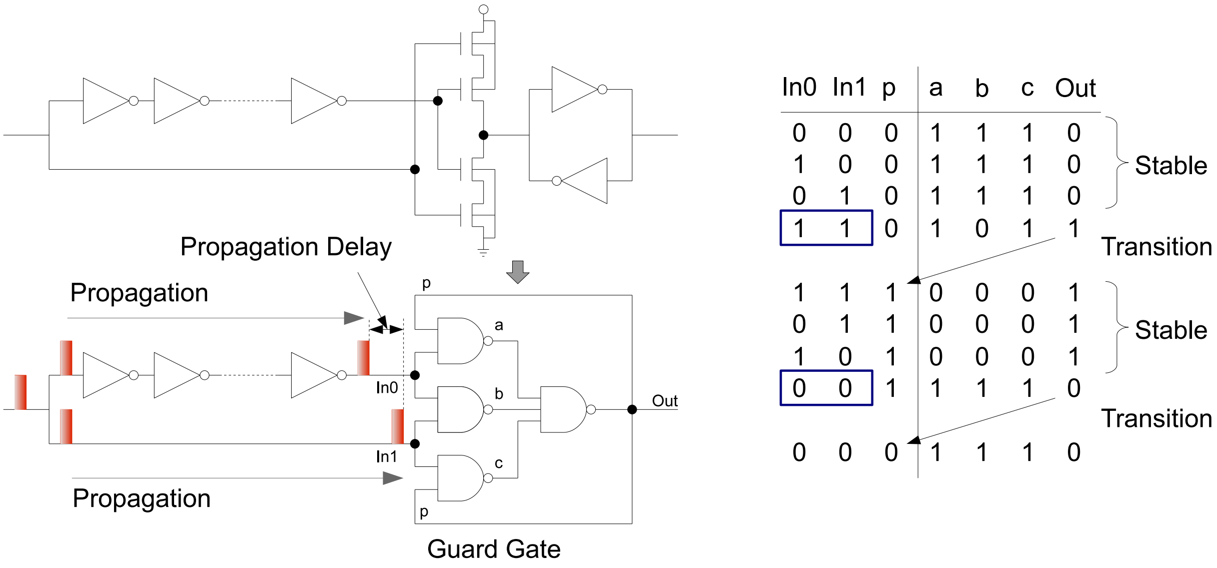


Figure 6.17.: Implemented TMR design into Flash-FPGA

6.5.2. Single event transient filtering

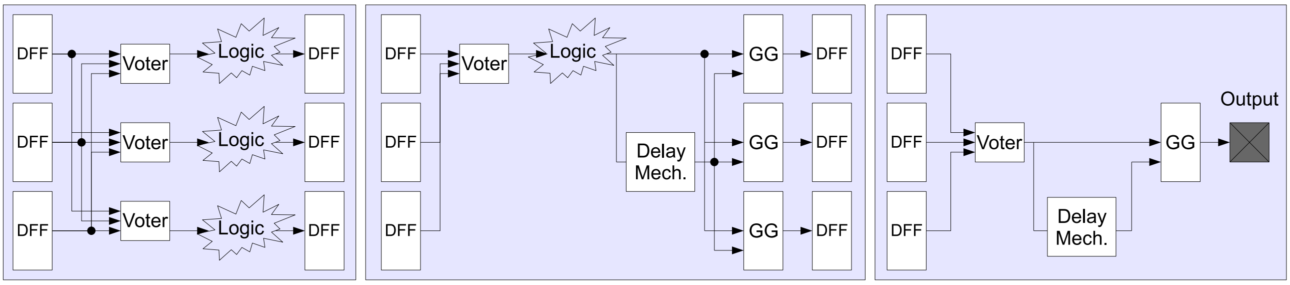
Because an SET is essentially a pulse signal with a very short width which propagates through the combinational logic, this can be filtered out introducing propagation delay elements. This SET filter consists of two main elements: 1) a SET transition delay mechanism and 2) guard gate (GG) which actually filters out SETs. The most simple delay mechanism, which can be implemented in the Flash-FPGA logic cells is the inverter logics. By connecting these an arbitrary number of inverters in serial, transition delay of SET can be adjusted. The second element GG is a combinational logic based on NAND logics. A GG produces one output value from two input values. The output of an GG is either one or zero and transition from one of them to the other happens only when both of the two input values shifted to the same new value as illustrated in Figure 6.18. In the figure, the logical combination of the inputs and the output is also illustrated. Every SET, which propagates through this mechanism, is separated into two waves, reaching to the GG at different time periods. If the amount of the delay is larger than that of the SET width, the SET can be filtered out at the GG [118].



In0	In1	p	a	b	c	Out	
0	0	0	1	1	1	0	
1	0	0	1	1	1	0	Stable
0	1	0	1	1	1	0	
1	1	0	1	0	1	1	Transition
1	1	1	0	0	0	1	
0	1	1	0	0	0	1	Stable
1	0	1	0	0	0	1	
0	0	1	1	1	1	0	Transition
0	0	0	1	1	1	0	

Figure 6.18.: SET filtering mechanism

Applying this filter mechanism, the redundant modules in TMR can be omitted as illustrated in Figure 6.19 (a) and (b). Furthermore, SET effects on output signals can be filtered out just before the output interfaces as illustrated in Figure 6.19 (c). This method is very attractive for the SET mitigation of large combinational logics, for which TMR takes too much hardware resources. However, for the secure mitigation, one shall measure the actual SET width for estimated energy range of heavy ions. In addition to this, the operational frequency shall be as low as possible in order to keep the relative duration of SET sufficiently shorter. For the design of the CA of the *Flying Laptop* the TMR method is applied.



(a) Full TMR **(b) SET Filter** **(c) Output signal filtering**

Figure 6.19.: Possible design trade-off of SEE mitigation methods[118]

6.6. Control algorithm development environment

The design environment of the CA of the *Flying Laptop*' OBC is established as illustrated in Figure 6.20. The CA for Xilinx SRAM-FPGAs are coded based on the DK Design Suite. After the design verification by compiler simulation an EDIF netlist is generated. This becomes the input of the design synthesis activity by Xilinx ISE (Integrated Software Environment) tool. The generated hardware logic file is finally written into the configuration memories inside SRAM-FPGAs.

The CA for Actel Flash-FPGA is developed based on both DK Design Suite and Libero IDE (Integrated Design Environment). The CAs developed in the DK Design Suite environment can be exported as VHDL files, which can be inported into the Libero environment. The other CAs are directly developed in the Libero by using VHDL. Both of these design files can be stored as VHDL libraries, on demand. Based on the simulation capability and analysis tools inside the Libero, the hardware design can be optimized. Finally, the generated design files can be written in the configuration memories inside Flash-FPGAs. This environment build the basis of the CA development of the *Flying Laptop*'s OBC.

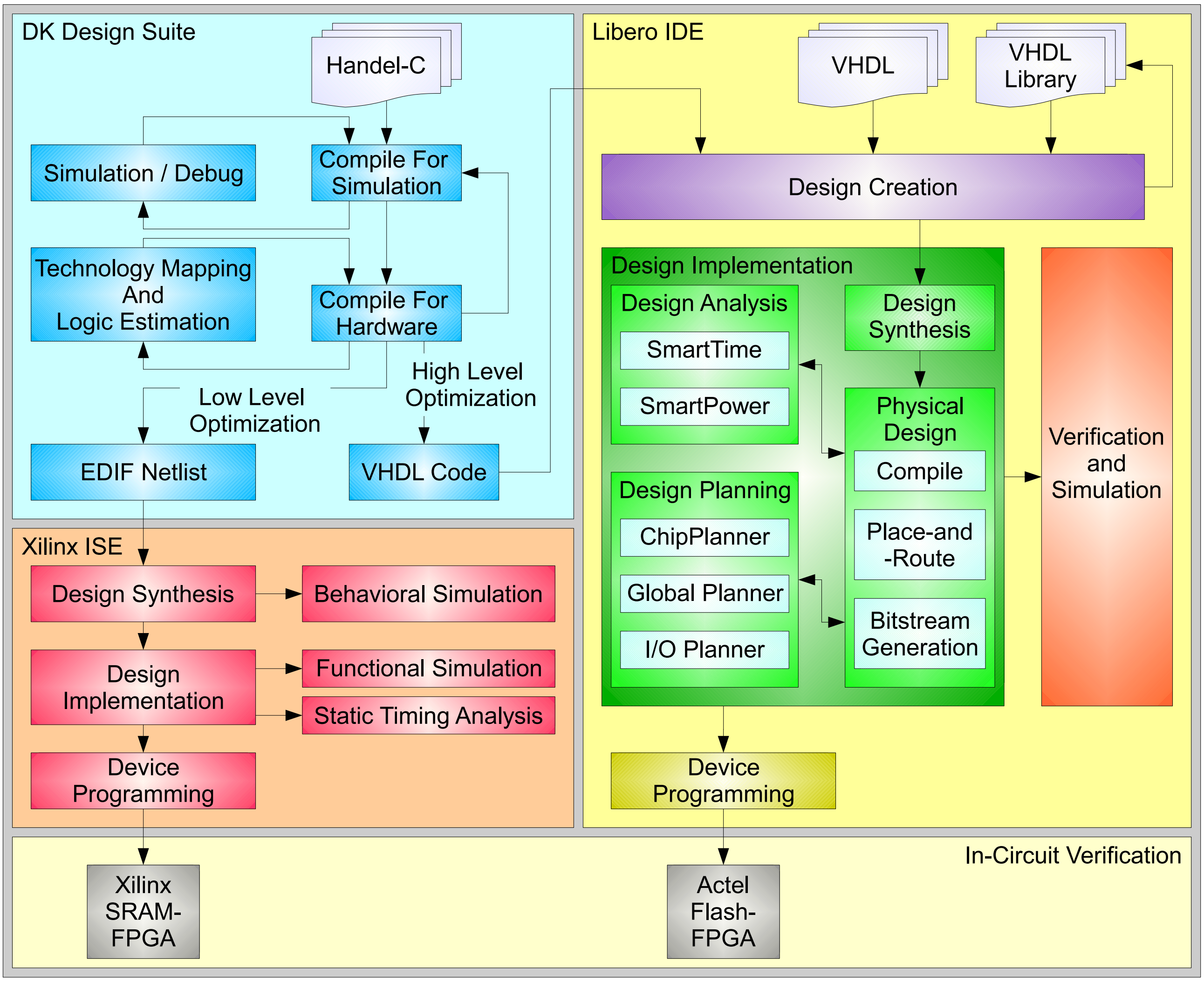


Figure 6.20.: Control algorithm development environment [119], [120]

6.7. Summary

In this chapter the development of the control algorithms of OBC was summarized. Based on the developed breadboard model of the OBC described in Chapter 5, the CA of the CPNs and the CDV are developed.

The design of the CA is a hardware development activity. Firstly, the internal hardware resources of SRAM- and Flash-FPGAs are briefly summarized, which are the target of the design trade-off. The CA of the CPNs is developed by means of Handel-C hardware description language. The CA of the CPNs is designed in a layered manner in order to realize a higher portability of the application level algorithms, which accelerates the entire design activities. After a brief discussion on parallel and sequential logic implementation into FPGAs, a CA architecture of compositional multi-agent system is conceptualized. In this architecture, functional unit of CA such as subsystems and components are implemented as communicating parallel running agents. Also the FDIR function is implemented based on fault tree structures utilizing state vector variables and bit operations of Handel-C. This method can maximize the hardware implementation efficiency. An implementation method of complex decision making logic, named as parallel asynchronous reactive system (PARS), is developed and is applied to the central control logic of the CA. The PARS can realize fast reactive decision making against different types and levels of environmental conditions. The behaviors of the agents are specified by means of state-machines. These state-machines are designed in a MathWorks Simulink-/Stateflow environment.

The main concern of the development of CA for the CDV is the secure radiation mitigation. The radiation effects and their mitigation methods based on triple module redundancy and single event transient filters, as well as their application methods are summarized. Finally, a control algorithm development infrastructure for both SRAM- and Flash-FPGAs were established by integrating development tools of DK Design Suite (Handel-C), Xilinx ISE (SRAM-FPGAs), and Actel Libero IDE (Flash-FPGAs).

In this part as a whole, the design and development of the FPGA-based on-board computer for the *Flying Laptop* were described based on the conceptualized application methods of reconfigurable FPGAs summarized in Part I. In Chapter 4 firstly, the system design of the whole *Flying Laptop* was conducted according to the system level requirements and constraints, so that all mission objectives can be fulfilled by the FPGA-based on-board computer. The conducted system design allows the actual implementation of the OBC as the central computing system of the satellite. Based on the results of the system design, the hardware of the FPGA-based OBC was designed and its breadboard model, as well as engineering models of some components were developed. By utilizing the breadboard model assembly, the control algorithms of the OBC were developed. Consequently, this part provided the complete information on the conceptualized application methods of reconfigurable FPGAs for on-board computers of space systems.

Part III.

Simulation and verification

In this part, the simulation and verification aspects of the developed control algorithm are described. Taking the parallel behavior of the hardware logic inside an FPGA into account, the functionalities of the developed control algorithm in Chapter 6 need to be verified in order to prove the validity of the design. In Chapter 7, the developed hardware-in-the-loop simulation and verification environment for FPGA-based on-board computers is described. The developed data exchange interface implemented in the extra hardware logic inside an FPGA allows the communication between an simulator to realize simulation and verification of the control algorithm implemented inside the target FPGA. In Chapter 8, the simulation results by means of the established environment are summarized. The results prove the validity of the control algorithm implementation and the feasibility of the whole concept proposed in this thesis. Finally, Chapter 9 summarizes the results of investigation on and development of formal verification methods of the control algorithms. Formal verification provides a concrete confidence of validity of the designed algorithm, which is desired for critical application fields such as space systems. The existing formal semantics of the Handel-C hardware description language is extended in terms of "signal" syntax in order to cover the all applied behavior of the control algorithm. This new formal semantics indicate the possibility of formal verification of control algorithms including signal syntax, which can be implemented into model check tools in the future. Consequently, this part illustrates the validity of the investigation and development described in Part I and Part II.

7. Establishment of a real-time FPGA hardware-in-the-loop simulation environment

Simulation environments are indispensable for satellite development. It also holds the truth for on-board computers based on FPGAs. The simulation environment of FPGA-based OBCs shall be capable of dealing with the parallel behavior of FPGAs. In this chapter, the development and implementation of an FPGA hardware-in-the-loop simulation and verification environment are described.

Over the past few years, model-based development and verification has been playing a very important role in satellite design processes of industry-led space projects [121]. Since 2001 EADS Astrium GmbH, for example, has developed a new, highly integrated model-based real-time system simulation infrastructure to support spacecraft development, on-board software verification/maintenance and spacecraft design validation [77]. This simulation infrastructure is called "Model-based Development and Verification Environment" (MDVE). As described in Section 3.1 Albert Falke has established an MDVE at the Universität Stuttgart with the support of EADS Astrium [78]. Though an MDVE offers attractive simulation and verification capabilities for satellite developments, it was originally designed for traditional processor-type on-board computing systems, i.e., there is no possibility to emulate parallel execution behaviors of FPGAs inside the simulator. In order to support the development of the innovative FPGA-based OBC architecture of the *Flying Laptop*, there was a need on establishing a simulator interface which allows FPGA hardware-in-the-loop simulations with the MDVE. This chapter describes the detail of the development and implementation of this interface. A summary of this subject can be found in the publication [122].

7.1. Model based development and verification environment

7.1.1. State of the art

The MDVE simulation environment provides a tool infrastructure allowing spacecraft models ranging from early pure virtual simulations via hybrid testbenches up to full FlatSat configurations. This largely minimizes cost for hardware models, and in parallel provides risk mitigation through stepwise verification of on-board software, on-board hardware, flight procedures and so forth. The MDVE is a real-time simulator with models of the space environment, spacecraft dynamics, thermal and electrical power models, and all satellite components. MDVE is, at the same time, an engineering environment which enables the developers to systematically develop spacecraft. Generally, the available MDVE testbed configurations are:

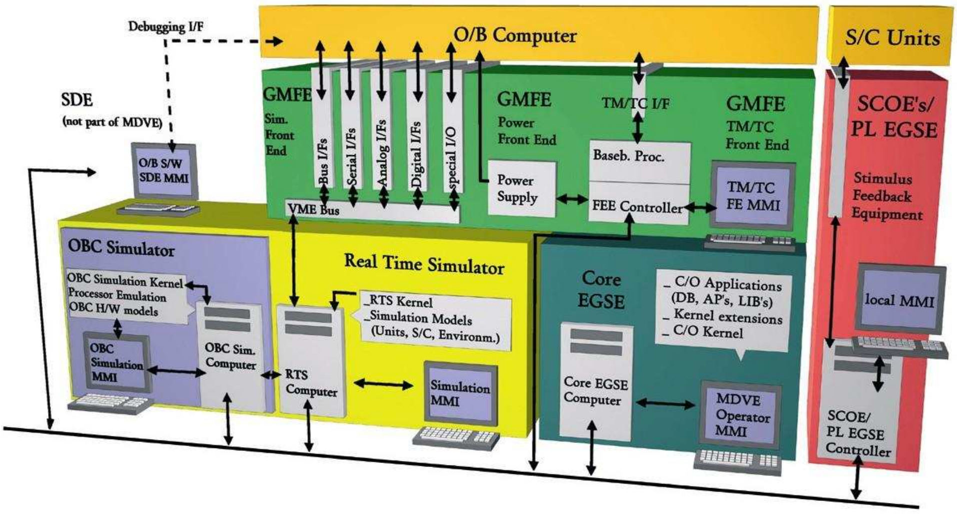


Figure 7.1.: Model based development and verification environment [123]

- Early system simulations and tests of the on-board software (yet without available spacecraft hardware and software)
- Extensive simulation and verification of an on-board computer software based on a processor emulation (before availability of the computer hardware).
- Verification of an on-board software with the real-time simulator and the Core EGSE in a hardware-in-the-loop configuration.
- Simulation and verification of whole satellite components in a flat-sat configuration.
- Software maintenance and operator training in the ground station.

The core element of the MDVE is an On-board Computer Simulator (OBC-Simulator) and a Real-Time Simulator (RTS) (Figure 7.1 [1]). OBC-Simulator is a simulator of the OBC software and hardware. It is based on a processor emulator and can be built both by simple functional model and complete detailed implementation of the OBC software. RTS is responsible for modeling the remaining equipment units of the spacecraft, spacecraft dynamics, space environments, as well as thermal and electrical 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, which is the man-machine interface to control all possible simulation configurations. The functional behavior of the satellite system and all interactions of the on-board equipments are modeled inside the MDVE. The Generic Modular Frontend (GMFE) consists of interface cards, which enables communication between software models in simulation environment and real hardware components connected to the simulator. The Special Check-out Equipment (SCOE) supports additional hardware interfaces with special test equipments with which the real sensor data can be reflected to simulations.

A major benefit of the model-based system development utilizing an MDVE is the possibility for early simulated satellite mission operations. 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.

---

[1] Figure used by courtesy of Astrium GmbH, Friedrichshafen, Germany ©Astrium GmbH

7.1.2. MDVE configurations

The developed interface and simulation environment enables a number of design, development, and verification tasks in different working environments in various project phases. The important MDVE configurations relevant to this thesis are summarized below.

Software verification facility

In this Software Verification Facility (SVF) configuration, basic functionalities of the OBC are verified with the satellite component models in the RTS. The data handling and the ACS algorithms, as well as test and operating procedures can be developed and verified. The simulated satellite is controlled through TC/TM to and from the simulated OBC. The developed hardware-in-the-loop simulation environment makes use of FPGA development boards for the simulation of the real hardware CA, while SVF is originally based on a processor emulation on which the on-board software can run like on the real OBC.

Real-time testbed

In this Real-time Testbed (RTB) configuration, the real OBC is in the loop with the RTS and the Core EGSE through the GMFE. In RTB the real functionality of the OBC is verified. System level tests can be performed with the RTB.

Flat-sat configuration

The flat-sat is an extended real-time testbed. Following the OBC, further real hardware components are integrated into the loop, which can be driven by special checkout equipment.

Software maintenance facility

This configuration is used as the CA maintenance facility for operations support after the launch of the spacecraft. Changes in on-board CA can be verified on the simulator before the updated.

7.2. Development of a hardware-in-the-loop environment

7.2.1. Implementation concept

To incorporate the real on-board CA into the MDVE closed loop simulation the MDVE requires FPGA chips for on-board CA execution. The assumptions made for this implementation are:

- Periodical execution rate of routine tasks of OBC is maximum 10 Hz
- RTS executes the attitude control simulation loop with a frequency of minimum 20 Hz, in order to guarantee a stable simulation
- The timing discrepancy between MDVE and FPGA-based OBC is negligible during the range of simulation duration of interest

According to these assumptions, it is estimated that computation speed of both the RTS and the simulating FPGA is fast enough, so that all parallel communications can be transported through serial communication lines within the required time step. This communication channel consists of a full-duplex RS-422 interface with a baud rate of 921600 [124].

7.2.2. Extended layered structure of control algorithm

The most important requirement of this simulation and verification environment is that the application level control algorithms remain the same in flight and verification configurations so that the verified CA can be directly ported into the real flight configuration. To fulfill this requirement the simulator interface is implemented into lower levels than PAL API as a part of the abstraction layers. The extended layered structure of the CA is illustrated in Figure 7.2.

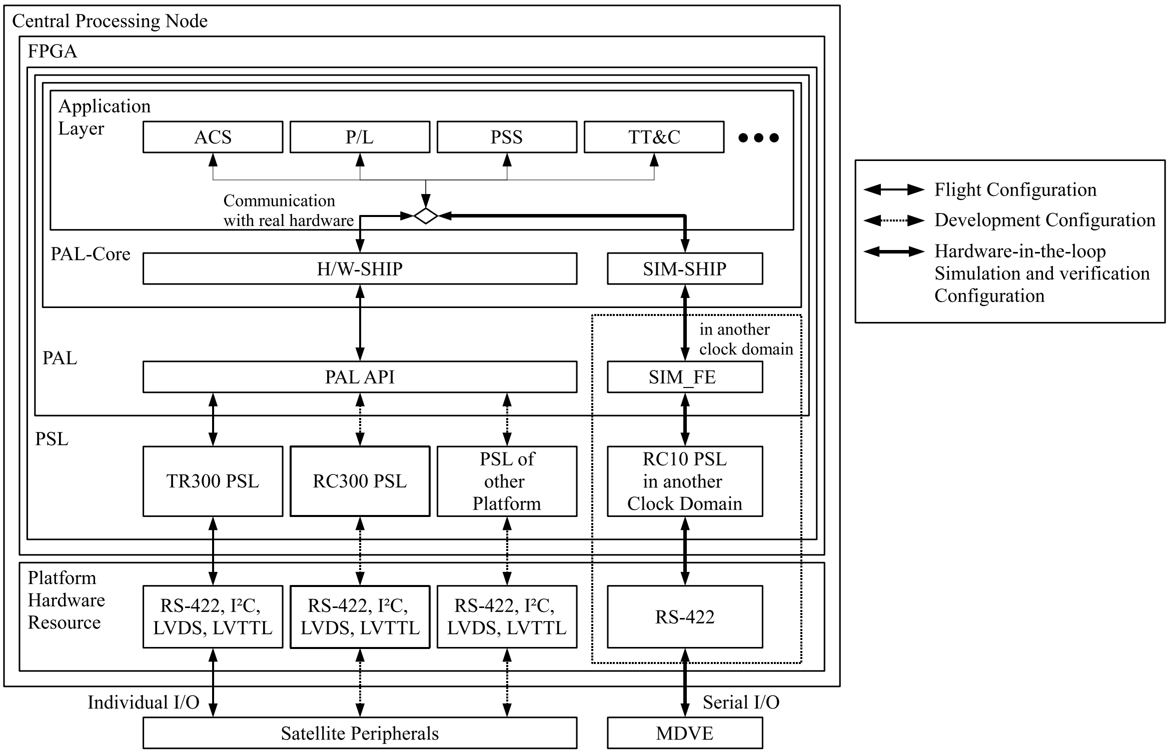


Figure 7.2.: Extended layered structure of control algorithm

7.3. Development of a simulator front-end

The simulator front-end (SimFE) is implemented as an extra CA function within the FPGA as illustrate in Figure 7.2. As described above, the SVF is intended to enable the development and verification of functionalities of the basic CA architecture, attitude control algorithms, commanding procedures and operational plans, as well as power balancing and thermal design. Within the scope of this thesis, in order to achieve these purposes, primary satellite components are implemented as described below. Since in this configuration no GMFE is present and the OBC is represented by an FPGA development board with less functionality than the real OBC, a SimFE needs to be implemented which realizes the communication with the RTS.

7.3.1. Communication link budget

Since the MDVE reflects hardware components down to electrical communication protocols the communication interface need to be bit for bit compatible with the real communication interface between hardware components. On the first look, the SimFE interface seems to be a bottleneck, because the communication of several independent hardware devices, taking place in parallel, have to be routed through a single serial communication line. However, as shown below, the total amount of demanded bandwidths is small enough, so that the the real-time simulation requirements can be fulfilled. On the one hand the simulation relies on the fact that it receives information from the OBC instantaneously, on the other hand, the timing of the incoming and outgoing communication should reflects the real behavior of satellite hardware components as precise as possible. In Table 7.1 and Table 7.2, the communication budgets of implemented relevant primary satellite components are summarized.

Table 7.1.: Simulator front-end uplink budget (to RTS)

Component	Quantity	Rate [Hz]	STF-Length [byte]	Bandwidth [bit/s]	Transmission time [μs]
MGM Req.	2	6	7	336	60.76
MGT Req.	1	10	7	560	60.76
MGT Com.	1	10	9	720	78.13
SuS Req.	2	1	8	64	69.44
PCDU Req.	1	1	7	56	60.76
PCDU Com.	1	1	17	136	147.57
			Sum:	5952	

Table 7.2.: Simulator front-end downlink budget (from RTS)

Component	Quantity	Rate [Hz]	STF-Length [byte]	Bandwidth [bit/s]	Transmission time [μs]
MGM	2	6	15	720	130.21
MGT	1	10	8	640	69.44
SuS	2	1	40	320	347.22
PCDU	1	1	53	424	460.07
			Sum:	7864	

7.3.2. Simulator transfer frame and DLE protocol

The communication between SimFE and RTS utilizes a packet format called simulator transfer frame (STF). An STF is composed of: the first byte standard STX followed by two bytes of length (n) information of the transfered data including the checksum in little endian order, unit ID which makes the routing in SimFE and RTS possible, n-1 bytes of user data, one byte of checksum based on a cyclic redundancy check, and the standard ETX (Figure 7.3).

Since the start and stop bytes of the STF (STX and ETX) shall be avoided inside the packet, these control bytes are masked with the data link escape (DLE) protocol. The STX, ETX, CR,

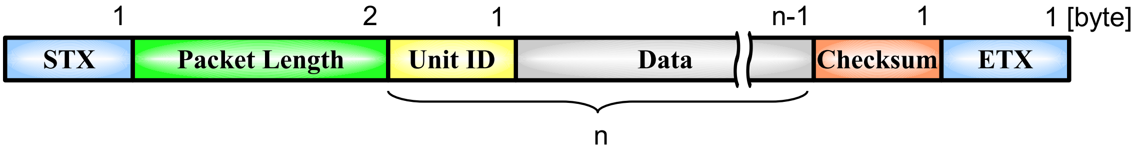


Figure 7.3.: Simulator transfer frame

or DLE byte will be masked by replacing them with a combination of a DLE and an replacement byte as summarized in Table 7.3. Each entity listed in above tables has its own unit ID for the identification, which is included in the STF for the internal routing purposes.

Table 7.3.: Control characters of DLE protocol

Control Characters	Hexadecimal	Replacement Characters
STX	0x02	0x10 0x42
ETX	0x03	0x10 0x43
CR	0x0D	0x10 0x4D
DLE	0x10	0x10 0x10

7.3.3. Data transfer strategies

The transportation of STFs are implemented in an asynchronous manner. This allows the CA to be left nearly untouched, since the communications between the FPGA and the satellite peripheral components in the real case take place asynchronously. A synchronous implementation, on the other hand, requires a considerable amount of additional timing information to be transported in order to deal with the asynchronous behavior, which has a great impact on the simulation speed. In this way, the SimFE simply acts as a router, wrapping every piece of communication into an STF and sending it to the RTS in serial as quickly as possible. On the way back the SimFE it analyzes the sanity of the incoming STF, unwraps and distributes it to the intended instance of CA. This approach reflects the parallelism of the FPGA much better and produces less informational overhead. The communication speed should be fast enough to achieve short transfer time even when more than one communications shall be conducted at the very same time, thus achieving a hardware-like timing behavior.

7.3.4. Clock domains

The SimFE runs in parallel to the satellite CA and is placed in another clock domain with a clock frequency of 96 MHz, eight times faster than the core CA which runs with a frequency of 12 MHz. This enables a sufficiently high data processing throughput and a clear modular design to keep the application level CA unchanged. The much faster processing of wrapping and routing of information ensures stable communication between an FPGA and the RTS. The faster clock rate results in harder timing constrains, and therefore, the logical depth of the SimFE is carefully designed to fulfill the timing requirements.

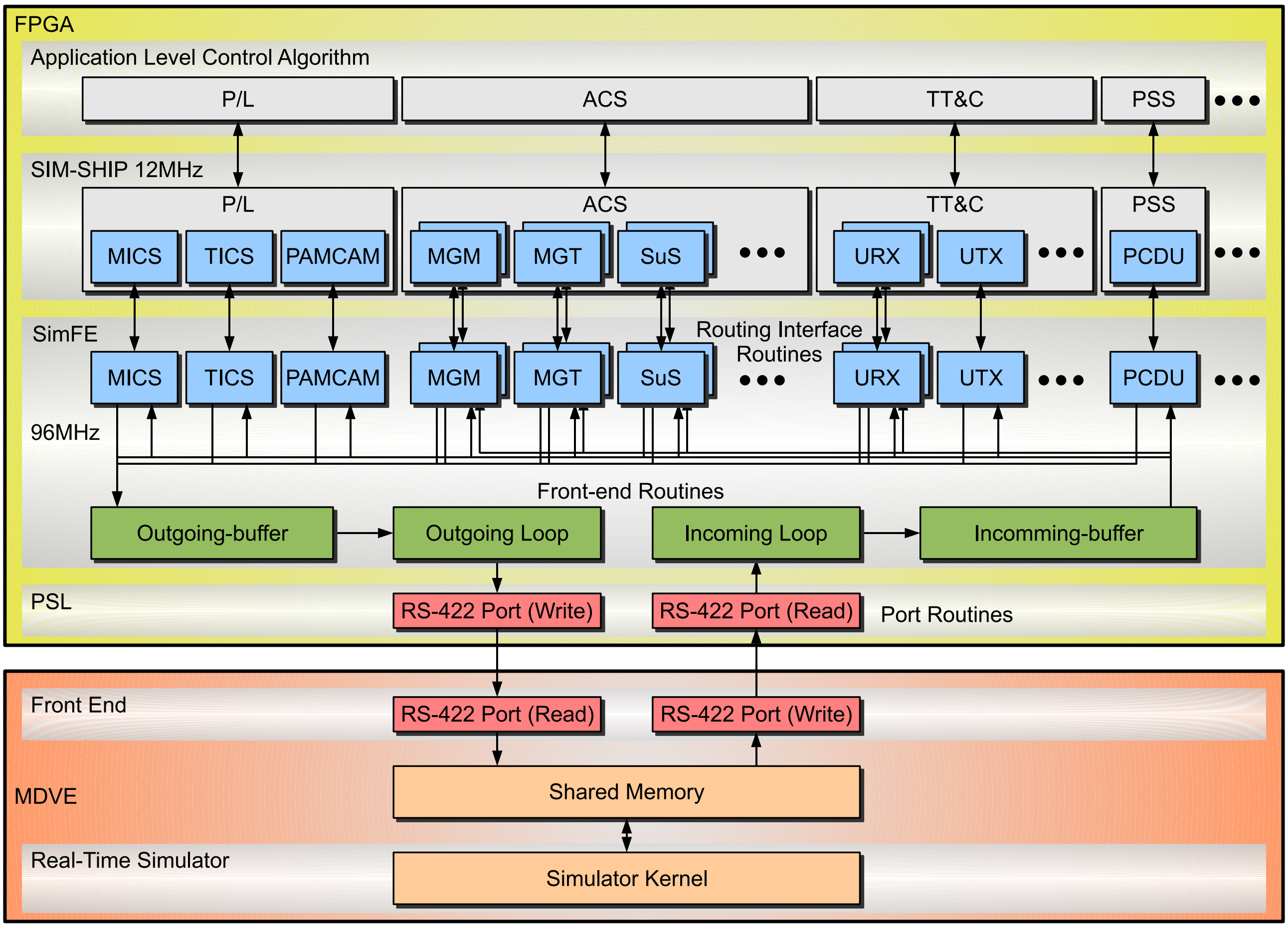


Figure 7.4.: Internal architecture of simulator front-end

7.3.5. Structure of simulator front-end

The SimFE is located in the PAL Core. SimFE consists of three levels of routines: routing interface routines (RIR), front-end routines, and port routines. In addition to this, the H/W-SHIP was slightly modified to SIM-SHIP so that it communicates with the SimFE instead of accessing hardware components. The structure of the SimFE is illustrated in Figure 7.4.

Port routines: The write and read port routines are implemented as PSL functions and actually drive hardware components on the board. This layer ensures platform independence of the SimFE. Low level DLE protocol is implemented in this layer.

Front-end routines: The front-end routines consist of outgoing- and incoming- loops and buffers, where the latter are accessible from RIRs. Front-end routines are responsible to the serialization of data streams. Every time a data packet is stored in the outgoing buffer by one of the RTRs with raising a flag, other interface routines are blocked from the buffer. The outgoing loop calculates the checksum for the data packet and wraps into an STF before sending it to the write port routine. The incoming loop once stores the received DLE packet into the incoming buffer by proving the checksum at the same time. If the packet is not corrupted it raises a flag to inform RTRs that a packet has been received.

Routing interface routines: The routing interface routines are the interface between SimFE and SIM-SHIP. Each CA instance in SIM-SHIP has the corresponding RTR in SimFE. Each RTR is identified by the unit ID and performs storing of data packets into the outgoing-buffer and extracting from the incoming-buffer in the front-end routines. The communication between the SIM-SHIP is realized by channels, which ensures a timing and information consistency since the receiver of a channel has to wait until the data is sent, so it is assured that timing consistency is given and no byte is accidentally skipped. An RTR of a component consists of a output loop and a input loop. The output loop stores outgoing data packets into the outgoing-buffer and set a flag if it is not occupied by other routines. The input loop waits until a data packet with its unit ID is stored in the incoming-buffer and routes it to the SIM-SHIP.

7.4. Timing analysis

In order to ensure the validity of the developed SimFE, the response time of communications between MDVE and FPGA-based OBC for relevant components are measured and statistically analyzed. The values are measured for each component: magnetometer (MGT), magnetic torquer (MGT), sun sensor (SuS), and power control and distribution unit (PCDU) for 60 minutes. Figure 7.5 illustrates the measured response time of magnetometer communication along the simulation time, which needs the longest response time among the components. Figure 7.6 illustrates the histograms of the measured response time of the four components in concern. The number of packets is plotted against the measured response time.

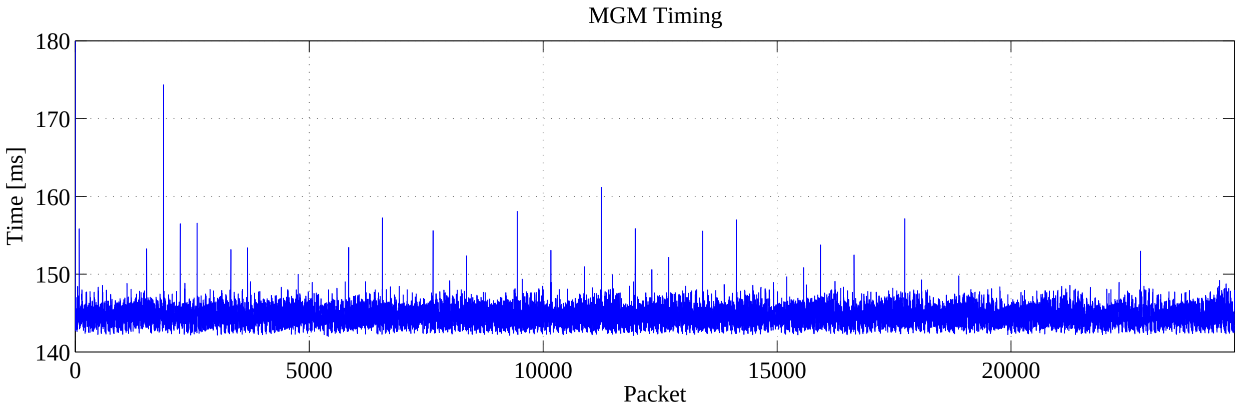


Figure 7.5.: Measured timing of magnetometer communication

Those values illustrated in Figure 7.6 shall be as similar as possible to the behaviors of the real hardware components. The response time of the magnetometer is specified as 142 ms due to its integration time of the Earth's magnetic field, while those of the other components shall be as short as possible. This time delay of magnetometer is deliberately implemented in the software model in the RTS. Figure 7.6 illustrates that most of the magnetometer communications have taken place within 148 ms. It is estimated that the delay of approximately 6 ms from the desired value is due to the transmission time through the communication lines and disturbances by other internal processes and other packets. It can be seen in Figure 7.5 that there are many peaks with the values of up to around 175 ms. Because these peaks also happen to the other components at the same moment, it is assumed that this phenomenon is caused by the temporal overload

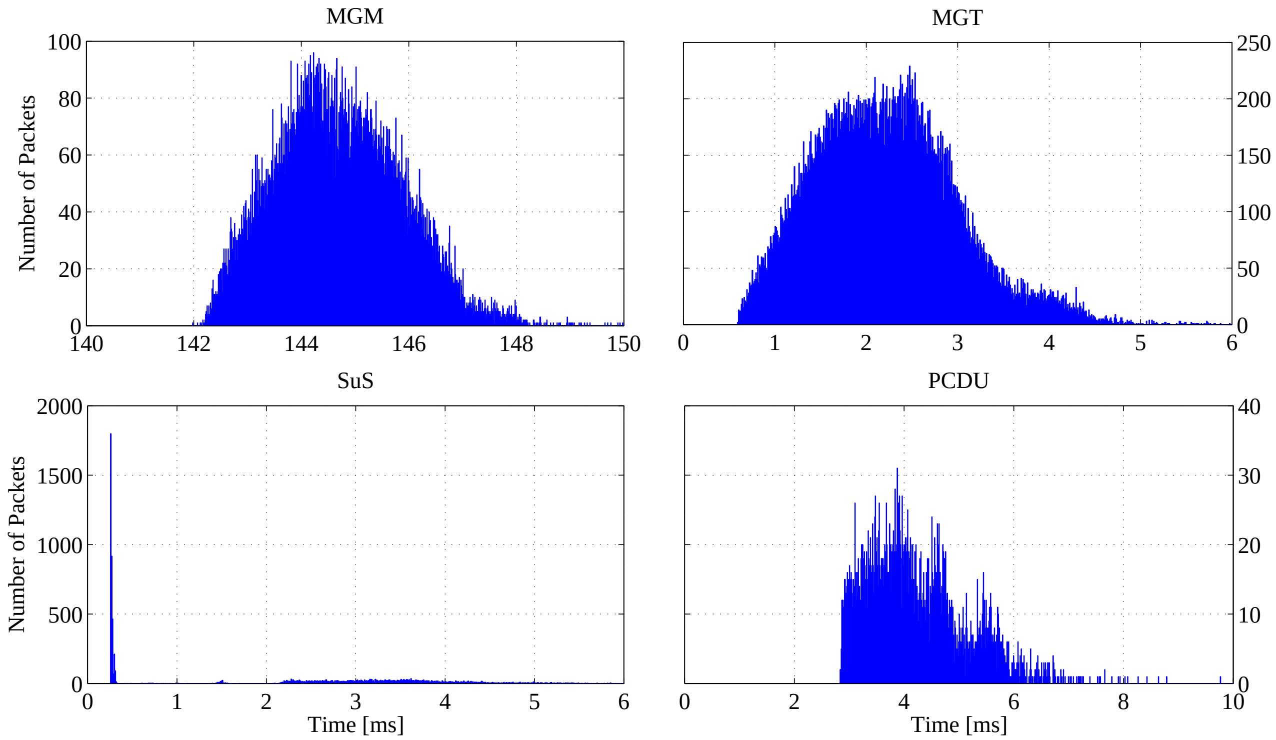


Figure 7.6.: Statistics of measured communication timing

on simulator kernel. The lower limit of response time of the PCDU is because of the larger size of its STF. Response time of sun sensors are significantly short because of the simple processing of their sensor data at the RTS side.

It can be seen that the implemented hardware specific time delay of magnetometer is consistent with the required value in the presence of acceptable distribution of around ±3 ms. There is also a possibility of improvement by optimizing the software model in the RTS by tuning this value. The other communications also take place in a sufficiently short time, which enables the update of sensor data within the required time periods under the assumptions described above. What shall be also mentioned is that these communication take place individually, without waiting for other processes. According to these results, it is proved that the communication speed between MDVE and FPGA-based OBC is high enough so that the parallel asynchronous communication of FPGA can be successfully serialized by the SimFE.

7.5. Developed hardware-in-the-loop simulation environment

The developed FPGA hardware-in-the-loop real-time simulation environment is illustrated in Figure 7.7. The SVF is supported by a central control system (CCS), a test scripting engine, MDVE development PCs and a server. For the mission control system, SCOS-2000 is selected and implemented in the CCS. The RTS on the SVF can be controlled by SCOS-2000. A standalone proxy application between the SCOS and RTS, running inside the SVF, enables conversion and forwarding of data from SCOS-2000 to the simulator. Spacecraft TC and TM can be also directly routed between the SCOS and the FPGA development board through the application. The test scripting engine is based on a test procedure editor and executer MOIS,

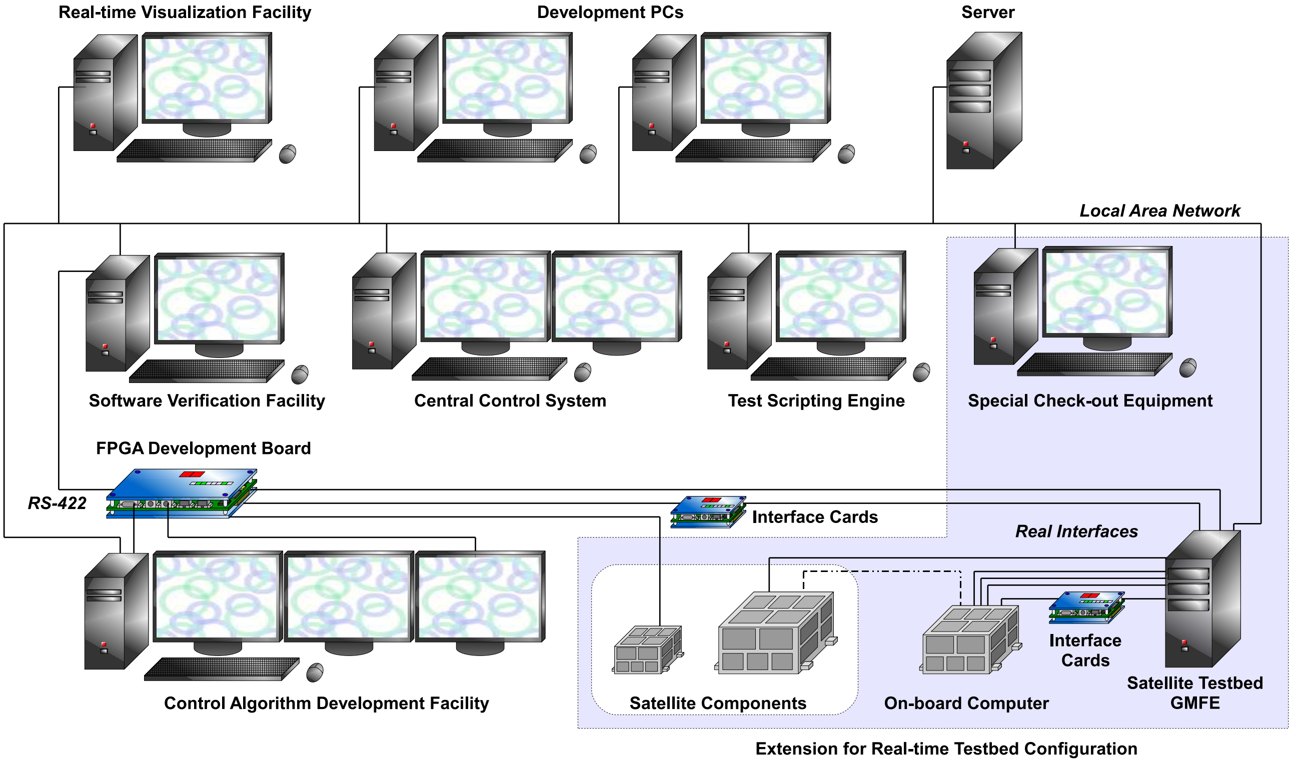


Figure 7.7.: FPGA hardware-in-the-loop simulation environment

which enables test case generation and management for their reproducibility [79]. MOIS can create test cases using flowcharts. Being supported by this test scripting engine, any test scenario can be executed automatically. Information required for the simulations, such as spacecraft database, project documentation, source codes, control algorithms for FPGAs etc., is stored in the central server.

The control algorithms for the on-board computer including those for the development boards are developed by the Control Algorithm Development Facility (CADF). After the simulator is set to ready, the development board is initialized and then both of RTS and FPGA start real-time simulation synchronously. The exchanged TM data, e.g., the actual attitude control mode, state of charge of the battery and so forth, can be routed to SCOS-2000 and then either numerically or graphically displayed. For the real-time visualization purpose of the simulation results, especially the satellite's attitude and orbit conditions, a real-time visualization facility (RVF) is developed. RVF is based on a visualization software Celestia, which communicates with the RTS via a port and actualizes the visualization.

For the real-time testbed configuration of the MDVE, the facility is extended with SCOE and GMFE. The real OBC, as well as satellite components can be connected with the facility via GMFE. Satellite components can be either connected with the OBC or GMFE or both of them based on simulation and verification purposes. For example, camera units can be connected directly with the OBC to be controlled or some sensor data can be generated and sent to the OBC via GMFE. For the interfaces which is not widely supported by standard PC interface cards, additional FPGA development boards can be used as a part of the GMFE.

7.6. Control algorithm development facility

As the consequent of above development activity, the development environment of the on-board control algorithms is established as illustrated in Figure 7.8. Starting from the mathematical modeling of the attitude control algorithms, the whole satellite control algorithm, which shall be finally implemented into the FM of the FPGA-based OBC can be developed in this environment. MATLAB/Simulink is used for the initial control algorithm development and optimization, whose results can be also visualized and evaluated/validated in the environment. The MATLAB codes are then manually ported into Handel-C codes partially supported by a modular concept. The portability into Handel-C codes and performance analyzed by DK simulation can be fed back to the original MATLAB codes and the numerical model and codes are optimized. The functionalities of the hardware logic designed in Handel-C can be simulated and verified in the FPGA hardware-in-the-loop simulation environment based on the MDVE as described above. The simulation results can be visualized in real-time by the RVF. The real hardware of the OBC can be integrated in the RTB configuration together with the satellite components and the verified final CA can be implemented into the FM of the OBC. Even after the launch, the CA can be maintained, modified, verified based on this environment. This facility provides comprehensive supports for the CA development.

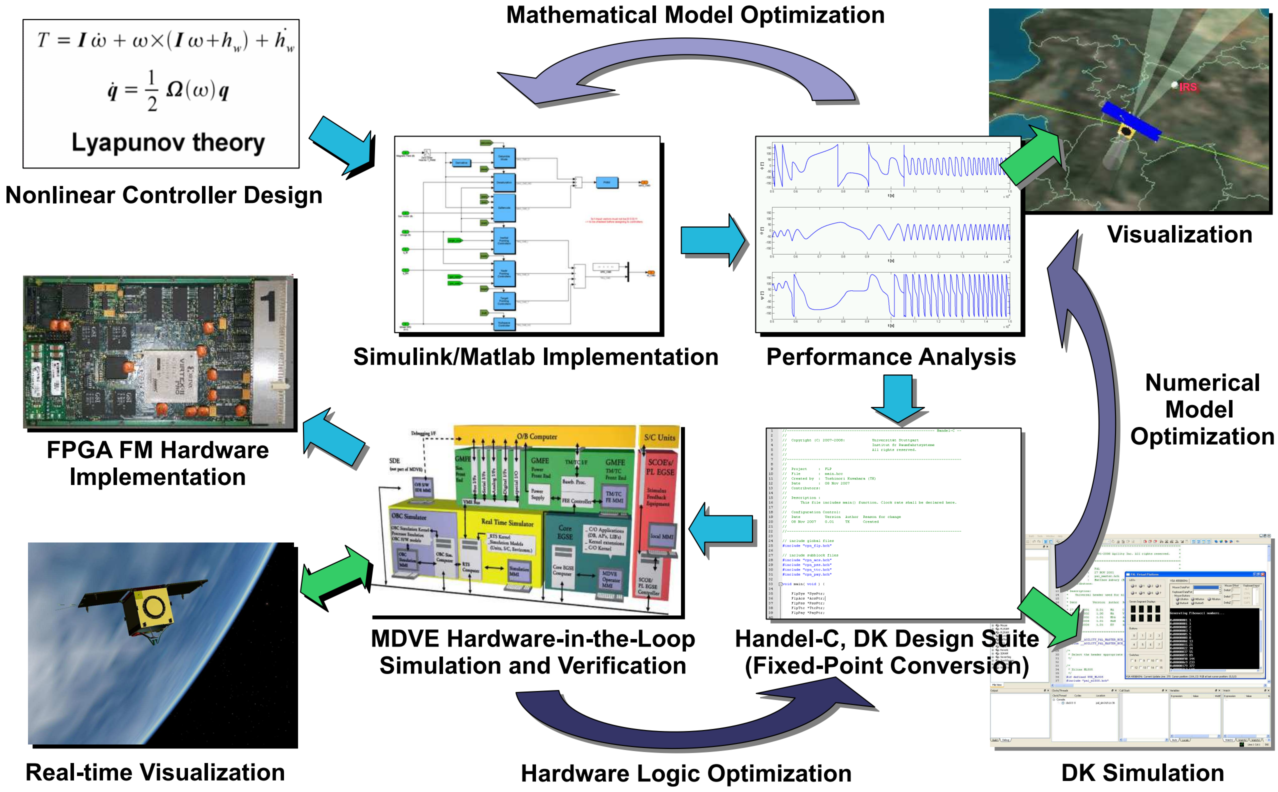

Mathematical Model Optimization
$T = \boldsymbol{I}\dot{\omega} + \omega\times(\boldsymbol{I}\omega+h_w) + \dot{h}_w$
$\dot{q} = \frac{1}{2}\,\boldsymbol{\Omega}(\omega)\boldsymbol{q}$
Lyapunov theory
Nonlinear Controller Design
IRS
Visualization
Simulink/Matlab Implementation
Performance Analysis
Numerical Model Optimization
FPGA FM Hardware Implementation
O/B Computer
S/C Units
Real Time Simulator
OBC Simulator
MDVE Hardware-in-the-Loop Simulation and Verification
Handel-C, DK Design Suite (Fixed-Point Conversion)
Real-time Visualization
Hardware Logic Optimization
DK Simulation

Figure 7.8.: Control algorithm development environment and procedures

8. FPGA Hardware-in-the-loop simulation results

This chapter summarizes the simulation and verification results based on the developed hardware-in-the-loop simulation environment as described in Chapter 7. Section 8.1 describes the simulation and verification results of attitude control algorithms of detumbling and safe modes implemented into the FPGA hardware logic, as well as the functionalities of the developed simulation interface SimFE. Section 8.2 describes the simulation results of the LEOP scenario, which validates the operational concept in this phase in terms of power budget. Section 8.3 summarizes the power budget simulation results, which verify the validity of system design of the *Flying Laptop* in terms of electrical design, power balancing design, operational scenario, and attitude control strategies. Finally, Section 8.4 summarizes simulation and verification results of a normal operation scenario including FDIR functions.

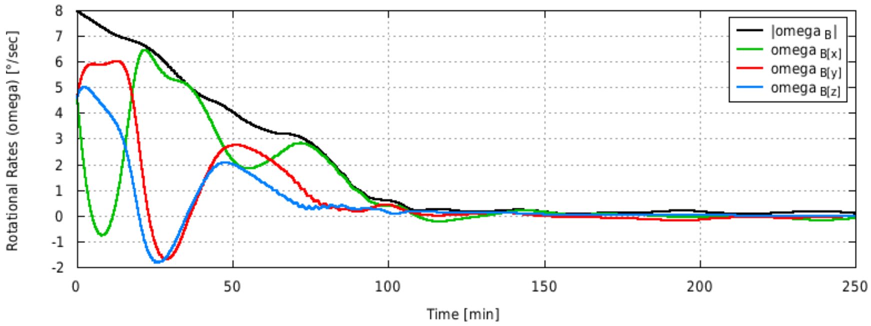
8.1. Attitude control algorithm simulation

The attitude control algorithms of the detumbling mode and the safe mode are simulated in the FPGA hardware-in-the-loop simulation facility. In order to verify the functional performance of the developed environment, the simulation results are evaluated by comparing with a mathematical model based on Mathworks MATLAB/Simulink. At the time of simulation, the moment of inertia of the *Flying Laptop* satellite was not yet available, and therefore, the inertia matrix and principal axis in Equation (D.15) and Equation (D.16) were used for the both simulations. The environmental model of the MDVE is much more precise than that of MATLAB model including additional effects of the residual magnetic dipole moment of the satellite, the atmospheric drag and the albedo effect. However, the albedo effect is deactivated in this section in order to enable a better qualitative verification of the control algorithms.

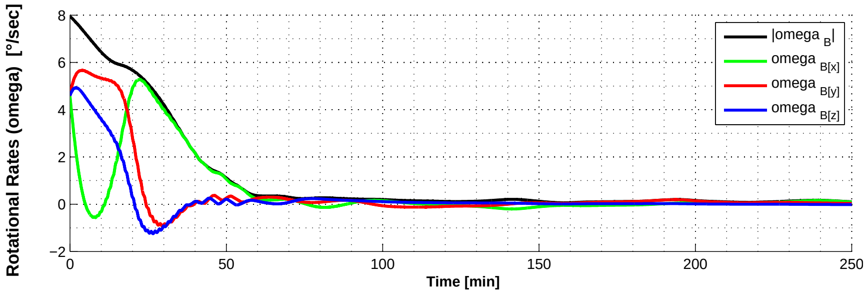
During the simulation, FPGA-based OBC measures the Earth's magnetic field and sun vector by accessing the magnetometer and sun sensor models in the RTS and sends back calculated commands to the magnetic torquer models in the RTS. RTS then propagates the satellite attitude and orbital position based on the induced torques and prepares sensor information for the next step. The reference orbit 1 (see Table 4.2) is selected for both simulations. Every simulation starts just after the satellite entered into the sun light out of the umbra.

8.1.1. Detumbling mode simulation

Figure 8.1 illustrates the angular rate of the satellite body and Figure 8.2 illustrates the induced torques after the separation from the launcher with an estimated initial angular rate of 8 °/s in both simulation environment, respectively.

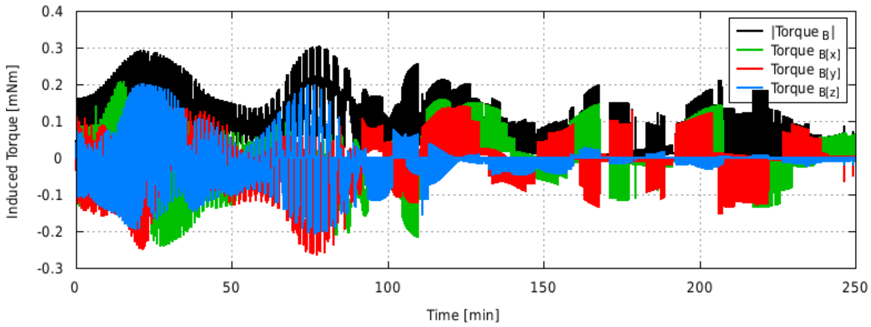


(a) MDVE hardware-in-the-loop simulation

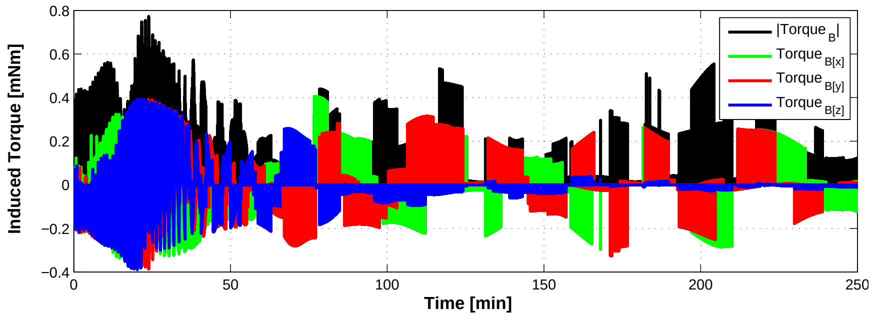


(b) MATLAB simulation

Figure 8.1.: Rotational rate in detumbling mode



(a) MDVE hardware-in-the-loop simulation



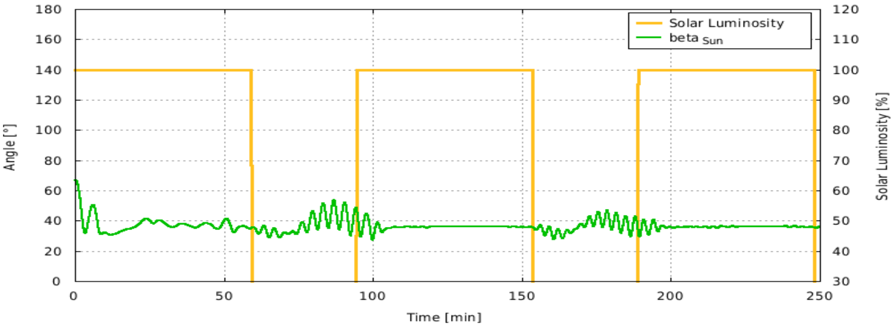
(b) MATLAB simulation

Figure 8.2.: Induced torque in detumbling mode

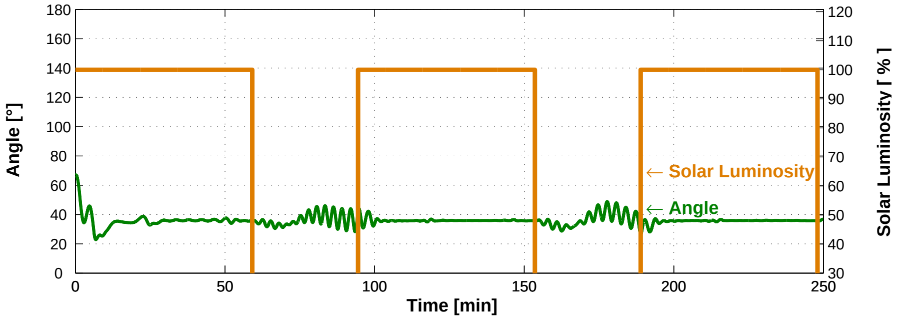
In this mode, the angular rate is reduced by means of magnetic torquers. It can be seen that the angular rates around the three axes are reduced near to zero in both simulations. The absence of the sun light does not play any role for the detumbling mode, because the rotational rate information is based only on the magnetometer measurement. The time duration needed for the sufficient detumbling is slightly more than 100 minutes in the MDVE simulation and around 60 minutes in the MATLAB model. This difference is caused by the greater disturbance forces applied in MDVE, thus deviation of attitude takes much longer to damp. Even though there is a quantitative difference between these figures, the comparison of them clearly shows the qualitative consistency between the two simulations. At the same time, the comparison of these figures shows that the FPGA-based OBC and magnetometer and magnetic torquer models in MDVE communicate with each other properly.

8.1.2. Safe mode simulation

Once the satellite initiated the safe mode, it spins itself up around its principle axis with the maximum moment of inertia in order to stabilize the attitude with coarsely orienting the solar panel normal toward the sun. Figure 8.3 illustrates the angle between solar panel normal and sun vector with the indication of sun light and umbra portion. The offset angle between solar panel normal and sun vector is about 38 °, because the principal axis, which is pointed toward the sun, is not aligned with the solar panel normal. When the satellite enters into the umbra, the pointing accuracy becomes worse because the satellite loses the sun vector information. The pointing stability becomes gradually better in later orbits. Figure 8.4 illustrates the rotational rate and Figure 8.5 illustrates the induced torques just like the case of the detumbling mode.

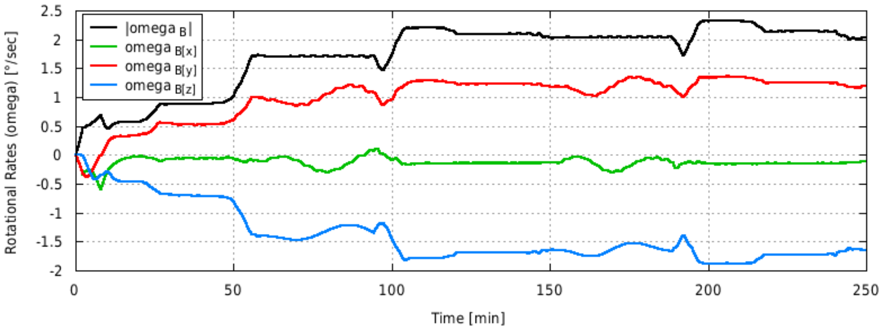


(a) MDVE hardware-in-the-loop simulation

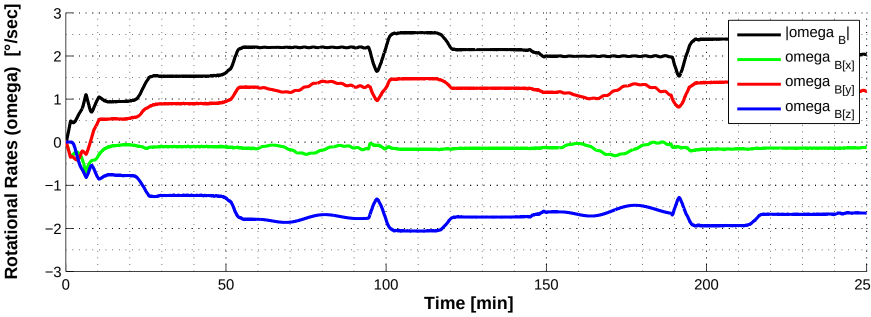


(b) MATLAB simulation

Figure 8.3.: Sun vector angle against solar panel normal in safe mode

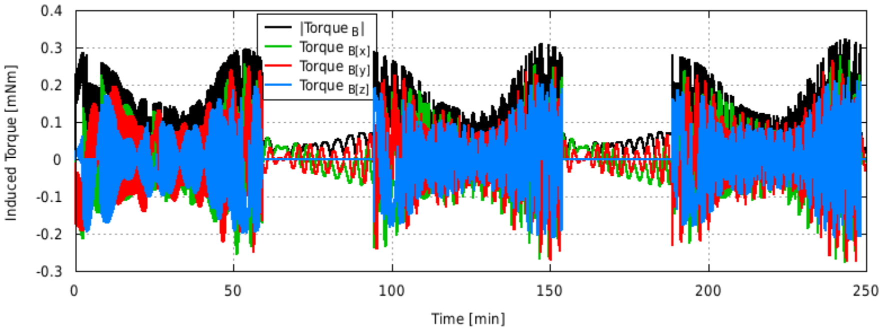


(a) MDVE hardware-in-the-loop simulation

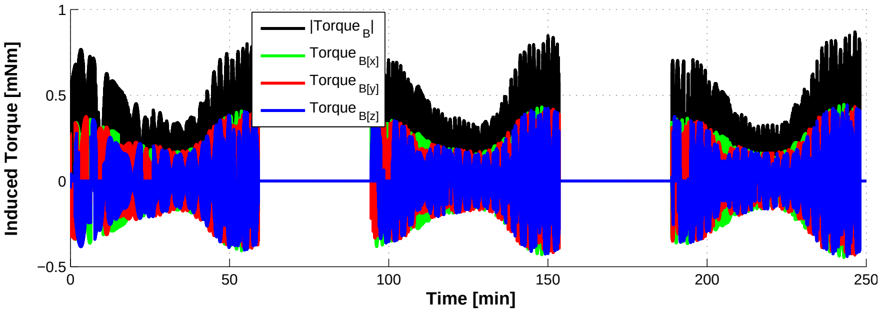


(b) MATLAB simulation

Figure 8.4.: Rotational rate in safe mode

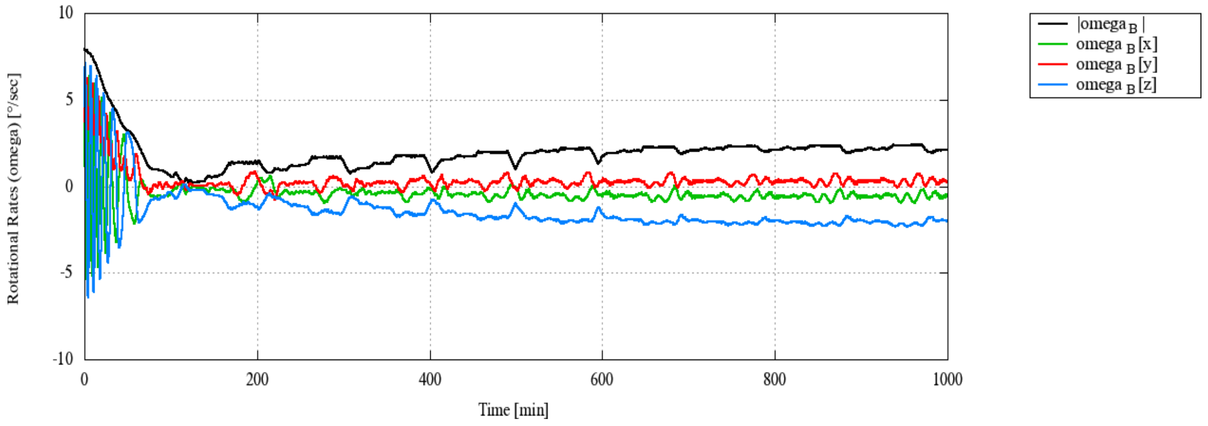


(a) MDVE hardware-in-the-loop simulation

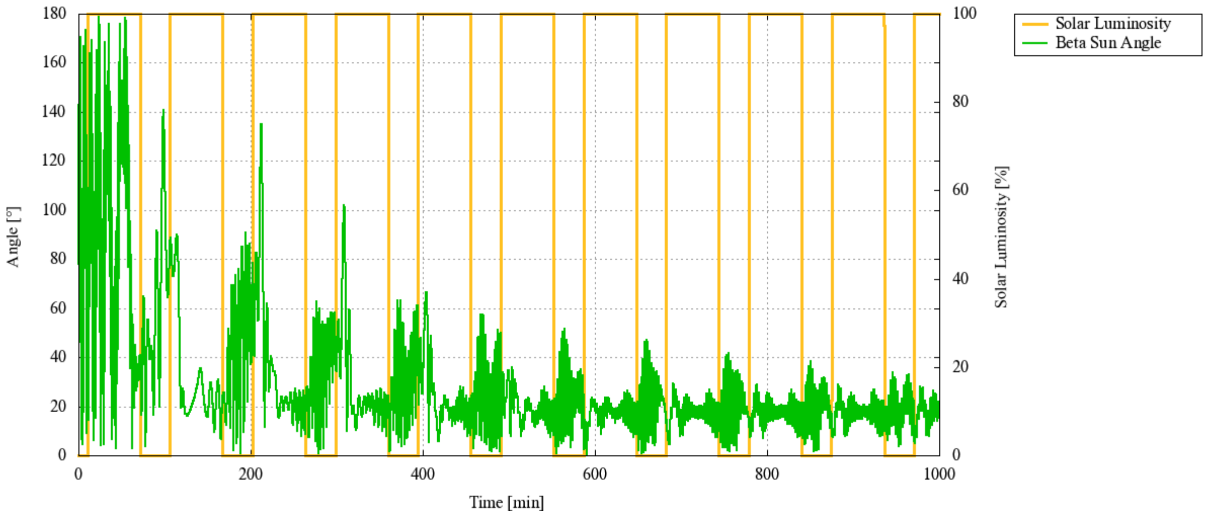


(b) MATLAB simulation

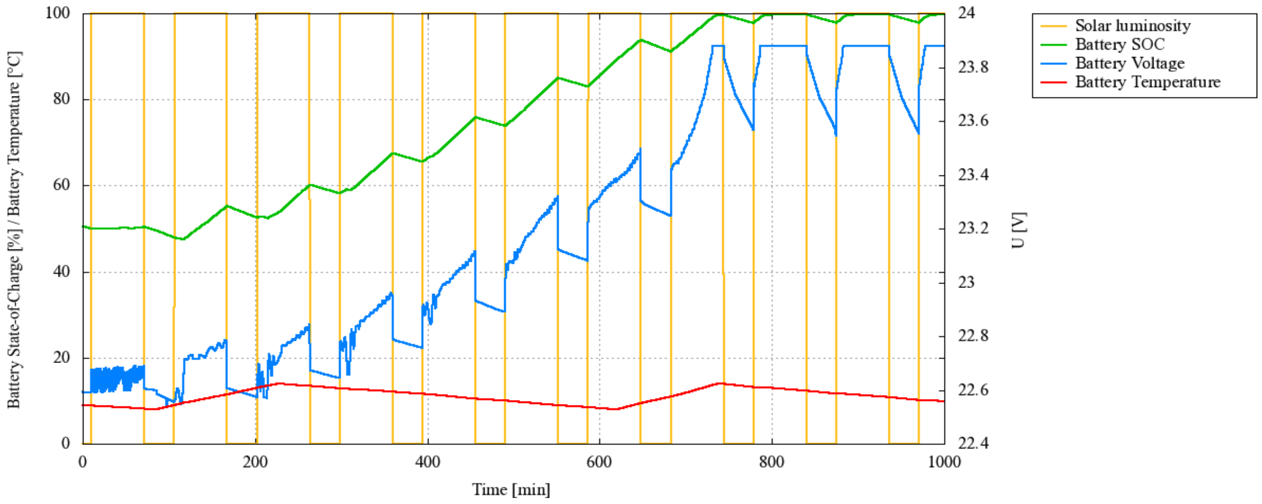
Figure 8.5.: Induced torque in safe mode



(a) Angular velocity



(b) Angle between sun vector and solar panel normal



(c) Battery SOC

Figure 8.6.: LEOP simulation

From Figure 8.4 it can be seen that the satellite gradually spins itself up at the beginning of the simulations. Because the estimated residual magnetic dipole moments of the satellite is implemented in the MDVE, the induced torque is not equal to zero even during the umbra (Figure 8.5). Comparisons of these figures prove the followings:

- all communication between OBC and software models in MDVE took place correctly.
- hardware control algorithm implemented in the FPGA performs correct attitude control same as mathematical model in MATLAB model.
- all component software models in MDVE are correctly implemented and work as same as real hardware components in the simulation environment.

According to the above consideration and evaluation, the validity of the implementation of the hardware-in-the-loop simulation and verification environment in terms of satellite's attitude behavior is verified. This developed simulation environment is capable of performing a real-time attitude simulation of the satellite, as well as evaluation and verification of the attitude control algorithms implemented in FPGA hardware logic.

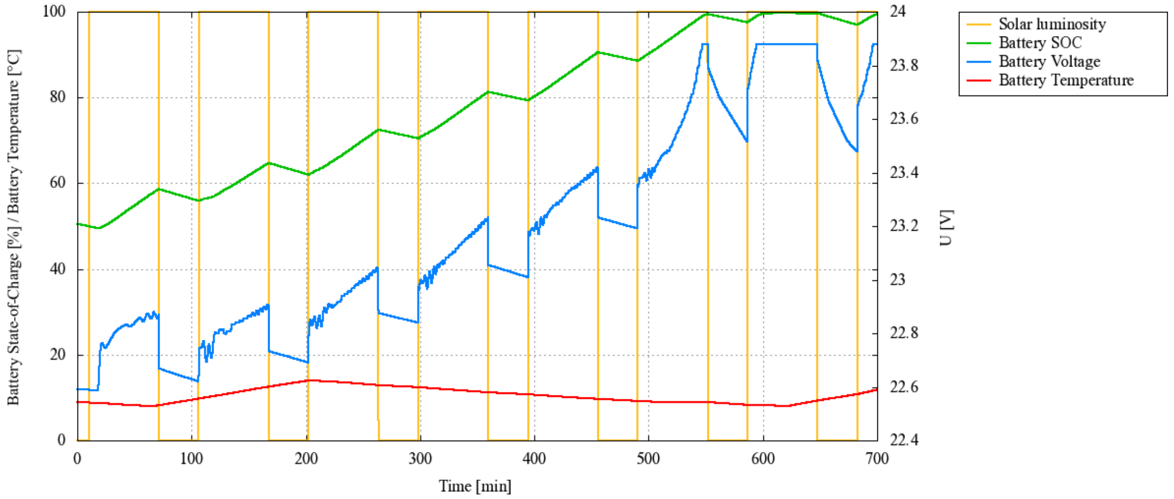
8.2. Launch and early orbit phase simulation

The satellite operation in the LEOP is simulated and the results are summarized in Figure 8.6. After the separation from the launcher, the satellite performs detumbling mode at first. Then it automatically moves into safe mode at the moment the angular rate becomes lower than a predefined value of 0.3 °/s. The solar panels are closed at the initial detumbling phase and then deployed at the same time the safe mode is initiated. For this simulation and all later ones, the albedo effect is included in the environmental model and also the actual moment of inertia of the *Flying Laptop* is used (see Appendix D.3).

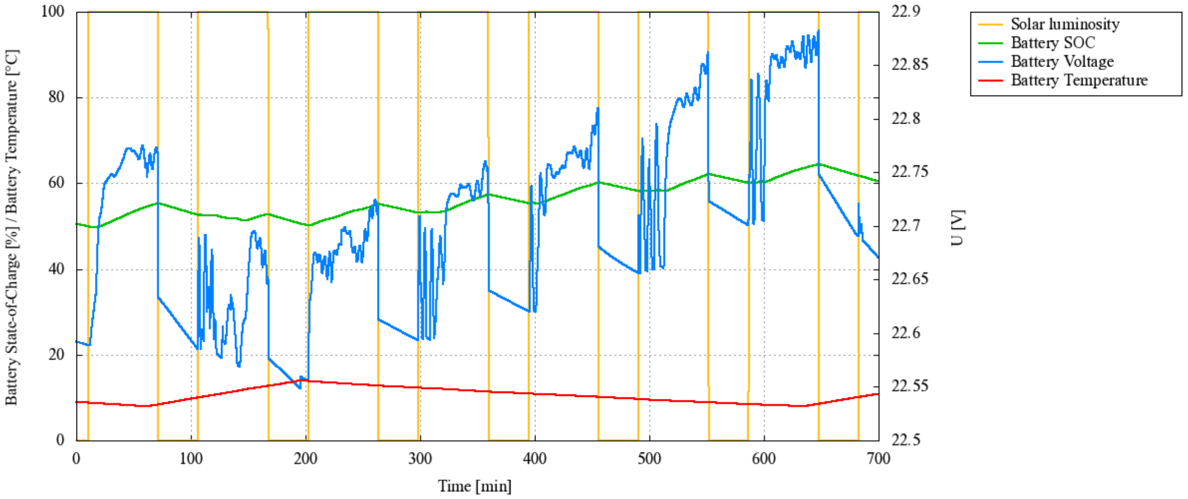
In this simulation, the initial angular rate of around 8 °/s is detumbled within about 80 minutes after the launcher separation and the satellite moves into safe mode deploying the solar panels at the same time (Figure 8.6(a)). The battery SOC remains almost the same until the safe mode is initiated. In the safe mode, the satellite spins itself up to the reference angular rate of 2 °/s. The angle between the solar panel normal and the sun vector becomes around 18 ° at the end of simulation, which comes from the offset between the reference principle axis and the solar panel normal (Figure 8.6(b)). Because of the albedo effect, the pointing stability is less than the result with no albedo effect illustrated in Figure 8.3(a). Finally, the battery state of charge successfully rises up to 100 % (Figure 8.6(c)), which ensures a secure establishment of the satellite's initial condition in the LEOP. Consequently, these simulation results ensure the validity of the LEOP concept of the *Flying Laptop*.

8.3. Power budget simulation

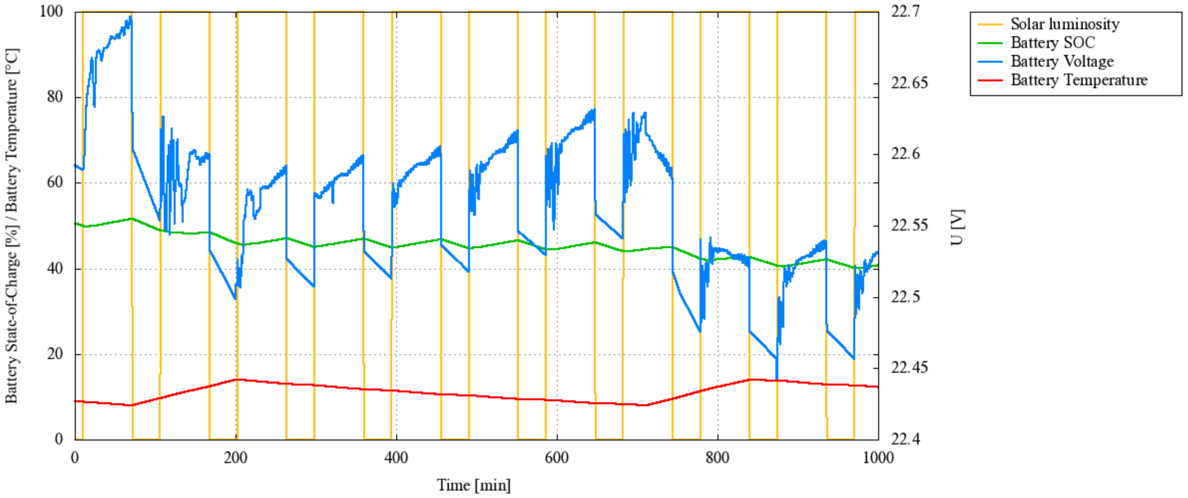
In order to verify the power balance design of the *Flying Laptop* described in Section 4.5, a set of power budget simulations for three solar panel deployment configurations A, $\mathrm{B_R}$, and C (see Figure 4.11) have been conducted in safe mode with the albedo effect and the actual moment of inertia of the *Flying Laptop*. With the same reason described in Section 4.5, the initial value of the battery state of charge is set to 50 %.



(a) Configuration A



(b) Configuration B_R



(c) Configuration C

Figure 8.7.: Power balance simulation

Figure 8.7 illustrates the simulation results for the three configurations. Figure 8.7(a) illustrates that the battery SOC rises up to 100 % in 6 orbits. Considering the fact that the simulation result in MATLAB model in Figure 4.12 required only 5 orbits to charge the battery to 100 % after the initiation of the safe mode, the modeling in MATLAB can be regarded as an optimistic one. This is because the MDVE includes not only electrical but also thermal models of all components, as well as the satellite structures and heaters. Therefore the simulation result by MDVE can be thought to be closer to the real behavior. This can be seen, e.g., in the figure that the battery temperature is continuously kept within the range of between 8 and 14 °C. The dependency of the battery voltage on the battery temperature and SOC can be also recognized. Furthermore, the albedo effect implemented inside the MDVE causes worse pointing accuracy of the solar panels toward the sun, which results in less power generation. According to these considerations, the simulation results in the hardware-in-the-loop simulation environment ensure the validity of the power balancing design of the satellite system in the configuration A.

Figure 8.7(b) and Figure 8.7(c) illustrate the simulation results in configuration $\mathrm{B_R}$ and C, respectively. The battery can be safely charged up to 100 % even in the configuration $\mathrm{B_R}$. Because all missions of the *Flying Laptop* satellite are single-shot missions, this simulation result indicates that the satellite can perform all missions from time to time by charging the battary during sufficient amout of time intervals. Even the Ka-band communication can be performed if the battery SOC once becomes near to 100 %. On the other hand, the latter figure indicates that the battery SOC continuously decreases in the configuration C, while the corresponding simulation result by MATLAB model suggested the opposite (Figure 4.14). Therefore, both of the side solar panels shall be deployed as early as possible after the separation from the launcher. However, based on the single-failure tolerant design concept (only one deployment failure either the left or right solar penel is considered), this configuration C is not a driving requirement for the satellite design. If this scenario actually happens, the on-board control algorithm shall be changed to use another attitude control strategy, which wins higher pointing accuracy of the solae panel toward the sun at the price for using none high-rel components, for example, such as fiber optic gyros. The configuration C is only enough to keep the SOC for several orbits.

8.4. Satellite operation simulation

The purpose of this simulation is to demonstrate the capability of standard operation of the *Flying Laptop* satellite, as well as to verify the functionalities of the developed on-board control algorithms, including the operability of BRDF measurement experiment and Ka-band high-speed communication, the commanding concept and structure, and FDIR algorithms. This simulation results shall verify the development activities of the thesis.

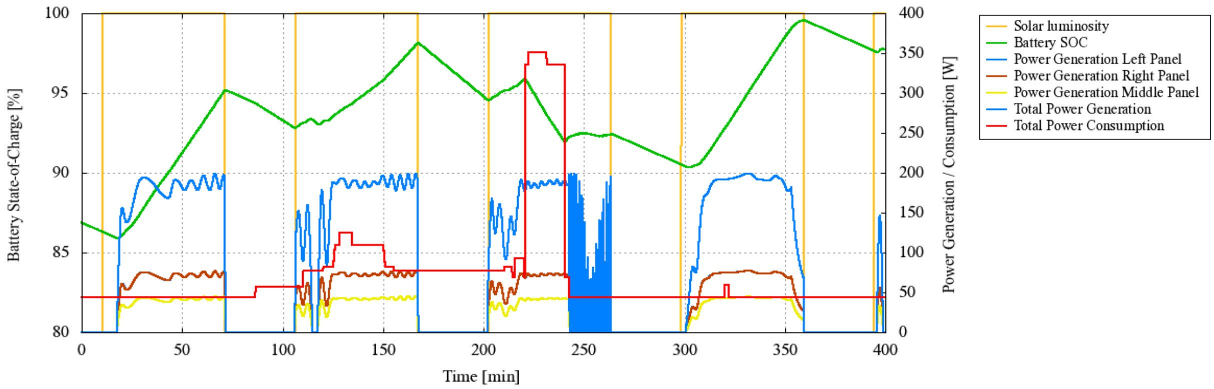
The relevant subsystems to this simulation are the ACS, P/L and TT&C subsystems. The state-machines of the system, ACS, and TT&C are illustrated in Figure 4.8, Figure 4.10, and Figure C.1, respectively. The state-machine of the P/L camera system has only two modes: Off and Operation mode, while the MICS, TICS, and PAMCAM themselves have three modes: off, idle, and operation modes. Other components have only on and off modes. The operational scenario of this simulation is summarized in Table 8.1. The satellite is at first in safe mode with an initial angular velocity of 0 °/s and a battery SOC of around 86 %. The satellite establishes a stable attitude with solar panels pointed toward the sun in about 100 minutes.

Table 8.1.: Satellite operation scenario

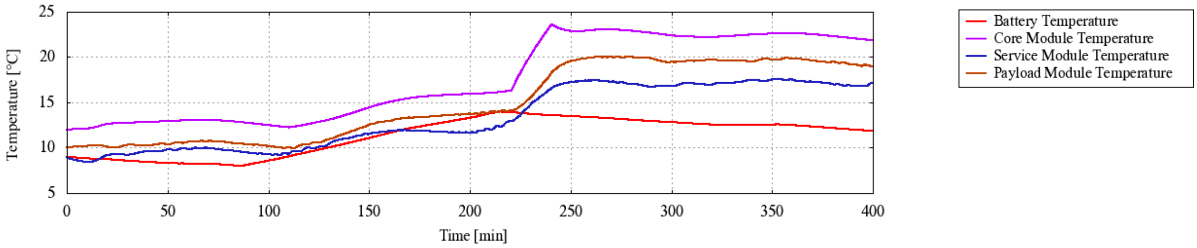
Time [min]	Sys. event	ACS event	P/L event	TT&C event
0	Safe mode (4)	Safe mode (1)	Off mode (0)	Stand-by mode (0)
110	Idle mode (5)	Idle mode (2)	|	|
115	|	|	Operation mode (1)	|
120	Active op. (6)	Nadir P. mode (4)	|	|
125	|	|	Camera Idle (1)	|
130	|	Target P. mode (5)	Camera Operation (2)	|
145	|	Nadir P. mode (4)	Camera Idle (1)	|
150	|	|	Camera Off (0)	|
155	Idle mode (5)	Idle mode (2)	Off mode (0)	|
210	Active op. (6)	Nadir P. mode (4)	|	Standard contact mode (1)
215	|	|	|	S-Band LG On (1)
220	|	|	|	High-speed com. mode (3),
|	|	|	|	S-Band LG Off (0),
|	|	|	|	Ka-Band On (1)
225	|	Target P. mode (5)	|	|
240	|	|	|	Stand-by mode (0),
|	|	|	|	Ka-Band Off (0)
242	Safe mode (4)	RW malfunction	|	|

After coming into the sun light from the umbra, the satellite moves into idle mode to prepare for the BRDF measurement. At 115 minutes, the P/L moves into operational mode. Then the satellite performs nadir pointing mode from 120 minutes in order to start the target pointing mode at 130 minutes. P/L turns camera instruments on at 125 minutes. At 130 minutes, the satellite initiates the BRDF measurement performing the target pointing and image aquisitions for 15 minutes, before ACS moves back to nadir pointing mode at 145 minutes. P/L cameras go back to idle mode then turned off at 150 minutes, and P/L itself remains in operation mode 5 minutes longer until the satellite moves into idle mode. At 210 minutes, ACS initiates nadir pointing mode and the TT&C goes into standard contact mode. At 215 minutes, the S-band Low Gain is turned on until the TT&C moves into the high-speed Ka-band communication mode at 220 minutes and turns Ka-band instruments on in order to heat up amplifiers for a stable communication. From 225 minutes, the satellite performs high-speed communication for 15 minutes. At 240 minutes, TT&C goes back to stand-by mode. Finally, at 242 minutes, ACS experiences malfunction of a reaction wheel and detects a higher angular velocity than the pre-defined maximum value of 3 °/s. The FDIR function automatically decide to move back into the detumbling mode imediately and turns all components off, which are not needed in the system safe mode. After the angular velocity becomes smaller than 0.3 °/s, the ACS moves into safe mode again and stabilizes the attitude. All of above mode changes are conducted by sending commands from the ground station based on the SCOS-2000.

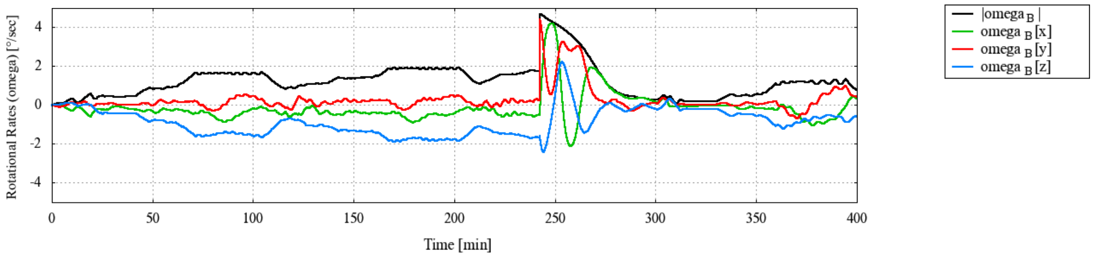
In Figure 8.8, the simulation results of the above operational scenario is summarized. Since only two attitude control algorithms were converted into OBC control algorithms for FPGAs at the time of writing, the idle mode and three pointing modes of the ACS perform the safe mode. From the amount of the total power consumption in Figure 8.8(a), one can see that all components are correctly turned on and off according to the commands received. Several heaters are also automatically turned on and off. It also illustrates that the battery SOC decreases about 4 % during the high-speed communication, and is charged back to 100 % in the next orbits even in the safe mode.



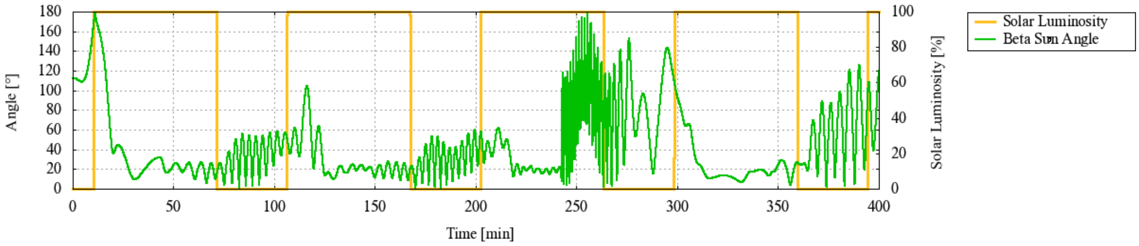
(a) Power balance and battery SOC



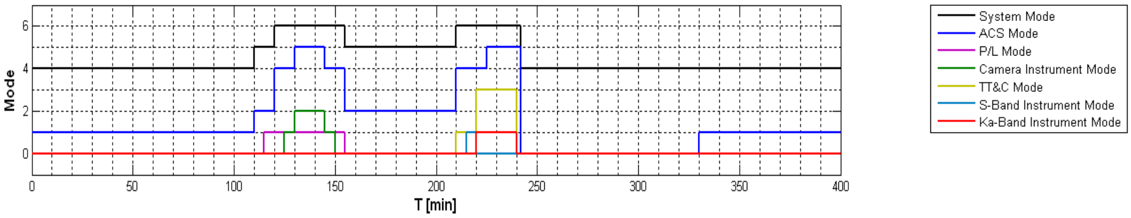
(b) Temperatures of core, service, and payload modules and battery



(c) Angular velocity



(d) Angle between sun vector and solar panel normal



(e) Mode transitions

Figure 8.8.: Normal operation scenario simulation

From Figure 8.8(b), one can also see that the temperature of the core module dramatically rises during the Ka-Band high-speed communication, where the traveling wave tube amplifier is mounted on. The other modules also receives the heat flux with a certain time delay. After the communication, the temperatures gradually decrease, again. From Figure 8.8(c) and Figure 8.8(d), it is easy to notice that the satellite experienced a disturbance at 242 min. The rotational rate became more than 4 °/s. As the reaction against this contingency event, the satellite system went back to the safe mode and let the ACS perform detumbling mode to reduce the rotational rate. Around at 330 min, the ACS successfully finished detumbling and initiated the safe mode. This is also clearly summarized in Figure 8.8(e). The figure illustrates the mode transitions of the system, subsystems, and components took place correctly as they are designed. Though some attitude control algorithms were not available, this simulation illustrates the secure operational capability and correct functionalities of the developed satellite control algorithm implemented inside the on-board computer breadboard model.

8.5. Summary

This chapter summarized simulation results by the hardware-in-the-loop simulation and verification environment, from which the following conclusions can be derived.

- The developed hardware-in-the-loop simulation and verification environment is capable of simulating the behavior of FPGA-based on-board computers in real-time to verify the whole satellite control algorithms to be implemented in FPGAs. The developed special simulation interface SimFE works correctly and realizes asynchronous communications between the CA inside an FPGA and the RTS.
- The developed simulation environment is capable of attitude control algorithm simulations supported by environmental models in the MDVE. Comparison with the reference results by mathematical MATLAB model qualitatively verified its correct simulation capability. This simulation environment becomes the basis of attitude control algorithm development.
- Attitude control algorithms of detumbling mode and safe mode implemented in the FPGA with fixed point conversion were qualitatively verified. This is the world's first prove of the correct real-time functionality of satellite attitude control algorithms implemented inside an FPGA.
- Validity of the LEOP concept of the *Flying Laptop* was verified. The simulation result illustrated the safe initialization of the satellite after the orbit insertion.
- Validity of the power balancing design together with the electrical component design, operational design, and attitude control strategies of the *Flying Laptop* was verified.
- Operability of the whole satellite system down to the subsystems and components by telecommanding was verified. A sample operational scenario including a BRDF measurement and a high-speed Ka-band communication was demonstrated. This result verifies the validity of the satellite's state-machine design, mode design, and commanding structure design.
- Preliminary FDIR function implemented in the FPGA was demonstrated, which verifies the developed implementation strategy of the FDIR functions. Also an automatic mode transitions and solar panel deployment were demonstrated and their correct functionalities were verified.

9. Formal verification of Handel-C program

It is often the case that software and hardware systems which require high safety and/or security are subject to be formally verified in industrial fields. A satellite system is also one of those critical systems for which extremely high safety and security is required. Formal verification supported by formal methods has capability of providing mathematically-reasoned specification, development and verification for such systems.

This also applies to the control algorithms which shall be implemented into FPGAs as hardware logic. FPGA based reconfigurable satellite on-board computing systems require hardware logics to be implemented and updated just like software for usual computers based on processors. Moreover, the algorithms implemented inside an FPGA are parallel running concurrent processes. On the one hand, the strong logic design capability of the Handel-C enables flexible and rapid hardware design, on the other hand, the produced algorithms shall be proven as failure free by some means. The possible methods of verifying this control algorithm is to test the functionality in the simulated environment in a form of point-to-point test. Unfortunately, this method can prove fulfillment of defined requirements, but it can not ensure existence or absence of certain properties, e.g., dead-lock. In order to verify these special properties of control algorithms, formal verification is the only way of proving. Thinking about the fact that the control algorithms in FPGAs are combinational parallel running hardware logics, testing of all possible combinational behaviors every time a new one is developed is not feasible at all. Formal verifications provide not only verification of the developed algorithms, but also the specification of algorithm design activities. For the realization of formal verification of Handel-C codes, there is a clear need for both a formal semantics of Handel-C as well as an appropriate methodology and tool support. In this chapter, the formal verification methods of Handel-C is investigated in detail and the formal semantics of the Handel-C which also incorporates the signal construct is established. The notation references used in this chapter are summarized in Appendix F.

9.1. Formal verification of concurrent systems

The study of formal verification methods of concurrent systems has actively started in 1980's after C. A. R. Hoare has published his *Communicating Sequential Processes* (CSP) in 1985 [125]. A. W. Roscoe has then provided comprehensive and partly extended version of the CSP in his publication [126]. It is also described that SCP can be analyzed and solved by Failures/Divergence Refinement (FDR, now FDR2) model checking tool over a wide range of application areas. As Handel-C came to the world in around early 1996, it was considered that it is a hybrid of CSP and C [127], [128]. Indeed, some of the syntax of Handel-C are based on that of CSP, e.g., those of channels "$c!e$" (see Appendix F). The study of formal verification of

Handel-C program has been started by several researchers since the beginning of 21st century. An extensive investigation has been done by Butterfield [129]. As he mentioned in [110] and [130], it was not possible to handle "prialt" (prioritized alternative) syntax of Handel-C because of the difficulty treating the priorities in concurrent processes in CSP. One of the main issues is that the asynchronous nature of CSP formalisms makes it very difficult to establish when prialts are "coming together," in order for their communication requests to be resolved. Focusing on the presence of a synchronizing clock in Handel-C, Butterfield treated the behavior of the prialt as static problem of resolving priorities of a known collection of prialts, and presented complete formal semantics for prialt in [110]. At this point, channels are treated as a prialt with only one guard condition. Recently, the hardware compiler semantics, one of the formal semantics of Handel-C, is to be translated into a UTP (Unifying Theories of Programming) framework for realization of formal verification [129].

The main concern here was the dead-lock caused by priority in channel communication in prialt. A prialt selects the highest channel communication (the channel communication listed at the top of the list in prialt) out of those which are ready in the clock cycle (see Appendix E.10). The following example with synchronized two processes is the typical case of a dead-lock, because the priority between the two communication channels forms a loop.

```
chan unsigned 1 ch1, ch2; // shared by both processes
unsigned 1 x, y;
//process A                          //process B
prialt {                             prialt {
    ch1!1: break;                        ch2!1: break;
    ch2?x: break;                        ch1?y: break;
}                                    }
```

One important point, however, is that these previous research have focused on the channel communications as the standard communication method between two concurrent processes and did not include signal constructs (see Appendix E.7). Because signals allow broadcasting of information from one to many processes, while channels allow only one-to-one communication, signals enable simultaneous synchronization among multiple processes. Moreover, the signal lines are implemented as wires in FPGAs, which is very efficient in terms of hardware resources. It is now a great concern whether the signal construct can be included in the formal semantics described above. In this chapter, an investigation on including signal construct into operational semantics and hardware compiler semantics is described in detail. Furthermore, the previously proposed formal semantics by other authors are limited in a single clock domain. In Section 9.5, one possible method of extending the scope to multiple clock domains is investigated.

9.2. Scope of Handel-C semantics

The scope of the Handel-C semantics can be summarized in following four components:

Types: A range of Handel-C data types ultimately reduce down to specification of bit strings of fixed length.

Synchronous "cores": Regions of hardware under the control of a single clock. Operational semantics described here is about the behavior of these cores.

Priority: The “channel” communication constructs of Handel-C are provided in the form of prialt-statements, which requires all choices between communication events to be prioritized.

Asynchronous environment: The synchronous cores can be regarded that they communicate with each other and with the environment via asynchronous interfaces.

9.2.1. Handel-C abstract syntax

The operational semantics suggested by Butterfield does not include the signal construct. Here, the complete abstract syntax, including the signal construct is developed. Following the methodology given in [110], firstly, two given sets of identifiers can be defined: one for variables ($v, x, y, z \in Var$) and the other for channels ($c, d \in Ch$). Also provided are expressions built from constants, variables and various operators and functions ($e, s, b \in E$). In addition to these, the target sets of signals can be also defined as ($sig \in Sig$). Variables and expressions includes both booleans and integers. According to these definitions, the abstract syntax can be expanded in terms of signal construct as follows.

$$
\begin{array}{rcll}
p \in P :: & = & \mathbf{0} & \text{Skip} \\
 & | & \mathbf{1} & \text{Delay} \\
 & | & v := e & \text{Assignment} \\
 & | & sig :\bumpeq d \leftarrow e & \text{Signal Assignment} \\
 & | & p_1 \lhd c \rhd p_2 & \text{Conditional} \\
 & | & s \blacktriangleright [p_i] & \text{Selection} \\
 & | & b * p & \text{While} \\
 & | & p_1 ; p_2 & \text{Sequential Composition} \\
 & | & p_1 \parallel p_2 & \text{Parallel Composition} \\
 & | & \langle g_i \rightarrow p_i \rangle & \text{Prioritized Choice (prialt)} \;.
\end{array} \tag{9.1}
$$

The unit delay (1) statement does nothing, but takes one complete clock cycle to do it. The assignment statement ($v := e$) evaluates the expression e consulting the current value of any variables it references. The variable v is then updated at the end of the clock cycle. The signal statement ($sig :\bumpeq d \leftarrow e$) holds the default value d in sig unless the expression e is not assigned at the beginning of the clock cycle. If it is assigned with expression e, the value of the sig changes at the very beginning of the clock cycle before any selection happens. The conditional statement ($p_1 \lhd c \rhd p_2$) evaluates its condition at the beginning of the clock-cycle, and then executes p_1 if c is evaluated to true, otherwise it turns p_2. The selection (switch or case) statement ($s \blacktriangleright [p_i]$) evaluates the expression s, which should always result in a value in the range $1..n$. The process p_i for which $i = s$ then starts executing immediately. The iteration (while) statement ($b * p$) evaluates the boolean expression b. If false, the statement terminates immediately, otherwise the process p has terminated. This activity continues until b is evaluated to false. The sequential composition ($p_1 ; p_2$) of two statements simply indicates that the first process runs to completion, and then the second starts up immediately afterward. Parallel composition ($p_1 \parallel p_2$) involves both processes starting simultaneously, and it terminates when both processes have finished. The prialt statement ($\langle g_i \rightarrow p_i \rangle$) can be viewed as a sequence of guard-process pairs, where the guards ($g \in G$) are either communication actions like input ($c?v$) or output ($c!e$) or a default guard ($!?$) to be invoked if no communication guard is enabled. Hence:

$$g \in G ::= c?v \mid c!e \mid !? \; . \tag{9.2}$$

9.3. Extended operational semantics

This section summarizes the extended operational semantics which includes signal construct.

9.3.1. Operational phase

The Handel-C has sequential and parallel constructs, global variable assignment, and channel communication. For the establishment of the operational semantics for Handel-C, it is necessary to understand the process behavior of all constructs during a clock cycle. For the inclusion of the signal construct, the behavior of them shall be analyzed. Typical signal assignment is described in Appendix E.7. The value assignment to a signal takes place at the very beginning of the clock cycle, before other first conditional processes are evaluated. Because the scope of atomic actions of the former operational semantics by Butterfield starts from the conditional evaluation, atomic actions by signal shall be inserted before that. Because this phase is related to the signal propagation, it can be named as "signal propagation" (*pro*) phase. The next following phase to this is the "select" (*sel*) phase, in which the conditional and while-loop expressions are evaluated based on the state of "signal" as well as the flow of control from the statements executed on the previous clock cycle. The next phase is the "request" (*req*) phase. In this phase, all the selected prialts load their corresponding requests with the environment. At the end of this phase, the system has global knowledge of communication request associated with this clock cycle. In the fourth "resolve" (*res*) phase, the system resolves all these requests and define which communication actions are going to be executed. The final "action" (*act*) phase is where all the atomic assignment and active communication actions take place. In this stage, the permanent change on system state is made, which persist across clock boundaries.

All processes in these phases are actually implemented into hardware logic. The first four phases are implemented as combinational logic, which finally makes decisions well within the time allocated to the clock cycle. The last phase is synchronized to the end of the cycle. Because the resolution determines the final communication actions between parallel processes which affect to consequent permanent state changes, the sequencing of these phases are very important. In the hardware implementation, this can be achieved by letting the resulting combinational logic have enough time to settle into a consistent state.

9.3.2. Phase transitions

Transition types: A transition type ($TType$) is one of the above mentioned five phases, therefore:

$$TType \mathrel{\widehat{=}} \{pro, sel, req, res, act\} \quad pro < sel < req < res < act \; . \tag{9.3}$$

Now statements can be classified by associating a transition type set ($tts(p)$):

$$
\begin{aligned}
tts &: P \to \mathbb{P}TType \\
tts(\mathbf{0}) &\mathrel{\widehat{=}} \{pro\} \\
tts(sig :\doteq d \leftarrow e) &\mathrel{\widehat{=}} \{pro\} \\
tts(p_1 \lhd c \rhd p_2) &\mathrel{\widehat{=}} \{sel\} \\
tts(s \blacktriangleright [p_i]) &\mathrel{\widehat{=}} \{sel\} \\
tts(b * p) &\mathrel{\widehat{=}} \{sel\} \\
tts(\langle g_i \rightarrow p_i \rangle) &\mathrel{\widehat{=}} \{req\} \\
tts({}^{+}\langle g_i \rangle) &\mathrel{\widehat{=}} \{req\} \\
tts(a\langle g_i \rangle \blacktriangleright [p_i]) &\mathrel{\widehat{=}} \{res\} \\
tts(w\langle g_i \rangle * p) &\mathrel{\widehat{=}} \{res\}, w = FALSE \\
tts(w\langle g_i \rangle * p) &\mathrel{\widehat{=}} \{act\}, w = TRUE \\
tts(\mathbf{1}) &\mathrel{\widehat{=}} \{act\} \\
tts(v := e) &\mathrel{\widehat{=}} \{act\} \\
tts(p_1 ; p_2) &\mathrel{\widehat{=}} tts(p_1) \\
tts(p_1 \parallel p_2) &\mathrel{\widehat{=}} tts(p_1) \cup tts(p_2) \\
ttype &: P \to TType \\
ttype(p) &\mathrel{\widehat{=}} min(tts(p)) \ .
\end{aligned}
\tag{9.4}
$$

Based on this transition type, the system state can be defined as a tuple:

$$(p, t, \rho, \gamma, B, \tau) : State \ , \tag{9.5}$$

where:

$p : P$	process syntax-state
$t : TType$	current transition type
$\rho : Id \xrightarrow{m} Val$	variable environment
$\gamma : Ch \xrightarrow{m} G^{+}$	active channels
$B : \mathbb{P}(G^{+})$	requested/brocked prialts
$\tau : \mathbb{N}$	current clock value

Given an initial program p_0, the initial system state can be provided as:

$$(p_0, pro, \theta, \theta, \emptyset, 0) \ , \tag{9.6}$$

where θ denotes an empty map, and $\emptyset$ an empty set.

Transition events: Formally, all events can be regarded as functions on global state:

$$Evt \mathrel{\widehat{=}} State \to State \ , \tag{9.7}$$

$$[\![\cdot]\!]_T : P \to Evt \ , \tag{9.8}$$

$$[\![x := e]\!]_T(\ldots,\rho,\ldots) \mathrel{\widehat{=}} (\ldots,\rho \dagger \{x \mapsto [\![e]\!]_\rho\},\ldots) \ , \tag{9.9}$$

$$[\![^+\langle g_i\rangle]\!]_T(\ldots,B,\ldots) \mathrel{\widehat{=}} (\ldots,B \cup \{\langle g_i\rangle\},\ldots) \ . \tag{9.10}$$

Transition relation: Given a state pair consisting of program (p) and transition type (t), transition conditions can be defined. By including the signal construct, the transition relation suggested by Butterfield can be extended by inserting corresponding two transition conditions. All transition conditions are summarized in Table 9.1. The system switches from one state to the next higher when no more processes are wishing to do transitions of lower type.

Table 9.1.: Extended transition conditions

transition	enable condition	changes
pro	$t = pro \wedge ttpype(p) = pro$	$p \xrightarrow{pro} p'$
$pro2sel$	$t = pro \wedge ttpype(p) \neq pro$	$t := sel$
sel	$t = sel \wedge ttpype(p) = sel$	$p \xrightarrow{sel} p'$
$sel2req$	$t = sel \wedge ttpype(p) \neq sel$	$t := req$
req	$t = req \wedge ttpype(p) = req$	$p \xrightarrow{req} p'$
$req2res$	$t = req \wedge ttpype(p) \neq req$	$t := res$
res	$t = res \wedge ttpype(p) = res$	$p \xrightarrow{res} p'$
$res2sel$	$t = res \wedge ttpype(p) = sel$	$t := sel$
$res2act$	$t = res \wedge ttpype(p) = act$	$t := act$
act	$t = act \wedge act \in tts(p)$	$p \xrightarrow{act} p'$
$act2pro$	$t = act \wedge act \notin tts(p) \wedge actttype(p) = pro$	$t := pro$
$act2sel$	$t = act \wedge act \notin tts(p) \wedge actttype(p) = sel$	$t := sel$

The complexity arises when $t = act$, and signal propagation occurs as the resulting process. This signal may again trigger other "select" conditions in different place, and therefore, the *pro* is selected as the next process. This can be seen in the following example code.

```
static signal unsigned 1 sig1 = 0; // shared by all processes
static signal unsigned 1 sig2 = 0; // shared by all processes
static signal unsigned 1 sig3 = 0; // shared by all processes
do {                   do {                        do {
    sig1 = 1;              if(sig1==1) {             if(sig1==1 && sig2==1) {
    delay;                     sig2 = 1;                   sig3 = 1;
} while(1);                } else {                    } else {
                               delay;                      delay;
                           }                           }
                       } while(1);                 } while(1);
```

The first process sets the value of $sig1$ in the first *pro* phase, the second and third process evaluates the "if" condition in the first *sel* phase. As the evaluation result, the second process sets the value of the $sig2$ as "1" in the first *act* phase. At this point, the third process selected the action of *delay*. However, after the propagation of the signal of $sig2$, the "if" condition of the third process is eventually evaluated as **TRUE**, and the value of the $sig3$ is set as "1". Which action the third process finally choose is defined at the end of the current clock cycle. For the detailed description about the behavior of the prialt see [110].

Transition rules: Transition rule of the signal construct can be defined as follows using the semantics suggested in [110].

$$\textbf{Sig-Assign}\frac{}{sig :\overset{\frown}{=} d \leftarrow e \xrightarrow[sig:\overset{\frown}{=}d\leftarrow[\![e]\!]_\rho]{pro} \mathbf{0}} \ . \tag{9.11}$$

After execution of the signal assignment, the value is kept as the result value of e based on the variable environment ρ as defined in Section 9.3.2, until the end of the current clock cycle. After the "tick," the value is set to the default value d.

Execution trace: An execution trace of a Handel-C program can be finally described as a sequence of transitions occurring with the following structure:

$$((pro^*; sel^*; req^*; res^*)^*; act^+)^+ \ , \tag{9.12}$$

where ";" denotes concatenation, and x^* and x^+ denote zero-or-more xs and one-or-more xs, respectively.

9.4. Extended hardware compiler semantics

As described in [110], nested prialts require us to iterate its request-resolve loop several times in any given clock cycle. Combined signal lines shall be also iteratively solved tracing the final result of combinational signal propagation. Managing this micro-cycles activity severely complicates the operational semantics. However, the underlying hardware does not iterate, as it just propagates signals through combinational logic and also make decision on which communication lines are to be active in any given clock cycle using combinational logic.

The key concept behind the hardware semantics is to recognize that the resulting hardware simply consists of a fixed bank of registers, connected by fixed combinatorial logic – in effect a large (finite-) state-machine [127]. New values of the register state are computed as a function of the current values on each clock cycle. From this point of view, the hardware semantics of a Handel-C program is therefore simply a fixed function, that is:

$$f : State \rightarrow State \ , \tag{9.13}$$

where $State$ denotes the contents of all registers. The relationship between operational and hardware compilation semantics can be seen in [127].

The contribution of this thesis is to extend the equations that model the behavior of the hardware in terms of signal constructs, to describe how f is computed. The hardware features that need to be modeled are: registers, multiplexers, input $start : \mathbb{B}$ and output $finish : \mathbb{B}$ signals to and from program statements (see Appendix E.2), clock cycle control tokens, as well as "signals." These shall be expressed using a set of equations Eqn. As the consequence, the overall system is described as a list of such equations:

$$Sys \mathrel{\widehat{=}} \mathbb{P}Eqn \ . \tag{9.14}$$

Following the convention suggested in [127], the above features can be described as building blocks. The extended description including "signals" is summarized as follows:

Multiplexers:

$$\begin{aligned} mux_n &: \mathbb{B}^n \to \mathbb{Z}^n \to \mathbb{Z} \\ mux_n(c_1, \ldots, c_n)(data_1, \ldots, data_n) &\widehat{=} data_i, if c_i \ . \end{aligned} \tag{9.15}$$

Registers:

$$register[load : \mathbb{B}, in : \mathbb{Z}] : \mathbb{Z} \ . \tag{9.16}$$

Clock cycle control token:

$$wait[finish : \mathbb{B}] : \mathbb{B} \ . \tag{9.17}$$

Synchronous block:

$$sync_n[finish_1 : \mathbb{B}, \ldots, finish_n : \mathbb{B}] : \mathbb{B} \ . \tag{9.18}$$

Signal:

$$signal[load : \mathbb{B}, set : \mathbb{Z}, default : \mathbb{Z}] : \mathbb{Z} \ . \tag{9.19}$$

9.4.1. Hardware compilation semantics

Hardware compilation semantics provides mapping of processes into sets of hardware equations, that is:

$$[\![-]\!] : P \to Eqn \ . \tag{9.20}$$

Here the extended semantics including signal constructs are summarized as follows:

$$[\![l :: Skip]\!] \widehat{=} finish_l = start_l \ , \tag{9.21}$$

$$[\![l :: Delay]\!] \widehat{=} finish_l = wait[start_l] \ , \tag{9.22}$$

$$\begin{aligned} [\![l :: x := e]\!] \ \widehat{=} \ & in.x = mux_1(start_l)([\![e]\!]); \\ & load.x = \bigvee\{start_l\}; \\ & x = register[load.x, in.x]; \\ & finish_l = wait[start_l] \ , \end{aligned} \tag{9.23}$$

$$\begin{aligned} [\![l :: x :\widehat{=} d \leftarrow e]\!] \ \widehat{=} \ & set.x = mux_1(start_l)([\![e]\!]); \\ & load.x = \bigvee\{start_l\}; \\ & x = signal[load.x, set.x, d]; \\ & finish_l = wait[start_l] \ , \end{aligned} \tag{9.24}$$

$$\begin{aligned}[\![l :: (m :: p_1 \lhd c \rhd n :: p_2)]\!] \;\widehat{=}\;& [\![m :: p_1]\!] \uplus [\![n :: p_2]\!] \uplus \\ & start_m = start_l \wedge [\![c]\!]; \\ & start_n = start_l \wedge \neg[\![c]\!]; \\ & finish_l = finish_m \vee finish_n \ ,\end{aligned} \tag{9.25}$$

$$\begin{aligned}[\![l :: (s \blacktriangleright [m :: p_i])]\!] \;\widehat{=}\;& [\![m :: p_1]\!] \uplus \ldots \uplus [\![m :: p_i]\!] \uplus \\ & start_{m_i} = start_l \wedge i = s; \\ & finish_l = \bigvee_i \{finish_{m_i}\} \ ,\end{aligned} \tag{9.26}$$

$$\begin{aligned}[\![l :: (b * m :: p)]\!] \;\widehat{=}\;& [\![m :: p]\!] \uplus \\ & start_m = [\![b]\!] \wedge (start_l \vee finish_m); \\ & finish_l = \neg[\![b]\!] \wedge (start_l \vee finish_m) \ ,\end{aligned} \tag{9.27}$$

$$\begin{aligned}[\![l :: (m :: p_1 ; n :: p_2)]\!] \;\widehat{=}\;& [\![m :: p_1]\!] \uplus [\![n :: p_2]\!] \uplus \\ & start_m = start_l; \\ & start_n = finish_m; \\ & finish_l = finish_n \ ,\end{aligned} \tag{9.28}$$

$$\begin{aligned}[\![l :: (m :: p_1 \parallel n :: p_2)]\!] \;\widehat{=}\;& [\![m :: p_1]\!] \uplus [\![n :: p_2]\!] \uplus \\ & start_m = start_l; \\ & start_n = start_l; \\ & finish_l = sync_2[finish_m, finish_n] \ .\end{aligned} \tag{9.29}$$

Finally, the running program is characterized by a sequence of states generated on successive clock-cycles by the repeated use of f on some starting state $s_0 : State$, therefore:

$$s_0, f(s_0), f^2(s_0), f^3(s_0), \ldots \quad . \tag{9.30}$$

9.5. Formal verification of logics in multiple clock domains

The past studies about formal semantics of Handel-C concentrated to only the case with a single clock domain with one master clock, and so far, the communications between outer world were regarded as asynchronous communications. The truth is, however, that some of the effective design methods enables synchronized communication between processes in different clock domains. For these cases, the combination of several clock domains can be regarded as a single complex combinational logic and becomes the target of combinational formal verification. The only problem here is that the language reference manual of the Handel-C [131] does not include detailed information about the duration of channel communication between two clock domains. It just notes that those communications take at least one clock cycle, possibly more than one. For the following discussion, we make two assumptions:

- if a pair of communication channels ("in" and "out") is ready in both clock domains, the communication takes place immediately.
- the beginning of clock cycles of all clock domains are aligned with no phase shift, which is technically well possible.

9.5.1. Introduction of observation clock domain

For multiple clock domains with each frequency F_i, an observation clock domain with a Least Common Multiple (LCM) frequency F_{LCM} can be introduced and the communication can be observed from this newly generated clock domain. For example, for two clock domains with different clock frequencies of $F_1 = 100\,\mathrm{MHz}$ and $F_2 = 150\,\mathrm{MHz}$, respectively, LCM frequency of $F_{LCM} = 300\,\mathrm{MHz}$ can be generated as illustrated in Figure 9.1. Provided that an output channel is ready in the second clock cycle in the clock domain 2 and waiting for the corresponding input channel in the clock domain 1. Because the channel communications take place immediately after resolution, if the input channel becomes ready in the first clock cycle of the clock domain 1, then the communication takes place in the third clock cycle of the LCM observation clock domain (A in Figure 9.1). On the contrary, if the input channel becomes ready in the second clock cycle of the clock domain 1, then the communication takes place in the forth clock cycle of LCM observation clock domain (B in Figure 9.1). On the other hand, synchronous communications between above two clock domains can be performed with a frequency of up to 50 MHz. This maximum value is the highest common factor of the clock domains. Clearly, these concepts can be also applied for more than two different clock domains.

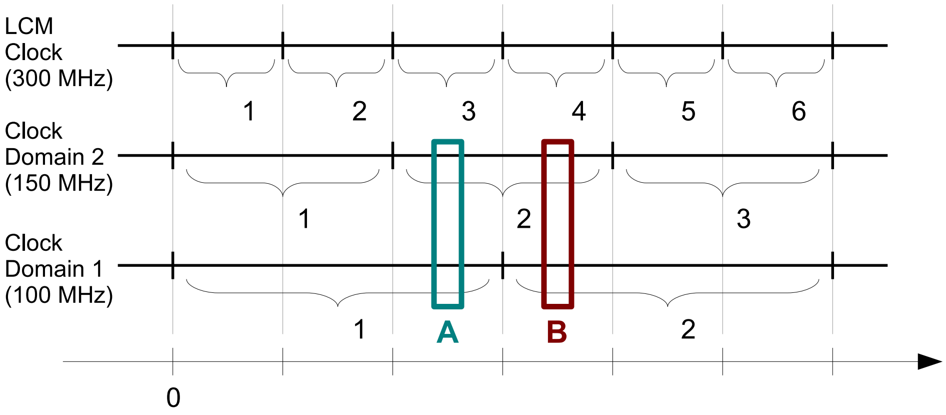


Figure 9.1.: Multiple clock domains

9.5.2. Derivation of synchronization function

Now define a delay function (df) for each i-th clock domain relative to the frequency of the observation clock domain, such that:

$$df_i = \frac{F_{LCM}}{F_i} . \tag{9.31}$$

Now a clock domain synchronization function *domSync* can be introduced modifying the above defined *wait* function in Equation (9.17), that is:

$$domSync[df : \mathbb{Z}, fini : \mathbb{B}] : \mathbb{B} \ , \tag{9.32}$$

where

$$domSync[df, fini] \mathrel{\widehat{=}} \underbrace{wait[wait[wait[fini]]]}_{df} \ . \tag{9.33}$$

Replacing the *wait* function with *domSync*, the corresponding atomic statements can be obtained as:

$$[\![l :: Delay]\!] \mathrel{\widehat{=}} finish_l = domSync[start_l] \ , \tag{9.34}$$

$$\begin{aligned} [\![l :: x := e]\!] \mathrel{\widehat{=}} in.x &= mux_1(start_l)([\![e]\!]); \\ load.x &= \bigvee\{start_l\}; \\ x &= register[load.x, in.x]; \\ finish_l &= domSync[start_l] \ , \end{aligned} \tag{9.35}$$

$$\begin{aligned} [\![l :: x :\doteq d \leftarrow e]\!] \mathrel{\widehat{=}} set.x &= mux_1(start_l)([\![e]\!]); \\ load.x &= \bigvee\{start_l\}; \\ x &= signal[load.x, set.x, d]; \\ finish_l &= domSync[start_l] \ . \end{aligned} \tag{9.36}$$

As the result, by introducing the notion of LCM frequency, delay function, and synchronization function, hardware compilation semantics in Section 9.4.1 can be extended to include multiple clock domains. This method has advantage against introducing explicit clock variables in the equations, in terms of simplicity and easy extendability.

9.6. Formal verification with UTP

In the above sections, formal semantics of operational and hardware compilation semantics of Handel-C are extended to include signal syntax into the scope of formal verification. Though the formal semantics written in CSP can be verified with tools such as FDR2, there is unfortunately no tool available verifying the formal semantics developed in this chapter at the time of writing. However, as suggested in [127], transformation of above Handel-C hardware compilation semantics into Unifying Theories of Programming (UTP) framework [132] is possible, by regarding the whole as a group of complex state-machines. It will be the future work to apply this technique to realize formal verification of Handel-C codes. It shall be also specified what is the full range of Handel-C laws that can be verified using the hardware semantics. It is the future work to strictly clarify the scope and also constrains of formal verification of Handel-C codes, so that design specification of control algorithms can be derived from, and developed control algorithms can be verified by formal methods.

9.7. Discussion on using signal constructs

Signal is the only construct with which one can (manually) generate direct electrical signal lines connecting combinational logics. Because the signal allocation command line can be nested inside the select conditions, the complete behavior of more than one “signals” shall be iteratively solved, and therefore, one shall take the resulting combinational hardware logic into account. Similar to the shared variables in Handel-C, which can not be written by more than one assignments, there is a constraint for using signals called “combinatorial cycles”.

```
static signal unsigned 1 sig1 = 0; // shared by both processes
static signal unsigned 1 sig2 = 1; // shared by both processes
//process A                          //process B
if (sig2 == 1) {                     if (sig1 == 1) {
    sig1 = 1;                            sig2 = 0;
} else {                             } else {
    delay;                               delay;
}                                    }
```

In the above example, the two signals $sig1$ and $sig2$ have default values of “0” and “1”, respectively. Provided that both processes start at the same time, the first process evaluates the if condition as **TRUE** setting the value of $sig1$ as “1”, and the second process evaluates the if condition as **FALSE** and performs *delay* at the very beginning of the clock cycle. However, because the values of $sig1$ eventually propagates up to 1 within the clock cycle, the if condition in the second process is evaluated as **TRUE** setting the value of $sig2$ to “0”, which leads again the change of the $sig1$. This behavior is called combinatorial cycle, which can be classified as live-lock. The compiler of DK Design Suite can detect this condition and report an error message. This behavior is often called as zero-delay combinatorial cycles.

The important point here, is that a “signal” can be further connected with other signals nested in select conditions, but they may not be connected with each other resulting in combinatorial cycles. As the compiler can detect this failure, it is assured that this kind of combinatorial cycles of signals do not exist in the designed control algorithms. This condition is necessary to ensure that extended formal semantics with signals described above can be surely solved. In other words, the resolution process of formal semantics shall be iteratively performed until all sequential combination of signal propagation lines are evaluated.

Synchronization of multiple clock domains

Channel communications can be executed only when the both input and output channels became ready. Due to this nature of channel construct, it can not be used for synchronization of more than two processes. To see the problem, consider the processes explained in the below example. Provided that all processes start at the same time, the process A intends channel communication trying to synchronize the other two processes B and C. However, because only process B is ready to receive the data, only the $ch1$ communication takes place in the first clock cycle. In the next clock cycle, process B performs statement1, while process C performs channel communication through $ch2$. Consequently, two statement1 and statement2 can not be synchronized.

```
chan unsigned 1 ch1; // shared by all processes
chan unsigned 1 ch2; // shared by all processes
unsigned 1 a, b;
//process A                    //process B               //process C
do {                           do {                      do {
    par {                          ch1?a                    delay;
        ch1!1;//1st clock          statement1;               ch2?b;
        ch2!1;//2nd clock          delay;                    statement2;
    }                          } while (1);              } while (1);
    delay;
} while (1);
```

However, this problem can be solved by using the signal as described in the example codes below. In the example, the process A sets the value of $sig1$ as “1” in the second clock cycle, by performing the statement1 at the same time. This assignment of the signal value is not dependent on the other processes any more. Process B and C wait until the value of the $sig1$ becomes “1” and if the value changes to “1”, they performs statement2 and statement3, respectively in the very same clock cycle. In this way, all three statements can be synchronized.

```
static signal unsigned 1 sig1 = 0; // shared by all processes
//process A                    //process B               //process C
do {                           do {                      do {
    delay;                         while(sig1!=1){           while(sig1!=1){
    par {                              delay;                    delay;
        sig1 = 1;                  }                         }
        statement1;                statement2;               statement3;
    }                          } while (1);              } while (1);
} while (1);
```

As described in this example, “signals” provide attractive features for synchronizations of multiple parallel running processes, as well as for simultaneous information exchanges between them, which can realize an effective use of hardware logic inside FPGAs and convenient logic design implementations.

9.8. Conclusion

In this chapter, the existing formal semantics of the Handel-C hardware description language is extended in terms of “signal” syntax in order to cover all aspects of applied control algorithms implemented into the on-board computer of the *Flying Laptop* in Part II. The formal semantics are also extended to deal with multiple clock domains inside an FPGA. This method introduces an observation clock domain to solve the behavior of hardware logic. The combinational use of the developed formal semantics and the UTP will be able to realize formal verification of Handel-C codes in order to assure the fault-free functionality of designed control algorithms which shall be implemented into FPGA-based OBCs for future space systems.

Part IV.

Conclusions and appendices

10. Conclusions and outlook

The application of “reconfigurable computing” based on FPGA technologies provides ideal solutions for fulfilling high demands on computational capabilities of recent and future space systems. The extremely high throughput of FPGAs due to their internal parallel processing promises an innovative leap of the capability of computing systems aboard spacecraft. Other features of reconfigurable FPGAs – dense interface implementation capability, lower power consumption, low cost, high flexibility, and small hardware size – are very attractive, especially for the application fields of small satellites. However, due to their radiation susceptibilities, reconfigurable FPGAs have not yet been used in practical space applications. This thesis conceptualized an application method of ground-based reconfigurable FPGAs for space systems, with the approach of a combinational use of SRAM-FPGAs and Flash-FPGAs for radiation mitigation. Within the scope of this thesis an FPGA-based on-board computer was developed for the demonstration small satellite *Flying Laptop*. This is a purely FPGA-based and the only on-board computer of the satellite, in which all satellite control functions shall be implemented. Accordingly, this thesis realizes following three innovations:

- Application of ground-based reconfigurable FPGAs for space systems.
- Demonstration of an FPGA-based on-board computer on the small satellite *Flying Laptop*.
- Demonstration of the implementation of the whole satellite control functions into the control algorithms inside FPGAs.

The contributions of this thesis are:

- Extensive investigation on radiation effects on existing reconfigurable FPGA devices and failure rate estimation in LEO.
- Conceptualization of reconfigurable FPGA-based on-board computers for space systems applying a combinational use of SRAM- and Flash-FPGAs for the radiation mitigation.
- Reliability modeling and establishment of an operational strategy of the proposed on-board computer concept reflecting its repairable features and degree of redundancy.
- System design of the demonstration small satellite *Flying Laptop* in order to realize the implementation of the on-board computer.
- Development of the hardware breadboard model of the on-board computer as well as partial engineering models of its components.
- Development of the hardware logic (control algorithm) inside the FPGAs which shall replace the software functions of traditional satellites and shall conduct the whole control functions of the *Flying Laptop* satellite.
- Development of a simulation interface inside FPGAs for hardware-in-the-loop simulation and verification purposes.
- Simulation and verification of the functionalities of the implemented control algorithms by means of the established simulation and verification environment.

- Investigation on formal verification methods of hardware logic inside FPGAs and extension of formal semantics in terms of “signal” syntax and multiple clock domains.

First of all, an extensive investigation about radiation effects on different types of FPGAs has been conducted. Experimentally obtained characteristics against radiation effects were surveyed for SRAM- and Flash-based FPGA devices. According to the available experimental data, mainly from scientific publications, manufacturers, and the NASA’s Office of Logic Design, the state-of-the-art SRAM-FPGA devices Virtex-II Pro and Virtex-4QV, as well as Flash-FPGA devices ProASIC3 and RTProASIC were selected as candidates. Failure frequencies estimated by means of the industry-standard radiation effects model CREME96 indicate the possibility of realizing SRAM-FPGA-based OBCs with continuous repairs against SEEs in case of modest radiation conditions. The strong tolerance of SRAM-FPGAs against TID effects allows the assumption that the radiation effects due to SEEs can be repaired by reconfigurations. It also indicated that under heavy radiation conditions, such as those caused by coronal mass ejections, existing SRAM-FPGAs can not survive even few seconds without an error (it can be repaired/reconfigured). This fact derives the requirement that the FPGA-based OBC shall possess the capability of being completely shut down in case of heavy radiation conditions. On the contrary, Flash-FPGAs revealed their tolerance against SEEs with adequate mitigation methods.

Based on the achievements above, an application method of reconfigurable FPGAs for the space radiation environment with a combinational use of SRAM-FPGAs and Flash-FPGAs was conceptualized. This method applies multiple SRAM-FPGAs for multi-chip redundancy (2-out-of-m) in order to mitigate SEEs monitoring their behavior using Flash-FPGAs as the voting function. The Flash-FPGAs do not need to be reconfigured in orbit. The reliability modeling of these conceptualized repairable redundant systems has been conducted. Investigations on radiation effects and the reliability analysis revealed that 2-out-of-4 system based on Virtex-II Pro XC2VP50 (which was actually applied for the OBC of the *Flying Laptop*) provides a sufficient high reliability against the radiation effects for the mission life time of 2 years in LEO, whereas 2-out-of-3 system is estimated to fail around every 2 months. The required repair rate was identified as a realistic value of 5 s. It was also illustrated that periodical reconfigurations shall ensure a sufficiently high reliability against SEEs. Reliability against TID can be evaluated separately and is not repairable. The selected RTProASIC RT3PE3000L can provide a sufficient reliability for the mission life time of around 10 years in LEO, assuming an annual dose of about 1.5 krad with a 3 mm of aluminum shielding. The reliability against SEEs depends on the duration after the most recent start-up, while the reliability against TID depends only on the duration after the launch. Consequently, applying 2-out-of-4 redundant SRAM-FPGAs with Flash-FPGAs as the voter, a system reliability of ≈ 1 can be achieved in 2 years after 30 days of operational time of the node system. In case of 2-out-of-3 system, this reliability decreases to about 0.49. The quad-redundant system can also absorb a permanent failure of one SRAM-FPGA chip resulting in a 2-out-of-3 system.

The above conceptualized application method has been applied to the design of the OBC of the small satellite *Flying Laptop*. The *Flying Laptop* is a scientific Earth observation and technology demonstration satellite and is equipped with various scientific instruments, such as camera instruments as well as communication instruments. All on-board electrical components shall be controlled by the OBC. First of all, the system design of the whole satellite was conducted within the scope of the thesis. The electrical architectures of the satellite system were established and the electrical interfaces were specified. By means of a real-size mock-up model of the satellite, harness cabling was designed in order to define realistic hardware interfaces of the OBC. The functional parameters of the TT&C and PSS subsystems were defined and the satellite's operational concept, including commanding, data handling, and contingency operation has been established. The state-machine of the system and subsystems were designed. A power balancing design was conducted reflecting the target orbit, attitude control modes, and solar panel configurations, which ensures the secure operation of the satellite.

Derived from the system design of the satellite, requirements of the OBC have been defined. The design of 2-out-of-4 system was selected to secure a higher reliability. The hardware breadboard model of the OBC was developed and assembled and engineering models of the central processing node were developed. The control algorithms were developed using the breadboard model. The layered architecture of the control algorithm realizes a higher portability. A compositional multi-agent programming method and layered FDIR architecture were established for the implementation of subsystem and component control algorithms. The main central control logics were implemented based on the developed concept of parallel asynchronous reactive system (PARS). State-machines of subsystems were developed in the Mathworks MATLAB/Simulink environment. Furthermore, an integrated control algorithm development environment for both SRAM- and Flash-FPGAs were established.

For the verification purposes of the control algorithms inside FPGAs, a hardware-in-the-loop simulation environment was developed. A special simulation interface which allows communications between parallel processes inside an FPGA and the simulator was developed and implemented into the FPGA. This interface realizes a hardware-in-the-loop simulation and verification environment based on the model-based development and verification environment established at the IRS with the support of EADS Astrium. By means of this environment, extensive simulations have been conducted. The simulation results illustrate validities of the implementation methods of control algorithms, their functionalities, and the system design of the satellite, in terms of power balancing design, operational design, commanding architecture design, state-machine design, and FDIR design.

Finally, an investigation on formal verification methods of the hardware description language Handel-C was conducted. The existing formal semantics of the Handel-C were extended in terms of "signal" syntax in order to cover all aspects of applied control algorithms. The formal semantics were also extended to deal with multiple clock domains inside an FPGA, introducing an observation clock domain. The combinational use of the developed formal semantics and the UTP will be able to realize formal verification of Handel-C codes in order to assure fault-free functionalities of designed control algorithms for FPGA-based OBCs of future space systems.

As a future work, radiation tests shall be performed as soon as the engineering model of the OBC becomes available. After the qualification of the engineering model, the flight model shall be developed. The control algorithms shall be continuously improved/developed based on the breadboard model and the hardware-in-the-loop simulation and verification environment. Starting from the EM, satellite peripheral components shall be integrated with the OBC and their combinational functionalities shall be verified. After the launch of the satellite, the OBC shall demonstrate the validity of its design, as well as the introduced application methods of reconfigurable FPGAs for space systems. Furthermore, if the mission life time allows, application capabilities of the OBC for intelligent space systems can be demonstrated by testing higher level autonomy functions. The OBC promises successful achievements of all scientific experiments and technology demonstrations of the *Flying Laptop* satellite.

Appendices

A. Failure frequency of SRAM-FPGAs

This appendix provides failure frequencies of the selected Xilinx SRAM-FPGAs (Virtex-II Pro XC2VP50 and Virtex-4QV XQR4VFX60), calculated by the radiation model CREME96 [56].

Table A.1.: Failure frequency of Virtex-II Pro XC2VP50 [device/s]

Case	CFG		BRAM		POR	
	PUP	HUP	PUP	HUP	PUP	HUP
Solar Max	1.15E-14	7.46E-5	2.73E-15	2.10E-5	5.95E-21	1.87E-10
Solar Min	4.83E-15	2.76E-5	1.14E-15	7.84E-6	2.48E-21	7.64E-11
Worst Week	1.04E-10	1.39E+1	2.32E-11	4.10	4.09E-17	3.58E-5
Worst Day	6.29E-10	6.76E+1	1.39E-10	2.00E+1	2.37E-16	1.76E-4
Peak-5min	2.51E-9	2.77E+2	5.54E-10	8.19E+1	9.34E-16	7.22E-4

Case	SMAP		JCFG		–	
	PUP	HUP	PUP	HUP	–	–
Solar Max	9.11E-21	2.00E-10	4.56E-21	2.49E-11		
Solar Min	3.82E-21	7.97E-11	1.91E-21	9.45E-12		
Worst Week	8.23E-17	3.26E-5	4.12E-17	4.76E-6		
Worst Day	4.97E-16	1.58E-4	2.49E-16	2.31E-5		
Peak-5min	1.99E-15	6.50E-4	9.94E-16	9.49E-5		

Table A.2.: Failure frequency of Virtex 4QV XQR4VFX60 [device/s]

Case	CFG		BRAM		F/F(1)		F/F(0)	
	PUP	HUP	PUP	HUP	PUP	HUP	PUP	HUP
Solar Max	1.05E-6	4.57E-5	3.05E-7	3.74E-5	1.20E-9	6.63E-7	3.61E-9	2.81E-7
Solar Min	4.50E-7	2.16E-5	1.27E-7	2.08E-5	5.01E-10	3.29E-7	1.50E-9	1.16E-7
Worst Week	4.02E-3	7.78	1.10E-3	5.55	4.35E-6	1.10E-1	1.30E-5	5.44E-2
Worst Day	2.35E-2	4.04E+1	5.90E-3	3.14E+1	2.33E-5	5.90E-1	6.97E-5	2.72E-1
Peak-5min	9.32E-2	1.66E+2	2.28E-2	1.30E+2	8.98E-5	2.44	2.69E-4	1.12

Case	POR		SMAP		GSIC		–	
	PUP	HUP	PUP	HUP	PUP	HUP	–	–
Solar Max	3.47E-12	1.48E-10	2.71E-12	1.17E-11	3.42E-12	5.21E-12		
Solar Min	1.46E-12	7.69E-11	1.14E-12	6.06E-12	1.45E-12	2.90E-12		
Worst Week	1.41E-8	2.90E-5	2.51E-8	2.37E-6	1.66E-8	9.58E-7		
Worst Day	8.06E-8	1.60E-4	1.51E-7	1.30E-5	9.83E-8	5.55E-6		
Peak-5min	3.18E-7	6.61E-4	6.03E-7	5.38E-5	3.91E-7	2.29E-5		

PUP:	Proton induced events	HUP:	Heavy ion induced events
CFG:	A SEU in the configuration memory	BRAM:	A SEU in the Block RAM
POR:	SEFI in the Power-on-Reset circuitry	SMAP:	A SEFI in the SelectMAP circuitry
JCFG:	A SEFI in the JTAG Configuration Access Port	GSIG:	Global Signal
F/F(1):	User flip-flops in state of “1”	F/F(0):	User flip-flops in state of “0”

B. Basic probability calculus

This appendix provides a fundamental background of basic probability calculus. For detailed information on basic probability calculus one can refer, e.g., [59].

B.1. Definition of distribution function and density function

In general, the *probability distribution function* of a random variable Z in an exponential case can be described as

$$F_Z(z) \equiv P(Z \leq z) \ . \tag{B.1}$$

In case $F_Z(z)$ is differentiable, its derivative

$$f_Z(z) \equiv \frac{d}{dz} F_Z(z) \tag{B.2}$$

is called the *probability density function* of Z. In case Z is a continuous random variable,

$$E(Z) = \int_{-\infty}^{\infty} z f_z(z) dz \tag{B.3}$$

is the expected value of Z. In case of non-negative random variables such as $Z \equiv L \leq 0$ for life time, the value of $E(L)$ can be described as Riemann integral approximation:

$$E(L) = \int_0^{\infty} t f_L(t) dt = \sum_{i=1}^{\infty} i\Delta t [F_L(i\Delta t + \Delta t) - F_L(i\Delta t)] + O(\Delta t) \ . \tag{B.4}$$

This is on the other side equal to the shaded area in Figure B.1 [59]. Consequently the following equation can be derived:

$$E(L) = \int_0^{\infty} t f_L(t) dt = \int_0^{\infty} [1 - F_L(t)] dt = \int_0^{\infty} \bar{F}_L(t) dt = \int_0^{\infty} R(t) dt \ , \tag{B.5}$$

where $\bar{F}_L(t)$ and $R(t)$ are called the survivor function and reliability of the random variable L at t. The normation condition of the probability distribution function has the form of

$$F_Z(\infty) = 1 \ \Leftrightarrow \ \int_{-\infty}^{\infty} f_z(z) dz = 1 \ . \tag{B.6}$$

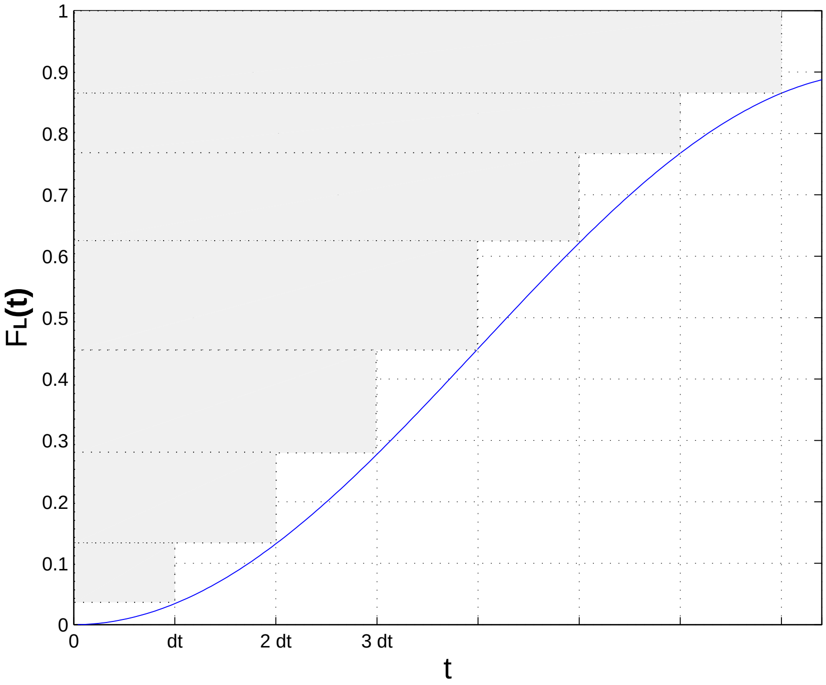


Figure B.1.: Non-negative random variable

B.2. Distribution of the sum of two random variables

A single distribution function which is a combination of two random variables can be defined by

$$F_{Z_1,Z_2} = P\{(Z_1 \leq z_1) \cap (Z_2 \leq z_2)\} \ . \tag{B.7}$$

For statistically independent random variables, this yields

$$F_{Z_1,Z_2} = F_{Z_1}(z_1)F_{Z_2}(z_2) \Leftrightarrow f_{Z_1,Z_2} = f_{Z_1}(z_1)f_{Z_2}(z_2) \ . \tag{B.8}$$

Distribution of the sum of two statistically independent variables $L_1, L_2 \geq 0$ can be now written as:

$$F_{L_1+L_2}(t) = \int\limits_0^t f_{L_1}(\tau)F_{L_2}(t-\tau)d\tau \ . \tag{B.9}$$

Differentiating this and applying the sign "⊗" indicating convolutional functions we get

$$f_{L_1+L_2}(t) = \int\limits_0^t f_{L_1}(\tau)f_{L_2}(t-\tau)d\tau \equiv f_{L_1} \otimes f_{L_2} \ . \tag{B.10}$$

For detailed derivation of above equations, see [59].

B.3. Basic Laplace transformation

Generally, Laplace transformation of the function $f(t)$ can be described as:

$$f^*(s) \equiv \mathcal{L}\{f(t)\} \equiv \int\limits_0^\infty f(t)e^{-st}dt \ . \tag{B.11}$$

where, $\mathcal{L}$ indicates Laplace transformation. Letting $\mathcal{L}^{-1}$ indicate inverse Laplace transformation, the relation can be also described as:

$$f(t) \equiv \mathcal{L}^{-1}\{f^*(s)\} \ . \tag{B.12}$$

From the definition,

$$\mathcal{L}\{1\} = \int\limits_0^\infty e^{-st}dt = \frac{1}{s} \ . \tag{B.13}$$

Prominent properties of the Laplace transformation can be described as follows.

Differentiation rule:

$$\mathcal{L}\left\{\frac{d}{dt}f(t)\right\} = sf^* - g(+0) \ . \tag{B.14}$$

Integration rule:

$$\mathcal{L}\left\{\int\limits_0^t f(\tau)d\tau\right\} = \frac{f^*(s)}{s} \ . \tag{B.15}$$

Multiplication rule:

$$\mathcal{L}\left\{\int\limits_0^\infty f_1(\tau)f_2(t-\tau)d\tau\right\} = f_1^*(s)f_2^*(s) \ . \tag{B.16}$$

As a practical application, one can transform Equation (B.10) for $\tau \rightarrow \infty$ obtaining

$$f^*{}_{L_1+L_2}(s) = f^*{}_{L_1}(s)f^*{}_{L_2}(s) \ . \tag{B.17}$$

B.4. Markov model of a repairable unit

For the Markov model of a single repairable unit, illustrated in Figure B.2, the $\dot{P}_1(t)$ can be described as

$$\dot{P}_1(t) \equiv \frac{dP_1(t)}{dt} = -\lambda P_1(t) + \mu P_2(t) \ . \tag{B.18}$$

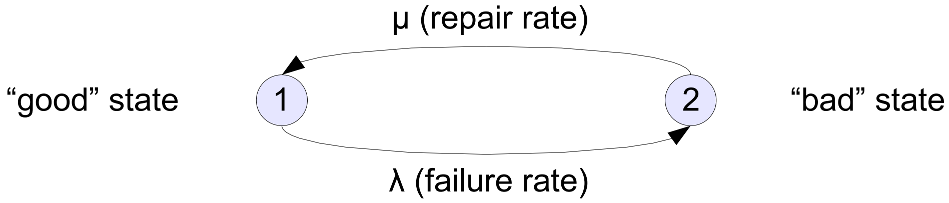


Figure B.2.: Markov graph of a single repairable unit

By the normation equation of probabilities

$$P_1(t) + P_2(t) = 1 \ , \tag{B.19}$$

Equation (B.18) can be written as

$$\dot{P}_1(t) + (\lambda + \mu)P_1(t) - \mu = 0 \ . \tag{B.20}$$

The general solution can described as

$$P_1(t) = \alpha + \beta \cdot e^{-\gamma t} \ , \tag{B.21}$$

and inserting Equation (B.21) to Equation (B.20) results

$$-\beta\gamma e^{-\gamma t} + (\lambda + \mu)\alpha + (\lambda + \mu)\beta e^{-\gamma t} - \mu = 0 \ . \tag{B.22}$$

This yields two equations, i.e., for $t = 0$

$$-\beta\gamma + (\lambda + \mu)(\alpha + \beta) - \mu = 0 \ , \tag{B.23}$$

and for $t \to \infty$ (for $Re(\gamma) \geq 0$),

$$(\lambda + \mu)\alpha - \mu = 0 \Rightarrow \alpha = \frac{\mu}{\lambda + \mu} \ . \tag{B.24}$$

with the initial condition $P_1(0) = 1$, Equation (B.21) yields

$$\alpha + \beta = 1 \ . \tag{B.25}$$

From Equation (B.24)

$$\beta = \frac{\lambda}{\lambda + \mu} \ . \tag{B.26}$$

With α and β known, γ can be determined as

$$\gamma = \frac{\lambda}{\beta} = \lambda + \mu \ , \tag{B.27}$$

and this acknowledges the above assumption $Re(\gamma) \geq 0$. Consequently, $P_1(t)$ can be determined as

$$P_1(t) = \frac{\mu}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} e^{-(\lambda+\mu)t} \ . \tag{B.28}$$

B.5. Basic Boolean functions

Provided that a component C has a binary property, it can be modeled by a Boolean indicator variable X

$$X = \begin{cases} 0, & \text{if the component has the certain property} \\ 1, & \text{else} \end{cases} \ . \tag{B.29}$$

Boolean function φ depending on $X_1, \ldots, X_n$ and resulting in X_φ can be described as

$$X_\varphi = \varphi(X_1, \cdots, X_n) \ . \tag{B.30}$$

Generally, Boolean algebra can be defined by means of three operations: conjunction operation (AND) with the operator $\wedge$, disjunction operation (OR) with the operator $\vee$, and negation operation (NOT) with the operator $\bar{}$. For complex calculations of Boolean functions with n not too small, the Boolean operators in Boolean functions can be further replaced by mathematical operations such that:

$$\bigwedge_i A_i = \prod_i A_i \ , \tag{B.31}$$

and

$$\bigvee_i A_i = 1 - \prod_i (1 - A_i) = 1 - \prod_i \bar{A}_i \ . \tag{B.32}$$

For Boolean indicator quantities A and B clearly

$$A \wedge B = AB \tag{B.33}$$

meaning concatenation (multiplication), and

$$A \vee B = A + B - AB = 1 - (1 - A)(1 - B) = A + \bar{A}B \ . \tag{B.34}$$

Negation can be replaced by

$$\bar{A} = 1 - A \ , \tag{B.35}$$

and idempotence law also holds as

$$A^2 = A \ . \tag{B.36}$$

C. Flying Laptop satellite subsystem design

Based on the system design activity functions and components are allocated to subsystems. In following chapters, relevant design aspects of the subsystems TT&C (Telemetry Tracking & Command) and PSS (Power Supply System) are described.

C.1. Telemetry, Tracking & Command subsystem design

As the components of TT&C subsystem, VHF-Band up-link, UHF-Band down/up-link, S-Band down/up-link are implemented. The S-Band is the main communication channel of the *Flying Laptop* and is equipped with high gain and low gain down-link antennas. The down-link baud rates are 2 Mbps and 150 kbps, respectively. For the redundancy, the receiver of the S-Band and UHF-Band are fully hot redundant and they are always powered on. The VHF channel is only for the up-link. These VHF and UHF channels will be partly opened for amateur radio community.

C.1.1. Link equation

The most basic link equation, which relates concerned parameters to design a digital data link with desired performances is

$$\frac{E_b}{N_0} = \frac{P_t L_l G_t L_s L_a G_r}{k T_s R_d} \ , \tag{C.1}$$

where E_b/N_0 is the ratio of received energy-per-bit to noise-density, P_t is the transmitter power, L_l is the transmitter-to-antenna line loss, G_t is the transmit antenna gain, L_s is the space loss, L_a is transmission path loss, G_r is the receive antenna gain, k is Boltzmann's constant (1.380×10^{-23} J/K), T_s is the system noise temperature, and R_d is the data rate [133]. L_a is a function of factors such as rainfall density and atmospheric attenuations.

The received power C at the antenna can be described as the power flux density W_f times the effective receiver antenna aperture area A_r:

$$C = W_f A_r \ , \tag{C.2}$$

where W_f and A_r can be described as

$$W_f = \frac{P_t L_l G_t L_a}{4\pi S^2} \ , \tag{C.3}$$

$$A_r = \frac{\pi D_r^2}{4} \cdot \eta \ . \tag{C.4}$$

Here, S is the path length between the two antennas in meter, and η is the antenna efficiency of receive antenna. The maximum distance in case of *Flying Laptop* is calculated with the elevation angle of 7 ° for the orbit with an altitude of 900 km. The part of parameters which can be measured by means of sensors, that is P_t, L_l, and G_t are also called as effective isotropic radiated power $EIRP$. Therefore, W_f can be also described as

$$W_f = \frac{(EIRP)L_a}{4\pi S^2} \ . \tag{C.5}$$

The receiver antenna gain can be defined as the ratio of its effective aperture area A_r to the effective area of a hypothetical isotropic antenna $\lambda_w^2/4\pi$, where λ_w is the wavelength of the transmitted signal, such that

$$G_r = \left(\frac{\pi D_r^2 \eta}{4}\right)\left(\frac{4\pi}{\lambda_w^2}\right) = \frac{\pi^2 D_r^2 \eta}{\lambda_w^2} \ . \tag{C.6}$$

From above equations, the Equation (C.2) can be now described as

$$C = P_t L_l G_t L_a G_r \left(\frac{\lambda_w}{4\pi S}\right)^2 \equiv P_t L_l G_t L_s L_a G_r \equiv (EIRP) L_s L_a G_r \ , \tag{C.7}$$

where

$$L_s = (\lambda_w/4\pi S)^2 = (c/4\pi S f)^2 \tag{C.8}$$

is defined as the space loss. Here, f is the carrier signal frequency in Hz and $c = 3 \times 10^8$ m/s is the speed of light.

The received energy per bit E_b is equal to the received power times the bit duration, therefore

$$E_b = \frac{C}{R_d} \ . \tag{C.9}$$

The total received noise power N is the noise spectral density N_0 times the received noise bandwidth B

$$N = N_0 B \ , \tag{C.10}$$

where N_0 is related to the system noise temperature T_s by:

$$N_0 = kT_s \ . \tag{C.11}$$

N_0 is in W/Hz, N is in W, T_s is in K, and B is in Hz.

Because the link equations are products of successive terms, they are conventionally expressed in terms of decibels (dB). Equation (C.1) can be described as

$$E_b/N_0 = P_t + L_l + G_t + L_s + L_a + G_r + 228.6 - 10 \log T_s - 10 \log R_d \ , \tag{C.12}$$

and the carrier-to-noise-ratio C/N results in

$$\begin{aligned} C/N &= E_b R_d / N_0 B \\ &= E_b/N_0 \times R_d/B \\ &\equiv P_t + L_l + G_t + L_s + L_a + G_r + 228.6 - 10\log T_s - 10\log B \ . \end{aligned} \tag{C.13}$$

Pointing loss: For the link design of the *Flying Laptop*, only the pointing losses of directional antennas have been taken into account. For circular directional antenna beam, the half-power beam width θ is the angle across which the gain is within 3 dB of the peak gain. As described in [133], the half-power beam width of a receiver antenna is empirically,

$$\theta = \frac{21}{f_{GHz} D_r} \ , \tag{C.14}$$

where f_{GHz} is the carrier frequency in GHz and D_r is the receiver antenna diameter in meter, θ is in degrees. A receiver antenna might not be located at the center of the transmitter antenna beam, or vice versa. The reduction from peak gain, due to the pointing error e can be estimated by the following equation.

$$L_\theta = -12(e/\theta)^2 \ . \tag{C.15}$$

System noise: A system noise temperature represents communication loss against noises from a number of individual noise sources. Dividing these sources into two groups, antenna noise temperature T_{ant} originating ahead of the antenna aperture and receiver noise temperature T_r originating from between the antenna terminal and the receiver output, the system noise temperature can be described as:

$$T_s = T_{ant} + \left(\frac{T_0(1-L_r)}{L_r}\right)\left(\frac{T_r}{L_r}\right) \ , \tag{C.16}$$

where, the L_r is the line loss between the antenna and receiver, T_0 is a reference temperature, usually set as 290 K. The relation between T_r and T_0 is defined using noise figure F_{noise}:

$$F_{noise} = 1 + \frac{T_r}{T_0} \ . \tag{C.17}$$

According to the empirical numbers in [133], antenna noise (K), line loss noise (K), and receiver noise figure (dB) can be summarized in Table C.1.

Table C.1.: System noise sources

Noise Temperature	Frequency (GHz)				
	Downlink			Uplink	
	0.2	2–12	20	0.2–20	40
Antenna Noise (K)	150	25	100	290	290
Line Loss Noise (K)	35	35	35	35	35
Receiver Noise Figure (dB)	0.5	1.0	3.0	3.0	4.0

Transmission path loss: The transmission path loss caused by the Earth's atmosphere, L_a, is a function of frequency. According to [133], provided that the zenith attenuation $L_{a,zenith}$ is available, the loss at elevation angle β of above 5° can be obtained by

$$L_a = \frac{L_{a,zenith}}{sin\beta} \ . \qquad \text{(C.18)}$$

In [133], one can find values of zenith attenuation in different carrier frequencies. The L_a of the communication channels of the *Flying Laptop* have been obtained as summarized in Table C.2 and Table C.3. The attenuation by rain $L_{a,rain}$ is also counted in the transmission path loss. This effect is negligible for lower frequencies than about 10 GHz. The estimated value of the rain attenuation based on the Crane model [134] are summarized in [133]. These values for the *Flying Laptop* are also listed in the tables below. Other atmospheric attenuation effects are described in detail in, e.g., [135], [136], and [137]. However, these effects are small relative to the others and have not been taken into account for the link design of the *Flying Laptop*.

C.1.2. Link budget of Flying Laptop

Finally, the designed link budgets of communication systems of the *Flying Laptop* are summarized in following tables. Link budgets of nondirectional communication channels are summarized in Table C.2 and those of directional communications including the experimental transmitters in Ka and Ku-Bands are summarized in Table C.3.

C.1.3. State-machine of the TT&C subsystem of Flying Laptop

The designed state-machine of the TT&C subsystem is illustrated in Figure C.1.

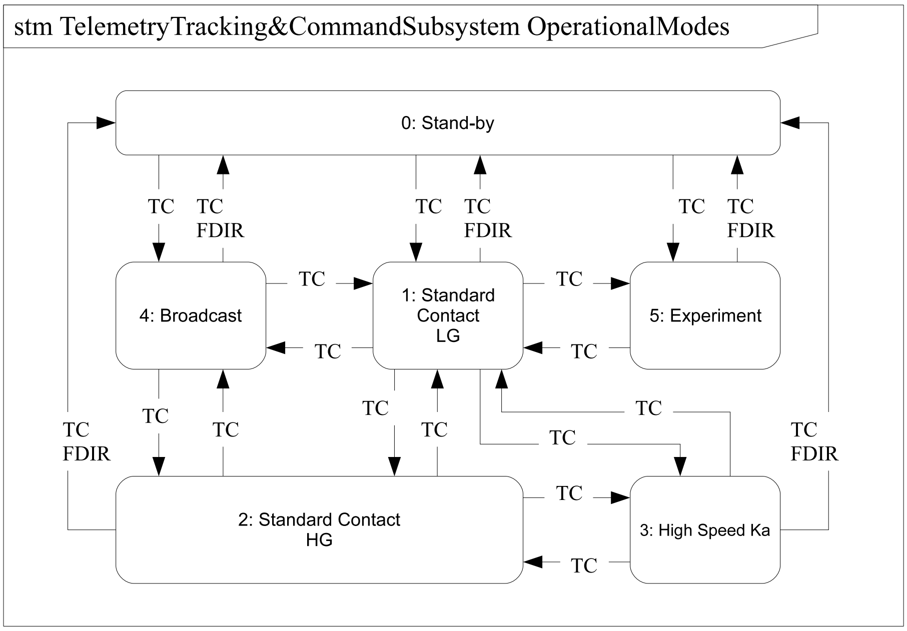


Figure C.1.: State-machine of TT&C subsystem of *Flying Laptop*

Table C.2.: Link budget of omnidirectional communications

Parameter	Unit	VHF-Up	UHF-Up	UHF-Down	S-Up	S-Down LG
P_t	W	1.00E+00	5.00E+00	5.00E+00	5.00E+00	5.00E+00
f	Hz	1.45E+08	4.50E+08	4.50E+08	2.03E+09	2.20E+09
B	Hz	3.00E+04	1.25E+04	5.00E+04	3.00E+04	2.50E+05
R_d	bps	1.92E+04	4.80E+03	3.80E+04	1.92E+04	1.50E+05
G_t	dB	7.00E+00	1.00E+01	0.00E+00	3.23E+01	0.00E+00
G_r	dB	0.00E+00	0.00E+00	1.00E+01	0.00E+00	3.30E+01
L_l	dB	1.00E+00	3.00E+00	1.00E+00	3.00E+00	2.00E+00
L_a	dB	5.59E-01	5.59E-01	5.59E-01	1.00E+00	1.00E+00
$L_{\theta,G.S.}$	dB	–	–	–	7.01E-01	7.01E-01
L_s	dB	1.45E+02	1.55E+02	1.55E+02	1.68E+02	1.68E+02
T_s	K	6.14E+02	6.14E+02	1.00E+03	6.14E+02	1.50E+02
E_b/N_0	dB	1.67E+01	2.08E+01	1.11E+04	2.29E+01	2.11E+01
C/N	dB	1.47E+01	1.67E+01	1.05E+01	2.09E+01	1.88E+01
M_{E_b/N_0}	dB	6.65E+00	1.08E+01	1.11E+04	1.29E+01	1.11E+01
$M_{E_b/N_0,Rain}$	dB	6.65E+00	1.08E+01	1.11E+04	1.29E+01	1.11E+01
$M_{C/N,Rain}$	dB	1.47E+01	1.67E+01	1.05E+01	2.09E+01	1.88E+01

Table C.3.: Link budget of directional communications

Parameter	Unit	S Down-HG	Ka-Up	Ka-Down HG	Ka-Down LG	Ku
P_t	W	5.00E+00	2.00E+00	5.60E+01	1.00E+00	1.00E-02
f	Hz	2.20E+09	3.00E+10	2.00E+10	2.00E+10	1.20E+10
B	Hz	2.50E+06	1.50E+07	8.00E+08	1.00E+05	1.40E+06
R_d	bps	2.20E+06	1.00E+07	5.00E+08	6.00E+04	1.00E+03
G_t	dB	1.00E+01	5.73E+01	3.73E+01	2.00E+01	2.00E+01
G_r	dB	3.30E+01	4.08E+01	5.38E+01	5.38E+01	4.93E+01
L_l	dB	2.00E+00	1.00E+00	1.00E+00	1.00E+00	1.00E+00
L_a	dB	1.00E+00	1.94E+00	2.89E+00	2.89E+00	1.00E+00
$L_{\theta,G.S.}$	dB	7.01E-01	2.20E+00	9.80E-01	9.80E-01	3.53E-01
$L_{\theta,FLP}$	dB	4.80E-01	4.96E-02	5.51E-01	3.00E-02	4.80E-01
L_s	dB	1.68E+02	1.91E+02	1.87E+02	1.87E+02	1.83E+02
T_s	K	1.50E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02
E_b/N_0	dB	1.89E+01	3.67E+01	3.04E+01	3.54E+01	3.53E+01
C/N	dB	1.80E+01	3.50E+01	2.84E+01	3.32E+01	3.81E+00
M_{E_b/N_0}	dB	8.91E+00	2.67E+01	2.04E+01	2.54E+01	2.53E+01
$L_{a,rain}$	dB	0.00E+00	2.20E+01	2.00E+01	2.00E+01	2.00E+01
$M_{E_b/N_0,Rain}$	dB	8.91E+00	4.72E+00	4.40E-01	5.42E+00	5.28E+00
$M_{C/N,Rain}$	dB	1.80E+01	1.30E+01	8.40E+00	1.32E+01	-1.62E+01

C.2. Power supply system deisgn

The PSS of the *Flying Laptop* satellite consists of the following major components: Power Control and Distribution Unit (PCDU), three solar panels, battery management system and battery with six battery cells. The solar panel configuration of the *Flying Laptop* is wing-shaped. A body mounted solar array configuration was also considered because it simplifies the design of the satellite and improves system reliability. However, for the body mounted configuration, we need to rotate the satellite around its Y axis and camera systems are forced to see the sun. Because of this reason, wing-shaped solar panels are selected. For the deployment of the solar panels electrical solar panel deployment mechanisms are developed. In Figure C.2 the configuration of the solar panels and solar cells on them are illustrated.

Solar panels

Each of the 7 strings on the side panels consists of serial 15 solar cells (GaAs) and each of the 4 strings on the middle panel consists of 16 cells in series. The additional one solar cell in each string of the middle solar array compensates for the lack of the string total voltage resulting from the cells' functional degradation because of the higher operational temperature of the middle body mounted solar panel relative to the side solar panels. A pair of 8 solar cells mounted on outboard of the middle solar panel is connected into one string in series. The string colored with gray on the middle solar panel is a test string with 16 test solar cells with higher voltages. The developed flight model of the middle solar panel is illustrated in Figure C.3.

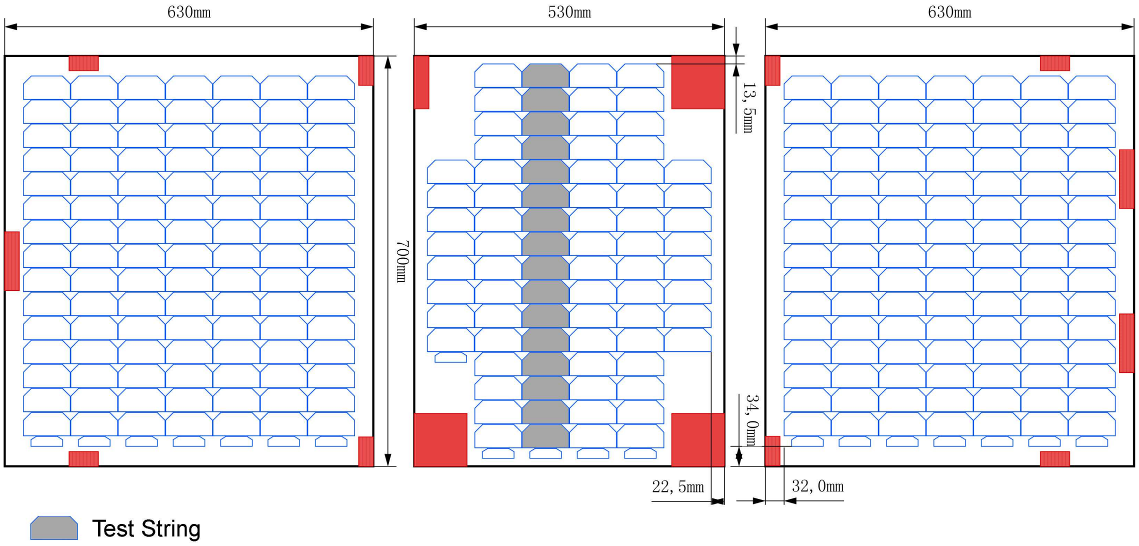


Figure C.2.: Solar panel configuration

Power distribution

As shown in Figure C.4, four redundant shunt regulators are implemented on the PCDU to distribute the power. The forth regulator is redundant and can bear the function of a failed regulator. The power lines to the satellite electronics are divided into four lines as illustrated

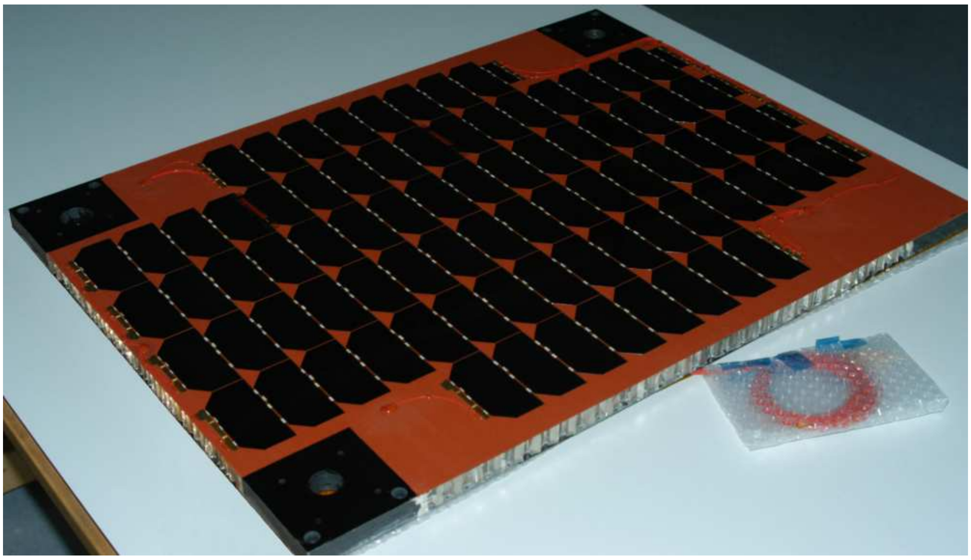


Figure C.3.: Flight model of the middle solar panel

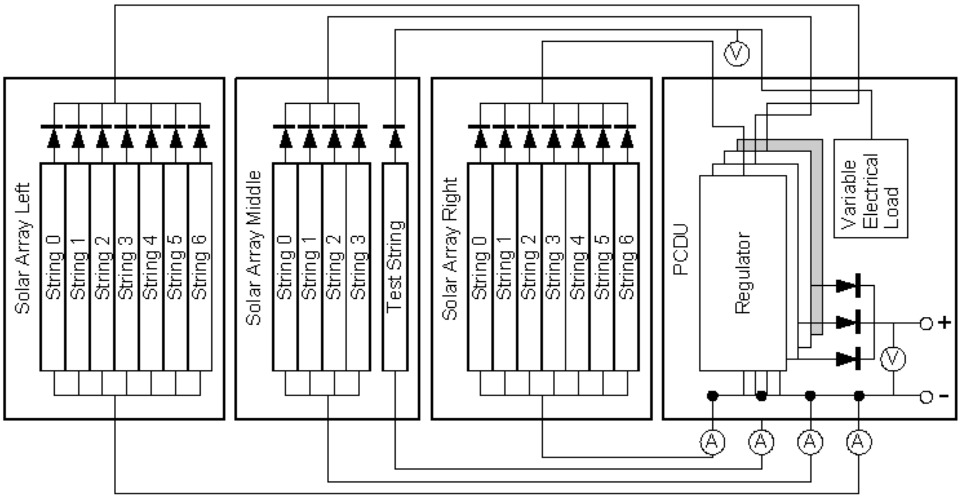


Figure C.4.: Configuration of solar panels and PCDU

in Figure 4.6. The power source is distributed into two inverters for the most of the electrical satellite components, one high voltage line for the Ka-Band Traveling Wave Tube Amplifier (TWTA), which consists of TWT and Electric Power Controller (EPC), and one nonregulated voltage lines for solar panel deployment mechanisms and heaters which are protected by protection switches, respectively. As shown in Figure 4.6 most of the redundant components, reaction wheels, magnetic torquers, magnetometers, UHF/S-Band receivers, central processing nodes, and command decoder and voter, are divided into two groups each supplied from one of the two main inverters. The solar panel deployment mechanisms are arranged as full redundant and their power supply are cross coupled. The most of the heaters are mounted as paired and supplied with power independently. The detailed description of the PCDU can be seen in [138].

Battery

The battery of the *Flying Laptop* consists of six Li-ion battery cells (Figure C.5), which is connected in series, and battery management system. Li-ion battery cells are chosen because of its higher energy storage density. The characteristic of the battery is listed in Table C.4. The PCDU even operates under breakdown of one battery cell, that is with voltage of about 16 V. The solar panel deployment mechanisms and heaters have less influence from battery breakdown because they are supplied unregulated power from the main power bus.

The Battery Management System conducts battery cell balancing and ensures that all battery cells are supplied with constant current/constant voltage power. It also prevents battery cells from over charging, protects from over/under–voltage, and monitors supplied current and voltage for cells and temperatures of the battery. The temperature control of battery cells are accomplished with redundant bi-metal heaters.

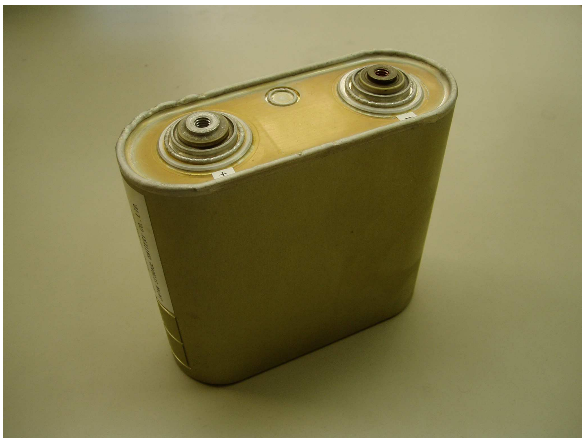


Figure C.5.: Li-Ion battery cell

Table C.4.: Battery cell characteristics

Parameter	Performance
Minimum (BOL) cell capacity	50 Ah
Minimum (EOL) cell capacity	30 Ah
Battery Configuration	6 cells in series
Depth of Discharge	20 % max. BOL
Discharge Voltage (Battery total)	21.6 V

D. Attitude control algorithms

In this appendix, mathematical descriptions of detumbling mode and safe mode of the *Flying Laptop* satellite are presented. For detailed information see [89].

D.1. Detumbling mode

The objective of the detumbling mode is to reduce the angular velocity ω after the launcher separation. It is also used for rate damping in case the rotational rate accidentally exceeds the maximum limit. This mode plays an important role for safe satellite operation [139]. Due to high angular rates in this mode, the measured magnetic field vector changes mainly as a result of the satellite's rotation and not because of the orbit variation of the Earth's magnetic field. Thus, the derivative of the magnetic field vector is a good approximation for the magnitude of the satellite's angular velocity and detumbling can be performed with magnetometer measurements alone. Because a rotation about the Earth's magnetic field vector does not change its derivative, only vector components perpendicular to the magnetic field can be measured at every given point of time. Nevertheless, the magnetic field variation over a whole orbit allows a full 3-axis rate damping.

The control law of the detumbling mode is based on the common B-dot control, such that

$$\mathbf{m}_{Det} = -k_b \frac{\dot{\mathbf{B}}}{\|\mathbf{B}\|} \;, \tag{D.1}$$

where m_{Det} is the controlling magnetic dipole moment, $\mathbf{B}$ is the measured magnetic field vector, k_b is a positive scalar gain.

D.2. Safe mode

The sensors for this mode need to have high reliabilities and sensor outputs shall be available all the time. For this reason, this mode uses only sun sensors and magnetometers. The control torque is generated by means of magnetic torquers.

In this mode, during the sun phase, the reference axis $\boldsymbol{s}_{ref}$, which is equal to the negative principal axis, is aligned with the sun direction $\boldsymbol{s}$, and at the same time, an additional spin around this axis is built up. Due to this spin stabilization, the satellite attitude can be maintained during eclipse phases without the sun sensor information.

Sun acquisition is achieved by:

$$\mathbf{T}_{sun} = k_s(\mathbf{s} \times \mathbf{s}_{ref}) - k_{\perp}(\dot{\mathbf{s}} \times \mathbf{s}) \;. \tag{D.2}$$

The first term controls the deviation from the desired attitude relative to the sun and the second controls the rate perpendicular to the sun vector to zero. k_s and $k_\perp$ are positive scalar gains. The spin rate along the reference axis is approximated as

$$\omega_\parallel \approx -\frac{(\mathbf{B} \times \mathbf{s}_{ref})\dot{\mathbf{B}}}{\|\mathbf{B} \times \mathbf{s}_{ref}\|^2} \ . \tag{D.3}$$

The spin-up control law can be written with a positive scalar gain $k_\parallel$ as:

$$\mathbf{T}_{spin} = k_\parallel(\omega_{ref} - \omega_\parallel)\mathbf{s} \ . \tag{D.4}$$

As the result, The controlling magnetic dipole moment is calculated as

$$\mathbf{m}_{Safe} = \frac{\mathbf{B} \times (\mathbf{T}_{sun} + \mathbf{T}_{spin})}{\|\mathbf{B}\|^2} \ . \tag{D.5}$$

D.3. Inertia matrices

The inertia matrix **I**, principal moments of inertia $\mathbf{I}_p$, principal axes **A** of the *Flying Laptop* satellite in configuration A, B$_\mathrm{R}$, and C at the time of writing is summarized in followings:

Configuration A:

$$\mathbf{I}_{FLP,Config.A} = \begin{bmatrix} 8.710976 & -0.265904 & 0.26067 \\ -0.265904 & 8.223424 & -0.147369 \\ 0.26067 & -0.147369 & 9.719775 \end{bmatrix} \mathrm{kg \cdot m^2} \tag{D.6}$$

$$\mathbf{I}_{p_{FLP,Config.A}} = \begin{bmatrix} 8.105861 \\ 9.810634 \\ 8.73768 \end{bmatrix} \mathrm{kg \cdot m^2} \tag{D.7}$$

$$\mathbf{A}_{FLP,Config.A} = \begin{bmatrix} 0.394957 & -0.258783 & -0.881499 \\ 0.91848 & 0.132194 & 0.372718 \\ 0.020076 & -0.956847 & 0.289898 \end{bmatrix} \tag{D.8}$$

Configuration B$_\mathbf{R}$:

$$\mathbf{I}_{FLP,Config.B_R} = \begin{bmatrix} 8.652003 & -0.27986 & 0.679485 \\ -0.27986 & 7.39654 & -0.161595 \\ 0.679485 & -0.161595 & 8.951864 \end{bmatrix} \mathrm{kg \cdot m^2} \tag{D.9}$$

$$\mathbf{I}_{p_{FLP,Config.B_R}} = \begin{bmatrix} 7.336762 \\ 9.540287 \\ 8.123358 \end{bmatrix} \mathrm{kg \cdot m^2} \tag{D.10}$$

$$\mathbf{A}_{FLP,Config.B_R} = \begin{bmatrix} 0.201585 & -0.629008 & -0.750808 \\ 0.979382 & 0.13976 & 0.145868 \\ 0.013181 & -0.764732 & 0.644213 \end{bmatrix} \tag{D.11}$$

Configuration C:

$$\mathbf{I}_{FLP,Config.C} = \begin{bmatrix} 8.588082 & -0.25981 & 0.262629 \\ -0.25981 & 6.572013 & -0.18192 \\ 0.262629 & -0.18192 & 8.191257 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{D.12}$$

$$\mathbf{I}_{pFLP,Config.C} = \begin{bmatrix} 6.52571 \\ 8.763774 \\ 8.061868 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{D.13}$$

$$\mathbf{A}_{FLP,Config.C} = \begin{bmatrix} 0.11316 & -0.457686 & 0.881883 \\ 0.989471 & -0.028695 & -0.141857 \\ 0.090232 & 0.888651 & 0.44962 \end{bmatrix} \tag{D.14}$$

Following inertia matrix and principal axis $\mathbf{A}_z$ were used for simulation purposes before the actual value of the *Flying Laptop* satellite became available.

$$\mathbf{I}_{SIM} = \begin{bmatrix} 3.8965 & -0.03007 & 0.02355 \\ -0.03007 & 4.0906 & -0.25174 \\ 0.02355 & -0.25174 & 4.2621 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{D.15}$$

$$\mathbf{A}_{zSIM} = \begin{bmatrix} -0.0667305 \\ 0.5818428 \\ -0.8105591 \end{bmatrix} \tag{D.16}$$

E. Hardware logic design with Handel-C

Handel-C is based on ANSI C syntax. Programming with Handel-C is a hardware design activity. It has extensions and restrictions for hardware design such as to produce parallelism, arbitrary bit width/operation, synchronization, and hardware interfaces. In order to produce high performance hardware logics, following programming aspect shall be taken into account.

E.1. Types and operators

The basic type in Handel-C is integer and no floating types are supported. Integers can be of any width as signed or unsigned. For n-bits integer, Most Significant Bit (MSB) is bit n-1 and Least Significant Bit (LSB) is bit 0. These variables actually behave like registers in the sense that they receive new values on the clock cycle following an assignment. Handel-C operators include bitwise operators, shift operators, bit manipulation operators, and arithmetic operators as well as relational operators and logical operators. Handel-C does not allow side-effects in expressions. The following assignment is implemented in a logic between D-type Flip-Flops (FFs) as illustrated in Figure E.1 (see Figure 6.3 for hardware resources of a Slice).

```
unsigned 1 a, b;
a = b;
```

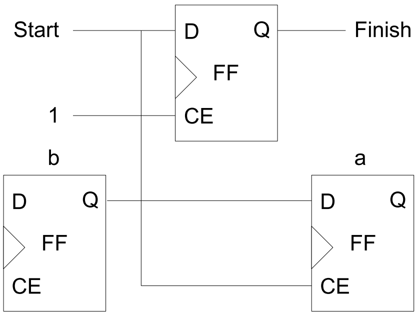


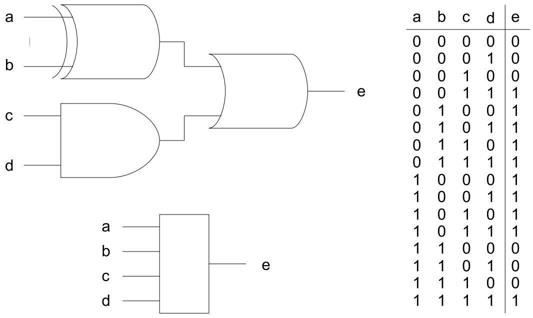
Figure E.1.: Hardware logic of variable allocation

Shift operator: Shift operators are implemented as wires and shift bits in variables to right (>>) or left (<<). Bit manipulation operators "Take" (<-) or "Drop" (\\), which are also implemented as wires, enable bitwise assignment from one variable to another. Take operator returns the n LSB of a value and the drop operator returns all but n LSB bits of a value.

Arithmetic operator: Arithmetic operators are Add (+), Subtract (-), Multiply (*), divide (/) and modulus (%). Because / and % produce deep hardware logic, their use shall be prevented.

Bitwise operator: Bitwise operators enable AND (&), OR (|), NOT (~), and XOR (^) bitwise operation. Following exemplary operation is implemented in a LUT as illustrated in Figure E.2.

```
unsigned 1 a, b, c, d, e;
e = (a ^ b) | (c & d);
```



a	b	c	d	e
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Figure E.2.: Combinational logic in LUT

Conditional operator: Handel-C also supports conditional operator as described below. If the "condition" evaluates to 0, then the results is "expression2", otherwise the result is "expression1".

```
condition ? expression1 : expression2
```

Assignment of conditional operator takes one clock cycle to complete. Both expressions are indeed evaluated in parallel, while "if" statements, which is described later, select which statement to execute. Following conditional operator is implemented in hardware logic as illustrated in Figure E.3.

```
  c = (a == b) ? (x * y) : z;
```

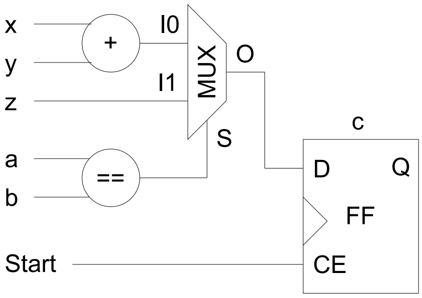


Figure E.3.: Conditional operator implementation

Concatenation operator: Concatenation operator (@) joins two sets of bits together into a set of bits, whose width is the sum of the widths of the two operands. Following example results in x = 0b10.

```
unsigned 2 x; unsigned 1 a, b;
a = 0b1; b = 0b0;
x = a @ b;
```

Bit selection operator: Bit selection operator [:] selects out either individual or a range of bits from a value. Bit 0 is the LSB. Following example results in b = 0b1100, c = 0b0.

```
unsigned 8 a; unsigned 4 b; unsigned 1 c;
a = 0b10110000;
b = a[5:2];
c = a[6];
```

E.2. Parallelism

Handel-C is implicitly sequential and each assignment takes one clock cycle. In the hardware level, each logic block has an input signal Start and an output signal Finish. When a specific statement is to be executed, its Start signal goes high for one clock cycle. In this way, multiple blocks are executed in sequences. Handel-C introduces “par” statement to implement parallel processes. On the contrary, sequential processes may be explicitly declared as “seq” as illustrated below.

```
// Sequential Block
seq { // 2 Clock Cycles
    a = 1;
    b = 2;
}
```

```
// Parallel Block
par { // 1 Clock Cycle
    a = 1;
    b = 2;
}
```

The propagation of Start signal to Finish signal in a sequential case is illustrated in Figure E.4.

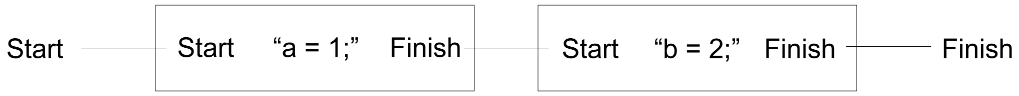


Figure E.4.: Sequential processes

It is important to note that this par statement enables both individual statements to be declared exactly in parallel at a clock cycle (fine-grained parallelism) and functions in parallel (coarse-grained parallelism). If declared as par, the block completes when longest parallel branch completes. This is implemented in hardware logic in the way illustrated in Figure E.5.

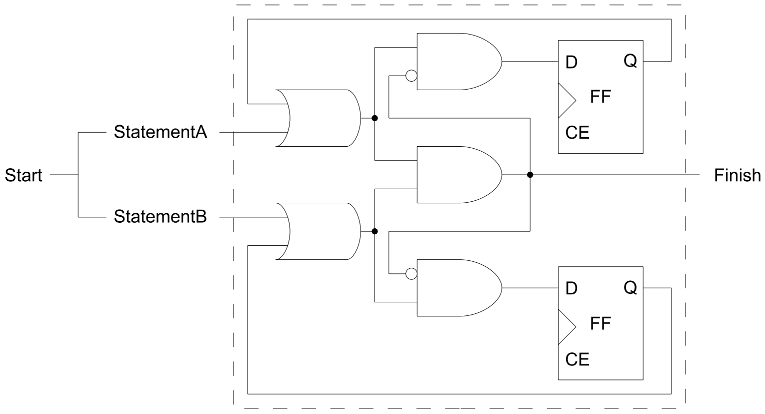


Figure E.5.: Synchronization block of parallel logics

The important constrains of Handel-C coding are:

- one can read from a variable from multiple points in the code, which is implemented as parallel wires from the register in the actual hardware.
- one can not write to the same variable in parallel, otherwise secure operation in hardware logic can not be assured.

It is also a significant difference from ANSI C, that one does not need extra temporary variables to swap values, that is the following exemplary code is valid, performing swap of the values of a and b in a single clock cycle. This is clear because the values of variables are kept at the output signal of FFs in Slices, two output signals can be cross-connected to the input port of the FFs, assigning swapped new values at the next clock cycle. This is illustrated in Figure E.6.

```
unsigned 1 a, b;
a = 1;
b = 0;
par {
    a=b;
    b=a;
}
```

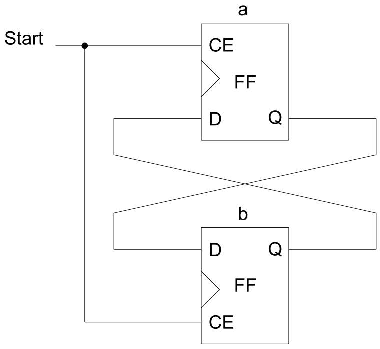


Figure E.6.: Variable swap implementation

E.3. Conditional logic

“if” statement

The expressions in conditions are evaluated in 0 clock cycle. The important point here is that one shall insert “delay;” statement even if there is nothing to do to keep the balance of execution time. In the following example, if the a is actually equal to 0, then the assignment a++ is performed, therefore it takes one clock cycle. However, without the delay statement, if a is not equal to 0, then this conditional branch does not produce any clock cycle. This may cause uncertain “combinational cycle” in the code, which can not assure correct implementation. This example is implemented in hardware as illustrated in Figure E.7.

```
if (a == 0) {
    a++;
} else {
    delay;
}
```

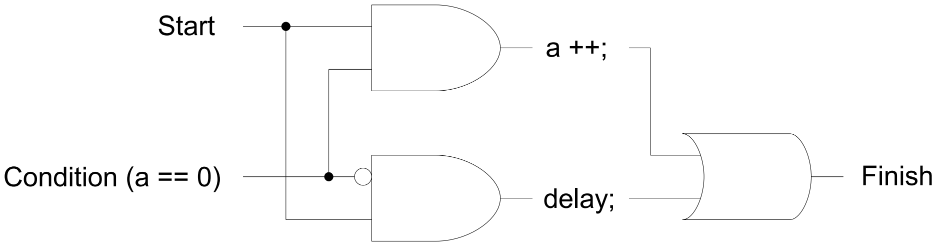


Figure E.7.: Hardware implementation of “if” condition

“do{ } while” loops

This statement is the most hardware efficient implementation of loop logics. The example codes below are implemented in the hardware logic as illustrated in Figure E.8. There is a delay statement “delay;” in order to produce one clock cycle delay. The most remarkable point of parallelism is that each function, which shall be conducted periodically and repeatedly, is implemented as “while(1)” loop, which runs forever. This idea is completely different than usual software processes. Because of their hardware level inefficiency, “for” loops shall not be used.

```
do {
    statement;
} while(Condition);
```

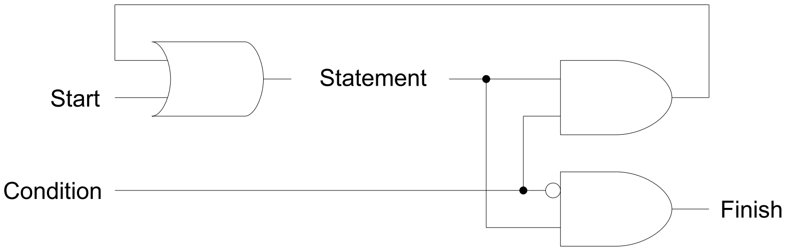


Figure E.8.: Hardware implementation of “do-while” condition

E.4. Arrays, RAMs and ROMs

Arrays can be multi-dimensional. Even though using a variable as an array index is possible, this can lead inefficient hardware implementation, and therefore constants shall be used as an array index. All elements of Arrays can be accessed in parallel.

RAMs and ROMs can only read from or write to one location in a clock cycle. ROM has no write data bus. Handel-C implements these to distributed RAM by default. RAM shall be used for Random access and large data storage. RAM can not be read, modified, and written in the same clock cycle. It is possible to use Multi-Port RAM. In this mode, devices have entirely independent read/write ports and these can be used on the same clock cycle, which is useful for, for example, line buffers in image processing. ROM can be used for, for example, LUTs of coefficients.

E.5. Initialization

FPGA can initialize variables and memories at start-up, and this can be specified in Handel-C without any logic overhead. Only static and global variables can be initialized. If a variable is not initialized, one can not assume that it will be 0 on start-up. If no initializer is specified, a static variable will be initialized to 0.

E.6. Arithmetic operation

Although arithmetic operation can be implemented into normal CLBs, it is very costly in terms of hardware logic. Xilinx Virtex offers special embedded Arithmetic LUT (ALUs) which is specially designed for efficient arithmetic operations. 18-bit multiplier blocks of Xilinx Virtex-II Pro can be utilized for mathematical operations of satellite control functions such as attitude control algorithms. Because the amount of resources are limited, efficient coding is necessary. Handel-C has the capability of mapping the design to these special elements.

E.7. Communication between logics

The most indispensable key technique of designing efficient hardware logics based on parallelism is the implementation of communication mechanisms between parallel running processes. There are three possible ways in Handel-C: via channels, via signals, and via variables.

Channels

Channels enable communication between two processes running in parallel. They block the execution of both processes until both sides, read (?) and write (!) are ready. Channels can be also used in FIFO (First In First Out) mode. Channels allow communication between processes even in different clock domains. Channels can be used for synchronization between two processes. In the following exemplary codes, two parallel running processes communicate through the channel "myChannel". Starting at the same clock cycle, the first process try to send the value of the variable a. The communication, however, actually occurs in the second clock cycle, after the second process becomes ready to read the myChannel. In this way, the two processes can exchange information through the channel, and at the same time, the two program lines are synchronized. Channels can be used between only two processes in a point-to-point manner, and do not allow multi-point communications.

```
chan unsigned 2 myChannel; // shared by both parallel processes
unsigned 2 a;                       unsigned 2 b;
do {                                do {
    myChannel ! a; // send              delay;
    delay;                              myChannel ? b; // receive
} while(1);                         } while (1);
```

Signals

Signals behave like wires and take the assigned value for the current clock cycle only. Comparison between register implementation and signal implementation are described below.

```
// implementation with variables    //implementation with signals
unsigned 2 a[4], c;                 unsigned 2 a[4], c;
unsigned 2 b[2];                    signal unsigned 2 b[2];
do { // 2 Clock Cycles              do { // 1 Clock Cycles
    par{                                par {
        b[0] = a[0] + a[1];                 b[0] = a[0] + a[1];
        b[1] = a[2] + a[3];                 b[1] = a[2] + a[3];
    }                                       c = b[0] + b[1];
    c = b[0] + b[1];                    }
} while(1);                         } while (1);
```

In the case of with variables, the value of c is assigned at the second clock cycle, and becomes actualized in the third clock cycle as the sum of the all component of the array a. On the contrary, in the case with signals, the value of c is assigned at the first clock cycle and becomes

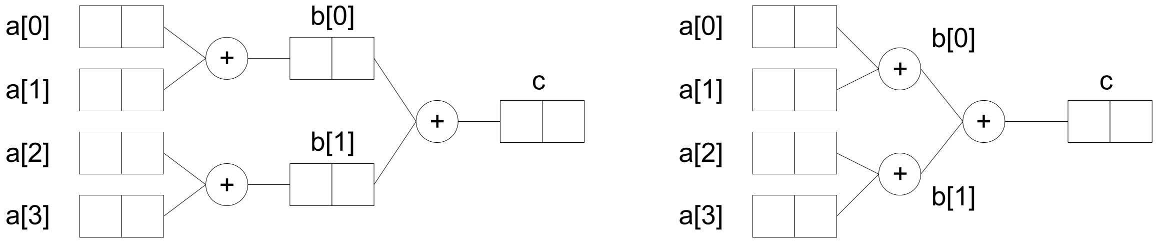


Figure E.9.: Pipelining of design

actualized already in the second cycle (Figure E.9). Because signals are implemented as simple wires, they can be connected with multiple processes. In this way, signals enable instantaneous multi-point communications between parallel processes in the very clock cycle, which is attractive for synchronization of processes, simultaneous exchange of information, and establishment of data consistency.

E.8. Pipelining

Combinational use of par and variables enables pipelining of data stream processing, as exemplary described below. In case the value of a is continuously updated, the value of c is also continuously updated with the sum of all component of the array a with a time delay of one clock cycle. This programming technique is called pipelining and is very useful for efficient and fast image and digital signal processing.

```
unsigned 2 a[4], c;
unsigned 2 b[2];
do { // 2 Clock Cycles
    par{
        b[0] = a[0] + a[1];
        b[1] = a[2] + a[3];
        c = b[0] + b[1];
    }
} while(1);
```

E.9. Timing

Each Handel-C program shall have input clock signal. This can be defined in the main source file. Each assignment in Handel-C codes takes one clock cycle. Because the duration of one clock cycle is fixed, the logic propagation through hardware elements shall meet the timing constrains. For example, each piece of hardware logic with a clock frequency of 250 MHz shall meet 4 ns logic propagation. Handel-C allows existence of more than one main source files, i.e., more than one clock domains within a design.

E.10. Prialt

The prialt statement of Handel-C can select the first channel ready to communicate from a list of channel cases. The syntax of a prialt is

```
prialt {
    case ChannelStatement1:
        statement1
        break;
    case ChannelStatement2:
        statement2
        break;
    ....
    default:
        statementDefault
        break;
}
```

where the “ChannelStatement” shall be either read (?) or write (!) channel statement (see Section E.7). The “case” whose communication statement is the first to be ready to exchange data executes the channel communication. Following statements till the next break statement will then be executed.

Priority

If two or more channels become ready in the same clock tick, the one which is listed earlier (upper) receives the higher priority.

Default case

If no communication statement is ready in the clock tick, the default case is executed. The default case is optional and if so, execution halts until one of the communication statement becomes ready.

F. Notation reference for formal verification of Handel-C code

The notations used in Chapter 9 are summarized based on those described in [110] and [127].

$\mathbb{P}A$	set of A
$\times$	cross product
$\in$	membership
$\cup$	union
A^*	zero or more As
A^+	one or more As
$A \xrightarrow{m} B$	finite partial map from A to B
$\{x \mapsto y\}$	singleton map from x to y
$\dagger$	override
$A \rightarrow B$	total function from A to B
$A \rightarrow B \rightarrow C$	curried function from A to $B \rightarrow C$
$\widehat{=}$	"is defined equal to"
$(c!e)$	output request on c of value e
$(c?v)$	input request on c into variable v
?	in
!	out
!?	default guard
τ	timestamp
$\uplus$	equation join operator ($\mathbb{P}Eqn \times \mathbb{P}Eqn \rightarrow \mathbb{P}Eqn$)

Bibliography

[1] Gokhale, M. and Graham, P. S., *Reconfigurable Computing - Accelerated Computing with Field-Programmable Gate Arrays*, Springer, 2005, ISBN 978-0-387-26105-8.

[2] Conde, R. F., Darrin, A. G., Dumont, F. C., Luers, P., Jurczyk, S., Bergmann, N., and Dawood, A., "Adaptive Instrument Module - A Reconfigurable Processor for Space Applications," Military and Aerospace Programmable Logic Devices Conference, Laurel, Maryland, USA, Sept. 1999.

[3] Bergmann, N., "Adaptive Instrument Interfacing for Satellites using FPGAs," Annual Conference on Commercialization of Military and Space Electronics, Los Angeles, USA, Feb. 2001.

[4] Dawood, A. and Bergmann, N., "An Adaptive Instrument Module (AIM) for Satellite Systems," International Symposium on Signal Processing and its Applications, Brisbane, Australia, Aug. 1999.

[5] Chu, P. P. and Kifle, M., "A Reconfigurable Communications System for Small Spacecraft," NASA Tech. Rep., TM-212534, 2004.

[6] Barnhart, D. J., Vladimirova, T., and Sweeting, M. N., "Very-small-Satellite Design for Distributed Space Missions," *Journal of Spacecraft and Rockets*, Vol. 44, No. 6, 2007, pp. 1294–1306.

[7] Corpino, S., Chiesa, S., and Viola, N., "PICPOT Program: Lessens Learned," 58th International Astronautical Congress, Paper IAC-07-B4.6.02, Hyderabad, India, Sept. 2007.

[8] Santoni, F. and Piergentili, F., "Analysis of the UNISAT-3 Solar Array In-orbit Performance," *Journal of Spacecraft and Rockets*, Vol. 45, No. 1, 2008, pp. 142–148.

[9] Uryu, A., van der Ha, J. C., Röser, H.-P., and Kuwahara, T., "QSAT Mission Analysis and Operation Plan Design," 59th International Astronautical Congress, Paper IAC-08-B4.3.7, Glasgow, United Kingdom, Sep.-Oct. 2008.

[10] Funase, R., Nakasuka, S., Sako, N., Fujiwara, T., Tsuda, Y., Ukawa, S., Kimura, S., Hashimoto, H., Yoshihara, K., and Yamamoto, T., "On-board Experiment of Vision-based Motion Estimation and Tracking of Tumbling Objects in Space," *Transactions of the Japan Society for Aeronautical and Space Sciences*, Vol. 50, No. 168, 2007, pp. 97–104.

[11] Baxter, E. and Bill, L., "Topsat: Lessons Learned from a Small Satellite Mission," *International Symposium on Small Satellite for Earth Observation Selected Proceedings*, 2007, pp. 377–384.

[12] Xilinx, Inc., "Website," Available from http://www.xilinx.com/, last visited 19 Mar. 2009.

[13] Brewer, J. E. and Gill, M., *Nonvolatile Memory Technologies with Emphasis on Flash*, IEEE Press Series on Microelectronic Systems, Hoboken, New Jersey, USA, 2008.

[14] Tietze, U. and Schenk, C., *Electronic Circuits*, Springer-Verlag, 2nd ed., 2008.

[15] Altera Corporation, “Website,” Available from http://www.altera.com/, last visited 19 Mar. 2009.

[16] Actel Corporation, “Website,” Available from http://www.actel.com/, last visited 19 Mar. 2009.

[17] Actel Corporation, “Radiation-Tolerant ProASIC3 Low-Power Space-Flight Flash FPGAs,” Tech. Rep. RTPA3-DS, Actel Corporation, Sep. 2008.

[18] Actel Corporation, “ProASIC3 Flash Family FPGAs,” Tech. Rep. PA3-DS, Actel Corporation, Feb. 2009.

[19] Wang, J. J., “Radiation Effects in FPGAs,” *Proceedings of the 9th Workshop on Electronics for LHC Experiments*, Amsterdam, The Netherlands, Sep.-Oct. 2003, pp. 34–43.

[20] Coffrey, M., Graham, P., Johnson, E., Wirthin, M., and Carmichael, C., “Single-Event Upsets in SRAM FPGAs,” Los Alamos National Laboratory LA-UR-02-6138, 2002.

[21] MacQueen, D. M., Gingrich, D. M., Buchanan, N. J., and Green, P. W., “Total Ionizing Dose Effects in a SRAM-based FPGA,” IEEE Radiation Effects Data Workshop, Norfolk, USA, July 1999.

[22] Wang, J. J., Samiee, S., Chen, H.-S., Huang, C.-K., Cheung, M., Borillo, J., Sun, S.-N., Cronquist, B., and McCollum, J., “Total Ionizing Dose Effects on Flash-based Field Programmable Gate Array,” *IEEE Transactions on Nuclear Science*, Vol. 51, No. 6, 2004, pp. 3759–3766.

[23] Actel Corporation, “HiRel SX-A Family FPGAs Data Sheet,” Tech. Rep., v5.3, 2007.

[24] Rezgui, S., “Radiation Characterization and Mitigation of RTAX-S FPGAs,” Actel Space Forum, Noordwijk, Netherlands, March 2009.

[25] Sandor, M., Davarpanah, M., Soliman, K., Suszko, S., and Mackey, S., “Field Programmable Gate Arrays: Evaluation Report for Space-Flight Application,” Tech. Rep., NASA JPL Report, NASA CR-192796, Sep. 1992.

[26] Wang, J. J., Katz, R. B., Sun, J. S., Cronquist, B. E., McCollum, J. L., Speers, T. M., and Plants, W. C., “SRAM Based Re-programmable FPGA for Space Applications,” *IEEE Transactions of Nuclear Science*, Vol. 46, No. 6, 1999, pp. 1728–1735.

[27] Rezgui, S., Wang, J. J., Sun, Y., Cronquist, B., and McCollum., J., “New Reprogrammable and Non-Volatile Radiation Tolerant FPGA: RTA3P,” *Proceedings of the IEEE Aerospace Conference*, Big Sky, 2008, pp. 1–11.

[28] Fabula, J. J. and Bogrow, H., “Total Ionizing Dose Performance of SRAM-based FPGA and supporting PROMs,” Military and Aerospace Programmable Logic Devices Conference, Maryland, USA, Sept. 2000.

[29] Vera, A., Llamocca, D., Pattichis, M., Kemp, W., Shedd, W., Alexander, D., and Lyke, J., “Dose Rate Upset Investigations on the Xilinx Virtex IV Field Programmable Gate Array,” *IEEE Radiation Effects Data Workshop*, 2007, pp. 172–176.

[30] Smith, F. and Mostert, S., “Total Ionizing Dose Mitigation by Means of Reconfigurable FPGA Computing,” *IEEE Transactions on Nuclear Science*, Vol. 54, No. 4, 2007, pp. 1343–1349.

[31] Fabula, J. J., DeJong, J. L., Lesea, A., and Hsieh, W.-L., "The Total Ionizing Dose Performance of Deep Submicron CMOS Processes," Military and Aerospace Programmable Logic Devices Conference, Maryland, USA, Sept. 2008.

[32] George, J., Rezgui, S., Swift, G., and Carmichael, C., "Initial Single-Event Effects Testing and Mitigation in the Xilinx Virtex II-Pro FPGA," Military and Aerospace Programmable Logic Devices Conference, Washington, D. C., USA, Sept. 2005.

[33] Speers, T., Wang, J. J., Cronquist, B., McCollum, J., Tseng, H., Katz, R., and Kleyner, I., "0.25 μm Flash Memory Based FPGA for Space Applications," Military and Aerospace Programmable Logic Devices Conference, Maryland, USA, Sept. 1999.

[34] Snyder, E. S., McWhorter, P. J., Dellin, T. A., and Sweetman, J. D., "Radiation Response of Floating Gate EEPROM Memory Cells," *IEEE Transactions on Nuclear Science*, Vol. 36, No. 6, 1989, pp. 2131–2139.

[35] Allen, G., McClure, S., Rezgui, S., and Wang, J. J., "Total Ionizing Dose Characterization Results of Actel ProASIC3, ProASIC3L, and IGLOO Flash-based Field Programmable Gate Arrays," Military and Aerospace Programmable Logic Devices Conference, Maryland, USA, Sept. 2008.

[36] Fuller, E., Caffrey, M., Blain, P., Carmichael, C., Khalsa, N., and Salazar, A., "Radiation test results of the Virtex FPGA and ZBT SRAM for Space Based Reconfigurable Computing," Los Alamos National Laboratory LA-UR-99-5166, 1999.

[37] Fuller, E., Caffrey, M., Salazar, A., Carmichael, C., and Fabula, J., "Radiation Characterization, and SEU Mitigation, of the Virtex FPGA for Space-Based Reconfigurable Computing," Los Alamos National Laboratory LA-UR-00-3345, 2000.

[38] Swift, M., Jet Propulsion Laboratory, and California Institute of Technology, "Xilinx Single Event Effects 1st Consortium Report Virtex-II Static SEU Characterization," Tech. Rep., Jan. 2004.

[39] Koga, R., George, J., Swift, G., Yui, C., Edmonds, L., Carmichael, C., Langley, T., Murray, P., Lanes, K., and Napier, M., "Comparison of Xilinx Virtex-II FPGA SEE Sensitivities to Protons and Heavy Ions," *IEEE Transactions on Nuclear Science*, Vol. 51, No. 5, 2004, pp. 2825–2833.

[40] Yui, C., Swift, G., and Carmichael, C., "Single Event Upset Susceptibility Testing of the Xilinx Virtex II FPGA," Military and Aerospace Programmable Logic Devices Conference, Maryland, USA, Sept. 2002.

[41] Napier, M., Moore, J., Rezgui, S., Carmichael, C., George, J., and Swift, G., "Single Event Effect (SEE) Analysis, Test & Mitigation of the Xilinx Virtex-II Input Output Block (IOB)," Military and Aerospace Programmable Logic Devices Conference, Washington, D. C., USA, Sept. 2004.

[42] Yui, C. C., Swift, G. M., Carmichael, C., Koga, R., and George, J. S., "SEU Mitigation Testing of Xilinx Virtex II FPGAs," *IEEE Radiation Effects Data Workshop*, 2003, pp. 92–97.

[43] Foster, C. C., O'Neil, P. M., and Kouba, C. K., "Monte Carlo Simulation of Proton Upsets in Xilinx Virtex-II FPGA Using a Position Dependent Q_{crit} with PROPSET," *IEEE Transactions on Nuclear Science*, Vol. 53, No. 6, 2006, pp. 3494–3501.

[44] Edmonds, L. D. and Irom, F., "Extension of a Proton SEU Cross Section Model to Include 14 MeV Neutrons," *IEEE Transactions on Nuclear Science*, Vol. 55, No. 1, 2008, pp. 649–655.

[45] George, J., Koga, R., and McMahan, M., "Neutron Soft Errors in Xilinx FPGAs at Lawrence Berkeley National Laboratory," *IEEE Radiation Effects Data Workshop*, 2008, pp. 118–123.

[46] NASA - Goddard Space Flight Center, "Xilinx Virtex-II Pro PowerPC Proton Test Results," Tech. Rep., I101705_V2Pro, Oct. 2005.

[47] Hiemstra, D. M., Chayab, F., and Mohammed, Z., "Single Event Upset Characterization of the Virtex-4 Field Programmable Gate Array Using Proton Irradiation," *IEEE Radiation Effects Data Workshop*, 2003, pp. 105–108.

[48] George, J., Koga, R., Swift, G., Allen, G., Carmichael, C., and Tseng, C. W., "Single Event Upsets in Xilinx Virtex-4 FPGA Devices," *IEEE Radiation Effects Data Workshop*, 2006, pp. 109–114.

[49] Allen, G., Swift, G., and Carmichael, C., "Virtex-4QV Static SEU Characterization Summary," NASA Tech. Rep., April 2008.

[50] Quinn, H., Morgan, K., Graham, P., Krone, J., and Caffrey, M., "Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device," *IEEE Radiation Effects Data Workshop*, 2007, pp. 177–184.

[51] Czajkowski, D., Samudrala, P., and Pagey, M., "SEU Mitigation for Reconfigrable FPGAs," IEEE Aerospace Conference, Big Sky, Montana, USA, March 2006.

[52] Adell, P. and Allen, G., "Assessing and Mitigating Radiation Effects in Xilinx FPGAs," NASA Tech. Rep., 2008.

[53] Xilinx, "Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet," Tech. Rep., DS083 (v4.6), 2007.

[54] Xilinx, "Radiation-Tolerant Virtex-4 QPro-V Family Overview," Tech. Rep., DS653 (v1.2), 2008.

[55] Tylka, A. J., J. H. Adams, J., Boberg, P. R., Brownstein, B., Dietrich, W. F., Flueckiger, E. O., Petersen, E. L., Shea, M. A., Smart, D. F., and Smith, E. C., "CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code," *IEEE Transactions on Nuclear Science*, Vol. 44, No. 6, 1997, pp. 2150–2160.

[56] Naval Research Laboratory, "CREME96: Cosmic Ray Effects on Micro-Electronics," Available from https://creme96.nrl.navy.mil/, last visited 27 Jan. 2009.

[57] European Space Agency, Belgian Institute of Apace Aeronomy, "SPENVIS - Space Environment Information System," Available from http://www.spenvis.oma.be/, last visited 27 Jan. 2009.

[58] Rezgui, S., Wang, J., Tung, E., Cronquist, B., and McCollum, J., "NEW Methodologies for SET Characterization and Mitigation in Flash-Based FPGAs," *IEEE Transactions on Nuclear Science*, Vol. 54, No. 6, 2007, pp. 2512–2524.

[59] Schneeweiss, W. G., *Reliability Modeling*, LiLoLe-Verlag GmbH, Hagen, Germany, 1st ed., 2001, ISBN 3-934447-04-X.

[60] US Department of Defense, "Electronic Reliability Design Handbook," Tech. Rep., MIL-HDBK-338B, 1998.

[61] Dittmar, M., "Radiation Effects on FPGAs," Tech. Rep. FLP-AN-0000000-001-IRS, Institute of Space Systems, Stuttgart, Germany, Dec. 2009.

[62] Grillmayer, G. and Muhammad, Y., "Radiation Environment," Tech. Rep. FLP-TN-00000-004-IRS, Institute of Space Systems, Stuttgart, Germany, June 2006.

[63] Grillmayer, G., Lengowski, M., Walz, S., and et al., "Flying Laptop - Micro-Satellite of the University of Stuttgart for Earth Observation and Technology Demonstration," 55th International Astronautical Congress, Paper IAC-04-IAA.4.11.P.08, Vancouver, Canada, Oct. 2004.

[64] Grillmayer, G., Falke, A., and Röser, H.-P., "Technology Demonstration with the Micro-satellite Flying Laptop," *International Symposium on Small Satellite for Earth Observation Selected Proceedings*, Berlin, Germany, April 2005, pp. 419–427.

[65] Laufer, R., Röser, H.-P., and the Lunar Mission BW1 Project Team, "LUNAR MISSION BW1 - A Small Lunar Exploration ans Technology Demonstration Satellite," 1st European Planetary Science Congress (EPSC), EPSC2006-A-00488P, Berlin, Germany, Sept. 2006.

[66] Röser, H.-P., Auweter-Kurz, M., Wagner, H. P., Laufer, R., and Huber, F., "Challenges and innovative technologies for a low cost lunar mission," *Acta Astronautica*, Vol. 59, No. 8–11, 2006, pp. 1048–1051.

[67] Laufer, R. and Röser, H.-P., "Lunar Mission BW1 - An Academic Low-cost Small Lunar Exploration Satellite," 58th International Astronautical Congress, Paper IAC-07-A3.I.A.03, Hyderabad, India, Sept. 2007.

[68] Grillmayer, G. and Hirth, M., "ACS Matlab Model Description," Tech. Rep. FLP-TN-00000-006-IRS, Institute of Space Systems, Stuttgart, Germany, Nov. 2006.

[69] Grillmayer, G. and Hirth, M., "ACS Simulink Model Description," Tech. Rep. FLP-TN-00000-005-IRS, Institute of Space Systems, Stuttgart, Germany, Nov. 2006.

[70] Böhringer, F. and Grillmayer, G., "Orbit Definition for Simulation Purposes," Tech. Rep. FLP-SP-00000-001-IRS, Institute of Space Systems, Stuttgart, Germany, Sept. 2006.

[71] Grillmayer, G. and Böhringer, F., "μASC Operation Analysis," Tech. Rep. FLP-TN-00000-003-IRS, Institute of Space Systems, Stuttgart, Germany, Sept. 2006.

[72] Grillmayer, G. and Muhammad, Y., "Data Transfer Protocol of ACS Simulation in FPGA-Matlab Environment," Tech. Rep. FLP-TN-00000-009-IRS, Institute of Space Systems, Stuttgart, Germany, Nov. 2006.

[73] Walz, S., "Description of the BRDF Measurement," Tech. Rep. FLP-TN-00000-008-IRS, Institute of Space Systems, Stuttgart, Germany, May 2006.

[74] Walz, S., Lengowski, M., and von Schönermark, M., "Payload and Scientific Investigation of the Flying Laptop," *International Symposium on Small Satellite for Earth Observation Selected Proceedings*, Berlin, Germany, April 2005, pp. 153–160.

[75] Lengowski, M., Röser, H.-P., Haarmann, R., Beyermann, U., and Gebel, G., "Mechanical Design of the Micro-Satellite Flying Laptop," 6th International Symposium on Small Satellite for Earth Observation, Berlin, Germany, April 2007.

[76] Falke, A., Grillmayer, G., Walz, S., Hesselbach, F., Eickhoff, J., and Röser, H.-P., "LED In-flight calibration and Model-based Development of ACS Algorithms for the University Micro-satellite FLYING LAPTOP," 4th Small Satellite Systems and Services Symposium, Chia Laguna, Italy, Sept. 2006.

[77] Eickhoff, J., Falke, A., and Röser, H.-P., "Model-based design and verification - State of the art from Galileo constellation down to small university satellites," *Acta Astronautica*, Vol. 61, No. 1–6, 2007, pp. 383–390.

[78] Brandt, A., Kossev, I., Falke, A., Eickhoff, J., and Röser, H.-P., "Preliminary System Simulation Environment of the University Micro-satellite Flying Laptop," *International Symposium on Small Satellite for Earth Observation Selected Proceedings*, Berlin, Germany, April 2007, pp. 231–243.

[79] Fritz, M., "Simulated operations of the microsatellite Flying Laptop with a Mission Operation System," Study Thesis IRS-08-S-37, Institute of Space Systems, Universität Stuttgart, Stuttgart, Germany, 2008.

[80] Fritz, M., Falke, A., Kuwahara, T., Röser, H.-P., Pearson, S., Witts, A., and Eickhoff, J., "A Commercial Procedure Execution Engine Completing the Command Chain of a University Satellite Simulation Infrastructure," *Acta Astronautica*, Vol. 66, No. 5–6, 2009, pp. 950–953.

[81] von Schönermark, M., Geiger, B., and Röser, H.-P., *Reflection Properties of Vegetation and Soil with a BRDF Data Base*, Wissenschaft und Technok Verlag, 2004, ISBN 3-89685-565-4.

[82] Kuwahara, T., "Flying Laptop Mission Success Level," Tech. Rep. FLP-RQ-00000-001-IRS, Institute of Space Systems, Stuttgart, Germany, Oct. 2006.

[83] Kuwahara, T., Huber, F., Falke, A., Lengowski, M., Walz, S., Grillmayer, G., and Röser, H.-P., "System Design of the Small Satellite Flying Laptop, as the Technology Demonstrator of the FPGA-based On-board Computing System," 58th International Astronautical Congress, Paper IAC-07-B4.6.08, Hyderabad, India, Sept. 2007.

[84] Hesselbach, F., "Development of a Calibration Concept with LEDs for the Multispectral Camera System MICS of the Small Satellite Flying Laptop," Study Thesis IRS-06-S-48, Institute of Space Systems, Universität Stuttgart, Stuttgart, Germany, Dec. 2006.

[85] Kirchgäßner, U., Kuwahara, T., and von Schönermark, M., "Improved Land Surface Temperature Retrieval Method for the Small Satellite Flying Laptop," 59th International Astronautical Congress, Paper IAC-08-B1.4.9, Glasgow, United Kingdom, Sep.-Oct. 2008.

[86] Urbainczyk, C., "Characterization and Testing of the Infrared Instrument TICS for the micro-satellite Flying Laptop," Study Thesis IRS-08-S-07, Institute of Space Systems, Universität Stuttgart, Stuttgart, Germany, March 2008.

[87] Waidmann, M., Waidmann, C., Saile, D., Grillmayer, G., and Wolter, V., "Use of New Developments of Attitude Control Sensors for the Micro satellite FLYING LAPTOP," 57th International Astronautical Congress, Paper TBC, Valencia, Spain, Oct. 2006.

[88] Hauschild, A., Grillmayer, G., Montenbruck, O., Markgraf, M., and Vörsmann, P., "GPS Based Attitude Determination for the Flying Laptop Satellite," *International Symposium on Small Satellite for Earth Observation Selected Proceedings*, Berlin, Germany, April 2007, pp. 211–220.

[89] Grillmayer, G., Hirth, M., Huber, F., and Wolter, V., “Development of an FPGA based Attitude Control System for a Micro-Satellite,” AIAA Paper 2006-6522.

[90] Flying Laptop Team, “Technical Note on Flying Laptop,” Tech. Rep. FLP-TN-00000-001-IRS, Institute of Space Systems, Stuttgart, Germany, May 2005.

[91] Kuwahara, T., Falke, A., and Röser, H.-P., “Effective Project Management of Small Satellite Projects from the System Engineer’s Point of View, An Example of the Small Satellite Flying Laptop Project,” *Transactions of Japan Society for Aeronautical and Space Sciences, Space Technology Japan*, Vol. 7, No. ists26, 2009, pp. Pt_1–Pt_8.

[92] Öztürk, S., “Development of a Harness concept for the Flying Laptop and Construction of a “Harness Model”,” Study Thesis IRS-06-S-40, Institute of Space Systems, Universität Stuttgart, Stuttgart, Germany, Oct. 2006.

[93] Kuwahara, T., “Technical Note on On-board Computer Interconnections,” Tech. Rep. FLP-TN-04000-001-IRS, Institute of Space Systems, Stuttgart, Germany, Oct. 2006.

[94] Kuwahara, T., Böhringer, F., Falke, A., Eickhoff, J., Huber, F., and Röser, H.-P., “FPGA-based Operational Concept and Payload Data Processing for the Flying Laptop Satellite,” *Acta Astronautica*, Vol. 65, No. 11–12, 2009, pp. 1616–1627.

[95] EADS Astrium GmbH - Business Division Equipment & Subsystems, “Technical Description of Flying Laptop Solar Panels,” Tech. Rep. FLP-ASO-DS-1000-0001, EADS Astrium GmbH, Feb. 2007.

[96] GS Japan Storage Battery, “Lithium Ion Cells for Space Applications,” Tech. Rep., GS Japan Storage Battery, May 2003.

[97] DIEHL & EAGLE PICHER Batterie-Systeme, “Procurement Specification of the Li-Ion Batteries Assemblies,” Tech. Rep. SPE/IRS/001/DEP, DIEHL & EAGLE PICHER, Oct. 2006.

[98] Roche, G. L., *Solar Panel for the Space (in German)*, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 1997.

[99] Halaczek, T. L., *Batteries and Charging Concept (in German)*, Franzis-Verlag GmbH, 1996.

[100] Pletner, S., “Detailed Specification of FLP-Node,” Tech. Rep. FLP-FIRST-SP-01, Issue 1.4, Fraunhofer Institute of Computer Architecture and Software Technology, Berlin, Germany, Jan. 2007.

[101] Steinbeis Transferzentrum Raumfahrt (TZR), “Website,” Available from http://www.tz-raumfahrt.de/, last visited 19 Mar. 2009.

[102] Xilinx, “Data Recovery,” Tech. Rep., XAPP224 (v2.5), 2005.

[103] ISO/IEC, “Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures,” Tech. Rep., ISO/IEC 13239:2002(E), 2002.

[104] Oldham, T. R., Suhail, M., Friendlich, M. R., Carts, M. A., Ladbury, R. L., Kim, H. S., Berg, M. D., Poivey, C., Buchner, S. P., Sanders, A. B., Seidleck, C. M., and LaBel, K. A., “TID and SEE Response of Advanced 4G NAND Flash Memories,” IEEE Radiation Effects Data Workshop, Tucson, USA, July 2008.

[105] Oldham, T. R., Friendlich, M. R., J. W. Howard, J., Berg, M. D., Kim, H. S., Irwin, T. L., and LaBel, K. A., "TID and SEE Response of an Advanced Samsung 4Gb NAND Flash Memory," IEEE Radiation Effects Data Workshop, Honolulu, USA, July 2007.

[106] Cochran, D. J., Buchner, S. P., Poivey, C., LaBel, K. A., Ladbury, R. L., O'Bryan, M., J. W. Howard, J., Sanders, A. B., and Oldham, T. R., "Compendium of Current Total Ionizing Dose Results and Displacement Damage Results for Candidate Spacecraft Electronics for NASA," IEEE Radiation Effects Data Workshop, Honolulu, USA, July 2007.

[107] Cellere, G., Paccagnella, A., Visconti, A., Bonanomi, M., Beltrami, S., Schwank, J. R., Shaneyfelt, M. R., and Paillet, P., "Total Ionizing Dose Effects in NOR and NAND Flash Memories," *IEEE Transactions on Nuclear Science*, Vol. 54, No. 4, 2007, pp. 1066–1070.

[108] 3D PLUS, "Detail Specification 16Gbit Flash NAND: MMFN08204804S-F," Tech. Rep. 3DPA2460-2, 3D PLUS, BUC, France, July 2008.

[109] SAMSUNG Electronics, "512M x 8 Bit / 1G x 8 Bit NAND Flash Memory," Tech. Rep. K9XXG08UXA, SAMSUNG Electronics, Korea, March 2006.

[110] Butterfield, A. and Woodcock, J., "prialt in Handel-C: an operational semantics," *Software Tools for Technology Transfer*, Vol. 7, No. 3, 2005, pp. 1–34.

[111] Tanenbaum, A. S., *Structured Computer Organization*, Prentice Hall, 5th ed., 2005, ISBN 978-0131485211.

[112] Wolter, V., "TZRSHIP ACS Interface Software," Tech. Rep. FLP-ICD-04000-001-TZR, Steinbeis Transferzentrum Raumfahrt, Stuttgart, Germany, March 2007.

[113] Weiss, G., editor, *Multiagent Systems*, Massachusetts Institute of Technology, 1st ed., 2000.

[114] Kayal, H., Barwald, W., Briess, K., Gill, E., Halle, W., and et al., "Onboard Autonomy and Fault Protection Concept of the BIRD Satellite," *Proceedings of IEEE International Conference on Recent Advances in Space Technologies*, Istanbul, Turkey, 2003, pp. 34–41.

[115] Brooks, R. A., "A Robust Layered Control System For A Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, 1986, pp. 14–23.

[116] Brooks, R., "Planning Is Just A Way Of Avoiding Figuring Out What To Do Next," Tech. Rep., 1987.

[117] Brooks, R. A. and Flynn, A. M., "Fast, Cheap and Out of Control: A Robot Invasion of the Solar System," *Journal of the British Interplanetary Society*, Vol. 42, 1989, pp. 478–485.

[118] Rezgui, S., "Radiation Characterization and Mitigation of RT ProASIC3 Flash-based FPGAs," Actel Space Forum, Noordwijk, Netherlands, March 2009.

[119] Actel Corporation, "Libero IDE v8.5 User's Guide," Tech. Rep., 5-02-9120-23, 2008.

[120] Xilinx, "Xilinx ISE 10.1 Design Suite Software Manuals and Halp - PDF Collection," Tech. Rep., 2008.

[121] Eickhoff, J., Hendricks, R., and Flemmig, J., "Model-based Development and Verification Environment," 54th International Astronautical Congress, Paper IAC-03-U.3.07, Bremen, Germany, Sep.-Oct. 2003.

[122] Kuwahara, T., Falke, A., Ziemke, C., Muhammad, Y., Eickhoff, J., and Röser, H.-P., "Development of a Hardware-in-the-Loop Simulation Environment on a MDVE for FPGA-based On-board Computing Systems," *Transactions of Japan Society for Aeronautical and Space Sciences, Space Technology Japan*, Vol. 7, No. ists26, 2009, pp. Pf_1–Pf_9.

[123] Eickhoff, J., "Systemsimulation in der Satellitenentwicklung I & II (System Simulation in Satellite Engineering I & II)," Annual lectures at Institute of Space Systems, Universität Stuttgart .

[124] Ziemke, C., "Model based development and verification of an on-board control algorithm architecture for the micro satellite "Flying Laptop"," Study Thesis IRS-08-S-16, Institute of Space Systems, Universität Stuttgart, Stuttgart, Germany, May 2008.

[125] Hoare, C. A. R., *Communicating Sequential Processes*, Prentice-Hall, 1st ed., 1985.

[126] Roscoe, A. W., *The Theory and Practice of Concurrency*, Prentice-Hall (Pearson) and Bill Roscoe, 3rd ed., 2005.

[127] Butterfield, A. and Woodcock, J., "A Hardware Compiler Semantics for Handel-C," *Proceedings of Third Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT)*, Dublin, Ireland, Aug. 2006, pp. 73–90.

[128] Butterfield, A. and Woodcock, J., "Semantic Domains for Handel-C," *Electronic Notes in Theoretical Computer Science*, Vol. 34, 2003, pp. 1–20.

[129] Butterfield, A., "A Denotational Semantics for Handel-C," *Formal Methods and Hybrid Real-Time Systems*, Vol. 4700, 2007, pp. 45–66.

[130] Butterfield, A. and Woodcock, J., "An Operational Semantics for Handel-C," *Electronic Notes in Theoretical Computer Science*, Vol. 80, 2003, pp. 1–16.

[131] CELOXICA, "Handel-C Language Reference Manual," Tech. Rep., RM-1003-4.3, 2006.

[132] Butterfield, A., Sherif, A., and Woodcock, J., "Slotted-Circus: A UTP-Family of Reactive Theories," *Integrated Formal Methods*, Vol. 4591, 2007, pp. 75–97.

[133] Wertz, J. R. and Larson, W. J., *Space Mission Analysis and Design*, Microcosm, El Segundo, CA, 3rd ed., 1999, ISBN 978-1881883104.

[134] Crane, R. K., *Electromagnetic Wave Propagation through Rain*, John Wiley & Sons, Inc., New York, 1996.

[135] Ippolito, L. J., *Radiowave Propagation in Satellite Communications*, Van Nostrand Reinhold Company Inc., New York, 1986.

[136] International Telecommunication Union, "Attenuation by Atmospheric Gases," Tech. Rep., Rec. ITU-R P.676-5, 2001.

[137] Zhang, W., Karhu, S. I., and Salonen, E. T., "Predictions of Radiowave Attenuations Due to a Melting Layer of Precipitation," *IEEE Transactions on Antennas and Propagation*, Vol. 42, No. 4, 1994, pp. 492–500.

[138] Kuwahara, T. and Lengowski, M., "Technical Note on Power Control and Distribution Unit," Tech. Rep. FLP-TN-05000-001-IRS, Institute of Space Systems, Stuttgart, Germany, Sept. 2008.

[139] Chang, Y., Lee, B., Choi, J., Yun, M., and Kang, S., "Rapid Initial Detumbling Strategy for Micro/ Nanosatellites Using a Pitch Bias Momentum System," *Transactions of the Japan Society for Aeronautical and Space Sciences*, Vol. 50, No. 167, 2007, pp. 63–69.

Acknowledgements

The Stuttgart small satellite program is a fantastic and outstanding academic satellite development program, which has been offering students opportunities in hand-on activities, scientific research, and engineering education. This doctoral research has been conducted within the scope of the first small satellite *Flying Laptop* project. It is my great honor that I could partake in this project as the systems engineer of the satellite. I express my sincere thanks to Professor Dr. Hans-Peter Röser for giving me this precious opportunity, for inviting me from abroad, for his trust in my work, and for his generous supervision of my doctoral research.

My cordial thanks go to all the participants of the *Flying Laptop* project. Especially, I thank Albert Falke, Georg Grillmayer, Michael Lengowski, Yasir Muhammad, and Sebastian Walz for their creative discussions and cooperative work, with whom I started my research together as a team. I also thank Mr. Manfred Hartling for his warm support in the field of electrical engineering. I also want to thank students, namely Alexander Uryu, Claas Ziemke, Marek Dittmar, Mario Kobald, Michael Fritz, Rouven Witt, and Quang Hoa Le who worked with me for their study and diploma theses under my supervision or as my research assistants. They achieved great contributions to the project. I thank Michael Lachenmann for his proofreading and reviewing of the manuscript of my dissertation.

I would like to thank Professor Dr. Rudolf Benz and Dr. Jens Eickhoff of EADS Astrium GmbH for their support in project management and in making technical decisions.

I would also like to thank Professor Dr. Tetsuo Yasaka for being the co-referee of my dissertation, who was the supervisor for my master thesis at the Kyushu University in Japan. Thank you very much for giving me valuable advice and feedback.

For financial support for a large part of my research activities, I thank the German Academic Exchange Service. Thanks to this support, my research could achieve fruitful accomplishments. I also thank the office of international affairs at the Universität Stuttgart for their kind support in lifestyle matters.

I would like to extend my thanks to my family, especially to my parents and parents-in-law, who kindly understood my intentions to do my doctorate in Germany and have warmly supported me the whole time during the past over four years. I now feel very happy to have successfully lived up to their expectations.

Finally, I would like to take this opportunity to sincerely thank my wife Shuo, who has always stood by my side and has kept our life so colorful and wonderful even during times I needed to work intensively and until very late. It is also true that our newborn daughter gave me a great emotional power whilst I was finalizing my doctorate. Thank you very much, Umi.

Curriculum Vitae of Author

Personal data

Name:	Toshinori Kuwahara
Date of birth:	August 5, 1981
Place of birth:	Kumamoto
Nationality:	Japanese

Academic studies

since 10/2005	Scientific staff at the Institute of Space Systems, Universität Stuttgart Project member of the small satellite Flying Laptop project
04/2000 – 03/2005	**Master of Engineering, Kyushu University, Fukuoka, Japan** Master Thesis at the Space Systems Dynamics Laboratory within the scope of the small satellite program “QTEX”: “Development of a low shock separation mechanisms for small satellites and application for the Micro-tether-satellite QTEX” Early entry into Kyushu University graduate school via an advanced placement examination (Bachelor study absolution) Participation in an experimental small satellite project “CANSAT” at Kyushu University and development of the attitude determination and control system and command and data handling system
03/2000	Chikushigaoka high school graduation

Internship

08/2002 – 09/2002	Internship at ShinMaywa Industries, Ltd. Flying Boats Division, Kobe, Japan

Educational activities

06/2007 – 08/2007	Participation in International Space University Summer Session Program, Beijing, China. Member and technical staff of the project “On-orbit servicing concepts, technology options and road-map”
08/2003	Small satellite launch experiment with the ARLISS (American Rocket Launch for International Student Satellites), Nevada, USA

Publications of Author

Reviewed articles

T. Kuwahara, F. Böhringer, A. Falke, J. Eickhoff, F. Huber, H.-P. Röser, “FPGA-based Operational Concept and Payload Data Processing for the Flying Laptop Satellite,” *Acta Astronautica*, Vol.65, No.11–12, 2009, pp. 1616–1627.

M. Fritz, A. Falke, T. Kuwahara, H.-P. Röser, S. Pearson, A. Witts, J. Eickhoff, “A Commercial Procedure Execution Engine Completing the Command Chain of a University Satellite Simulation Infrastructure,” *Acta Astronautica*, Vol.66, No.5–6, 2009, pp. 950–953.

T. Kuwahara, A. Falke, C. Ziemke, J. Eickhoff, H.-P. Röser, Y. Muhammad, “Development of a Hardware-in-the-Loop Simulation Environment on an MDVE for FPGA-based On-board Computing Systems,” *Transactions of Japan Society for Aeronautical and Space Science (JSASS) Space Technology Japan*, vol. 7, No. ists26, 2009, pp.Pf_1–Pf_9.

T. Kuwahara, A. Falke, H.-P. Röser, “Effective Project Management of Small Satellite Projects from the System Engineer’s Point of View, An Example of the Small Satellite Flying Laptop Project,” *Transactions of Japan Society for Aeronautical and Space Science (JSASS) Space Technology Japan*, vol. 7, No. ists26, 2009, pp.Pt_1–Pt_8.

Book contributions

T. Kuwahara, Team Doctor, “Developing On-Orbit Servicing Concepts Technology Options and Roadmap,” Final Report of Team Project at ISU Summer Session Program, Strasbourg, France: International Space University Publications, Aug. 2007.

T. Kuwahara, C. Ziemke, M. Fritz, J. Eickhoff, H.-P. Röser, “Asynchronous Parallel Reactive System for Intelligent Small Satellite On-board Computing Systems,” *Small Satellites for Earth Observation - Selected Contributions*, edited by R. Sandau, H.-P. Röser, A. Valenzuela, Springer, Berlin, Germany, 2009.

Conference contributions

T. Kuwahara, M. Dittmar, C. Ziemke, H.-P. Röser, “FPGA-based On-board Computers for Reconfigurable Computing on Space Systems,” 60th International Astronautical Congress (IAC-09-B4.6A.9), Daejeon, Republic of Korea, Oct. 12–16 2009.

T. Kuwahara, M. Lengowski, U. Beyermann, A. Uryu, H.-P. Röser, “Ka-band High-speed Communication Systems on Small Satellites for Future Advanced Communication Networks and Earth Observations,” 27th International Symposium on Space Technology and Science (2009-j-10), Tsukuba, Japan, June 6–10 2009.

T. Kuwahara, C. Ziemke, M. Fritz, M. Kobald, M. Dittmar, H.-P. Röser, “Programmability of FPGAs with the High-level Hardware Description Language Handel-C for Space Applications,” 27th International Symposium on Space Technology and Science (2009-f-09), Tsukuba, Japan, June 6–10 2009.

T. Kuwahara, C. Ziemke, M. Fritz, J. Eickhoff, H.-P. Röser, “Asynchronous Parallel Reactive System for Intelligent Small Satellite On-board Computing Systems,” 7th International Symposium on Small Satellite for Earth Observation, Berlin, Germany, May 2009.

T. Kuwahara, F. Böhringer, A. Falke, J. Eickhoff, F. Huber, H.-P. Röser, “Operational Design and On-board Payload Data Processing of the small satellite “Flying Laptop” with an FPGA-based On-board Computer System,” 59th International Astronautical Congress (IAC-08-B4.3.1), Glasgow, United Kingdom, Sept. 29 – Oct. 3 2008.

U. Kirchgäßner, T. Kuwahara, M. v. Schönermark, “Improved Land Surface Temperature Retrieval Method for the Small Satellite Flying Laptop,” 59th International Astronautical Congress (IAC-08-B1.4), Glasgow, United Kingdom, Sept. 29 – Oct. 3 2008.

A. Uryu, J. van der Ha, H.-P. Röser, T. Kuwahara, “QSAT Mission Analysis and Operation Plan Design,” 59th International Astronautical Congress (IAC-08-B4.3.7), Glasgow, United Kingdom, Sept. 29 – Oct. 3 2008.

T. Kuwahara, A. Falke, C. Ziemke, J. Eickhoff, H.-P. Röser, Y. Muhammad, “Development of a Hardware-in-the-Loop Simulation Environment on a MDVE for FPGA-based On-board Computing Systems,” 26th International Symposium on Space Technology and Science (2008-f-28), Hamamatsu, Japan, June 1-8 2008.

T. Kuwahara, F. Huber, A. Falke, M. Lengowski, S. Walz, G. Grillmayer, H.-P. Röser, “System Design of the Small Satellite Flying Laptop, as the Technology Demonstrator of the FPGA-based On-board Computer System,” 58th International Astronautical Congress (IAC-07-B4.6.08), Hyderabad, India, Sept. 24–28 2007.

M. v. Schönermark, H.-P. Röser, F. Huber, S. Walz, G. Grillmayer, M. Lengowski, A. Falke, T. Kuwahara, “Earth Remote Sensing with the Stuttgart’s Small Satellites,” Meteorologentagung 2007 (DACH2007-A-00010), Hamburg, Germany, Sept. 10-14 2007 (German).

T. Kuwahara, A. Falke, G. Grillmayer, H.-P. Röser, “Challenges of Small Satellites: New Technology Demonstration of Field Programmable Gate Array On-board Computing System,” International Space University Annual Conference 2007, Beijing, China, Aug. 3–4 2007.

T. Kuwahara, “Use of Advanced Ion-engine Vehicles to Reduce the Cost of On-orbit Servicing,” International Space University Annual Conference 2007, Beijing, China, Aug. 3–4 2007

T. Kuwahara, T. Yasaka, T. Hanada, H. Hirayama, Y. Sakamoto, T. Itahashi, “Kyushu University Micro-Satellite QTEX Project,” 55th International Astronautical Congress (IAC-04-IAA.4.11.5.06), Vancouver, Canada, Oct. 4–8 2004.

T. Kuwahara, “Kyushu University CANSAT 2003 Project Report,” University Space Engineering Consortium Workshop 2003 (UNISEC 03-20), Sapporo, Japan, Dec. 13–14 2003.