AN EXTENSIBLE SELF-ARBITRATING ARCHITECTURE FOR REDUNDANT ONBOARD COMPUTERS IN NANOSATELLITES

UDAI BINDRA

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE

YORK UNIVERSITY

TORONTO, ONTARIO

June 2021

ABSTRACT

This thesis explores the use of a novel N-modular cold redundancy architecture using Commercial-Off-the-Shelf onboard computers in Nanosatellites. The proposed architecture explores a software-based decentralization of the arbitration logic. This eliminates the need for an external supervising device to perform the arbitration amongst the onboard computers (OBC). Instead, the architecture is reliant only on a memory-less watchdog timer to reset the spacecraft power bus in case an OBC becomes unresponsive. Moreover, the architecture assesses the health of each redundant OBC and provides precedence to the critical capabilities of the OBC. Finally, the proposed redundancy architecture enables extensibility and is near platform agnostic, allowing reusability across missions that utilize different OBC platforms, without the need for major modifications in the arbitration mechanism.

The arbitration mechanism of proposed architecture was successfully validated in the lab by emulating faults using Raspberry Pi Zero W modules in triple modular redundancy. An additional OBC was added to demonstrate extensibility of the proposed redundancy architecture.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor George Zhu, for his continued mentorship & support throughout my tenure as a graduate student and for providing me with the opportunities to work on several amazing projects including the DESCENT CubeSat mission. I would also like to extend my acknowledgments to my supervisory committee, professor Franz Newland and professor Regina Lee, for providing valuable inputs to my research.

I would like to extend my acknowledgements to my colleagues: Latheepan Murugathsan, Jude Furtal, Gangqiang Li, Junjie Kang and Chonggang Du for their collaboration and support on the development of this work.

Finally, I would like to thank my parents for their unwavering support in all my endeavors and without whom, none of my achievements would have been possible.

TABLE OF CONTENTS

ABSTRACT… ii

ACKNOWLEDGEMENTS iii

TABLE OF CONTENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

LIST OF SYMBOLS xi

LIST OF ABBREVIATIONS xiv

CHAPTER 1 INTRODUCTION 1

1.1 Background 1

1.1.1 Ionizing Radiation and its Effects 3

1.1.2 Traditional Space Grade Onboard Computers 6

1.2 Motivation To Employ Commercial OBCs 8

1.3 Research Scope and Objectives 9

1.3.1 Research Scope 9

1.3.2 Research Objectives 10

1.4 Methodology 11

1.5 Thesis Structure 12

1.6 List of Publications 13

CHAPTER 2 LITERATURE REVIEW 15

2.1 Redundant OBC Architectures 15

2.1.1 Centralized Watchdog Timers 16

2.1.2 Decentralized Redundancy 18
2.2 Extensible Architectures 20
2.3 COTS Components In Space 20
2.4 Gaps Identified In Current Literature 21
CHAPTER 3 PLATFORM AGNOSTIC HEALTH ASSESSMENT 22
3.1 Module Health Metrics 22
3.1.1 Categorizing Health Metrics 23
3.2 Ranking The Health Metrics 24
3.2.1 Equally Ranked Boolean Metrics 25
3.2.2 Equally Ranked Non-Boolean Metrics 26
3.2.3 Equally Ranked Mixed Metrics 27
3.2.4 Health Metric Rank Indices 28
3.3 Evaluating The Health Quotient 28
3.3.1 Health Metric Precedence 32
3.3.2 Health Quotient Uniqueness 35
3.4 Considerations For Practical Implementation 40
3.4.1 Avoiding Arithmetic Overflows 40
3.4.2 Handling Health Metric Outliers 42
3.5 Summary 44
CHAPTER 4 EXTENSIBLE ARBITRATION ARCHITECTURE 45
4.1 Architecture Overview 45
4.1.1 The Arbitration Process 45
4.1.2 Spacecraft Bus Layout 49
4.1.3 State Machine Representation 49
4.2 Module Initialization Phase 51
4.2.1 The Bootup State 53
4.2.2 The Configuration State 53

4.3 Handshake Phase 55
4.3.1 The Role Determination State 59
4.3.2 The Process Supervisor Branch 60
4.3.3 The Redundant Module Branch 67
4.3.4 The Process Supervisor's Self Assess State 74
4.4 Health Assessment Phase 76
4.4.1 The Process Supervisor Branch 78
4.4.2 The Redundant Module Branch 82
4.5 Arbitration Resolution Phase 85
4.6 System Performance vs Scale 89
4.6.1 System Reliability 89
4.6.2 Multi-Domain Clock Deviation 93
4.7 Summary 97

CHAPTER 5 DESIGN IMPLEMENTATION AND VALIDATION 98

5.1 Ground Validation 98
5.1.1 Experimental Setup 98
5.1.2 Architecture Implementation 101
5.2 Test Cases 107
5.2.1 Nominal Operations 110
5.2.2 Unresponsive Redundant Module 113
5.2.3 Deterioration of the Health Quotient 116
5.2.4 Unresponsive Process Supervisor 119
5.2.5 Minimizing Module Switchover 121
5.2.6 Extensibility 126
5.3 The DESCENT Mission 130
5.3.1 Mission Overview 130
5.3.2 Dual Redundant OBC Architecture 133
5.3.3 Current Status 138

CHAPTER 6 CONCLUSIONS AND FUTURE WORK 141

6.1 Summary of Contributions 141

6.1.1 Self-Arbitrating Cold Redundancy Architecture 141

6.1.2 Precedence-Based Unique Health Quotients 142

6.1.3 Extensible Redundancy Architecture.....142

6.2 Conclusion.....143

6.3 Future Work 144

6.3.1 Potential Areas of Application.....144

6.3.2 Current Limitations and Future Improvements.....145

REFERENCES.....148

APPENDIX A: THE EFFECTS OF IONIZING RADIATION.....160

LIST OF TABLES

Table 4-1: Arbitration Phase Name Designations 50

Table 4-2: Module Role Designations 51

Table 5-1: Health Metrics Used in Ground Validation Tests 103

Table 5-2: Platform-Based Arbitration Parameters 106

Table 5-3: Arbitration Timeline Color Codes 108

Table 5-4: OBC Health Quotients for Nominal Test Case 110

Table 5-5: OBC Health Quotients for Unresponsive Redundant Module Test Case 113

Table 5-6: Health Metric Evaluations for Deteriorated Health Quotient Test Case 116

Table 5-7: OBC Health Quotients for Degraded Performance of Process Supervisor 117

Table 5-8: OBC Health Quotients for Unresponsive Process Supervisor Test Case 119

Table 5-9: OBC System Time at Start of Arbitration Cycles 124

Table 5-10: Health Metric Evaluations During Persistence Metric Test 124

Table 5-11: Using Persistence Metric in OBC Health Quotient Computation 126

Table 5-12: Updated Normalized Bounds for the Health Metrics 128

Table 5-13: OBC Health Quotient Values for the Extensibility Test Case 128

Table 5-14: Health Metrics Used Onboard DESCENT 135

Table 5-15: Configuration of Platform-Based Arbitration Parameters On DESCENT 136

LIST OF FIGURES

Figure 2-1: External Arbiter in a Cold Redundant Architecture [47] 16

Figure 4-1: The Arbitration Phases 46

Figure 4-2: Arbitration Process State Template 50

Figure 4-3: The Module Initialization Phase 52

Figure 4-4: The Handshake Phase 57

Figure 4-5: The Self Assess State of the Handshake Phase 59

Figure 4-6: The Health Assessment Phase 77

Figure 4-7: The Arbitration Resolution Phase 86

Figure 4-8: Arbitration Architecture Finite State Diagram 88

Figure 4-9: Reliability of a Dual Modular Redundant System 92

Figure 4-10: System Reliability for Proposed Redundancy Architecture 93

Figure 4-11: Maximum Allowable Clock Rate Deviation 96

Figure 4-12: Parallel Redundancy of Proposed Architecture 97

Figure 5-1: Block Diagram for Ground Test Setup 99

Figure 5-2: Abstraction of the Arbitration Logic 102

Figure 5-3: Ground Test Setup 109

Figure 5-4: Nominal Arbitration Cycle 111

Figure 5-5: Arbitration Cycle for Unresponsive Redundant OBC 115

Figure 5-6: Arbitration Cycle for Deteriorated Health Quotient of the Process Supervisor 118

Figure 5-7: Arbitration Cycle with Unresponsive Process Supervisor 120

Figure 5-8: Arbitration Cycle Utilizing Persistence Metric 125

Figure 5-9: Updated Block Diagram for Ground Test Setup..... 127

Figure 5-10: Nominal Arbitration Cycle Using Four Redundant OBCs 129

Figure 5-11: Concept Art for DESCENT 131

Figure 5-12: Mechanical Layout of the Mother Satellite..... 132

Figure 5-13: Mechanical Layout of the Daughter Satellite 132

Figure 5-14: Dual Redundant OBCs on PC-104 Form Factor Host PCB 133

Figure 5-15: The DESCENT Spacecraft..... 139

LIST OF SYMBOLS

δ_{max}	=	Maximum tolerance in clock rate deviation
η	=	Scaling factor between reliabilities of an OBC and an external arbiter
λ	=	Constant failure rate of a component
$b_{k(p)}$	=	Evaluation of the p^{th} Boolean health metric with rank index k
$g_{k(p)}$	=	Evaluation of the p^{th} non-Boolean health metric with rank index k
$h_{k(p)}$	=	Evaluation of the p^{th} health metric with rank index k
H	=	OBC Health Quotient
$H_{k(p)}$	=	Normalized Evaluation of the p^{th} health metric with rank index k
$H_{\langle k \rangle}$	=	Aggregate normalized evaluations for health metrics with rank index k
H_0	=	Module rank of the OBC
$N_{CycleCount}$	=	Number of consecutive incomplete arbitration cycles
$N_{EOPAcks\_R}$	=	Total End-of-Phase acknowledgements issued during Handshake phase
$N_{EOPMessages\_S}$	=	Total End-of-Phase messages issued during Handshake phase
N_H	=	Largest rank index of a health metric
$N_{HDSKRequests}$	=	Maximum number of handshake requests
$N_{HQRequests}$	=	Maximum number of health assessment requests
N_m	=	Total number of redundant OBCs
$N_{MaxCycleCount}$	=	Maximum number of consecutive incomplete arbitration cycles allowed
$N_{SyncMessages\_R}$	=	Total synchronization acknowledgements in Health Assessment phase

$N_{SyncMessages\_S}$	=	Total synchronization messages in Health Assessment phase
$\|\|N_k\|\|$	=	Number of health metric with rank index *k* or lower
R(t)	=	Component reliability over time
$s_{k(p)}$	=	Original step-size of the p^{th} health metric with rank index *k*
$\boldsymbol{S_k}$	=	Normalized step-size of the p^{th} health metric with rank index *k*
$T_{AssessmentExecution}$	=	Maximum time to execute the Assessment state
$T_{BaseValue\_R\_HDSK}$	=	Base value of redundant module watchdog in Handshake phase
$T_{BaseValue\_R\_HLTH}$	=	Base value of redundant module watchdog in Health Assessment phase
$T_{BaseValue\_S}$	=	Base value of the process supervisor watchdog in Handshake phase
T_{Delay}	=	Maximum execution delay for the Configuration state
T_{EOPAck_R}	=	Time required to broadcast an End-of-Phase acknowledgement
$T_{EOPInterval\_R}$	=	Interval between consecutive End-of-Phase acknowledgements
$T_{EOPInterval\_S}$	=	Interval between consecutive End-of-Phase messages
$T_{EOPMessage\_S}$	=	Time required to broadcast an End-of-Phase message
$T_{HealthAssessment\_S}$	=	Maximum time to execute process supervisor Health Assessment phase
T_{HDSK_R}	=	Redundant module watchdog timer in Handshake phase
T_{HDSK_S}	=	Process supervisor watchdog timer in Handshake phase
$T_{HDSKRequest}$	=	Time required to broadcast handshake request
$T_{HDSKRequestInterval}$	=	Interval between consecutive handshake requests
$T_{HDSKSync\_S}$	=	Execution time of the process supervisor Handshake Sync state
$T_{HDSKToHLTH\_R}$	=	Redundant module's Handshake to Health Assessment transition time

$T_{HDSKToHLTH\_S}$	=	Process Supervisor's Handshake to Health Assessment transition time
T_{HLTH_R}	=	Redundant module watchdog timer in Health Assessment phase
T_{HLTH_S}	=	Process supervisor watchdog timer in Health Assessment phase
$T_{HLTHSync\_S}$	=	Execution time of the process supervisor Health Assessment Sync state
$T_{HLTHToRESL\_R}$	=	Redundant module's Health Assessment to Resolution transition time
$T_{HLTHToRESL\_S}$	=	Process Supervisor's Health Assessment to Resolution transition time
$T_{HQBroadcast}$	=	Time required to broadcast health assessment
$T_{HQCompute}$	=	Time required to perform health assessment
$T_{HQRequest}$	=	Time required to broadcast health assessment request
$T_{k(p)}$	=	Total steps within the interval of the p^{th} health metric with rank index k
$T_{OUTCOME\_R}$	=	Redundant module watchdog timer in Resolution phase
$T_{OUTCOME\_S}$	=	Process supervisor watchdog timer in Resolution phase
T_{PDM}	=	Maximum time allowed to command the power distribution module
$T_{SyncMessage\_R}$	=	Time required to broadcast a synchronization acknowledgement
$T_{SyncMessage\_S}$	=	Time required to broadcast a synchronization message
$T_{SyncInterval\_R}$	=	Interval between consecutive synchronization acknowledgements
$T_{SyncInterval\_S}$	=	Interval between consecutive synchronization messages

LIST OF ABBREVIATIONS

BJT	=	Bipolar Junction Transistor
C&DH	=	Command and Data Handling
CME	=	Coronal Mass Ejection
COTS	=	Commercial-off-the-Shelf
DDD	=	Displacement Dose Damage
EDAC	=	Error Detection and Correction
EPS	=	Electrical Power System
ESD	=	Electrostatic Discharge
GCR	=	Galactic Cosmic Rays
FPGA	=	Field Programmable Gate Array
FPU	=	Floating Point Unit
FPV	=	Functional Performance Value
GPIO	=	General-Purpose Input/Output
IEEE	=	Institute of Electrical and Electronics Engineers
MCU	=	Microcontroller Unit
MOSFET	=	Metal Oxide Semiconductor Field Effect Transistor
OBC	=	Onboard Computer
PDM	=	Power Distribution Module
QML	=	Qualified Manufacturers List
QPL	=	Qualified Parts List
SAA	=	South Atlantic Anomaly

SEB	=	Single Event Burnout
SEE	=	Single Event Effect
SEGR	=	Single Event Gate Rupture
SEL	=	Single Event Latchup
SEP	=	Solar Event Particle
SET	=	Single Event Transient
SEU	=	Single Event Upset
SoC	=	System on Chip
SOI	=	Silicon on Insulator
SOS	=	Silicon on Sapphire
TID	=	Total Ionizing Dose
WDP	=	Watchdog Processor
WDT	=	Watchdog Timer

CHAPTER 1 INTRODUCTION

Summary: This chapter presents the motivation for this research and defines its research scope and objectives. A summary of the proposed methodology to achieve these objectives is also presented.

1.1 BACKGROUND

Satellites experience a harsh launch environment and operate in an equally unforgiving environment. The mechanical stresses experienced by onboard electronics are primarily from the launch loads and thermal fatigue. However, the biggest challenges the satellite systems face in orbit are from the interactions between the semiconductor devices and incident radiation or particles. All of these factors can cause varying degrees of mission failure, from bit flips & electrical surges that result in data corruption to mechanical and/or thermal fatigue that can cause complete component failure [1] [2] [3] [4]. To prevent these failures, systems engineers strive to build a robust and resilient spacecraft bus by employing proven spaceflight technologies and often reinforcing them with numerous redundancies. However, these space grade components impose their own unique constraints on the design and development of a mission. Due to their specialized fabrication requirements and the fact that they reside in a very niche market, these components are very expensive, have long acquisition lead times (from component order till delivery), and the technologies themselves often lag behind commercially available alternatives [4] [5]. Spacecraft Onboard Computers (OBC) are no exception to this limitation of using space grade components. In fact, their criticality to the mission and the complexity of their design means that the space grade

OBCs often feature many more protection mechanisms and redundancies compared to other spacecraft systems, which further drives up their costs.

The various protection mechanisms and redundancies integrated into space grade OBCs are crucial for many applications such as communication satellites, global navigation systems and human space flight. However, the benefits of such failsafe mechanisms do not patently outweigh the aforementioned limitations, especially for small form factor missions such as NanoSats and MicroSats. These platforms primarily serve university teams and smaller research groups who have limited financial resources and constrained timelines for mission development. One potential solution for such platforms could be the use of Commercial-off-the-Shelf (COTS) OBCs. Although these components do not feature the various protection mechanisms built into space grade OBCs, they do overcome the programmatic challenges faced by many small-scale missions. Moreover, the COTS OBCs often feature up to date processors providing them with a performance advantage over their space grade counterparts.

The focus of this research is the development of a novel self-arbitrating redundant OBC architecture to enable the use of COTS OBCs in space. Redundancy is a widely utilized logical radiation hardness technique that can offset the lack of space heritage in COTS OBCs. This chapter briefly discusses the challenges posed by ionizing radiation as well as the processes used to address these challenges & qualify components for spaceflight. Subsequent sections of this chapter explore the motivation for this research in greater detail and expand on the significance of demonstrating the proposed architecture using OBCs instead of other sub-systems on the spacecraft bus. The latter sections of this chapter define the scope of this research, formalize the objectives for the proposed architecture, and briefly discuss the methodology for designing the proposed

architecture. The chapter concludes with a brief layout of this thesis and provides an overview of the subsequent chapters.

1.1.1 Ionizing Radiation and its Effects

Perhaps the most challenging issue faced by spacecraft OBCs is ionizing radiation, even though it is not the only threat posed by the space environment. As mentioned before, mechanical and/or thermal fatigue may also lead to component failure. However, ionizing radiation is a non-trivial issue as it can degrade and damage electronics through numerous mechanisms. The different mechanisms by which ionizing space radiation affects the functionality and performance of semiconductor devices onboard spacecrafts will be discussed here.

Incident ionizing radiation deposits large amounts of energy into the semiconductors and this interaction results in system performance degradation. There are three main sources of ionizing radiation for satellites in Earth orbit, namely Galactic Cosmic Rays (GCR), Solar Energetic Particles, and trapped charges in the Van Allen Belts. Galactic Cosmic Rays (GCR) originate outside the solar system and primarily constitute heavy ions travelling at high velocities. GCRs are the most energetic form of ionizing radiation with particles reaching energy levels of up to 100 GeV. During active periods of solar sunspot cycle, the shielding effects of the Solar wind against GCR increases and the flux of GCR radiation experienced by satellites decreases. Solar Energetic Particles (SEP), also called Solar Proton Events, are particles generated during solar events. SEPs can be further classified into solar flares which contain heavy ions, and Coronal Mass Ejections (CME) which contain high energy electrons and protons. The radiation particles typically reach energy levels between 10 MeV and 1 GeV, and the frequency of solar events increases with an increase solar in the solar sunspot cycle. Lastly, the Van Allen Belts consist of two belts of charged particles around the Earth that are trapped by the Earth's magnetic field. The inner belt

extends roughly from one Earth radius out to three Earth radii above the Earth's surface. Trapped protons, with energies reaching up to 100 MeV, are the dominant source of ionizing radiation within the inner Van Allen Belt. Due to the non-uniformity of the Earth's magnetic field, the inner belt dips closer to Earth over a region known as the South Atlantic Anomaly (SAA). The SAA covers the majority of the South Atlantic Ocean and South American continent. Satellites transiting over the SAA in Low Earth Orbit experience a higher flux of radiation compared to other parts of their orbit at the same altitude. The outer Van Allen belt extends from roughly three Earth radii out to ten Earth radii above the Earth's surface and predominantly contains trapped electrons with energies up to 10 MeV [1] [3] [6] [7].

The manifestation of the degradation caused by the various types of ionizing radiation can be instantaneous or gradual, while its extent can vary from being a temporary interruption in mission operations to irreversible system damage. The effects of radiation of spacecraft electronics can be categorized into three groups, namely dose damage, single event effects and spacecraft charging. The different mechanisms of radiation damage that fall under these categories are briefly discussed here. A more detailed discussion of these effects along with a survey of missions that have experienced these affects is provided in Appendix A.

Total Ionizing Dose (TID) and Displacement Damage Dose (DDD) are the two mechanisms that fall under dose damage by which ionizing radiation can cause irreversible damage to spacecraft electronics. Dose damage manifests gradually, but its effects are permanent. Total Ionizing Dose (TID) is the accumulated radiation that is absorbed by any electronic device onboard the spacecraft over its mission lifetime. Over time, the accumulated radiation increases the number of trapped charges in the semiconductor's gate oxide. This causes alterations in carrier concentrations and lifetimes, shifts the voltage threshold, and leads to current leakage [3] [8] [9].

Although the degradation is gradual, the may system eventually fail beyond recovery. Displacement Damage Dose (DDD) has similar effects to TID although the underlying mechanism is slightly different. Displacement damage is caused by protons, neutrons and high energy electrons that knock atoms out of the semiconductor lattice. The displaced atoms are trapped in interstitial spaces in the lattice increasing the lattice imperfections. As the lattice imperfections grow, the performance of the semiconductor degrades resulting in current leakage [3] [9] [10]. Radiation hardness levels for electronics are expressed as the total dose limits before the system begins to exhibit performance degradation. Space grade electronics that are radiation tolerant are typically rated for total doses between 100 to 1000 krad. In contrast, commercially available components are usually rated between 5 and 20 krad [4] [11].

Unlike dose damage, Single Event Effects (SEE) cause instantaneous change in the functionality of an electrical circuit of any spacecraft system. A Single Event Effect is a broad umbrella term used for a variety of soft errors and hard errors caused by a high energy particle travelling through the semiconductor. The particle leaves an ionized trail in the body of the semiconductor which causes sporadic off-nominal effects that maybe transient or permanent [6, 7, 10, 12]. SEEs such as Single Event Upsets (SEU), Single Event Transients (SET) and Single Event Latchups (SEL) cause transient changes in the hardware or software state of the device and are usually non-destructive. Power cycling the affected devices often resolves these affects. On the other hand, Single Event Burnouts (SEB) and Single Event Gate Rupture (SEGR) usually result in permanent component failure, however, it is worth noting that these effects are almost exclusively observed in power circuitry using high power Metal Oxide Semiconductor Field Effect Transistors (MOSFET) and Bipolar Junction Transistors (BJT).

The final category of ionizing radiation effects is spacecraft charging. Electrostatic Discharge (ESD) is a result of differential surface charging, internal charging of insulating material or high charge build up on the spacecraft relative to the ambient plasma. While radiation contributes to the build-up of this charge, other mechanisms such as photoelectric emission can also result in differential charging. The build-up of charge culminates in a sudden discharge arc across spacecraft surfaces or from the spacecraft to the ambient plasma. This sudden discharge dissipates large amounts of energy which can damage any electrical circuits that may be electrically coupled to the surfaces experiencing the discharge arc. There is also evidence showing that high velocity impacts with space debris can induce electrostatic discharge in spacecraft [13, 14]. This makes spacecraft electronics another potential victim to the ever-growing threat of space debris, which is not only capable of causing structural damage, but can also affect the electronics protected within the spacecraft chassis.

1.1.2 Traditional Space Grade Onboard Computers

In order to ensure reliable system performance in orbit, uniquely fabricated components are utilized in conjunction with system redundancies and flight qualification procedures. These systems usually wear the badge of space heritage i.e., the components and mechanisms implemented within these systems have been demonstrated as viable solutions for counteracting the harmful effects of the space environment. These components rely on a combination of component design qualification and system design qualification processes.

In component design qualification, the component vendor ensures that certain performance requirements are attained via standardized fabrication processes. There are several reliability programs that establish requirements for component performance and audit manufacturers that claim to conform with the established requirements. In the United Stated, the Qualified

Manufactures List (QML) and Qualified Parts List (QPL) are two such reliability programs. The Capability Approval is a reliability program similar to QML and is used throughout Europe [15]. Class V components under the QML are often marketed as space grade components [16]. Manufacturers often also implement physical radiation hardness mechanism to protect the system from the harmful effects of ionizing radiation. Physical radiation hardening techniques primary rely on the specialized fabrication processes & materials, and therefore typically undertaken by the component manufacturer. Common physical radiation hardening techniques include Silicon on Insulator (SOI) or Silicon on Sapphire (SOS) fabrication and spot shielding [17].

In system design qualification, mission engineers qualify and validate the overall system instead of the constituent components. This involves integrating system recovery mechanisms and backups. In the worst case when certain components of the system cannot be recovered, the design allows graceful degradation of the system which allows the spacecraft to continue executing mission operations. In regard to ionizing radiation, the system integrates logical radiation hardness techniques to preserve functional integrity. Logical radiation hardening focusses on counteracting the manifestation of the radiation effects, rather than mitigating the effects themselves. This requires feedback from the system being hardened to detect performance degradation. Since performance feedback can be added to the system after fabrication, it is possible for mission engineers to implement custom logical radiation hardening mechanisms. Common methods for building logical radiation hardness include using redundant systems, watchdog timers, Error Detection and Correction (EDAC) mechanisms, and immunity aware programming [17].

Traditional space grade OBCs heavily rely on component design qualification, utilizing QML/QPL certified radiation tolerant components. Protection mechanisms fabricated within the OBC include, but are not limited to, hardware memory scrubbers, latching current limiters and

watchdog timers. The specialized fabrication requirements for integrating these protection mechanisms leads to increased costs and long lead times for these components. Moreover, the capabilities of space grade OBCs must be demonstrated in orbit to achieve space heritage. This necessarily implies that space grade components will not feature the latest processor technologies available in the market.

1.2 MOTIVATION TO EMPLOY COMMERCIAL OBCS

Commercially available onboard computers provide several performance and programmatic advantages over their space grade counterparts.

CubeSat OBCs with space heritage typically cost $5,000 - $10,000 [18] [19] [20]. In contrast, commercial-off-the-shelf onboard computers are inexpensive, readily available, and provide several configurations within a cost range of $10 - $150 [21] [22] [23] [24] [25]. This helps increase the sparing philosophy for spacecraft components, which not only enables development of engineering models for the spacecraft systems, but also allows a greater number of team members to work in parallel with the flight hardware. The latter is of great importance for educational missions such as those developed by undergraduate and graduate teams where access to space grade components is usually limited to a few graduate students and professors. This not only limits the educational experience for majority of the team, but also dramatically slows down the development timeline.

COTS OBCs have faster component update cycles and feature faster processors and larger volatile & non-volatile memory modules. Commonly available CubeSat OBCs offer single core processors with clock speeds of less than 100 MHz [18] [19] [26], while COTS variants feature multi-core processors with clock speeds of up to 1.5 GHz [21] [22] [23] [24] [25].

Owing to limited mass and volumetric budgets available in CubeSats, their OBCs have very small form factors [18] [19] [20] [26] [27]. COTS OBCs provide an advantage here as well, as their form factors are much smaller than CubeSat standard OBCs [21] [22] [23] [24] [25]. This saving in mechanical budgets allows integration of multiple redundant COTS OBCs within small form factor satellites while occupying similar mass and volume as their space grade counterparts.

However, commercially available OBCs do not feature any of the protection mechanisms or radiation hardness that allow space grade OBCs to operate reliably in orbit. The biggest challenge of using commercially available components in space missions is their lack of lot traceability and product homogeneity [4] [16]. Although COTS components have been used on board spacecraft [12] [28], they typically constitute only a part of the onboard computer. The COTS components are often supplemented with radiation-hardened components and external arbiters [29] [30]. Moreover, these supplementary systems are tuned to the platform they are protecting i.e., switching to a different OBC or adding additional redundant modules requires major modifications to the redundancy architecture [31].

1.3 RESEARCH SCOPE AND OBJECTIVES

1.3.1 Research Scope

Deploying COTS OBCs in space requires a multi-faceted solution that addresses resilience across hardware and software. However, the most critical aspect to this solution is redundancy which allows failed OBCs to be swapped out for functional backup OBCs. This research explores the use of a novel N-modular cold redundant architecture of the spacecraft OBC. The proposed architecture utilizes software-based decentralization of the arbitration logic. This eliminates the need for an additional supervising device to perform OBC arbitration, and the architecture relies only on a memory-less watchdog timer to initiate the arbitration process in case the OBC in

command of the spacecraft stops responding. This thesis also explores strategies to make the redundant architecture scale agnostic, allowing reusability across missions without the need for major modifications to underlying arbitration logic.

Aspects of the proposed architecture implemented within flight software will themselves be prone to the effects of ionizing radiation. Fault isolation and recovery from software execution errors or data corruption requires the use of complex EDAC mechanisms and immunity aware programming techniques. However, the development of these mechanisms is beyond the scope of this research. In order to address software related faults, the proposed architecture can utilize existing design strategies to improve software resilience including, but not limited to, data redundancy, code execution duplication and watchdog timers [17] [32] [33] [34]. These mechanisms can be complemented with the proposed architecture to ensure its reliable execution, as well as that of the rest of the flight software. Moreover, common strategies such as spot shielding [17] [35] [36] and conformal coating can be used to increase hardware robustness [37] [38] [39].

1.3.2 Research Objectives

Given the scope of this research, the proposed architecture achieved the following objectives:

i. The architecture arbitrates between the redundant OBCs without the need for any external arbiter, except for a memory-less watchdog timer. The capability of this watchdog timer was limited to resetting the spacecraft power if an OBC becomes unresponsive.

ii. The architecture is capable of assigning precedence to individual health metrics. This health metric precedence addresses partial failures of an OBC that affect mission critical capabilities.

iii. The arbitration architecture is extensible, that is, the arbitration logic abstracts away the scale of the redundancy architecture. This extensibility trait also applies to the selection of health metrics, that is, the architecture is agnostic to the actual health metrics used for OBC health characterization.

1.4 METHODOLOGY

The proposed redundant COTS OBC architecture resembles N-modular cold redundant systems, with the exception that it does not have an external arbiter to monitor device health and swap control between the available OBCs. This eliminates the single point of failure found in existing cold redundant architectures. Instead of utilizing the external arbiter, the proposed architecture decentralizes the arbitration logic is decentralized within software. Each device quantifies its own health by aggregating various health metrics into a single unique health parameter, known as the OBC's Health Quotient. A novel strategy was proposed to normalize any health metric based on its relative priority amongst the list of all health metrics. The proposed strategy iteratively computes the normalized range for the health metrics, starting from the lowest prioritized metric. The process forces the step-size of any normalized health metric to be larger than the largest possible aggregate of all normalized health metrics with lower priorities. This ensures that metric priorities can be assigned independent of the original range or resolution of the health metric. The normalization of health metrics is setup such that larger values of a health

metric, and therefore of the OBC's Health Quotient, indicate a healthier state of the OBC. Each OBC is also assigned a unique module rank which serves as a precedence metric to resolve arbitration in the event that two or more redundant OBCs have equivalent state of health. The proposed methodology was designed to be agnostic to the health metrics themselves, allowing the exact metrics to be determined based on mission needs.

The proposed arbitration process for the computation and exchange of this Health Quotient defines one of two roles to each of the available OBCs, namely process supervisor or redundant module. Only one OBC is assigned the role of the process supervisor at any time, and it is responsible for arbitrating control of the spacecraft bus amongst all OBCs to allow them to compute their Health Quotients. Once an OBC computes its Health Quotient, it broadcasts this value to the other redundant modules for a comparative health assessment. All other OBCs are assigned the role of redundant module. This architecture design allows extensibility of the architecture, as any additional OBC added to the spacecraft bus will assume the role of a redundant module and execute the same logical functionality as existing redundant modules. The assignment of roles for the arbitration process is dynamic, and the process does not assume that any particular OBC (process supervisor or redundant module) will be functional when the arbitration process commences. Failsafe mechanisms were included to ensure that the arbitration process will resolve itself even in the event of complete or partial failure of one more OBCs. A proof-of-concept of the proposed architecture was tested in the lab to demonstrate its viability.

1.5 THESIS STRUCTURE

The following chapters of the thesis elaborate on the design of the proposed architecture and demonstrate its implementation. In all, there are six chapters in this thesis including the Introduction. Chapter 2 provides a comprehensive literature review on the use of redundant

onboard computers as well as the use of COTS components in space missions. The chapter also explores existing health characterization strategies for onboard systems. The next two chapters layout the details of the proposed architecture. Chapter 3 focusses on the characterization of OBC health and provides details on generating a unique precedence-based Health Quotient. Chapter 4 provides details on an arbitration mechanism that is architecture scale independent and allows computation and exchange of this Health Quotient. Chapter 5 details the ground-based validation tests conducted by deploying the proposed architecture using redundant Raspberry Pi Zero modules. Chapter 6 summarizes the work, highlights the contributions of this research and discusses the potential areas where the proposed architecture can be improved to expand its application scope.

1.6 LIST OF PUBLICATIONS

In addition to the research work performed to develop the proposed redundancy architecture, substantial work was done towards the design and development of a 2U tethered CubeSat mission called DESCENT. The redundancy architecture proposed in this thesis was implemented on this mission, and the constraints of implementing this redundancy architecture on an actual CubeSat platform were fed back to improve the design of the proposed architecture. Significant contributions were also made towards the development of a ground-based air-bearing testbed which was used to perform tether deployment tests for the DESCENT mission. The list of publications detailing this work is presented below.

1. Z. H. Zhu, J. Kang and **U. Bindra**, "Validation of CubeSat tether deployment system by ground and parabolic flight testing", *Acta Astronautica*, vol. 185, pp. 299-307, 2021.

2. **U. Bindra** and Z. H. Zhu, "Ground based testing of space tether deployment using an air bearing inclinable turntable", *International Journal of Space Science and Engineering*, vol. 4, no. 1, pp. 1-17, 2016.
3. L. Murugathasan, **U. Bindra**, C. Du, Z. H. Zhu, and F. T. Newland " A Software and Hardware Redundancy Architecture for Using Raspberry Pi Modules as Command & Data Handling Systems for the DESCENT Mission", in *Canadian Aeronautics and Space Institute ASTRO*, Quebec, 2018.
4. J. Kang, Z. H. Zhu, **U. Bindra**, L. Murugathasan, J. Furtal and G. Li, "Deployment Mechanism for DESCENT Mission", in *Canadian Aeronautics and Space Institute ASTRO*, Quebec, 2018.
5. V. Jain, **U. Bindra**, L. Murugathasan, F. T. Newland and Z. H. Zhu, "Practical Implementation of Test-As-You-Fly for the DESCENT CubeSat Mission", in *2018 SpaceOps Conference*, Marseille, 2018.
6. **U. Bindra**, L. Murugathasan, V. Jain, G. Li, J. Kang, C. Du, Z. H. Zhu, F. T. Newland, M. Alger, O. Shonibare, A. de Ruiter, "DESCENT: Mission Architecture and Design Overview", in *AIAA SPACE*, Orlando, 2017.
7. **U. Bindra** and Z. H. Zhu, "Development of an Air-Bearing Inclinable Turntable for Testing Tether Deployment", in *AIAA Guidance, Navigation, and Control Conference*, San Diego, 2016.
8. **U. Bindra** and Z. H. Zhu, "Experimental Validation for the Deployment Behaviour of Orbiting Tethers Using an Air-Bearing Turntable", in *5th International Tethers in Space Conference*, Michigan, 2016.

CHAPTER 2 LITERATURE REVIEW

Summary: This chapter presents the literature review on existing redundant OBC architectures. The chapter also summarizes the use of COTS components in spacecraft command and data handling systems. Finally, the gaps identified in current research & development, which are addressed in this thesis, have been summarized.

2.1 REDUNDANT OBC ARCHITECTURES

Redundant architectures can be broadly categorized as hot, warm, or cold redundant. In a hot redundant system, all backup OBCs operate in parallel with one primary device and concurrently generate output commands [30] [40] [41]. An independent voting mechanism is used to mask faults occurring on any one of the redundant devices prior to issuing the commands. Warm redundant architectures are similar to hot redundant architecture, and also have the backup devices powered up in parallel to the primary. However, the backup devices in warm redundant architectures do not generate any command outputs [42] [43]. This reduces the complexity of the system as external voting circuitry is not required. Finally, cold redundant architectures keep all backup devices powered down until the active module experiences a failure mode [41] [42] [43]. This reduces operational stress on the redundant devices and tempers the effects of Total Ionizing Dose (TID) [8] [9], while also limiting the overall power budget requirements. Cold redundant architectures are therefore ideal for small form factor satellites.

Redundant architectures utilize watchdog timers (WDT) that guard against off-nominal behavior by resetting the OBC or the spacecraft power if the timer overflows [44] [45]. These timers are required in single OBC architectures as well, to recover from SEEs or software loops.

Complex redundant OBC architectures require additional logic implemented on an external arbiter that can not only reset the system, but also arbitrate control between the redundant modules [46]. Figure 2-1 shows a high-level block diagram of such an architecture, modified from [47].

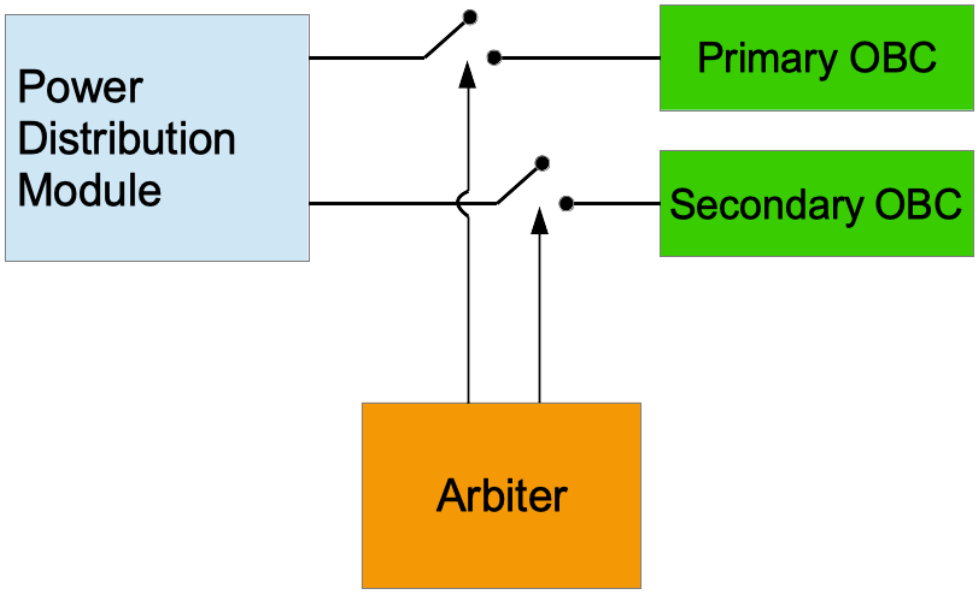

Figure 2-1: External Arbiter in a Cold Redundant Architecture [47]

2.1.1 Centralized Watchdog Timers

A commonly used strategy to arbitrate between redundant OBCs is to use heartbeat signals generated by the OBC to reset an external arbiter. If the active OBC experiences a fault, then the heartbeat signal stops and the watchdog timer either resets the OBC or switches to the backup [48] [49] [50]. The reliability of the redundant architecture can be improved by employing hierarchal watchdogs [51]. In this architecture, an external supervising device such as a watchdog processor (WDP) arbitrates between the redundant OBCs, and an independent watchdog timer monitors the

watchdog processor to ensure its nominal operations. This removes the single point failure of external watchdogs from the redundant systems.

The ZDPS-1A CubeSat [52] [53] implemented dual modular OBCs in warm redundancy. The architecture used an FPGA to route control of the spacecraft bus between the OBCs. An independent watchdog timer monitors the active OBC and initiates an autonomous switchover to the redundant module in the event that the primary module fails.

A similar architecture of dual modular OBCs in warm redundancy was implemented on the UWE-3 CubeSat [53] [54]. Two independent analog isolation switches, one connected to each OBC, were used to control access to the spacecraft bus. The isolation switches are monitored by a Toggle Watchdog Unit (TWU) that activates the redundant module via a flip-flop circuit in the event of a malfunction. Additionally, a cascaded-timeout watchdog implementation is used on the power unit to power cycle the toggle watchdog unit.

The ESTCube-1 CubeSat [55] [56] implemented dual modular Microcontroller Units (MCU) in cold redundancy. The control of the spacecraft bus was toggled between both MCUs by the spacecraft Electrical Power System (EPS). Independent isolation switches were used for each peripheral to isolate the MCUs from the spacecraft bus.

The X-Sat microsatellite [30] also implements a dual modular OBC in cold redundancy. The active OBC sends heartbeat signal to the spacecraft power supply unit that maintains a watchdog timer. If the OBC fails to reset the watchdog timer, the power supply unit resets the active OBC. After a pre-determined number of resets, the power supply unit switches to the backup OBC.

Most redundant architectures that integrate centralized watchdog timers focus on a dual modular architecture and require major modifications to scale up the architecture and

accommodate a greater number of redundant OBCs. Moreover, these implementations do not perform any quantitative health analysis of the OBCs. Instead, the watchdog timers perform a switchover to the redundant module as soon as the primary OBC hangs-up. This method results in an "all or nothing" performance characterization of the OBC and cannot determine the relative ranking of OBC state-of-health if there are more than two redundant OBCs. Spacecraft health monitoring strategies that can provide a wholistic characterization of the OBC based on various functional capabilities typically utilize machine learning and data mining techniques [57] [58] [59]. An arbitration mechanism using an instantaneous state-of-health metric that is indicative of the OBC's functional capabilities has not been explored. Such a mechanism will also allow definition of quantifiable health metrics that can be used to assess the functional capabilities of all OBCs before performing a switchover.

2.1.2 Decentralized Redundancy

The BEESAT-1 CubeSat [53] [60] employed dual modular OBCs in cold redundancy. Each of the OBCs is connected to the entire spacecraft bus over a redundant pair of Controller Area Network (CAN) busses, such that both OBCs can command the entire spacecraft. This eliminates the need for external arbiters; however, the architecture limits all sub-systems on the spacecraft to the CAN interface.

The Delfi-C^3 [31] [61] and Delfi-n3xt [31] CubeSats utilized autonomous sub-system design instead of a redundant OBC architecture. Both missions only employed a single OBC. On Delfi-C^3, each sub-system on the spacecraft monitors OBC activity. In the event of a timeout, the sub-systems are capable of performing mission critical tasks autonomously. On Delfi-n3xt, the spacecraft transceiver's telecommand decoder was equipped with critical OBC functionality. This functionality was executed by a microcontroller placed at the transceiver. A similar autonomous

sub-system architecture was implemented on the SNAP-1 [62] mission. A TTC node was implemented on a microcontroller integrated into the spacecraft's S-Band transceiver. The microcontroller was capable of polling critical health telemetry from other sub-systems on the spacecraft.

All of these architectures remove single point of failures from the Command and Data Handling (C&DH) unit of the spacecraft. However, the spacecraft functionality is reduced to its critical operations in all cases.

Zhu and Yu [40] demonstrated decentralization of the supervising unit hardware in a dual modular hot redundant system. Each OBC is connected to an independent Field Programmable Gate Array (FPGA). Each FPGA implements the arbitration logic and is capable of performing a transfer of control from the active module to the standby module. Fayyaz, Vladimirova, and Caujolle [29] demonstrated a similar decentralization of the arbitration logic in a distributed computing scheme. Each OBC is connected to an Adaptive Middleware for Fault Tolerance (AMFT) implemented on an FPGA. The AMFTs run membership algorithm for the OBCs. Each AMFT monitors its host OBC for faults in the processor, watchdog or memory. Once a fault is detected, the ID of the faulty OBC is propagated throughout the network of distributed computers through their respective AMFT module. However, both these decentralization techniques require specialized hardware connected to the OBCs.

These methods for decentralizing the arbitration architecture either limits the choice of the OBC platform, limits its functionality, or requires specialized hardware to support decentralization of the arbitration logic. A solely software-based decentralization of the arbitration mechanism that does not require specialized hardware (other than a memory-less watchdog timer) will be explored

in this thesis. Moreover, the arbitration mechanism will not utilize strategies that trade-off decentralization for limited functionality of the spacecraft.

2.2 EXTENSIBLE ARCHITECTURES

The discussion in literature of scalable architectures for Nanosatellite OBCs is limited. The X-Sat microsatellite [30] enabled scalability for the number of memory banks used by the spacecraft OBC, however, this scalability was not extended to the number of OBCs itself. The fault tolerant architecture proposed by Fayyaz, Vladimirova, and Caujolle [29] is size agnostic; however, it requires integration of an FPGA for every OBC that implements the arbitration logic via middleware.

2.3 COTS COMPONENTS IN SPACE

The use of COTS components in spacecraft sub-systems has drastically increased in recent years, enabling cost-efficient and rapid development of space missions [63].

The ESTCube-1 spacecraft [56] primarily used automotive or industrial grade COTS components for development of several onboard sub-systems. The spacecraft was launch in May 2013, and in-orbit operations of the spacecraft demonstrated the viability of these components in space. However, infant mortality of COTS components was recognized as an issue, which supports the use redundant components. The SNAP-1 mission [62] was developed with objective of using COTS components for all of its core sub-systems including the OBC, power and communications modules. The mission was launched in June 2000 and demonstrated successful operations in orbit. The X-Sat mission [30] utilized COTS memory banks in its dual redundant OBC architecture. The AAReST mission [28] proposed the use of dual redundant Raspberry Pi modules on the CubeSat.

These works demonstrate the increasing shift towards COTS components. The architecture presented in this thesis will be a stepping-stone in this direction and will broaden the use case of COTS OBCs by developing an extensible and near platform agnostic variant of the existing redundancy architectures.

2.4 GAPS IDENTIFIED IN CURRENT LITERATURE

The following gaps have been identified in existing literature on redundant OBC architectures:

1. Current arbitration mechanisms rely on the OBC to execute internal health assessments and only send heartbeat message to the external arbiter. An arbitration mechanism based on the comparative health assessment of all available OBCs using weighted health metrics has not been explored.
2. De-centralized arbitration mechanisms that do not rely on a single external arbiter utilize specialized hardware such as FPGAs to support the de-centralization. A de-centralized arbitration architecture that does not require any specialized hardware to execute the arbitration logic has not been explored.
3. Arbitration architectures utilized in nanosatellites are purpose built for each mission. A generalized architecture that supports extensibility and is near platform agnostic has not been explored.

This research addresses the aforementioned gaps identified in current literature.

CHAPTER 3 PLATFORM AGNOSTIC HEALTH ASSESSMENT

Summary: A novel strategy to aggregate the OBC health metrics into a unified health metric called the OBC Health Quotient is presented in this chapter. The discussion highlights how this strategy provides precedence to individual health metrics, while ensuring that the aggregated Health Quotient is unique for each OBC. Lastly, the limiting factors for computing the Health Quotient on an actual onboard computer are presented, along with suggestions to help circumvent these limitations.

3.1 MODULE HEALTH METRICS

As the complexity of the spacecraft bus and the system requirements for the onboard computer increase, the number of health assessments for an OBC will also increase to ensure that the module is capable of executing mission operations. Any such assessment is defined as a health metric. The exact metrics that must be evaluated to ensure that a particular OBC is capable of commanding the spacecraft bus will differ for each mission and will depend on the design of the spacecraft bus as well as the requirements for the command and data handling system. The following properties are defined for the evaluation result of any health metric:

P 3-1. A health metric shall evaluate to a Real scalar value.

P 3-2. A single health metric shall not evaluate to a vector of multiple values. If the assessment of any performance or functional capability for an OBC requires a vector of multiple values, it must either be reduced to a single equivalent health metric or each of those values must be defined as an independent health metric.

P 3-3. An optimal health assessment shall correspond to the maximization of the metric's evaluation.

P 3-4. The interval for a health metric is defined as all possible values for the metric across all available onboard computers. The lower bound and upper bound of this interval shall not be the same.

Properties (P 3-1) and (P 3-2) ensure that all health metrics are scalars that can be directly aggregated together into an onboard computer's Health Quotient. Property (P 3-3) implies that the healthiest available module will have the largest value of the Health Quotient. Lastly, Property (P 3-4) is used to demonstrate that each module computes a unique Health Quotient (discussed in section 3.3.2), therefore avoiding any ambiguity in picking the healthiest available onboard computer.

3.1.1 Categorizing Health Metrics

3.1.1.1 Boolean Health Metrics

Boolean health metrics are module health evaluations that always yield a binary result, indicating whether the module passed the evaluation for a given metric. Boolean health metrics may include, but are not limited to, computation validity, software execution validity, state configuration validity, or communication & memory interface functionality. A Boolean health metric *b* can be expressed as:

$$b = \begin{cases} 0, & evaluation\ failed \\ 1, & evaluation\ passed \end{cases} \tag{3.1}$$

3.1.1.2 Non-Boolean Health Metrics

Non-Boolean health metrics can evaluate to any real scalar quantity and provide granular information about the performance of an OBC with respect to the given metric. Non-Boolean health metrics may include metrics that compliment Boolean health metrics such as the frequency

of successful read/write operations on a communication or memory interface. Non-Boolean health metrics may also include independent metrics such as the amount of time the module has previously spent commanding the spacecraft bus or the amount of mission data stored onboard the module. The latter assists in providing precedence to an OBC if all available modules have identical health assessments for their capabilities.

3.1.1.3 Module Rank Metric

At mission epoch (first on-orbit power up), it is unlikely that any of the OBCs has experienced a failure mode i.e., the aggregate of all health metrics will result in the same Health Quotient. An immutable specialized rank metric is used to provide a unique rank for each available module and is used to establish a ranking order between all available OBCs. Note that even though the value of this metric is fixed for each module, property (P 3-4) implies that, on a spacecraft with N_m redundant onboard computers, the module rank metric $\in [1, N_m]$. If all modules are in identical states, property (P 3-3) dictates that the onboard computer with module rank of N_m will assume control of the spacecraft bus.

3.2 RANKING THE HEALTH METRICS

Every mission has a set of primary mission requirements that are valued higher than mission goals. The precedence of certain mission requirements over others is carried over to the spacecraft sub-systems involved in meeting those requirements. The onboard computer will likely be part of most mission requirements and goals; however, the precedence of specific mission requirements implies that certain capabilities of an onboard computer will be valued more than others. This can be reflected in the module's Health Quotient by ranking each of the selected health metrics.

Each health metric is assigned a whole number rank index which indicates its precedence over other health metrics. The least valued metric is assigned a rank index of 1 and the rank indices are incremented sequentially for higher valued health metrics. More than one health metric maybe assigned the same rank index if two or more OBC functionalities are equally valued. However, the rank indices must always increment by 1, even after assigning the same rank index to multiple equally ranked health metrics. Finally, the module rank metric is always assigned a rank index of 0 and is the only metric with this rank index.

It is possible to reduce the equally ranked metrics into a single equivalent health metric. This dataset reduction is useful for certain groupings of health metrics, especially if a large number of health metrics are being considered. However, it requires a tradeoff with computational resources of the onboard computer required to process these health metrics. This trade-off is discussed in section 3.4 and should be considered when deciding whether to reduce the dataset of equally ranked health metrics. The methodology for reducing various combinations of health metrics into a single equivalent health metric are presented in the following section, along with a discussion on when such dataset reductions are useful. The equivalent health metric retains the same rank index as its constituent health metrics' rank index.

3.2.1 Equally Ranked Boolean Metrics

Reducing two or more equally ranked Boolean health metrics is useful when the health metrics are either conjunct (all equally ranked metrics must pass their respective evaluations) or disjunct (sufficient for any one of the equally ranked metrics to pass its evaluation).

In both these cases, the health metrics can be combined into a single equivalent Boolean health metric. Let $b_{k(p)}$ be the evaluation result of an equally ranked Boolean health metric. The subscript $k(p)$ denotes the p^{th} equally ranked Boolean metric with a rank index of k. The equally

ranked metrics are always evaluated independently. In the case where design requirements dictate that all metrics must pass their respective evaluations i.e., the metrics are conjunct, logical conjunction operations are performed on the evaluation results of all such metrics. For a set of $m > 0$ equally ranked Boolean health metrics with a rank index of k, the equivalent Boolean health metric b_k can be expressed as:

$$b_k = b_{k(1)} \wedge b_{k(2)} \ldots \wedge b_{k(m)} \tag{3.2}$$

Similarly, in the case where it is sufficient for any one of the metrics to pass its evaluation, logical disjunction operations are performed on the evaluation results of all such health metrics. In this case, the equivalent Boolean health metric b_k can be expressed as:

$$b_k = b_{k(1)} \vee b_{k(2)} \ldots \vee b_{k(m)} \tag{3.3}$$

It is also possible to apply both logical operations within the same set of equally ranked Boolean metrics.

3.2.2 Equally Ranked Non-Boolean Metrics

Logical conjunction or logical disjunction operations cannot be performed to combine metrics with non-Boolean health metric evaluations. In fact, the metrics may have non-integer values and each metric might lie within a completely different interval. Taking the average of these metrics to reduce the dataset will skew the equivalent health metric and will always be dominated by changes in health metrics that have larger raw values. For instance, the amount of data stored on a 1 gigabyte non-volatile memory module could be measured in megabytes within the interval [0, 1024] or in kilobytes within the interval [0, 1048576]. In contrast, a metric measuring the ratio of the total successful read/write operation to the total number of attempted read/write operations in memory would always be bounded within the interval [0, 1]. Any fractional changes to this metric would pale in comparison to the metric measuring onboard data storage.

Considering that each metric may lie within unique intervals with unique step sizes, the unweighted geometric mean of the metrics can be used to compute an equivalent health metric. This reduction of equally ranked non-Boolean metrics into a single equivalent health metric is useful when relative change of the metric values is being considered. In other words, scaling any of the individual metrics by the same constant should reflect the same change in the value of the equivalent reduced health metric. Let $g_{k(p)}$ be the evaluation result of an equally ranked non-Boolean health metric. As before, the subscript *k(p)* denotes that it is p^{th} non-Boolean health metric with a rank index of *k.* For a set of $m > 0$ equally ranked non-Boolean health metrics with a rank index of *k*, the equivalent non-Boolean health metric g_k can be expressed as:

$$g_k = \left[\prod_{p=1}^{m} g_{k(p)} \right]^{\frac{1}{m}} \tag{3.4}$$

Prior to computing the geometric mean, all health metric evaluations must be normalized to values greater than 0. A health metric value of 0 will drive the entire geometric mean to 0, while any value less than 0 may result in imaginary values for the geometric mean.

3.2.3 Equally Ranked Mixed Metrics

As noted in section 3.1.1.2, non-Boolean health metrics may provide supplemental granular information for their corresponding Boolean health metrics. For instance, a Boolean health metric may indicate the functional capability of a communication interface while a corresponding non-Boolean health metric may indicate its performance capability as a measure of bit error rates on the interface. It is useful to reduce these corresponding mixed metrics by using the Boolean health metric as a weight for the non-Boolean health metric. For a Boolean health metric b_k and an equally

ranked non-Boolean health metric g_k, both with a rank index of k, the equivalent mixed metric M_k can be expressed as:

$$M_k = b_k \times g_k \tag{3.5}$$

Using this health metric reduction implies that the evaluation of the performance metric g_k is irrelevant if an OBC fails its corresponding capability metric evaluation b_k. Extending this to the example of the communication interface, if the interface itself is not functional then the bit error rate on that communication interface is irrelevant.

3.2.4 Health Metric Rank Indices

The smallest assignable rank index for the health metrics is 0 and is assigned only to the module rank metric. Each equally ranked health metric with the same rank index can be grouped into a set. The largest rank index that can be assigned within the overall list of all health metrics depends on the number of sets of equally ranked metrics, as well as the total number of metrics within each of those sets. Consider a total of n_H health metrics after all dataset reductions for equally ranked health metrics (excluding the module rank metric), with m_H health metrics that do not have a unique rank index and that have been grouped into k_H sets. Each of the k_H sets contains health metrics with the same rank index. In this case, N_H denotes the largest rank index that can be assigned to any health metric. N_H is evaluated as:

$$N_H = n_H - m_H + k_H;\ N_H \in [1,\ n_H] \tag{3.6}$$

3.3 Evaluating The Health Quotient

Once the health metrics have been selected and ranked, they are aggregated into a unified health indicator known as the module's Health Quotient. The challenge with simply aggregating or computing an average of all health metrics was discussed in section 3.2.2. This issue is even

more apparent when computing the Health Quotient as the health metrics with higher ranks may have smaller raw values. To address this issue, each health metric is normalized to a unique interval based on its assigned rank index. Once the dataset reductions have been performed on all equally ranked metrics, the health metrics are normalized to according to the following properties:

P 3-5. The lower bound of the normalized range for all health metrics must be 1.

P 3-6. Let P_k denote the difference between the aggregate of the upper bound and the aggregate of the lower bound of all the normalized metrics with a rank index less than *k*. The step size of the normalized range for a health metric with rank index $k > 0$ must always be greater than P_k.

P 3-7. The module rank metric always lies within the interval [1, N_m], where N_m is the total number of redundant onboard computer modules. Each OBC is assigned a unique rank and the step size of this metric will always be 1.

The rest of this section will demonstrate how these properties are utilized to generate a Health Quotient that provides precedence to health metrics based on their rank index. The discussion will also show that the Health Quotient is unique for each of the redundant OBCs, ensuring that a comparative health assessment generates an unambiguous result.

Min-max normalization is utilized to normalize a metric within a given interval. Consider an arbitrary health metric with a value *r* such that $r \in \mathbb{R}\ and\ r_{min} \leq r \leq r_{max}$. The metric value *r* can be normalized to $R \in [R_{min}, R_{max}]$ as follows:

$$R = R_{min} + \frac{(r - r_{min})(R_{max} - R_{min})}{r_{max} - r_{min}} \tag{3.7}$$

A health metric may or may not have a unique rank index. Let $h_{k(p)}$ be the original evaluation of the p^{th} metric with a rank index *k* such that $h_{k(p)} \in [{h_{k(p)}}^{min}, {h_{k(p)}}^{max}]$. Within this

interval, let the step size of this metric be $s_{k(p)}$. After normalization, let the value of the metric be $H_{k(p)} \in [{H_k}^{min}, {H_{k(p)}}^{max}]$ and let the new step size of the normalized metric be $\boldsymbol{S_k}$. It should be noted that all equally ranked health metrics will have the same minimum value and the same step size for their normalized interval (from properties (P 3-5) and (P 3-6) of normalization). However, the resolution and maximum value for each equally ranked health metric may differ, since it depends on the original interval and step size of that metric.

A new notation $H_{\langle k \rangle}$ is introduced to represent the aggregate of all equally ranked normalized health metrics with the rank index *k*. Consider a set of *m* equally ranked health metrics with a rank index of *k*. $H_{k(p)}$ represents the normalized value of the p^{th} health metric with a rank index *k*, where $p \in \mathbb{N}\ and\ 1 \leq p \leq m$. $H_{\langle k \rangle}$ can be expressed as:

$$H_{\langle k \rangle} = \sum_{p=1}^{m} H_{k(p)} \tag{3.8}$$

This notation is used to generalize the normalization strategies and subsequent proofs discussed in this chapter. For health metrics with unique rank indices, the value of *m* would evaluate to 1.

The limits of this new normalized range can be computed based on the normalization rules described above. The lower bound of the new interval is given by property (P 3-5) of normalization:

$${H_k}^{min} = 1$$

Since ${H_k}^{min}$ always evaluates to 1, ${H_{\langle k \rangle}}^{min}$ will evaluate to the number of health metrics with a rank index of *k*.

To compute the upper bound of the interval, the new step size of any metric with a rank index $k > 0$ must be computed within the new interval. Based on property (P 3-6) of normalization, the new step size $\boldsymbol{S_k}$ for a normalized metric with rank index $k > 0$ can be expressed as:

$$\boldsymbol{S_k} = \sum_{i=1}^{k-1} {H_{\langle i \rangle}}^{max} - \sum_{i=1}^{k-1} {H_{\langle i \rangle}}^{min} + 1 \tag{3.9}$$

As noted earlier, ${H_{\langle k \rangle}}^{min}$ evaluates to the number of health metrics with a rank index of k. It follows that the expression $\sum_{i=0}^{k} {H_{\langle i \rangle}}^{min}$ is the total number of health metrics with a rank index of k or less. This total is denoted by $\|N_k\|$. The expression for the step size $\boldsymbol{S_k}$ can be re-written as:

$$\boldsymbol{S_k} = \sum_{i=0}^{k-1} {H_{\langle i \rangle}}^{max} - (\|N_{k-1}\| - 1) \tag{3.10}$$

The step size for the module rank metric is always 1, as defined by rule (P 3-7) of normalization:

$$\boldsymbol{S_0} = 1$$

Using the step size for the normalized metric, the new upper bound of the metric can be scaled from its original range using the ratio of the normalized step size to the original step size:

$${H_{k(p)}}^{max} = \begin{cases} N_m, & k = 0 \\ {H_k}^{min} + \dfrac{\boldsymbol{S_k}}{s_{k(p)}} \left({h_{k(p)}}^{max} - {h_{k(p)}}^{min} \right), & k > 0 \end{cases} \tag{3.11}$$

The ratio of the metric's original range to its original step size provides the number of steps the original metric traverses between its minimum and maximum value. The number of steps $T_{k(p)} \in \mathbb{N}$ for the p^{th} metric with a rank index k is:

$$T_{k(p)} = \frac{({h_{k(p)}}^{max} - {h_{k(p)}}^{min})}{s_{k(p)}}$$

The expression for the upper bound of the normalized metric value can be re-evaluated as:

$$H_{k(p)}{}^{max} = \begin{cases} N_m, & k = 0 \\ H_k{}^{min} + \boldsymbol{S_k}\, T_{k(p)}, & k > 0 \end{cases} \tag{3.12}$$

Once the interval for normalizing the metric has been computed, the normalized metric is expressed as:

$$H_{k(p)} = H_k{}^{min} + \frac{\left(h_{k(p)} - h_k{}^{min}\right)[H_{k(p)}{}^{max} - H_k{}^{min}]}{h_{k(p)}{}^{max} - h_{k(p)}{}^{min}}$$

Since $H_k{}^{min}$ always evaluates to 1, and the ratio of the range for the metrics can be expressed as the ratio of their step sizes, the expression can be re-written as:

$$H_{k(p)} = 1 + \frac{\boldsymbol{S_k}}{s_{k(p)}}\left(h_{k(p)} - h_{k(p)}{}^{min}\right) \tag{3.13}$$

Once each metric has been normalized to their new intervals, the Health Quotient of the OBC can be computed by simply aggregating the values of all metrics. A module's Health Quotient H is expressed as:

$$H = \sum_{i=0}^{N_H} H_{\langle i \rangle} \tag{3.14}$$

where N_H is the highest assigned rank index to the list of all health metrics, as defined in section 3.2.4.

3.3.1 Health Metric Precedence

The selected normalization intervals for each of the health metrics allow any metric to override the combined effect of all metrics with smaller rank indices. The comparative health assessment of all available OBCs effectively boils down to a comparison of the highest ranked metric for which the evaluation results are different. This order of precedence established between

various health metrics will always hold and provides a predictable way of ranking the health of all available OBCs.

To demonstrate this order of precedence, the comparative health assessment between two arbitrary onboard computers is presented by working through the computation of the Health Quotient for both OBCs. From equation (3.14), the Health Quotients H^1 and H^2 for the two separate OBCs are expressed as:

$$H^1 = \sum_{i=0}^{N_H} H^1_{\langle i \rangle} = H^1_{\langle N_H \rangle} + H^1_{\langle N_H - 1 \rangle} + \cdots + H^1_0 \tag{3.15}$$

$$H^2 = \sum_{i=0}^{N_H} H^2_{\langle i \rangle} = H^2_{\langle N_H \rangle} + H^2_{\langle N_H - 1 \rangle} + \cdots + H^2_0 \tag{3.16}$$

Let the health metric with rank index *k* be the highest ranked metric where the evaluation result is not identical on both OBCs. The difference in the evaluation of this metric with rank index *k* on the two OBCs is minimized by setting it equal to the normalized step size of that metric:

$$H^1_{\langle k \rangle} - H^2_{\langle k \rangle} = \boldsymbol{S_k}$$

Or

$$H^1_{\langle k \rangle} = H^2_{\langle k \rangle} + \boldsymbol{S_k} \tag{3.17}$$

If the Health Quotient H^1 is greater than H^2, even by this minimum difference between the health metrics, the order of precedence between the health metrics must still hold true.

Substituting the Equation (3.17) back into Equation (3.15):

$$H^1 = H^1_{\langle N_H \rangle} + H^1_{\langle N_H - 1 \rangle} + \cdots + H^1_{\langle k+1 \rangle} + \left(H^2_{\langle k \rangle} + \boldsymbol{S_k}\right) + H^1_{\langle k-1 \rangle} + \cdots + H^1_0 \tag{3.18}$$

$$H^2 = H^2_{\langle N_H \rangle} + H^2_{\langle N_H - 1 \rangle} + \cdots + H^2_{\langle k+1 \rangle} + H^2_{\langle k \rangle} + H^2_{\langle k-1 \rangle} + \cdots + H^2_0 \tag{3.19}$$

Since the evaluation results of all metrics with rank indices greater than *k* are assumed to be identical on both OBCs, they can be replaced by a constant, denoted here as *L*, in Equations (3.18) and (3.19):

$$H^1 = L + \left(H^2_{\langle k \rangle} + \boldsymbol{S_k}\right) + H^1_{\langle k-1 \rangle} + \cdots + H^1_0$$

$$H^2 = L + H^2_{\langle k \rangle} + H^2_{\langle k-1 \rangle} + \cdots + H^2_0$$

Finally, the ability of the metric with index *k* to override the combined effects of all metrics with rank indices smaller than *k* must also be demonstrated. To do so, all normalized health metrics of H^1 with a rank index less than *k* are evaluated to their minimum possible values defined by their normalization interval. This also implies assigning a lower module rank index to the OBC with Health Quotient H^1. The same metrics on the OBC with Health Quotient H^2 are evaluated to their maximum possible values, such that:

$$H^1 = L + (H^2_k + \boldsymbol{S_k}) + H^{\min}_{\langle k-1 \rangle} + \cdots + H^{\min}_0 \tag{3.20}$$

$$H^2 = L + H^2_k + H^{\max}_{\langle k-1 \rangle} + \cdots + H^{\max}_0 \tag{3.21}$$

In such a case, the Health Quotient H^1 should still be strictly greater than the Health Quotient H^2. Subtracting Equation (3.21) from Equation (3.20):

$$H^1 - H^2 = S_k - \left(H^{\max}_{\langle k-1 \rangle} + \cdots + H^{\max}_0\right) + \left(H^{\min}_{\langle k-1 \rangle} + \cdots + H^{\min}_0\right)$$

$$\Rightarrow H^1 - H^2 = \boldsymbol{S_k} - \left(\sum_{i=0}^{k-1} H^{\max}_{\langle i \rangle} - \sum_{i=0}^{k-1} H^{\max}_{\langle i \rangle}\right) \tag{3.22}$$

From the definition of the step size for the interval based on property (P 3-6) of normalization, the step size $\boldsymbol{S_k}$ was expressed in Equation (3.9) as:

$$\boldsymbol{S_k} = \sum_{i=0}^{k-1} {H_{\langle i \rangle}}^{max} - \sum_{i=0}^{k-1} {H_{\langle i \rangle}}^{min} + 1$$

Substituting this expression for $\boldsymbol{S_k}$ back into Equation (3.22):

$$H^1 - H^2 = 1 \Rightarrow H^1 > H^2 \tag{3.23}$$

Hence, the proposed method of normalizing each health metric into unique intervals based on their relative ranks will always provide precedence to the onboard computer that performs better under the criteria for the highest ranked health metrics.

3.3.2 Health Quotient Uniqueness

It is imperative that the value of the health quotient computed by each available onboard computer is unique, so that there is no ambiguity in choosing the healthiest module. The specialized metric that assigns module ranks to each OBC ensures that even if the aggregate of all other normalized health metrics is equal on two separate health modules, the Health Quotient itself will be unique.

Lemma 3-1 shows that the step size for a normalized health metric with rank *k* is always greater than the step size for a normalized health metric with rank (*k* – 1). This intermediate result will be used to demonstrate the uniqueness of the Health Quotient.

Lemma 3-1: The step size $\mathbf{S_k}$ for a normalized health metric with a rank index of *k* is always greater than the step size $\mathbf{S_{k-1}}$ for a normalized health metric with a rank index of *k – 1*.

$$\mathbf{S_k} > \mathbf{S_{k-1}}, \quad \forall k \in \mathbb{N} : k > 0$$

Proof

From Equation (3.9), the step size $\mathbf{S_k}$ can be expressed as:

$$\mathbf{S_k} = \sum_{i=0}^{k-1} H_{\langle i \rangle}{}^{max} - \sum_{i=0}^{k-1} H_{\langle i \rangle}{}^{min} + 1$$

$$= \left(H_{\langle k-1 \rangle}{}^{max} + \sum_{i=0}^{k-2} H_{\langle i \rangle}{}^{max} \right) - \left(H_{\langle k-1 \rangle}{}^{min} + \sum_{i=0}^{k-2} H_{\langle i \rangle}{}^{min} \right) + 1 \tag{3.24}$$

Similarly, the step size $\mathbf{S_{k-1}}$ can be expressed as:

$$\mathbf{S_{k-1}} = \sum_{i=0}^{k-2} H_{\langle i \rangle}{}^{max} - \sum_{i=0}^{k-2} H_{\langle i \rangle}{}^{min} + 1 \tag{3.25}$$

Subtracting Equation (3.25) from Equation (3.24):

$$\mathbf{S_k} - \mathbf{S_{k-1}} = H_{\langle k-1 \rangle}{}^{max} - H_{\langle k-1 \rangle}{}^{min} \tag{3.26}$$

From property (P 3-4) of health metrics, the upper bound and lower bound of the metric cannot be equal and by definition, the upper bound of an interval cannot be smaller than its lower bound. So,

$$H_{\langle k-1 \rangle}{}^{max} - H_{\langle k-1 \rangle}{}^{min} > 0 \tag{3.27}$$

Substituting Equation (3.27) back into Equation (3.26):

$$\Rightarrow \mathbf{S_k} - \mathbf{S_{k-1}} > 0$$

$$\Rightarrow \mathbf{S_k} > \mathbf{S_{k-1}}$$

Lemma 3-1 can be used to demonstrate the uniqueness of the Health Quotients on any two OBCs. The Health Quotients H^1 and H^2 for two arbitrary OBCs are expressed as:

$$H^1 = \sum_{i=0}^{N_H} H_{\langle i \rangle}^1 = \sum_{i=1}^{N_H} H_{\langle i \rangle}^1 + H_0^1$$

$$H^2 = \sum_{i=0}^{N_H} H_{\langle i \rangle}^2 = \sum_{i=1}^{N_H} H_{\langle i \rangle}^2 + H_0^2$$

Taking the difference of the two Health Quotients:

$$H^1 - H^2 = \left[\sum_{i=1}^{N_H} H_{\langle i \rangle}^1 - \sum_{i=1}^{N_H} H_{\langle i \rangle}^2\right] + (H_0^1 - H_0^2) \tag{3.28}$$

The difference of the two specialized rank indices can never be 0 since module rank is. However, the aggregate of all other metrics may or may not evaluate to the same value on both modules.

3.3.2.1 Aggregate of All Health Metrics with Rank Index > 0 is the Same on Both Modules

In this case,

$$\sum_{i=1}^{N_H} H^1_{\langle i \rangle} = \sum_{i=1}^{N_H} H^2_{\langle i \rangle} \tag{3.29}$$

Substituting Equation (3.29) back into Equation (3.28):

$$\Rightarrow H^1 - H^2 = (H^1_0 - H^2_0)$$

Since the module rank is unique, its difference can never be 0. So,

$$(H^1_0 - H^2_0) \neq 0$$

$$\Rightarrow H^1 - H^2 \neq 0$$

3.3.2.2 Aggregate of All Health Metrics with Rank Index > 0 is Different on Both Modules

In this case,

$$\left[\sum_{i=1}^{N_H} H^1_{\langle i \rangle} - \sum_{i=1}^{N_H} H^2_{\langle i \rangle}\right] \neq 0$$

Or,

$$\left[\left(H^1_{\langle N_H \rangle} + H^1_{\langle N_H - 1 \rangle} + \cdots H^1_{\langle 1 \rangle}\right) - \left(H^2_{\langle N_H \rangle} + H^2_{\langle N_H - 1 \rangle} + \cdots H^2_{\langle 1 \rangle}\right)\right] \neq 0 \tag{3.30}$$

It can be deduced from Lemma 3-1 that the magnitude Equation (3.30) will be at least the step size $\boldsymbol{S_1}$, since the step size for all other normalized metrics will be greater than $\boldsymbol{S_1}$.

$$\left| \left(H^1_{\langle N_H \rangle} + H^1_{\langle N_H - 1 \rangle} + \cdots H^1_{\langle 1 \rangle}\right) - \left(H^2_{\langle N_H \rangle} + H^2_{\langle N_H - 1 \rangle} + \cdots H^2_{\langle 1 \rangle}\right) \right| \geq \boldsymbol{S_1} \tag{3.31}$$

From Equation (3.10), the value of $\boldsymbol{S_1}$ can be expressed as follows:

$$\boldsymbol{S_1} = {H_0}^{max} - (\|N_0\| - 1)$$

The module rank is the only health metric with a rank index of 0, so $\|N_0\| = 1$. The value of ${H_0}^{max} = N_m$ i.e., the total number of redundant modules deployed. Therefore,

$$\boldsymbol{S_1} = N_m \tag{3.32}$$

Substituting the Equation (3.32) back into Equation (3.31):

$$\left| \left(H^1_{\langle N_H \rangle} + H^1_{\langle N_H - 1 \rangle} + \cdots H^1_{\langle 1 \rangle}\right) - \left(H^2_{\langle N_H \rangle} + H^2_{\langle N_H - 1 \rangle} + \cdots H^2_{\langle 1 \rangle}\right) \right| \geq N_m$$

Or,

$$\left| \left[\sum_{i=1}^{N_H} H^1_{\langle i \rangle} - \sum_{i=1}^{N_H} H^2_{\langle i \rangle} \right] \right| \geq N_m \tag{3.33}$$

Since the assignment of which OBC has a Health Quotient H^1 and which OBC has Health Quotient H^2 is arbitrary, it can be assumed that the aggregate of the health metrics with a rank index k > 0 is larger on OBC 1. Therefore, Equation (3.33) can be re-written as:

$$\left[\sum_{i=1}^{N_H} H^1_{\langle i \rangle} - \sum_{i=1}^{N_H} H^2_{\langle i \rangle} \right] \geq N_m \tag{3.34}$$

Substituting Equation (3.34) back into Equation (3.28):

$$\Rightarrow H^1 - H^2 \geq N_m + (H^1_0 - H^2_0) \tag{3.35}$$

The magnitude of the difference between the module ranks of any two OBCs is at least 1. Therefore,

$$| H^1_0 - H^2_0 | \geq 1 \tag{3.36}$$

The upper bound of the difference between the module ranks can also be computed, since the metric is known to lie in the interval $[1, N_m]$

$$| H^1_0 - H^2_0 | \leq N_m - 1 \tag{3.37}$$

Even though the assignment of Health Quotients H^1 and H^2 was arbitrary, either one of the OBCs can have a larger module rank.

***Case 1*:**

OBC 1 has a larger module rank. Re-writing Equation (3.36):

$$H_0^1 - H_0^2 \geq 1 \tag{3.38}$$

Substituting Equation (3.38) back into Equation (3.35):

$$\Rightarrow H^1 - H^2 \geq N_m + 1 \tag{3.39}$$

***Case 2*:**

OBC 2 has a larger module rank. Re-writing Equation (3.36):

$$H_0^1 - H_0^2 \leq -1$$

From Equation (3.37), the lower bound of the difference in module ranks is also known:

$$H_0^1 - H_0^2 \geq 1 - N_m \tag{3.40}$$

Substituting Equation (3.40) back into Equation (3.35):

$$\Rightarrow H^1 - H^2 \geq N_m + 1 - N_m$$

$$\Rightarrow H^1 - H^2 \geq 1 \tag{3.41}$$

Equations (3.39) and (3.41) demonstrate that for all possible cases, the Health Quotients $H^1 \neq H^2$ for any arbitrarily chosen pair from the redundant onboard computers. Thus, the proposed methodology for normalizing the health metrics always results in a unique Health Quotient on each available onboard computer. The uniqueness property of Health Quotient combined with its established order of precedence can, therefore, be used to provide a deterministic comparison between any number of available onboard computers.

3.4 CONSIDERATIONS FOR PRACTICAL IMPLEMENTATION

3.4.1 Avoiding Arithmetic Overflows

The proposed formulation of an OBC's Health Quotient does not impose any restriction on the maximum number of health metrics since the precedence of individual health metrics as well as the uniqueness of the Health Quotient holds true for any number of health metrics. However, the number of health metrics and the maximum normalized ranges will be limited on the selected OBC's memory architecture along with any extensions applied in flight software [64]. If the Health Quotient or any of the intermediate results during it computation exceeds this limit, an arithmetic overflow will cause the result to wrap around the supported interval of the data type chosen to represent the Health Quotient. The supported interval for the selected data type must be assessed against the excepted interval of values for the Health Quotient. The same is also true for any intermediate results for computing the Health Quotient.

3.4.1.1 Bounding Interval of the Health Quotient

Since the minimum value for all normalized health metrics is 1, the lower bound for the Health Quotient is dependent on the number of health metrics (after all dataset reductions) as well the assigned value of the module rank (rank index 0). The deviation in the computed interval due to the module rank will not be relevant for choosing the correct data type, however, it can be used as a software guard to ensure each module is generating a Health Quotient within the defined limits. Using Equation (3.14), the lower bound of the Health Quotient H_L is expressed as:

$$H_L = \left(\sum_{i=1}^{N_H} H_{\langle i \rangle}{}^{min} \right) + H_0 = \left\| N_{N_H} \right\| - 1 + H_0 \tag{3.42}$$

N_H denotes the highest rank index assigned to the list of health metrics after all dataset reductions. Consequently, $\left\| N_{N_H} \right\|$ denotes the number of health metrics with a rank index of N_H

or less. Since this includes the module rank metric as well, its minimum value of 1 is subtracted from the expression and the value of the module rank metric H_0 is directly added in.

The upper bound for the Health Quotient of a module can be evaluated by setting all normalized health metrics to their maximum value. Extending Equation (3.12), the aggregate for the upper bound of all metrics with a rank index of $k > 0$ is expressed as:

$$H_{\langle k \rangle}{}^{max} = H_{\langle k \rangle}{}^{min} + \boldsymbol{S_k}\, T_{\langle k \rangle} \tag{3.43}$$

Using Equations (3.14) and (3.43), the upper bound for the Health Quotient H_U is expressed as:

$$H_U = H_0 + \sum_{i=1}^{N_H} H_{\langle i \rangle}{}^{max} = H_0 + \sum_{i=1}^{N_H} H_{\langle i \rangle}{}^{min} + \sum_{i=1}^{N_H} \boldsymbol{S_i}\, T_{\langle i \rangle} \tag{3.44}$$

Substituting Equation (3.42) into Equation (3.44), the upper limit for the Health Quotient can be re-expressed as:

$$H_U = H_L + \sum_{i=1}^{N_H} \boldsymbol{S_i}\, T_{\langle i \rangle} \tag{3.45}$$

Thus, bounding interval of the Health Quotient $H \in [H_L,\ H_U]$ is defined by Equations (3.42) and (3.45). This bounding interval should be used to select the appropriate storage types.

3.4.1.2 Geometric Mean Computation

Equally ranked Non-Boolean Health metrics are reduced to as single equivalent health metric by computing the geometric mean of the health metrics, as expressed in in Equation (3.4). The computation involves multiplication of all individual health metrics which themselves could be considerably large quantities. This could result in an arithmetic overflow on an intermediate calculation. Since arithmetic overflows cause the result to wrap around the supported range of

numbers, it can result in negative intermediate results and consequently imaginary values for the equivalent health metric.

An equivalent expression for the geometric mean is shown below [65] and can be used to compute the equivalent health metric and reduce the probability of an arithmetic overflow:

$$g_k = \left[\prod_{i=1}^{m} g_{k\_i}\right]^{\frac{1}{m}} = e^{\frac{1}{m}\left[\sum_{i=1}^{m} \ln g_{k\_i}\right]}$$

This expression can be expanded as:

$$g_k = e^{\frac{1}{m}\left[\ln g_{k\_m} + \ln g_{k\_m-1} + \cdots + \ln g_{k\_0}\right]}$$

Taking the natural logarithm of the original health metric value limits the growth rate of intermediate results and avoids arithmetic overflows when dealing with exceptionally large numbers. It should be noted that taking the natural logarithm of any number may result in rounding errors. The step size for the equivalent non-Boolean metric and its corresponding interval may need to be slightly adjusted to ensure that the number of steps between the minimum and maximum value of the equivalent metric belongs to the set of natural numbers.

3.4.2 Handling Health Metric Outliers

The normalization of health metrics uses feature scaling that relies on metrics values lying within the well-defined intervals of the original health metrics. If a health metric h_k with a rank index of *k*, evaluates to a value outside its assumed interval $[{h_k}^{min}, {h_k}^{max}]$, the normalized health metric H_k will also lie outside the desired interval $[{H_k}^{min}, {H_k}^{max}]$. This will upset the established order of precedence between the health metrics and the uniqueness of the Health Quotient will no longer be guaranteed.

To ensure the normalized health metrics always lie within the desired interval, the raw values of any outliers must be re-normalized to within their defined intervals. Simply clipping the value of the health metrics at the edge of the defined intervals may not reveal the true nature of the underlying error that forced the metric outside its defined range. For instance, the metric evaluation h_k can be larger than ${h_k}^{max}$ because of an arithmetic underflow of an unsigned integer. Setting the value of such a metric to ${h_k}^{max}$ may grossly overestimate the true value of this health metric. The same is true if the metric experiences an arithmetic overflow. Such metrics should be considered unreliable and should always be minimized to ${h_k}^{min}$. Even though it does not represent the true evaluation of that metric, the minimization is indicative of the fact that the OBC is not be performing nominally. This restores the order of precedence between the health metrics and allows other modules to assume command of the spacecraft bus.

An exception to this rule may be made for strictly monotonic health metrics that provide precedence for OBCs that have previously commanded the spacecraft bus. If fidelity of the health metric evaluation can be ensured, the value of this metric can be clipped at ${h_k}^{max}$. An example of such a metric could be one that measures mission lifetime, indicating the time a module has spent in command of the spacecraft bus. The value of such metric can be maximized, indicating the module has controlled the spacecraft bus for at least as long as the expected mission lifetime, and that the expected mission lifetime has elapsed. Precedence health metrics should, however, always have a lower rank index than any health metric that assess the module's functional or performance capabilities.

3.5 SUMMARY

A novel strategy to normalize any evaluated health metric, based on its relative priority, was presented in this chapter. The proposed methodology is agnostic to the range and step size of the evaluated health metric. Using the proposed methodology, a unified health metric called the Health Quotient is generated, which reduces the large dataset of individual health metrics to a single value. The proposed methodology allows relative prioritization of individual health metrics based on mission needs, and the generated Health Quotient is unique for each redundant OBC. This Health Quotient will be used to perform a comparative health assessment between all available onboard computers. The comparative analysis is encapsulated within the OBC arbitration process and will be discussed in Chapter 4

CHAPTER 4 EXTENSIBLE ARBITRATION ARCHITECTURE

Summary: This chapter introduces an extensible arbitration process that is executed autonomously after the spacecraft bus resets and mediates the computation and exchange of OBC Health Quotients. The arbitration process defines unidirectional transitions between each of its states to ensure resolution of the autonomous arbitration process. Lastly, the discussion highlights the scale agnostic nature of the arbitration logic and presents the impact of the scaling the redundancy architecture on the reliability of the system.

4.1 ARCHITECTURE OVERVIEW

4.1.1 The Arbitration Process

The arbitration process has four distinct phases, namely the Module Initialization phase, the Handshake phase, the Health Assessment phase, and the Arbitration Resolution phase. The sequential progression of these phases is shown in Figure 4-1. The Module Initialization phase constitutes the OBC's boot-up sequence and setup of its initial state configuration. The Handshake phase identifies all available OBC's that have successfully completed their initialization phase and are capable of taking part in the current arbitration cycle. The Health Assessment phase provides each available OBC an opportunity to compute its own Health Quotient and broadcast it to the other OBCs. Finally, the Health Quotients of all available OBCs are compared in the Arbitration Resolution phase and command of the spacecraft bus is assigned to the OBC with the greatest Health Quotient.

The background colors of the arbitration phases shown in Figure 4-1 indicate whether the flight software execution state of all available modules is identical. The execution times required to complete the boot up sequence for any given module will not be consistent through every arbitration cycle due to communication and cache memory latencies within the processor [66]. Every module exhibits this jitter in its boot-up time. As a result, the OBCs on the spacecraft bus are not necessarily in the same execution state during this phase. However, all available modules are synchronized during the Handshake phase of the arbitration cycle and thereafter, all available modules transition into subsequent phases within a well characterized window of time. The synchronization between all available modules ensures that no two modules attempt to issue conflicting power cycling commands on the spacecraft bus, which would likely lead to a non-deterministic resolution of the arbitration process.

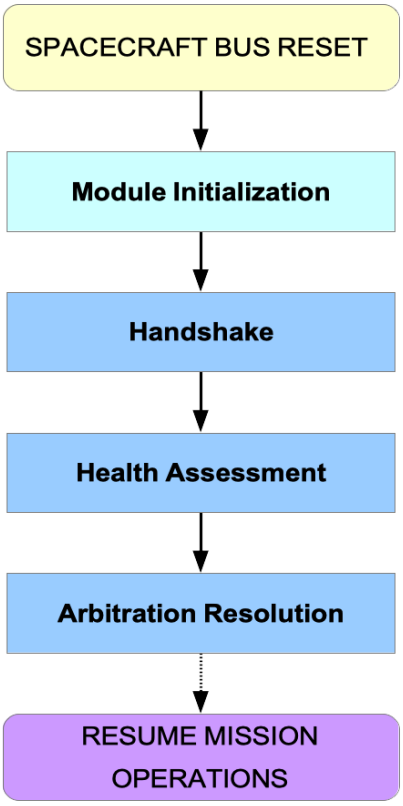

Figure 4-1: The Arbitration Phases

Although every module commences the arbitration process every time the spacecraft bus is reset, it might not complete all 4 phases of the cycle. Fault detection mechanisms on each module that track probable transient as well as permanent module failures might cause an OBC to terminate the arbitration cycle of other OBCs. Similarly, the detection of an unstable software configuration might result in the OBC terminating its own arbitration cycle. However, at the end of the arbitration cycle only one module transitions out of the Arbitration Resolution phase and assumes command

of the spacecraft bus. This distinction is represented in Figure 4-1 by a dashed arrow emerging out of the Arbitration Resolution phase. All other redundant modules will be powered down to reduce the long-term accumulated effects of ionizing radiation [3] [9].

Computing the Health Quotient alone after a spacecraft bus reset limits the assessment to the state of the OBC during a short window of a single arbitration cycle. It does not accommodate transient failures that might get resolved if the OBC resets, nor does it identify repeated failures that might occur shortly after the completion of the arbitration cycle. The arbitration process uses persistent configuration to maintain knowledge of these failure modes over multiple arbitration cycles.

A key attribute of the proposed arbitration process is that the OBCs conducting the arbitration process are themselves taking part in it as well i.e., there is no dedicated external supervisor for the arbitration process. Existing cold redundant architecture commonly integrate an external supervisor, for instance in the form of a low power microprocessor, which simplifies the arbitration cycle. The handshake process becomes trivial in such a case since the process supervisor can sequentially ping each OBC on the spacecraft bus to determine its state. It can also help in the Health Assessment phase of the arbitration process by mediating control of the spacecraft bus between all the available OBCs. However, an external process supervisor acts as a single point of failure in the redundancy architecture. If this external supervisor becomes unavailable at any point during the mission, the spacecraft will lose the ability to arbitrate between the OBCs, thus nullifying the benefits of deploying redundant Onboard Computers. Moreover, a failure mode need not be localized at the external process supervisor to interfere with its ability to conduct the arbitration process. For instance, a partial failure in the power distribution module

might cut-off power to this external supervisor and thus, render it unable to arbitrate between the OBCs, even though the process supervisor itself did not experience a failure mode.

Although an external supervisor is not a viable design choice, a supervisor for the arbitration process is required, nonetheless. As noted earlier, the supervisor helps determine which modules are available for the arbitration cycle during the Handshake phase, as well as mediates control of the spacecraft bus during the Health Assessment phase. If the arbitration process is carried out without any supervisor, each OBC will have to utilize a system of message broadcasting to inform the other modules of its current state, which can be highly unreliable. The primary issue with such a system is that a message broadcast by one of the modules might not necessarily be delivered successfully to all other modules, causing the modules to fall out of synchronization. The different operating states of the OBCs can result in conflicting commands being issued on the spacecraft bus, resulting in a non-deterministic resolution of the arbitration process.

To resolve this issue, one of the onboard computers is assigned the role of process supervisor for the arbitration process. The arbitration process allows dynamic re-assignment of this role during the Handshake phase such that if the current process supervisor of the arbitration cycle experiences a failure mode, the role is re-assigned to a different OBC at the start of the handshake process, thus ensuring a supervisor is always available to conduct the arbitration process.

All other modules are assigned the role of a redundant module during the arbitration process. Each redundant module progresses the through the same states within each phase of the arbitration process. This allows an arbitrary number of redundant modules to be deployed on the spacecraft, without the need to modify the underlying the arbitration process.

4.1.2 Spacecraft Bus Layout

Each redundant OBC is connected to an independent switched power line on the spacecraft's Power Distribution Module (PDM). The default state of each of these power lines is ON at mission epoch. As the mission progresses, malfunctioning OBCs can be permanently disabled by changing the default state of these power lines to OFF, such that a spacecraft bus reset will keep the OBCs powered down. This allows the architecture to remove failed OBCs from the arbitration process. The arbitration architecture expects an external watchdog timer to reset the power distribution module in case the OBC commanding the spacecraft bus experiences a transient or permanent failure. Note that this hardware watchdog does not act as an external supervisor for the arbitration process. It merely resets the spacecraft bus if an OBC fails, thereby initiating the arbitration process. After the bus reset, the OBCs carry out the arbitration process themselves. Spacecraft power systems include these watchdogs by default, even for single OBC platforms.

4.1.3 State Machine Representation

Each phase of the arbitration process is split up into states that represent the finite state machine of the arbitration process. Each state is encapsulated within a clearly defined transition boundary that identifies all possible logical paths entering or exiting from that state. During the arbitration process, the OBC cannot revisit a state it has already transitioned through. This unidirectional transition path ensures that an OBC does not loop within the arbitration process. The execution flow of each individual state is presented as a logical flowchart. Figure 4-2 shows the flowchart template for a state of the arbitration process.

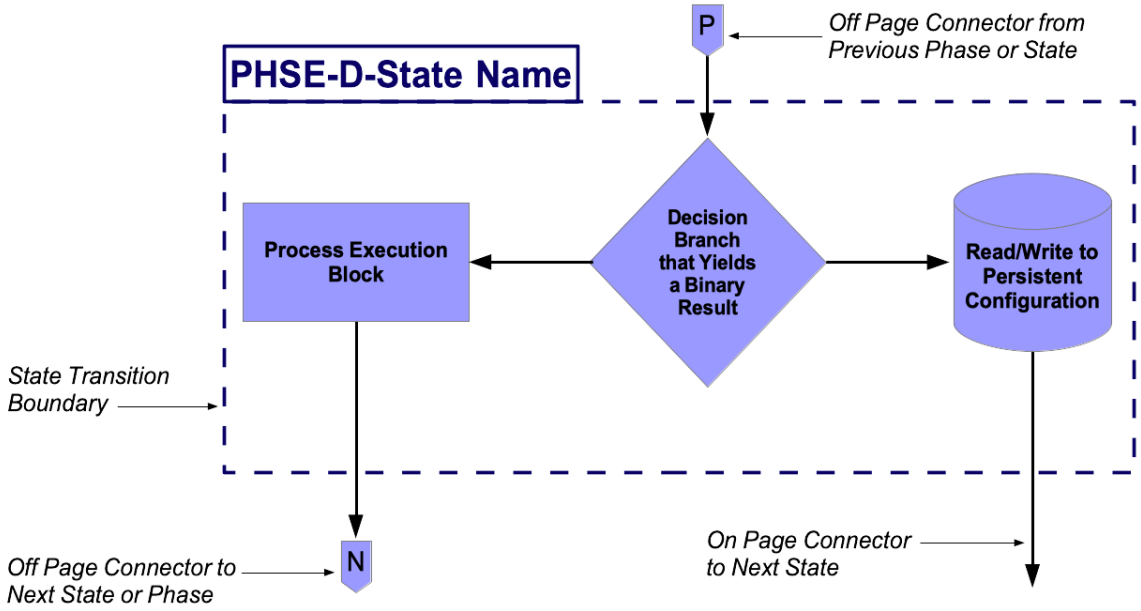

Figure 4-2: Arbitration Process State Template

The designation for each state has the following format:

PHSE-D-Sub Phase Name

The first part of the designation, denoted by **PHSE**, indicates the phase of the arbitration process within which the current state is implemented. The identifiers for each of the 4 phases of arbitration process are listed in Table 4-1.

Table 4-1: Arbitration Phase Name Designations

Arbitration Phase	Arbitration Phase Designation
Module Initialization	INIT
Handshake	HDSK
Health Assessment	HLTH

Arbitration Resolution	RESL

The next part of the designation, denoted by **D**, indicates whether the current state is executed by the OBC assigned the process supervisor role or one that has been assigned the redundant module role. It is also possible for a state to be executed by all OBCs, regardless of their role in the arbitration process. The possible identifiers for OBC role designations are listed in Table 4-2.

Table 4-2: Module Role Designations

OBC Role	Role Designation
Process Supervisor	S
Redundant Module	R
All OBCs	A

The logical flow diagrams for the process states are color coded according the OBC role responsible for executing it. The states executed by the process supervisor alone are shown in green, those executed only by the redundant modules are shown in orange, while those executed by all OBCs are shown in blue. The arbitration states in purple represent exit points from the current arbitration cycle.

4.2 MODULE INITIALIZATION PHASE

After the spacecraft bus resets, the boot code of the OBC initializes the hardware and its associated peripherals and commences execution of the flight software. The primary purpose of this phase is to drive the OBC into a safe initial configuration by suspending all scheduled and

sporadic tasks. This limits the flight software to the state transitions performed by the arbitration process. This phase also addresses any failure modes that were flagged in persistent storage during the previous arbitration cycle. The logical flowchart outlining the execution sequence of the Module Initialization phase is shown in Figure 4-3.

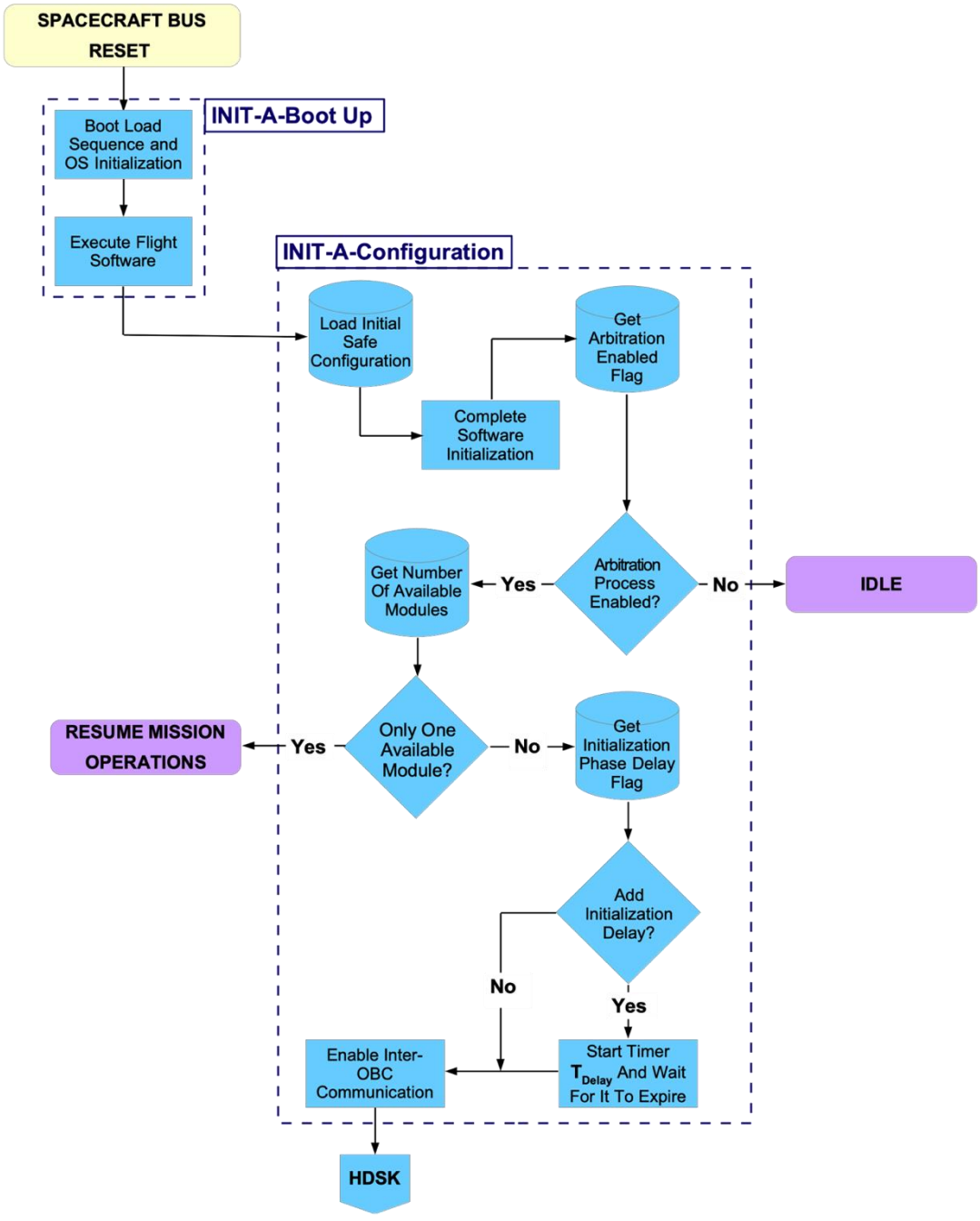

Figure 4-3: The Module Initialization Phase

4.2.1 The Bootup State

Whenever the spacecraft bus is powered up after a reset, all functional onboard computers will commence their boot-up sequence. The Module Initialization phase begins along with this boot-up sequence during which the underlying platform and its peripheral interfaces are initialized. There is no special consideration to be made for the arbitration cycle during this boot up sequence. As mentioned before though, inherent latencies within each OBC will result in variations in the time taken to transition into the next state. This variation in execution time will be resolved during the Handshake phase. Once the boot up sequence is complete, the OBC deploys the flight software binary and transitions into the *Configuration* state.

4.2.2 The Configuration State

Since there are multiple potential spacecraft bus masters powered up during the arbitration process, it is important to determine a safe initial configuration for the OBCs to avoid system damage or data corruption. Enforcing this initial configuration is the first step executed within the *Configuration* sub-phase. Since the OBCs are connected to the same physical spacecraft bus, it is crucial that all communication interfaces to the spacecraft bus are disabled. The flight software tasks are initialized and configured to be dormant. This ensures that consequent changes to the flight software state will only be driven by the arbitration process.

The *Configuration* state checks for failure modes that may have been flagged in persistent configuration during previous arbitration cycles. The first persistent configuration parameter identifies if the module's arbitration cycle is currently enabled and is utilized if the current process supervisor of the arbitration cycle is experiencing frequent resets during the arbitration cycle. Since this may stop other healthy modules from assuming control of the spacecraft, the process supervisor disables its own arbitration cycle. This causes the OBC to drop out of the arbitration

process and stay in an idle state. The redundant modules that are still executing the arbitration cycle assume that the current process supervisor has experienced a complete failure and thereafter attempt to assume control of the spacecraft bus. By default, the arbitration cycle is enabled for all OBCs.

The second parameter identifies the number of modules that can potentially command the spacecraft bus. In other words, this parameter tracks the number of modules that have been permanently disabled by switching the default state of their power lines to OFF. If there is only one module left which is capable of commanding the spacecraft bus, the arbitration cycle is skipped, and control is handed over to the only remaining module.

To determine the healthiest module available to command the spacecraft bus, the OBCs must be able to successfully communicate their Health Quotient with each other. However, this may not be possible if the communication channels between some or all OBCs experience frequent transients or complete failure. If the process supervisor cannot communicate with any other OBC and determines that its own Health Quotient is not sufficient to execute baseline mission tasks, it will enable a *Configuration* state delay in persistent memory and reset the spacecraft bus. During the next arbitration cycle, the OBC checks if this added delay was requested during the previous arbitration cycle. If so, a timer is set with a duration of T_{Delay} which blocks the progression of this sub-phase until that timer expires. If a redundant module is active on the spacecraft bus but was not able to communicate with the process supervisor, the added delay will force the redundant module to assume that the process supervisor has failed. The redundant module will subsequently take over as the process supervisor. This process, along with value of this delay timer, will be revisited during the Handshake phase (Section 4.3).

The last step of the *Configuration* state is to enable the communication interface used by the OBC to communicate with other available OBCs. As mentioned earlier, the jitter in bootup times between the various OBCs implies that their execution states are not synchronized during the Module Initialization phase and so, some of the OBCs will transition into the next phase of the arbitration process before others. Once in the next phase of the arbitration cycle, the process supervisor will try to initiate contact with the other available OBCs. If some of the other OBCs are still in the Module Initialization phase at point, attempting a handshake with them may cause them to prematurely transition into the next phase of the arbitration cycle. To avoid this and ensure that the transition from the Module Initialization phase to the Handshake phase is synchronous, the communication interface with the other OBCs is only enabled as the last step of *Configuration* state. Once the communication interface has been successfully enabled, the module transitions into the Handshake phase.

4.3 HANDSHAKE PHASE

The purpose of the Handshake phase is to identify all available modules on the spacecraft bus that are capable of taking part in the current arbitration cycle. A pre-established hierarchy based on the module ranks is utilized to choose the process supervisor of the arbitration cycle and is saved to persistent configuration. The process supervisor initiates a handshake request that is broadcast to all redundant modules. The redundant modules that successfully acknowledge this handshake request will transition into the next phase of the arbitration cycle, while the rest of the modules are powered down by the process supervisor.

Each state of the Handshake phase on every OBC is guarded by software watchdog timers running on every other OBC. Software watchdog timers are usually considered unreliable for detecting off-nominal behavior since the OBC's failure causes the watchdog timer to fail along

with it. However, the arbitration process relies on this attribute of software watchdog timers. Each OBC has a distinct timeout for its software watchdog timer based on its module rank. The values of these timeouts are setup such that they expire sequentially i.e., the watchdog timer on the module with the highest rank expires first while the timer on the module with the lowest rank expires last.

The process supervisor is the only OBC that can signal a reset of the watchdog timers on the redundant modules. Conversely, the watchdog timer on the process supervisor can only be reset if all redundant modules respond to the handshake request initiated by the process supervisor. This setup allows the process supervisor to identify unresponsive redundant modules and switch them off. On the other hand, if the watchdog timer expires on one of the redundant modules, the process supervisor itself has stopped functioning and the redundant module can attempt to take over as the process supervisor. The sequence in which the redundant module timers expire is based on immutable module ranks, which establishes a deterministic order between the modules for attempting to take over as the process supervisor. The logical flowchart outlining the execution sequence of the Handshake phase is shown in Figure 4-4.

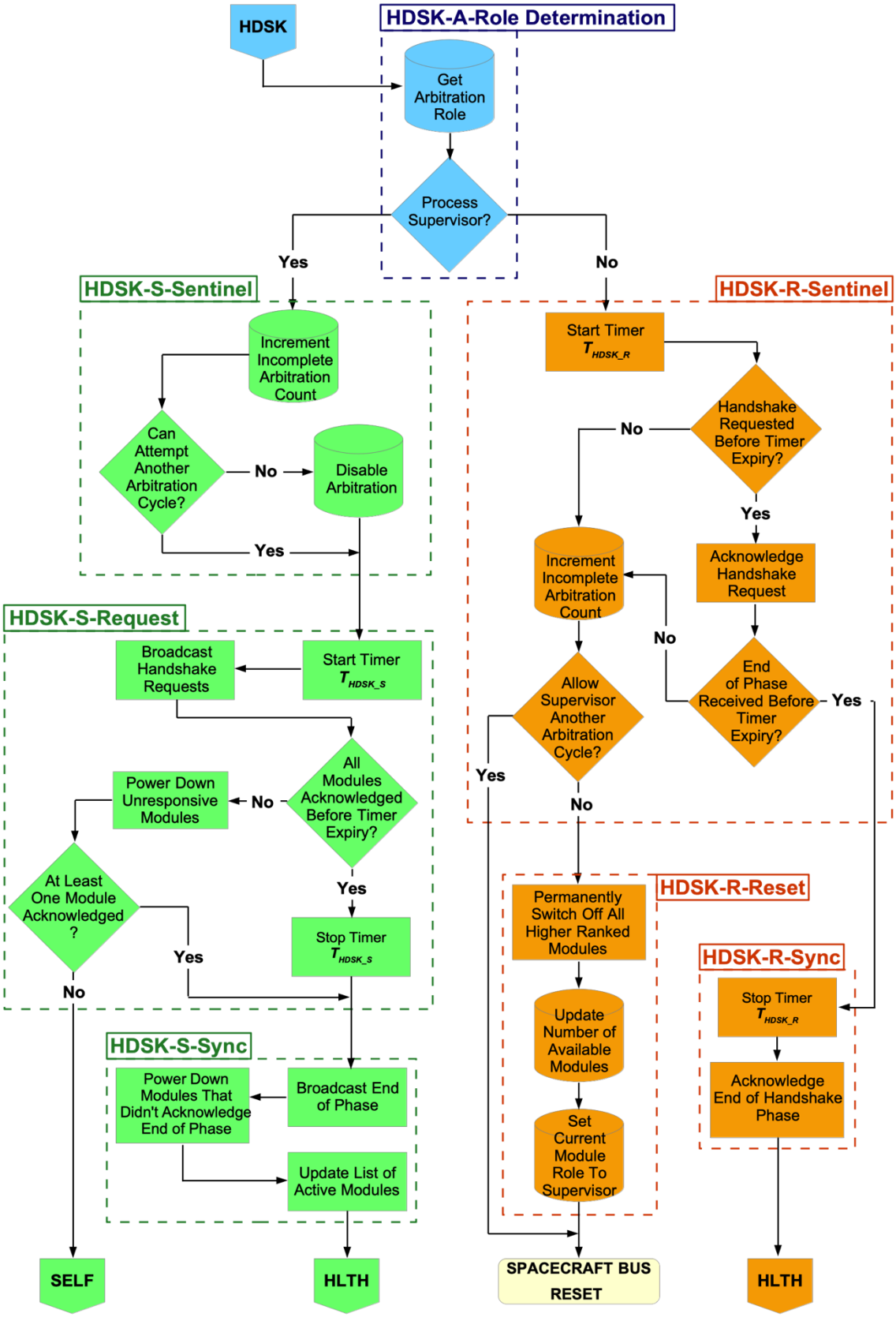

Figure 4-4: The Handshake Phase

The deterministic order in which the software watchdog timers expire can also be used to resolve the arbitration process in the event that the OBCs cannot communicate with each other. The process supervisor cannot distinguish between its watchdog timer expiring because all other modules have failed or because its communication channel with other modules has stopped functioning. In either case, the process supervisor will attempt to assume control of the spacecraft bus. However, the OBC will still assess its own Health Quotient. This is assessment is performed during the *Self Assess* state of the Handshake phase. The logical flowchart outlining the execution sequence of the *Self Assess* state is shown in Figure 4-5. The computed value of the Health Quotient can be compared to a preset threshold required for the OBC to perform the minimum set of tasks needed to continue mission operations. If the OBC fails to meet this threshold, the execution delay for the *Configuration* state is enabled and the spacecraft bus is reset. If the inter-OBC communication channel has in fact stopped working and other redundant modules are available, this execution delay allows the software watchdog timers on one of the redundant modules to expire before the current process supervisor's watchdog timer, allowing it to take over as the process supervisor.

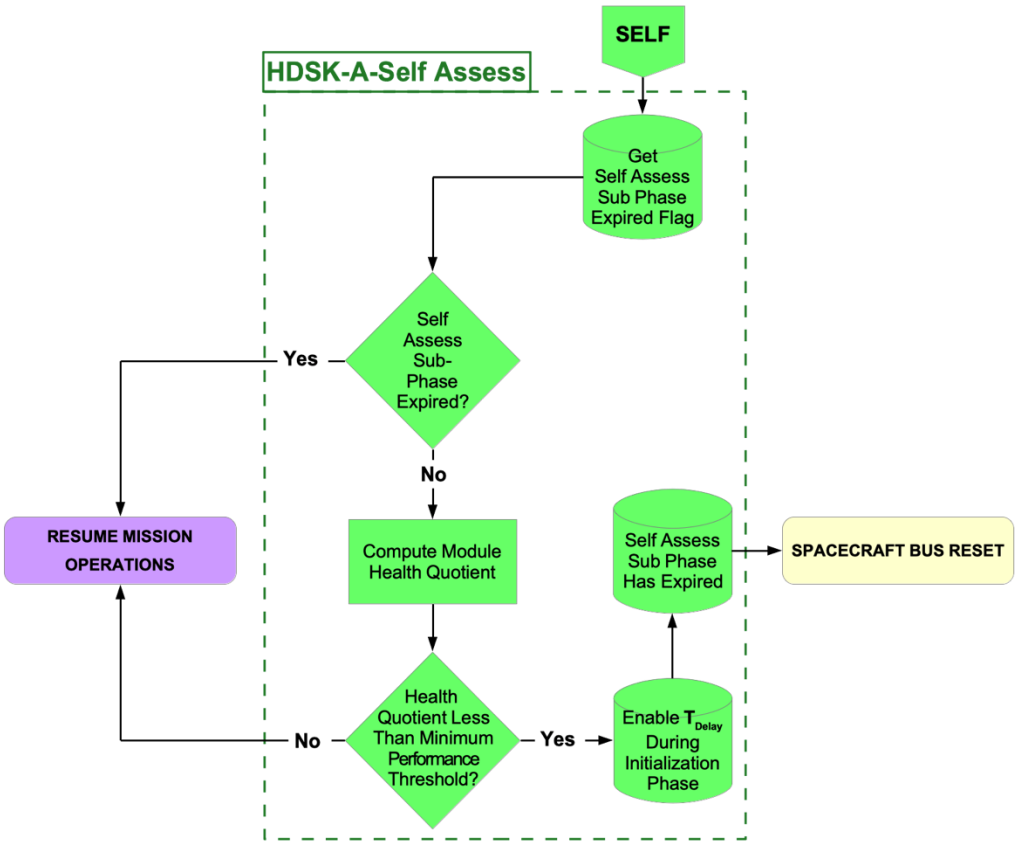

Figure 4-5: The Self Assess State of the Handshake Phase

4.3.1 The Role Determination State

During this state, the OBC establishes itself as the process supervisor or a redundant module in the arbitration cycle. Every module's role is stored in its persistent configuration. At mission epoch, the OBC with the highest module rank is assigned the process supervisor role, while all other OBCs act as redundant modules. The process supervisor maintains its role through subsequent arbitration cycles until it experiences a complete failure or is unable to conduct the arbitration process. To avoid conflicting commands being issued on the spacecraft bus, it is critical that any given arbitration cycle has exactly one process supervisor. Synchronization of module roles between all OBCs cannot be guaranteed because of unpredictable intermittent failures caused

by Single Event Effects. Thus, to ensure that the arbitration cycle has only one process supervisor, the current process supervisor is always permanently disabled before re-assigning that role to a functioning redundant module. In other words, the process supervisor always has the largest module rank out of all available redundant OBCs.

A side effect of this strategy is that a partial failure will also cause the current process supervisor to be permanently disabled if that partial failure hinders the process supervisor's ability to conduct the arbitration process. However, the partially working OBC is traded off in favor of a predictable and efficient arbitration process. Moreover, maintaining a redundant OBC whose module rank is higher than the current supervisor's module rank will lengthen the execution time of the arbitration cycle. This fact will be highlighted in several subsequent states of the arbitration process, where the software watchdog timers would terminate quicker if the process supervisor always has the highest module rank.

The module serving as the process supervisor for the current arbitration cycle transitions into the process supervisor branch, while all other modules transition into the redundant module branch. Both branches start with their respective *Sentinel* sub-phases. All redundant modules transition through the same states for the rest of the arbitration process, enabling an extensible architecture that can support greater (or fewer) number of redundant modules.

4.3.2 The Process Supervisor Branch

4.3.2.1 The Sentinel State

Since the processor supervisor is just one of the available OBCs on the spacecraft, it is as likely to experience a Single Event Effect as any one of the redundant modules, causing it hang up during the arbitration cycle. The SEE might resolve itself after a spacecraft bus reset, however, it is also just as likely to degrade the OBC's operational capacity causing it to repeatedly hang up

during the arbitration cycle or soon thereafter. As a result, the OBC might try to repeatedly conduct the arbitration cycle as the process supervisor but might fail to complete the cycle.

To prevent this, the process supervisor keeps track of the number of consecutive incomplete arbitration cycles, denoted by $N_{CycleCount}$. $N_{CycleCount}$ is incremented by 1 in persistent storage as soon as the process supervisor transitions into this state (the current arbitration cycle is counted as an incomplete cycle). The current value of $N_{CycleCount}$ is then compared to the preset maximum number of consecutive incomplete arbitration cycles allowed, denoted by $N_{MaxCycleCount}$.

The value of $N_{MaxCycleCount}$ is a tradeoff between the maximum number of resets allowed to resolve a potential SEE and the likelihood that at least one more redundant module is available to take over as the next process supervisor. As the mission progresses, some of the modules might experience complete failure thereby reducing the number of available redundant modules. A smaller value of $N_{MaxCycleCount}$ would allow quicker re-assignment of the process supervisor role and a faster recovery time. This configuration is ideal for a large cluster of redundant OBCs, where a redundant module is more likely to be available over the course of the mission. On the other hand, a larger value of $N_{MaxCycleCount}$ would afford the current process supervisor a greater number of resets to resolve a transient failure. This configuration is ideal for a small cluster of redundant OBCs, where a redundant module is less likely to be available during the later phases of the mission.

If the value of $N_{CycleCount}$ is equal to $N_{MaxCycleCount}$, then the process supervisor disables its arbitration process. Since the current arbitration cycle is also counted towards the number of incomplete arbitration cycles, the process supervisor still transitions to the next state and attempts to complete the current arbitration cycle. The arbitration process is only re-enabled if the process supervisor successfully completes the current arbitration cycle. If the process supervisor fails to

complete the arbitration cycle again and the spacecraft bus resets, then the module will terminate its next arbitration cycle during the Module Initialization phase.

After comparing the value of $N_{CycleCount}$ with that of $N_{MaxCycleCount}$ and, if required, disabling the next arbitration cycle, the process supervisor transitions into its *Request* state.

4.3.2.2 The Request State

During this state, the process supervisor broadcasts a handshake request to all redundant modules. Establishing a handshake with the other OBCs allows the process supervisor to identify the redundant modules that are still functional and can take part in the current arbitration cycle. The process supervisor also relays its own module rank to the redundant modules as part of the handshake request.

Since each OBC will likely enter the Handshake phase at different times after the spacecraft bus reset, the process supervisor must broadcast this handshake request more than once. The multiple broadcasts of the handshake request account for the account for the jitter in OBC bootup time. It also accommodates transient errors on the communication channel between the OBCs which may result in unsuccessful delivery of the handshake request or its acknowledgement.

After transitioning into this state, the process supervisor starts a software watchdog timer with a timeout value of T_{HDSK_S} and begins periodically broadcasting the handshake request message. The handshake requests are broadcast until all redundant modules acknowledge the request or the watchdog timer T_{HDSK_S} expires. The timeout value T_{HDSK_S} is computed as follows:

$$T_{HDSK\_S} = T_{BaseValue\_S} \times (N_m - H_0 + 1) \tag{4.1}$$

where

$$T_{BaseValue\_S} = N_{HDSKRequests} \times \left(T_{HDSKRequest} + T_{HDSKRequestInterval}\right) \tag{4.2}$$

$N_{HDSKRequests}$ is the total number of handshake requests that are broadcast by the process supervisor, $T_{HDSKRequest}$ is the time required to broadcast a single handshake request and $T_{HDSKRequestInterval}$ is the interval between subsequent handshake requests. N_m is the total number of OBCs deployed on the spacecraft and H_0 is the OBC's module rank, as defined in Chapter 3 . The base value of timer $T_{BaseValue\_S}$, represents step-size of the watchdog timer and the module rank of the OBC serves as a multiplier for this base value.

The number of handshake requests along with the interval between the start of each request forms the base timeout value of the watchdog timer, while the OBC's module rank acts as a multiplier for this timer. All the parameters used to compute the timeout value of T_{HDSK_S} are immutable, and therefore, this value is also immutable for any given module. As the mission progresses, the current process supervisor might experience a failure mode and get permanently disabled by one of the redundant OBCs. Lemma 4-1 shows that the different timeout values of T_{HDSK_S} for all redundant modules will be ordered such that an OBC with a lower module rank will have a larger timeout value. This lemma holds true for any watchdog timer will be used while setting up the watchdog timers for the redundant modules to ensure that they expire sequentially.

Lemma 4-1: If the watchdog timeout value for an OBC with a module rank of *k* has the form:

$$T^k = T_{Offset} + [T_{Base} \times (N_m - k + 1)]$$

where T_{Offset} and T_{Base} are constants and N_m is the total number of OBCs, then the watchdog timeout value of the same form will be smaller on an OBC with a module rank of *k+1*, such that:

$$T^k - T^{k+1} = T_{Base}$$

Proof

The watchdog timeout value for both OBCs can be expressed as:

$$T^{k+1} = T_{Offset} + [T_{Base} \times (N_m - (k + 1) + 1)] \quad (4.3)$$

$$T^k = T_{Offset} + [T_{Base} \times (N_m - k + 1)] \quad (4.4)$$

Subtracting Equation (4.4) from Equation (4.3):

$$T^{k} - T^{k+1} = T_{Base} \times [(N_m - k + 1) - (N_m - (k + 1) + 1)]$$

$$\Rightarrow T^{k} - T^{k+1} = T_{Base} \quad (4.5)$$

Lemma 4-1 shows the time afforded to the process supervisor for successfully communicating with the other redundant modules scales up linearly every time a redundant module with a lower module rank takes over as the process supervisor. Since the total number of available redundant modules reduces after this switchover, the longer timeout value provided by the process supervisor increases the chances that the remaining redundant modules can resolve any transient errors.

As mentioned earlier, having a redundant OBC with a larger module rank that that of the current process supervisor can extend the execution time of this sub-phase. In this case, the only way for the supervisor to confirm that all available modules have acknowledged the handshake is if ***all*** OBCs are functioning nominally. Else, the T_{HDSK_S} timer will always need to expire before the process supervisor can move forward with the arbitration cycle, forcing the worst-case execution time for this state. On the other hand, if the process supervisor has the highest module rank of all available OBCs, it can easily confirm that all redundant modules with lower module ranks have acknowledged the handshake request.

The process supervisor can transition into one of two states namely, the *Sync* state or the *Self Assess* state. If the watchdog timer expires before receiving an acknowledgement from all available redundant modules, the process supervisor attempts to power down the unresponsive OBCs for the current arbitration cycle. The process supervisor continues to command the power distribution module until all unresponsive modules have been powered down. This also powers down the redundant modules that did not acknowledge the handshake request because their inter-

OBC communication channel was not functioning, ensuring that their software watchdog timers are terminated. Finally, this step ensures that the process supervisor is capable of successfully commanding the power distribution module, which is essential not only for the arbitration cycle, but also for mission operations. The discussion of the redundant module's *Sentinel* state (Section 4.3.3.1) will demonstrate how the system can recover in the event the process supervisor is unable to successfully command the power distribution module. It should be noted that the redundant modules are not disabled permanently at this stage, since they might have experienced a SEE which might get resolved after the next bus reset. Moreover, permanently disabling one of the redundant modules would introduce synchronization issues for future process supervisors which might not know which modules will power up after the spacecraft bus is reset.

If the process supervisor succeeds in powering down all unresponsive redundant modules, then it checks the number of redundant modules that acknowledged the handshake request. If there is at least one other OBC capable of taking part in the arbitration cycle, then the process supervisor transitions into its *Sync* State. Alternatively, if all redundant modules acknowledge the handshake request prior to the expiry of the T_{HDSK_S} watchdog timer, then the process supervisor stops the timer. Even though the process supervisor did not have to command the power distribution module to switch off any of the redundant modules in this case, it should still validate its ability to successfully communicate with the power distribution module. Once the communication channel with the power distribution module has been validated, the process supervisor transitions into its *Sync* sub-phase.

Alternatively, if the process supervisor did not receive a handshake acknowledgement from any of the available redundant modules, it transitions into the *Self Assess* state. Since none of the

other modules acknowledged the handshake request, the current arbitration cycle terminates after the *Self Assess* state.

4.3.2.3 The Sync State

Once the process supervisor transitions into this sub-phase, it broadcasts an end-of-phase message to all redundant modules. This end-of-phase message synchronizes the transition of all OBCs into the next phase of the arbitration cycle i.e., the Health Assessment phase. The synchronization via the end-of-phase message is not meant to establish a strict lockstep between the OBCs. Instead, it ensures that all modules transition from the Handshake phase into the Health Assessment phase within a known window of time, denoted by $T_{HDSKToHLTH\_S}$. Similar to the Handshake phase, the Health Assessment phase also implements software watchdog timers to monitor off nominal behavior of the OBCs and resets the arbitration cycle if required. Since the window within which all modules transition into the Health Assessment phase is known, the watchdog timers in the Health Assessment phase can be setup independently of the execution time required to reach the end of the Handshake phase. The end-of-phase message also signals to the redundant modules that the current process supervisor has successfully tested the communication channel with the power distribution module during the *Request* state.

$T_{HDSKToHLTH\_S}$ is defined using the number of end-of-phase messages sent by the process supervisor, denoted by $N_{EOPMessages\_S}$, the time required to broadcast a single end-of-phase message, denoted by $T_{EOPMessage\_S}$, and the interval between each end-of-phase message, denoted by $T_{EOPInterval\_S}$. The transition window can be expressed as:

$$T_{HDSKToHLTH\_S} = N_{EOPMessages\_S} \times \left(T_{EOPMessage\_S} + T_{EOPInterval\_S}\right) \quad (4.6)$$

At this stage, the chances of not receiving an acknowledgement from the redundant modules is extremely low since the unresponsive modules were shut down during the *Request* sub-

phase. However, a SEE or a transient error on the communication bus might interfere with this end-of-phase message. So, a small number of end-of-phase messages are sent back-to-back. If a redundant module fails to acknowledge even one of these end-of-phase messages, then the process supervisor powers down that module. The maximum execution time of the *Sync* state $T_{HDSKSync\_S}$ is therefore represented by the summation of the phase transition interval and the maximum time given to the process supervisor to power down all unresponsive redundant OBCs, denoted by T_{PDM}:

$$T_{HDSKSync\_S} = T_{HDSKToHLTH\_S} + T_{PDM} \quad (4.7)$$

The transition window $T_{HDSKToHLTH\_S}$ and the execution time of the Sync state $T_{HDSKSync\_S}$, as expressed in Equations (4.6) and (4.7) respectively, will be used to setup the watchdog timers in the next phase of the arbitration cycle. The process supervisor updates the list of redundant modules that are transitioning into the next phase of the arbitration cycle i.e., the Health Assessment phase, after which the process supervisor itself transitions into the Health Assessment phase. Note that if the process supervisor doesn't need to command the power distribution module to power down any redundant OBCS, then an additional delay of T_{PDM} must be added to ensure a consistent transition time between the two phases.

4.3.3 The Redundant Module Branch

4.3.3.1 The Sentinel State

After the *Role Determination* state, all redundant modules transition into this state and wait for the handshake request to be initiated by the process supervisor. The primary purpose of this state is to reset the arbitration cycle if the process supervisor becomes unresponsive. Similar to the process supervisor, each of the redundant modules also use a software watchdog timer to guard against any off-nominal behavior exhibited by the process supervisor. Each redundant module sets

a unique timeout value for its watchdog based on its module rank. These timeout values, denoted by T_{HDSK_R}, have the following attributes:

P 4-1. The watchdog timer on the redundant OBC shall have a timeout value greater than the timeout value of the software watchdog timer running on the process supervisor.

P 4-2. The watchdog timer of a redundant OBC with a module rank *k* shall expire before any watchdog timer on redundant OBC with a module rank less than *k*.

Property (P 4-1) helps detect if the process supervisor has become unresponsive. If the redundant module does not receive a handshake request because its communication channel with the process supervisor is not working, the process supervisor's timer would expire before the redundant module's watchdog timer, and the redundant module will be powered down. However, if the process supervisor becomes unresponsive, its software watchdog timer will not expire by definition, therefore allowing the redundant module to reset the arbitration cycle.

Property (P 4-2) establishes a deterministic order in which the redundant modules attempt to reset the arbitration cycle which ensures that the redundant OBC with the highest module rank will take over as the next potential process supervisor.

The timeout value of the form expressed in Lemma 4-1 satisfies both these properties. However, the base value of the timer for the redundant modules must be increased since the redundant modules expect two messages from the process supervisor before they stop their watchdog timer, namely the handshake request and the end-of-phase message. This increase in the base value of the timer must also allow the process supervisor to attempt communication with the power distribution module. The software watchdog timeout value T_{HDSK_R} can therefore be expressed as:

$$T_{HDSK\_R} = T_{HDSKSync\_S} + \left[T_{BaseValue\_R\_HDSK} \times (N_m - H_0 + 1)\right] \quad (4.8)$$

$T_{HDSKSync\_S}$ represents the execution time of the *Sync* state for the process supervisor and is expressed in Equation (4.7). This constant offset accounts for the time required by the process supervisor to broadcast the end-of-phase messages. $T_{BaseValue\_R\_HDSK}$ represents the base value of the redundant module's watchdog timer and is expressed as:

$$T_{BaseValue\_R\_HDSK} = T_{BaseValue\_S} + T_{PDM} \tag{4.9}$$

$T_{BaseValue\_S}$ represents the base value of the process supervisor's watchdog timer, expressed in Equation (4.2). The addition of T_{PDM} to the base value of the redundant module accounts for the time required to reset the spacecraft bus if the process supervisor becomes unresponsive.

From Lemma 4-1, it follows that the minimum difference between the values of T_{HDSK_R} on any two redundant modules is at least $T_{BaseValue\_R\_HDSK}$, where OBCs with lower module ranks have larger timeout values. This satisfies property (P 4-2) of the watchdog timers for redundant modules. The timeout values of the redundant watchdogs and the process supervisor have similar forms, but different timer offsets and base values. All redundant OBCs have lower module rank than the current process supervisor. Consider that the current process supervisor has a module rank of *k+1*. From Lemma 4-1, the OBC with the smallest watchdog timeout value amongst all redundant modules has a module rank exactly one less than the current process supervisor i.e., a module rank of *k*. Lemma 4-2 shows an intermediary proof that will be used to demonstrate that the redundant watchdog timers will also adhere to property (P 4-1), that is, the redundant watchdog timer will always expire after the process supervisor watchdog.

Lemma 4-2: If the watchdog timeout value for a process supervisor with a module rank of *k+1* has the form:

$$T^{k+1} = T^{k+1}_{Offset} + \left[T^{k+1}_{Base} \times (N_m - (k+1) + 1)\right] \tag{4.10}$$

and the watchdog timeout value of a redundant OBC with module rank *k* has the form:

$$\mathrm{T}^{k} = \mathrm{T}^{k}_{\mathrm{Offset}} + \left[\mathrm{T}^{k}_{\mathrm{Base}} \times (\mathrm{N_m} - k + 1)\right] \tag{4.11}$$

where $\mathrm{T}^{k+1}_{\mathrm{Offset}}$, $\mathrm{T}^{k}_{\mathrm{Offset}}$, $\mathrm{T}^{k+1}_{\mathrm{Base}}$ and $\mathrm{T}^{k}_{\mathrm{Base}}$ are constants and N_m is the total number of OBCs, then:

$$\mathrm{T}^{k} > \mathrm{T}^{k+1}$$

so long as the following conditions are met:

$$\mathrm{T}^{k}_{\mathrm{Offset}} \geq \mathrm{T}^{k+1}_{\mathrm{Offset}} \; and \; \mathrm{T}^{k}_{\mathrm{Base}} \geq \mathrm{T}^{k+1}_{\mathrm{Base}} \tag{4.12}$$

Proof

Subtracting Equation (4.10) from Equation (4.11):

$$\mathrm{T}^{k} - \mathrm{T}^{k+1} = \mathrm{T}^{k}_{\mathrm{Offset}} + \left[\mathrm{T}^{k}_{\mathrm{Base}} \times (\mathrm{N_m} - k + 1)\right] - \mathrm{T}^{k+1}_{\mathrm{Offset}}$$

$$- \left[\mathrm{T}^{k+1}_{\mathrm{Base}} \times (\mathrm{N_m} - (k+1) + 1)\right]$$

Re-arranging the expression:

$$\mathrm{T}^{k} - \mathrm{T}^{k+1} = \left[\mathrm{T}^{k}_{\mathrm{Offset}} - \mathrm{T}^{k+1}_{\mathrm{Offset}}\right] + \left[\left(\mathrm{T}^{k}_{\mathrm{Base}} - \mathrm{T}^{k+1}_{\mathrm{Base}}\right) \times (\mathrm{N_m} - k + 1)\right] - \mathrm{T}^{k+1}_{\mathrm{Base}} \tag{4.13}$$

From the conditions given in Equation (4.12):

$$\mathrm{T}^{k}_{\mathrm{Offset}} \geq \mathrm{T}^{k+1}_{\mathrm{Offset}}$$

$$\Rightarrow \mathrm{T}^{k}_{\mathrm{Offset}} - \mathrm{T}^{k+1}_{\mathrm{Offset}} \geq 0 \tag{4.14}$$

Similarly,

$$\mathrm{T}^{k}_{\mathrm{Base}} - \mathrm{T}^{k+1}_{\mathrm{Base}} \geq 0 \tag{4.15}$$

Finally, the base value of any of the watchdog timers cannot be zero or negative:

$$\mathrm{T}^{k+1}_{\mathrm{Base}} > 0 \tag{4.16}$$

Adding Equations (4.14), (4.15) and (4.16):

$$\left[\mathrm{T}^{k}_{\mathrm{Offset}} - \mathrm{T}^{k+1}_{\mathrm{Offset}}\right] + \left[\left(\mathrm{T}^{k}_{\mathrm{Base}} - \mathrm{T}^{k+1}_{\mathrm{Base}}\right) \times (\mathrm{N_m} - k + 1)\right] - \mathrm{T}^{k+1}_{\mathrm{Base}} > 0$$

$$\Rightarrow \mathrm{T}^{k} - \mathrm{T}^{k+1} > 0$$

$$\Rightarrow T^{k} > T^{k+1} \quad (4.17)$$

Therefore, so long as the offset and base values on the redundant watchdog timer are larger than the offset and base timer values on the process supervisor respectively, the redundant watchdog timer will always expire after the process supervisor. For the handshake phase, the offset value of the process supervisor's watchdog is zero as shown in Equation (4.1), while the offset of the redundant modules is $T_{HDSKSync\_S}$ as shown in Equation (4.8), satisfying this condition by default. Equation (4.9) shows that the base value of the redundant module's watchdog is always greater than the base value of the process supervisor's watchdog. Therefore, the watchdog timers on the redundant modules will always expire after the watchdog timer on the process supervisor.

As the redundant modules transition into the *Sentinel* state, they each start the watchdog timer T_{HDSK_R} and wait for the handshake request from the process supervisor. Once a redundant module receives the handshake request, it acknowledges the request and subsequently waits for the end-of-phase message. While sending the handshake acknowledgement, the redundant OBC identifies itself using its module rank. Once the module receives the end-of-phase message, it transitions into its *Sync* State.

However, if the redundant module fails to receive either the handshake request or the end-of-phase message before its watchdog timer expires, it increments the number of consecutive incomplete arbitration cycles it has spent as a redundant module. The number of consecutive incomplete arbitration cycles, denoted by $N_{CycleCount}$ (and defined in section 4.3.2.1), is stored independently by each redundant module in their own persistent storage. This serves as a backup to the number of incomplete arbitration cycles stored by the process supervisor and allows the redundant modules to assume control even if a process supervisor fails to disable its own arbitration cycle.

If the current value of $N_{CycleCount}$ is equal to the maximum number of allowed consecutive incomplete arbitration cycles, denoted by $N_{MaxCycleCount}$ (also defined in section 4.3.2.1), the redundant module transitions into its *Reset* state where it attempts to take over as the process supervisor. Otherwise, the redundant module simply resets the spacecraft bus and allows the current process supervisor to attempt another arbitration cycle. The deterministic order in which the watchdog timers expire ensures that only one redundant OBC transitions into the *Reset* sub-phase at any given time.

4.3.3.2 The Reset State

If the current process supervisor fails to complete $N_{MaxCycleCount}$ consecutive arbitration cycles, a redundant OBC can transition into this state. Since all OBCs with higher module ranks have either failed to complete the arbitration cycle as the process supervisor or have themselves failed to take over as the next process supervisor, the redundant OBC in this state permanently disables all OBCs with higher module ranks. This ensures that the current process supervisor, and any other redundant OBC with a higher module rank, cannot take part in the next arbitration cycle. Once these OBCs have been successfully disabled, the redundant OBC updates the number of modules that can take part in the next arbitration cycle as well as its own role as the process supervisor of the arbitration cycle, and then resets the spacecraft bus. After the spacecraft bus reset, this OBC will take over as the process supervisor and initiate the handshake process with the remaining redundant modules. If this OBC is that last available module (module rank of 1), the arbitration cycle will end during the Module Initialization phase, and this OBC will directly assume command of the spacecraft bus.

It should be noted when an OBC is in the process of the resetting the spacecraft bus in this state, the watchdog timers on the redundant OBCs with lower module ranks are still counting down. Following from Lemma 4-1, the difference in time between two OBCs transitioning into

the *Reset* state will be at least $T_{BaseValue\_R\_HDSK}$. This base value of the watchdog timer includes the timeout to successfully communicate with the power distribution module. Therefore, the OBC currently in *Reset* phase has sufficient time to complete execution of this state before the watchdog timer expires on any other redundant module.

4.3.3.3 The Sync State

If the process supervisor successfully establishes a handshake with the redundant module and relays the end-of-phase message before the T_{HDSK_R} watchdog timer expires, the redundant module transitions into this state instead of the *Reset* state. In this state, the redundant module stops the T_{HDSK_R} watchdog timer and acknowledges the end-of-phase message. As noted in the Sync state of the process supervisor, an SEE might inhibit successful acknowledgment of the end-of-phase message. So, a small number of acknowledgements are sent back-to-back, much like the periodic transmission of the end-of-phase message itself.

Similar to the process supervisor, the transition window for a redundant module from the Handshake phase to the Health Assessment phase, denoted by $T_{HDSKToHLTH\_R}$, is based on the total number of end-of-phase acknowledgements $N_{EOPAcks\_R}$, the time required to broadcast a single end-of-phase message, denoted by T_{EOPAck_R}, and the interval between successive acknowledgement messages $T_{EOPInterval\_R}$. $T_{HDSKToHLTH\_R}$ is expressed as:

$$T_{HDSKToHLTH\_R} = \left[N_{EOPAcks\_R} \times \left(T_{EOPAck\_R} + T_{EOPInterval\_R}\right)\right] - T_{EOPInterval\_R} \quad (4.18)$$

The transition window for the redundant module considers one less acknowledgement interval period than is the number of messages transmitted since the redundant module does not wait after the transmission of the last end-of-phase acknowledgement.

The purpose of the *Sync* state is to ensure that all OBCs transition to the next phase of the arbitration cycle within a preset window of time. Since the process supervisor is responsible for

powering down the OBCs that do not acknowledge the end-of-phase, it is always the last OBC to transition into the next phase. In order to ensure that this transition condition holds true even if a redundant module acknowledges the last end-of-phase message transmitted by the process supervisor, the entire transition window for the redundant module must be smaller than the interval $T_{EOPInterval\_S}$ used by the process supervisor between successive end-of-phase messages, such that $T_{HDSKToHLTH\_R} < T_{EOPInterval\_S}$. After acknowledging the end-of-phase messages, the redundant module transitions into the Health Assessment phase.

4.3.4 The Process Supervisor's Self Assess State

If the process supervisor cannot establish a handshake with any of the other redundant OBCs before its watchdog timer expires, it powers down all redundant modules and transitions into this state. The failure to establish a handshake can be attributed to one of two factors, either all remaining redundant modules have become unresponsive, or the communication channel between the OBCs has stopped functioning. It is not trivial for the process supervisor to determine which one of these two factors inhibited the handshake. Moreover, if the communication channel has indeed stopped working, the process supervisor cannot determine whether it itself is the healthiest OBC to command the spacecraft bus. However, the process supervisor can determine its own Health Quotient. If the evaluation for all critical health metrics of the process supervisor is greater than their minimum performance thresholds, the OBC is deemed capable of executing critical mission operations, irrespective of whether it is the healthiest module on the spacecraft to do so. The minimum performance thresholds for the health metric are determined prior to launch based on mission requirements and system design.

If the process supervisor meets this threshold, it simply takes command of the spacecraft bus and continues with mission operations. However, if the process supervisor fails to meet this

threshold, it adjusts the next arbitration cycle to allow one of the redundant modules to take over. Since the process supervisor cannot be certain if any of the other redundant modules are still functioning nominally, the process supervisor does not remove itself from the arbitration process. Instead, the additional execution delay T_{Delay} is enabled for the *Configuration* state of the next arbitration cycle. This additional delay is set equal to the longest watchdog timeout on any of the redundant modules in the Handshake phase. Following from the Lemma 4-1, the longest timeout value will be on the OBC with a module rank of 1. Therefore, T_{Delay} is expressed as:

$$T_{Delay} = T_{HDSKSync\_S} + \left(T_{BaseValue\_R\_HDSK} \times N_m\right) \tag{4.19}$$

$T_{BaseValue\_R\_HDSK}$ represents the base value of the watchdog timer on redundant modules, expressed in Equation (4.9). The offset for the redundant module timers represents the execution time of the *Sync* phase.

Once the process supervisor enables this delay, it marks the *Self Assess* state as expired, saves both these parameter to persistent memory and resets the spacecraft bus.

After the bus reset, the current supervisor is suspended within the *Configuration* state until the T_{Delay} timer expires. If any other redundant module is still functional on the spacecraft bus, it would have transitioned into the Handshake phase without this additional delay and its watchdog timer will expire before the process supervisor transitions into the Handshake phase. Such a redundant module would eventually enter its *Reset* sub-phase, and permanently disable the current process supervisor. Subsequently, the redundant module would itself take over as the process supervisor of the next arbitration cycle.

On the other hand, if all other OBCs have become unresponsive, the process supervisor will continue executing the arbitration cycle after the T_{Delay} timer expires. It will subsequently transition into the Handshake phase, and similar to the previous arbitration cycle it will not be able

to establish a handshake with any of the redundant modules. Once its watchdog timer expires in the *Request* state, the process supervisor will once again power down all other OBCs and transition into the *Self Assess* state. Since this state had been marked as expired in persistent configuration, the process supervisor is now aware that no other module is available to command the spacecraft bus. Even though the module itself failed to meet its minimum threshold for the Health Quotient, it still takes command of the spacecraft bus as it is the only functioning OBC on the spacecraft.

4.4 HEALTH ASSESSMENT PHASE

Once all the OBCs participating in the current arbitration cycle have transitioned into the Health Assessment phase, the process supervisor commences the process of mediating access to the spacecraft bus which allows each OBC to compute its Health Quotient. The process supervisor begins by computing its own Health Quotient and broadcasts it to all other OBCs. Next, the process supervisor sequentially signals each of the redundant modules to compute their own Health Quotients. Similar to the process supervisor, the redundant modules also broadcast their Health Quotient to every other OBC.

As with the Handshake phase, the Health Assessment phase also relies on software watchdog timers for guarding against off-nominal behavior. While the Handshake phase would have already powered down all unresponsive OBCs during the Handshake phase, the remaining OBCs are still vulnerable to SEEs and transients along the communication lines. The watchdog timers ensure that these effects do not stall the arbitration cycle.

The watchdog timer used during the Handshake phase are disabled before an OBC transitions into the Health Assessment phase. Additionally, all OBCs transition from the Handshake phase into the Health Assessment phase within a known transition window. This transition time is incorporated into the timeout values of the watchdog timers used during this

phase which ensures that a watchdog timer cannot affect the arbitration process across the phase boundaries. In other words, the hierarchy in which the watchdog timers expire in the Health Assessment phase cannot be perturbed by any OBC that is still in its Handshake phase.

The logical flowchart outlining the execution sequence of the Health Assessment phase is shown in Figure 4-6.

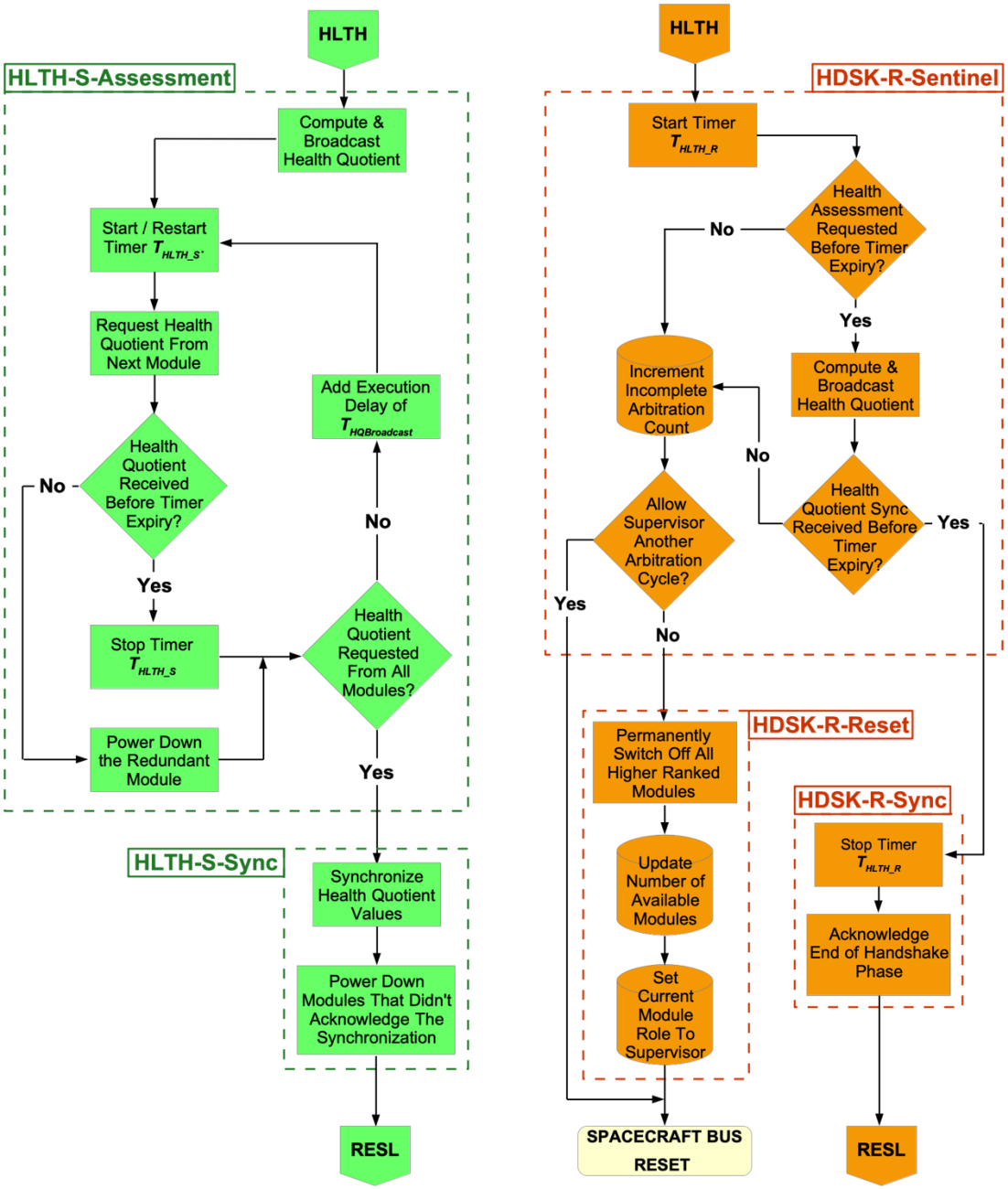

Figure 4-6: The Health Assessment Phase

4.4.1 The Process Supervisor Branch

4.4.1.1 The Assessment State

The process supervisor commences the Health Assessment phase by transitioning into the *Assessment* state and begins computing its Health Quotient. The process supervisor is always the first OBC to compute its Health Quotient which is carried out in three steps. The process starts by enabling communication interfaces to the spacecraft bus which were disable during the Module Initialization phase. The interfaces are only temporarily enabled so that the OBC can assess health metrics that require access to the spacecraft bus. The next step is the aggregation of all health metrics into the Health Quotient of the module (discussed in Chapter 3). The last step of this process disables the communication interfaces again, after which the process supervisor broadcasts its computed Health Quotient value to all redundant modules.

After broadcasting its own Health Quotient, the process supervisor begins mediating access to the spacecraft bus for the redundant modules. The process supervisor has a list of the redundant modules that had successfully acknowledged the end of the Handshake phase and transitioned into the Health Assessment phase. The process supervisor sequentially pings each of these modules and requests their Health Quotient. After requesting the Health Quotient from a redundant module, the process supervisor starts a software watchdog timer with a timeout value of T_{HLTH_S}. The timeout value denotes the maximum time allowed for any OBC to compute its Health Quotient, and can be represented as:

$$T_{HLTH\_S} = N_{HQRequests} \times \left(T_{HQRequest} + T_{HQCompute} + T_{HQBroadcast}\right) \quad (4.20)$$

$N_{HQRequests}$ represents the number of times that the process supervisor attempts to re-request the Health Assessment from the redundant module to account for any transient errors. $T_{HQRequest}$ represents the time required by the process supervisor to transmit a request to the

redundant OBC for its Health Quotient while $T_{HQCompute}$ represents the time required by an OBC to compute its Health Quotient. $T_{HQCompute}$ can be expressed as the summation of the time intervals required for computing individual health metrics that constitute the Health Quotient. If N_H is the largest rank index assigned to any health metric and $t_{\langle k \rangle}$ represents the maximum time interval required for determining every health metric with a rank index of *k*, the total time required to compute the OBC's Health Quotient is expressed as:

$$T_{HQCompute} = \sum_{k=0}^{N_H} t_{\langle k \rangle} \tag{4.21}$$

Finally, $T_{HQBroadcast}$ denotes the maximum time an OBC spends broadcasting its Health Quotient. It can be expressed in terms of the maximum number of Health Quotient broadcast messages, denoted by $N_{BroadcastMessages}$, the time required for a single transmission of the Health Quotient value, denoted by $T_{HQTransmit}$, and the interval between successive transmission of the Health Quotient, denoted by $T_{BroadcastInterval}$:

$$T_{HQBroadcast} = \left[N_{BroadcastMessages} \times \left(T_{HQTransmit} + T_{BroadcastInterval}\right)\right] - T_{BroadcastInterval} \tag{4.22}$$

Since the process supervisor is always the last OBC to transition into the Health Assessment phase, the transition times of the redundant modules do not need to be accounted within T_{HLTH_S}.

If a redundant module fails to broadcast its Health Quotient value before the T_{HLTH_S} timer expires, the process supervisor assumes that the module has either hung up or has experienced a latch-up on its communication interface. At this stage, it is not feasible to reset that specific OBC as it will transition into the Handshake phase after the reset and subsequently expect a handshake

request. Similarly, it is not feasible to reset the entire arbitration cycle just to reset one of the redundant OBCs. Therefore, the process supervisor powers down the redundant module and removes it from the current arbitration cycle. However, if the redundant module responds with its Health Quotient before the watchdog timer expires, then the process supervisor halts the timer.

In either case, the process supervisor proceeds to ping the next redundant module on its list and repeats the process of requesting the Health Quotient.

The T_{HLTH_S} watchdog timer is restarted every time to provide each redundant module the same amount of time to compute its Health Quotient. Since the process supervisor runs this watchdog timer on a loop, the maximum execution time of the *Assessment* sub-phase is expressed as:

$$T_{AssessmentExecution} = \left(T_{HLTH\_S} + T_{PDM} + T_{HQBroadcast}\right) \times H_S \qquad (4.23)$$

where H_S represents the module rank of the process supervisor. T_{PDM} accounts for the time interval to power down the redundant OBCs that do not respond to the health assessment request. The Health Quotient broadcast time $T_{HQBroadcast}$, as expressed in Equation (4.22), is also added in every loop before the process supervisor pings the next redundant module. The *Assessment* state is another part of the arbitration cycle where the execution time is optimized by ensuring that the process supervisor has the largest module rank since the number of redundant modules participating in the current arbitration cycle is guaranteed to be less than H_S. Once the process supervisor has requested the Health Quotient value from all functional redundant modules, it transitions into its *Sync* state.

4.4.1.2 The Sync State

The functionality of the *Sync* state in the Health Assessment phase is identical to that of the *Sync* state in the Handshake phase. It ensures that every OBC transitions into the next phase of

the arbitration process within a known transition window. However, instead of a broadcasting a generic end-of-phase message, the process supervisor synchronizes the list of Health Quotient values amongst all active OBCs. This accounts for any Health Quotient transmissions that some of the redundant modules may have dropped due to transients on the communication channel. It also accounts for any Health Quotient transmissions that the process supervisor may have dropped due to the same reason, but other redundant modules may have successfully received. Since the process supervisor always powers down any modules from which it does not receive the Health Quotient values, it is therefore the only OBC that is guaranteed to have the list of Health Quotient values for all OBCs that are active in the current arbitration cycle. Synchronization of this list ensures that all OBCs analyze the same list of Health Quotient values while performing the comparative health assessment.

Similar to the transition window of the Handshake phase expressed in Equation (4.6), the transition window to the next phase of the arbitration cycle (the Arbitration Resolution phase), denoted by $T_{HLTHToRESL\_S}$, can be expressed as:

$$T_{HLTHToRESL\_S} = N_{SyncMessages\_S} \times \left(T_{SyncMessage\_S} + T_{SyncInterval\_S}\right) \quad (4.24)$$

$T_{SyncMessage\_S}$ represent the maximum time required to relay the entire list of Health Quotient values. The synchronization message is repeated periodically after an interval of $T_{SyncInterval\_S}$ for a total of $N_{SyncMessages\_S}$ synchronization attempts. Similar to the transition between the Handshake phase and the Health Assessment phase, the process supervisor expects the redundant modules to acknowledge the synchronization of the Health Quotient values and transition themselves into the Arbitration Resolution phase. At the end of the $T_{HLTHToRESL\_S}$ transition window, the process supervisor powers down all redundant modules that do not send this acknowledgement.

Similar to the execution time of the *Sync* state for the Handshake phase, expressed in Equation (4.7), the maximum execution time of the *Sync* state for the Health Assessment phase, denoted by $T_{HLTHSync\_S}$, can be expressed as:

$$T_{HLTHSync\_S} = T_{HLTHToRESL\_S} + T_{PDM} \quad (4.25)$$

Unlike the *Sync* state in the Handshake phase however, the process supervisor no longer maintains the list of active modules since only one module will transition out of the next phase of the arbitration cycle. After powering down all unresponsive redundant modules, the process supervisor itself transitions into the Arbitration Resolution phase.

4.4.2 The Redundant Module Branch

The functionality of the redundant modules within the Health Assessment phase is almost identical to their functionality within the Handshake phase, discussed in section 4.3.3. However, there are a few key differences between the redundant module branches in both these phases. The first difference is the timeout value of the software watchdog timers used during this phase to monitor the process supervisor. The key difference is the response of the redundant modules to a successful ping from a nominally operating process supervisor. Instead of broadcasting a handshake acknowledgment, as done in the Handshake phase, the redundant modules compute and broadcast their Health Quotient values. Lastly, the transition window from the Health Assessment phase to the Arbitration Resolution phase is updated based on the time required for the process supervisor to synchronize the Health Quotient values. The discussion in this section will focus on these key differences.

4.4.2.1 The Watchdog Timer

A redundant OBC triggers its watchdog timer as soon as it transitions into the *Sentinel* state of this phase. The timeout value for the watchdog timer, denoted by T_{HLTH_R}, must adhere to

properties (P 4-1) and (P 4-2), as discussed in section 4.3.3.1, and will therefore utilize a similar expression to that presented in Lemma 4-1:

$$T_{HLTH\_R} = T_{HealthAssessment\_S} + \left[T_{BaseValue\_R\_HLTH} \times (N_m - H_0 + 1)\right] \quad (4.26)$$

$T_{HealthAssessment\_S}$ represents the total execution time of the process supervisor branch for the Health Assessment phase and is expressed as:

$$T_{HealthAssessment\_S} = T_{AssessmentExecution} + T_{HLTHSync\_S}$$

The execution time for the *Assessment* state $T_{AssessmentExecution}$ and that of the *Sync* state $T_{HLTHSync\_S}$, are expressed in Equations (4.23) and (4.25) respectively. Since the OBCs perform their health assessment sequentially and the process supervisor may become unresponsive at any time, the watchdog timers on each of the redundant modules must run at least for as long as it takes the process supervisor to transition out of the *Assessment* state transmit the end-of-phase messages. This is achieved by setting the maximum execution time of the process supervisor's *Assessment* state as the constant offset for the T_{HLTH_R} timer. This ensures the watchdog timers on the redundant modules will not expire before the process supervisor transitions out of its *Assessment* state. Recall that the process supervisor broadcasts its module rank during the Handshake process, which allows the redundant modules to compute the value of $T_{HealthAssessment\_S}$.

The base value of the redundant module's watchdog timer $T_{BaseValue\_R\_HLTH}$, must account for transition times between the Handshake phase and the Health Assessment phase, for both the process supervisor and the redundant OBCs. These transition windows denoted $T_{HDSKToHLTH\_S}$ and $T_{HDSKToHLTH\_R}$, are expressed in Equations (4.6) and (4.18) respectively. Accounting for the maximum time allocated for a redundant module to reset the spacecraft bus T_{PDM}, if the process supervisor becomes unresponsive, the base value of the redundant module's watchdog timer is expressed as:

$$T_{BaseValue\_R\_HLTH} = T_{PDM} + T_{HDSKToHLTH\_S} - T_{HDSKToHLTH\_R} \quad (4.27)$$

4.4.2.2 Health Quotient Acknowledgement

If the process supervisor successfully requests Health Quotient from a redundant module, the redundant module broadcasts its Health Quotient to all OBCs taking part in the current arbitration cycle. Similar to the process supervisor, the redundant module must also enable its communication interfaces prior to computing the Health Quotient and disable them again before broadcasting the Health Quotient value. Once the redundant module broadcasts its Health Quotient, it must wait for all other OBCs to complete their health assessment as well. The completion of the Health Assessment phase is signaled by the synchronization of the Health Quotient values initiated by the process supervisor.

If a redundant module receives the synchronization messages before its T_{HLTH_R} watchdog timer expires, the redundant module transitions into the *Sync* state and sends an acknowledgement to the process supervisor. The transition window for the redundant modules from the Health Assessment phase to the Arbitration Resolution phase, denoted by $T_{HLTHToRESL\_R}$, is analogous to their phase transition window between the Handshake phase and the Health Assessment phase. This transition window can be expressed as:

$$T_{HLTHToRESL\_R} =$$

$$\left[N_{SyncMessages\_R} \times \left(T_{SyncMessage\_R} + T_{SyncInterval\_R}\right)\right] - T_{SyncInterval\_R} \quad (4.28)$$

This is analogous to the transition window for the redundant modules from the Handshake phase to the Health Assessment phase, expressed by Equation (4.18). Once again, this transition window must be less than the interval between successive synchronization messages transmitted by the process supervisor, such that $T_{HLTHToRESL\_R} < T_{SyncInterval\_S}$.

If the redundant module's watchdog timer expires before it receives the synchronization message, it assesses the total number of the consecutive incomplete arbitration cycles $N_{CycleCount}$, and it either resets the spacecraft bus which re-starts the arbitration process or transitions into the *Reset* sub-phase, permanently disables the current process supervisor and attempts to take over as the process supervisor for the next arbitration cycle.

4.5 ARBITRATION RESOLUTION PHASE

This is the last phase of the arbitration process where the OBCs compare their Health Quotient values. Both the process supervisor branch as well as the redundant module branch only contain one state each, namely the *Outcome* state. The logical flowchart outlining the execution sequence of the Arbitration Resolution phase is shown in Figure 4-7.

Since the current process supervisor has successfully mediated the exchange of Health Quotients between the OBCs, every module resets its persistent parameter associated with the number of incomplete arbitration cycles conducted by the current process supervisor. Additionally, the process supervisor also re-enables its arbitration cycle if it was disabled during the Handshake phase. Each module sorts the list of Health Quotient values into a monotonic set and determines if it has the largest Health Quotient value. As discussed in Chapter 3 the evaluation of every OBC's Health Quotient will always be unique within the network of redundant OBCs deployed on the spacecraft, ensuring that every OBC can select an unambiguous winner of the arbitration process. If a module does have the largest Health Quotient, it powers down all other OBCs and transitions out of the arbitration process to resume mission operations. Otherwise, the OBC starts a small watchdog timer and waits idly for the arbitration winner to power it down. This watchdog timer

ensures that the arbitration cycle can be reset in case the arbitration winner happens to experience an SEE prior to powering down the other OBCs.

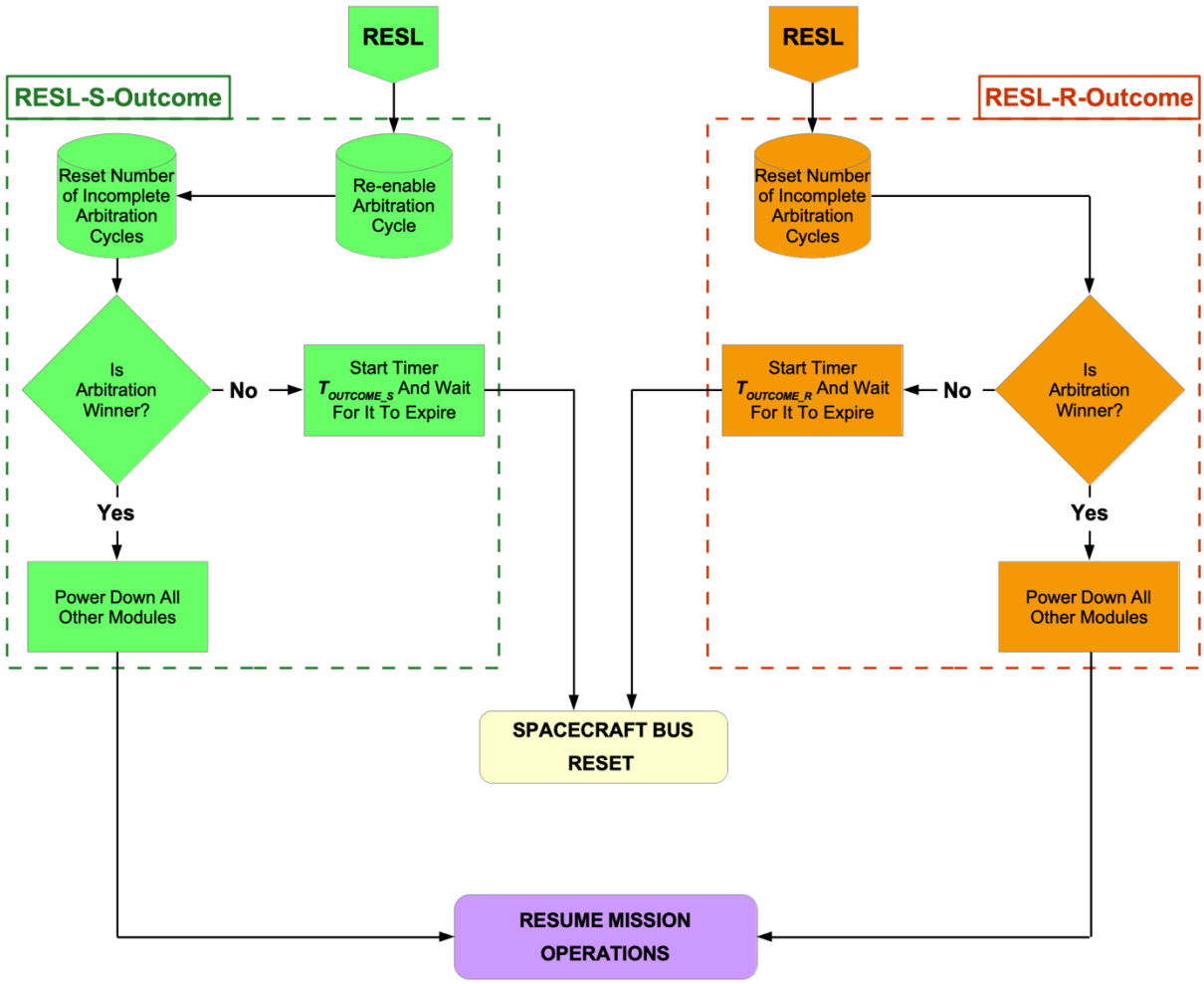

Figure 4-7: The Arbitration Resolution Phase

It should be noted that any one of the OBCs might have the largest Health Quotient and the arbitration winner might not be the current process supervisor. If the current process supervisor loses the arbitration cycle, then it starts a watchdog timer with a timeout value of $T_{OUTCOME\_S}$ while all redundant modules that lost the arbitration cycle start a watchdog timer with a timeout value of $T_{OUTCOME\_R}$. The timeout values are once again setup analogous to the expression used in Lemma 4-1. The maximum time afforded to the arbitration winner power down all other OBCs is denoted

as $T_{Resolve}$. All watchdog timers incorporate this maximum time allocated for resolving the arbitration process as the base value for their timers. The timeout value $T_{OUTCOME\_S}$ is expressed as:

$$T_{OUTCOME_S} = T_{Resolve} \times (N_m - H_0 + 1) \quad (4.29)$$

The redundant module's watchdog timer must account for transition times between the Health Assessment phase and the Arbitration Resolution phase, for both the process supervisor and the redundant OBCs. These transition windows denoted $T_{HLTHToRESL\_S}$ and $T_{HLTHToRESL\_R}$, are expressed in Equations (4.24) and (4.28) respectively. The timeout value $T_{OUTCOME\_R}$ is therefore expressed as:

$$T_{OUTCOME\_R} =$$

$$(T_{PDM} + T_{HLTHToRESL\_S} - T_{HLTHToRESL\_R}) + [T_{Resolve} \times (N_m - H_0 + 1)] \quad (4.30)$$

If any of the watchdog timers expire before the arbitration winner powers down the other OBCs, then the spacecraft bus will be reset, which would re-start the arbitration process. However, if the arbitration winner has not experienced a failure, then it will power down all other OBCs within the $T_{Resolve}$ window and assume command of the spacecraft bus. The arbitration winner is subsequently designated as the commanding OBC, until a module switchover occurs.

A caveat to consider in this phase is the case where a redundant OBC wins the arbitration cycle. Since the Process Supervisor will always be the last OBC to transition into the Arbitration Resolution, the redundant OBC must allow a delay equivalent to the transition windows. This enables the Process Supervisor to reset its persistent configuration (incomplete arbitration cycle count and arbitration enabled flag) before is powered down by the arbitration winner. The delay added is the same as the offset value of the $T_{OUTCOME\_R}$ timer, expressed in Equation (4.30).

Figure 4-8 shows the overall the finite state diagram of the arbitration process, along with transition triggers from one state to the next.

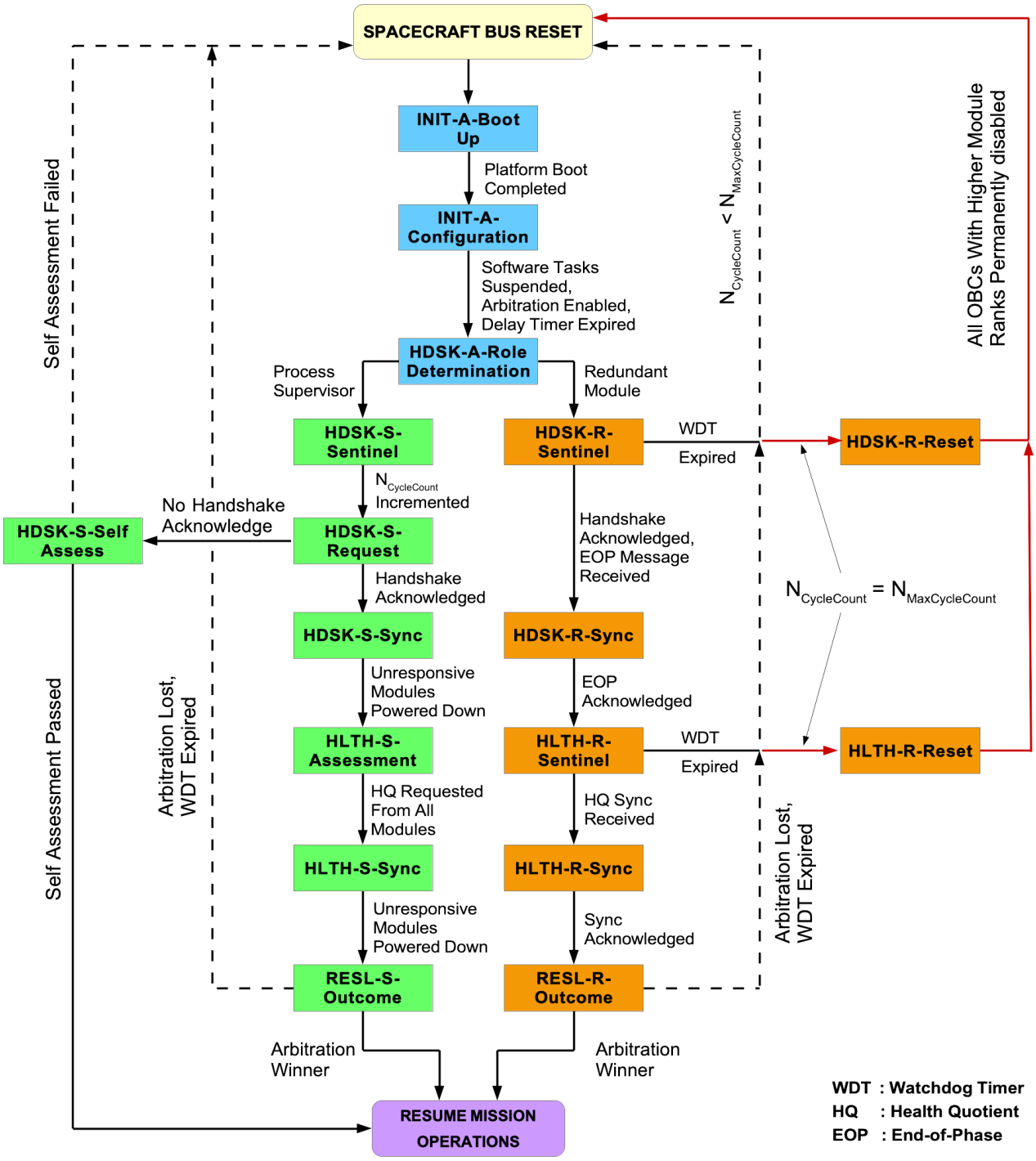

Figure 4-8: Arbitration Architecture Finite State Diagram

The diagram demonstrates how each phase of the arbitration process guards against off-nominal behavior by providing a path to reset the spacecraft bus. The path to resetting the arbitration cycle with the process supervisor is limited by $N_{MaxCycleCount}$, after which a redundant module takes over as the process supervisor. This prevents the architecture from looping due to incomplete arbitration cycles. The *Assessment* state iteratively requests the Health Quotient from each redundant module; however, this iterative loop is also bound by the total number of OBCs deployed within the architecture. The state diagram also shows that all redundant modules follow the same state transitions, irrespective of the number of redundant modules deployed within the architecture. This enables extensibility, allowing the number of redundant modules in the architecture implementation to be increased (or decreased) without changing the state diagram of the arbitration process. The architecture adjusts the watchdog timer values of each redundant module based on its module rank and the total number of OBCs deployed within the architecture.

4.6 SYSTEM PERFORMANCE VS SCALE

4.6.1 System Reliability

The reliability of the proposed redundant system can be modelled similar to that of a N-modular parallel cold redundant system. This reliability of the proposed architecture that relies only on a reset watchdog timer will be compared against the reliability of nominal N-modular redundant architecture that uses a reset timer and an external supervising unit for arbitration. This comparison will demonstrate the gain in reliability provided by the proposed architecture.

Since the system reliability presented in this section is only used for a comparative analysis between the redundancy architectures, a simplified model of component failure is assumed. The purpose of this comparative analysis is to demonstrate the fact that elimination of the external arbiter results in a gain in reliability, and that the reliability of any system that uses an external

arbiter will only asymptotically approach the reliability of a system without an external arbiter. The reliability analysis presented here does not characterize the exact reliability curve of any given redundancy architecture. Moreover, the choice of reliability model used to best characterize a system may change based on the underlying platform used for the redundancy architecture.

The reliability of a unit component over time, denoted by $\mathrm{R(t)} \in [0, 1]$, is commonly modelled as an exponential distribution. Assuming a constant failure rate for the unit, denoted by λ, the reliability of a single OBC over time is expressed as [67] [47]:

$$\mathrm{R(t)} = \mathrm{e}^{-\lambda \mathrm{t}} \tag{4.31}$$

The exponential decay indicates the deterioration in the OBC's reliability due to operational stress. The reliability for the external supervisor, denoted as $\mathrm{R}_{Supervisor}(\mathrm{t})$, can be similarly expressed using their own independent failure rate.

The reliability of a typical N-modular cold redundant system $\mathrm{R_{sys}(t)}$, where at least *m* out of the N devices must be operational, is expressed as [47] [68] [69]:

$$\mathrm{R_{sys}(t)} = \mathrm{R}_{Supervisor}(\mathrm{t}) \times \sum_{k\,=\,\mathrm{m}}^{\mathrm{N}} \frac{\mathrm{N!}}{\mathrm{k!\,(N-k)!}}\ \mathrm{R(t)}^{k}\,[1-\mathrm{R(t)}]^{\,\mathrm{N}-k} \tag{4.32}$$

This shows that the reliability of the external supervising units affects the total permuted reliability of the redundant system i.e., it is the dominating factor in determining the reliability of the system.

The inverse of the OBC's failure rate represents its Mean Time to Failure (MTTF) [47], denoted here as $\mathrm{MTTF_{OBC}}$. Re-writing the Equation (4.31):

$$\mathrm{R(t)} = \mathrm{e}^{-\frac{t}{\mathrm{MTTF_{OBC}}}} \tag{4.33}$$

Similarly, the reliability over time for a supervising unit in typical N-modular redundant architecture $R_{Supervisor}(t)$, is expressed as:

$$R_{Supervisor}(t) = e^{-\frac{t}{MTTF_{Supervisor}}} \tag{4.34}$$

The reliability of the supervising unit can be expressed in terms of the reliability of the units using a scaling factor η which defines the relation between the Mean Time to Failure of the unit and that of the arbiter:

$$MTTF_{Supervisor} = \eta \times MTTF_{OBC} \tag{4.35}$$

where the value of $\eta > 1$, that is, the supervisor has a higher reliability than that of the redundant OBCs themselves. The target redundant architecture only requires 1 out of N available OBCs to be functional. Substituting Equations (4.34) and (4.35) back into Equation (4.32):

$$R_{sys}(t) = R(t)^{\frac{1}{\eta}} \times \sum_{k=1}^{N} \frac{N!}{k!\,(N-k)!}\ R(t)^k\,[1-R(t)]^{N-k} \tag{4.36}$$

A key feature of the novel arbitration architecture presented in this thesis is that it is self-arbitrating. Since the redundant OBCs do not rely on an external arbiter for the proposed arbitration process, the reliability of the arbiter $R_{Supervisor}(t)$ is always 1, increasing the overall reliability of the redundant system. Substituting this reliability into Equation (4.32), the reliability of an N-modular redundant system that employs the proposed redundancy architecture, denoted by $R_{Proposed}(t)$, can be expressed as:

$$R_{Proposed}(t) = \sum_{k=1}^{N} \frac{N!}{k!\,(N-k)!}\ R(t)^k\,[1-R(t)]^{N-k} \tag{4.37}$$

Note that the memory-less watchdog timer responsible for resetting spacecraft power is not included in the reliability analysis, since these timers are present in single OBC architectures as

well, and their failure has equivalent consequences on radiation hardened platforms. The reliability of the proposed architecture provides an advantage over other redundant OBC architectures that require additional supervising units to store the arbitration logic and states. Such arbiters are part of the spacecraft OBC sub-system, and act as a single point of failure for the redundant architecture. The reliability analysis presented here reflects the same.

Figure 4-9 shows the reliability evolution of a dual redundant system. The figure shows that the proposed architecture provides a far superior reliability that than of typical cold redundant architectures that use external arbiters. The time scale has been normalized to multiples of the unit's Mean Time to Failure.

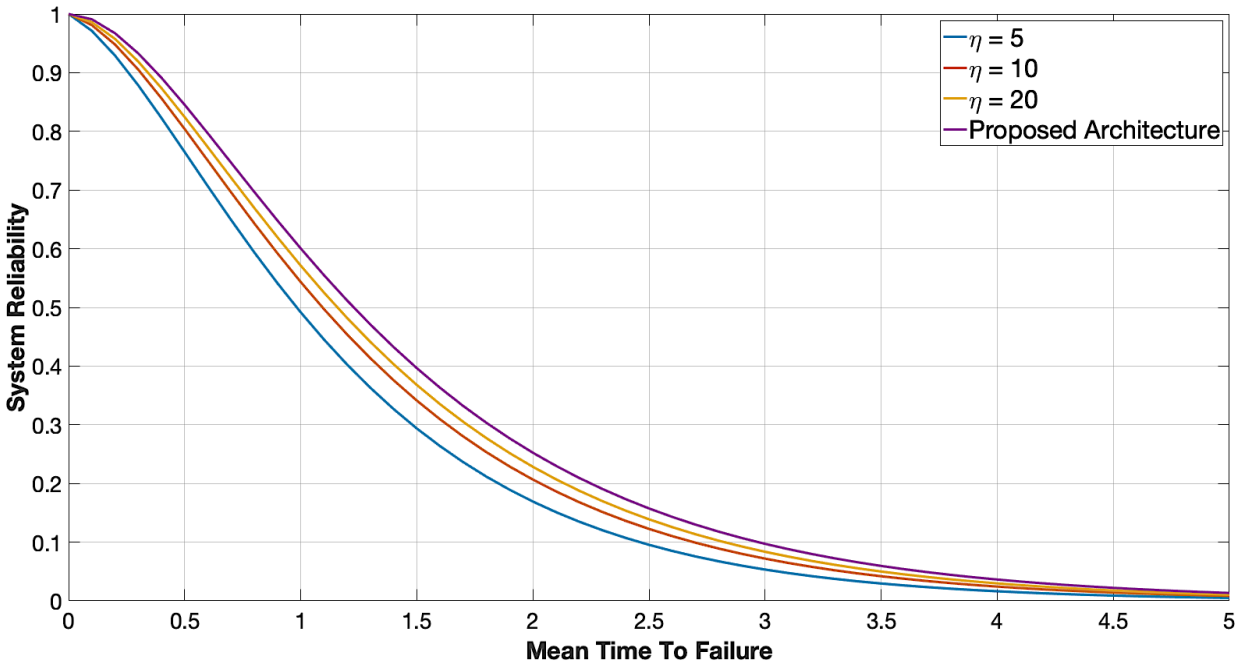

Figure 4-9: Reliability of a Dual Modular Redundant System

The extensibility of the architecture enables further gain in system reliability, as the number of redundant modules in the architecture is increased. Figure 4-10 shows the reliability evolution over time for various configurations of an N-Modular redundant system. The gain in system

reliability can be traded off against the scale of the redundant architecture that can be supported by the engineering budgets for any given mission.

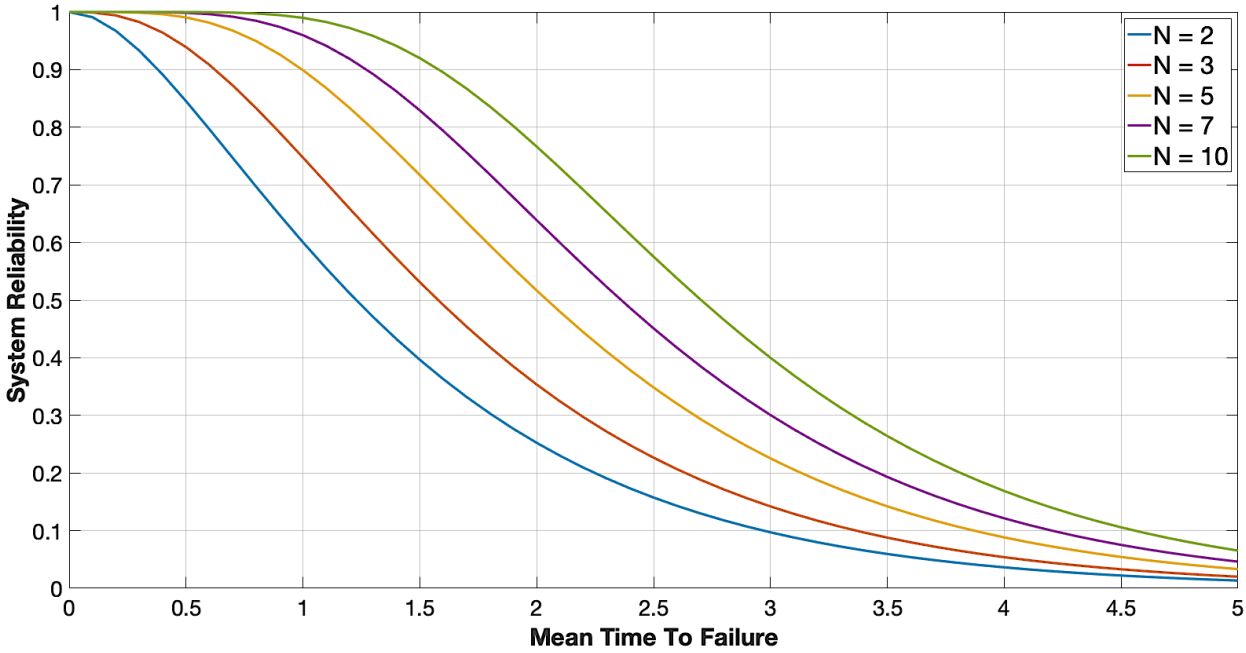

Figure 4-10: System Reliability for Proposed Redundancy Architecture

4.6.2 Multi-Domain Clock Deviation

The redundancy architecture relies on sequential expiration of the watchdog timers to ensure deterministic resolution of the arbitration process. The synchronization states at the end of each phase ensure that the watchdog timers on each OBC start within a defined window of time, and this window is accounted in the timeout value of the watchdog. However, the clocks on each OBC might not necessarily tick at the same rate. It is therefore important to characterize the maximum tolerance on the deviation in clock frequencies amongst the OBCs.

Consider two sequential redundant watchdog timers T^k and T^{k+1} on OBCs with module ranks *k* and *k+1* respectively. Lemma 4-1 demonstrates that difference between these two timers is expressed as:

$$T^k - T^{k+1} = T_{Base}$$

Re-arranging the expression:

$$T^k = T^{k+1} + T_{Base} \tag{4.38}$$

However, this property that ensures the watchdog timers expire sequentially will be violated if the timer T^k ticks faster (resulting in a smaller timeout with respect to a nominal clock) or if T^k ticks slower (resulting in a larger timeout with respect to a nominal clock). The maximum tolerance for clock rate deviation can be computed for the worst-case scenario where both these conditions are true. The clock rate deviation, denoted by δ, represents the ratio between the actual clock rate to expected clock rate of the OBC. This can be translated into the watchdog timer values as:

$$T_{Actual} = \delta\, T_{Expected} \tag{4.39}$$

The maximum allowable deviation in the clock rate of either OBC δ_{max}, must satisfy the following condition:

$$T^k\,(1 - \delta_{max}) = T^{k+1}(1 + \delta_{max})$$

Re-arranging the expression:

$$\frac{T^k}{T^{k+1}} = \frac{(1 + \delta_{max})}{(1 - \delta_{max})} \tag{4.40}$$

Substituting Equation (4.38) into Equation (4.40):

$$\frac{T^{k+1} + T_{Base}}{T^{k+1}} = \frac{(1 + \delta_{max})}{(1 - \delta_{max})}$$

$$\Rightarrow \frac{\mathrm{T_{Base}}}{\mathrm{T}^{k+1}} = \frac{(1+\delta_{max})}{(1-\delta_{max})} - 1$$

$$\Rightarrow \frac{\mathrm{T_{Base}}}{\mathrm{T}^{k+1}} = \frac{2\delta_{max}}{(1-\delta_{max})} \tag{4.41}$$

Substituting the expression for timer T^{k+1} from Equation (4.3) into Equation (4.41):

$$\frac{\mathrm{T_{Base}}}{\mathrm{T_{Offset}} + [\mathrm{T_{Base}} \times (\mathrm{N_m} - (k+1) + 1)]} = \frac{2\delta_{max}}{(1-\delta_{max})}$$

Re-arranging the expression and isolating for δ_{max}:

$$\delta_{max} = \frac{\mathrm{T_{Base}}}{2\,\mathrm{T_{Offset}} + 2\mathrm{T_{Base}}\,(\mathrm{N_m} - k) + \mathrm{T_{Base}}}$$

$$\Rightarrow \delta_{max} = \frac{1}{2\,\frac{\mathrm{T_{Offset}}}{\mathrm{T_{Base}}} + 2\,(\mathrm{N_m} - k) + 1} \tag{4.42}$$

The offset and base values of the timers can be easily tuned such that:

$$\frac{\mathrm{T_{Offset}}}{\mathrm{T_{Base}}} \ll 1$$

Therefore, Equation (4.42) can be re-written as:

$$\Rightarrow \delta_{max} \approx \frac{1}{2\,(\mathrm{N_m} - k) + 1} \tag{4.43}$$

The expression for the clock deviation shows that OBCs with larger module ranks *k* will tolerate a higher deviation in their clock rates. This follows from the fact that those OBCs will have smaller timer values, and consequently the same value of δ_{max} will result in a smaller deviation in the timer value. Therefore, the maximum tolerance in clock rate deviation for a given number of redundant modules will occur for the OBC with module rank 1.

$$\Rightarrow \delta_{max} \approx \frac{1}{2\,\mathrm{N_m} - 1} \tag{4.44}$$

Figure 4-11 shows the maximum clock rate deviation that can be tolerated by the arbitration mechanism as the redundancy architecture is scaled up. The accuracy and stability of typical crystal oscillators in microprocessors and microcontrollers are on order of 10 ppm or better [70] [71] [72], and are therefore not a cause of concern here. Deviations in the execution time of the arbitration instructions may also impact on limiting the scale of the redundant architecture. However, in practical use cases these deviations will be small. The impact on the scale of the architecture imposed due to tolerances in clock rate deviation will be far smaller than the impact of the engineering budget (power and mechanical) of a spacecraft.

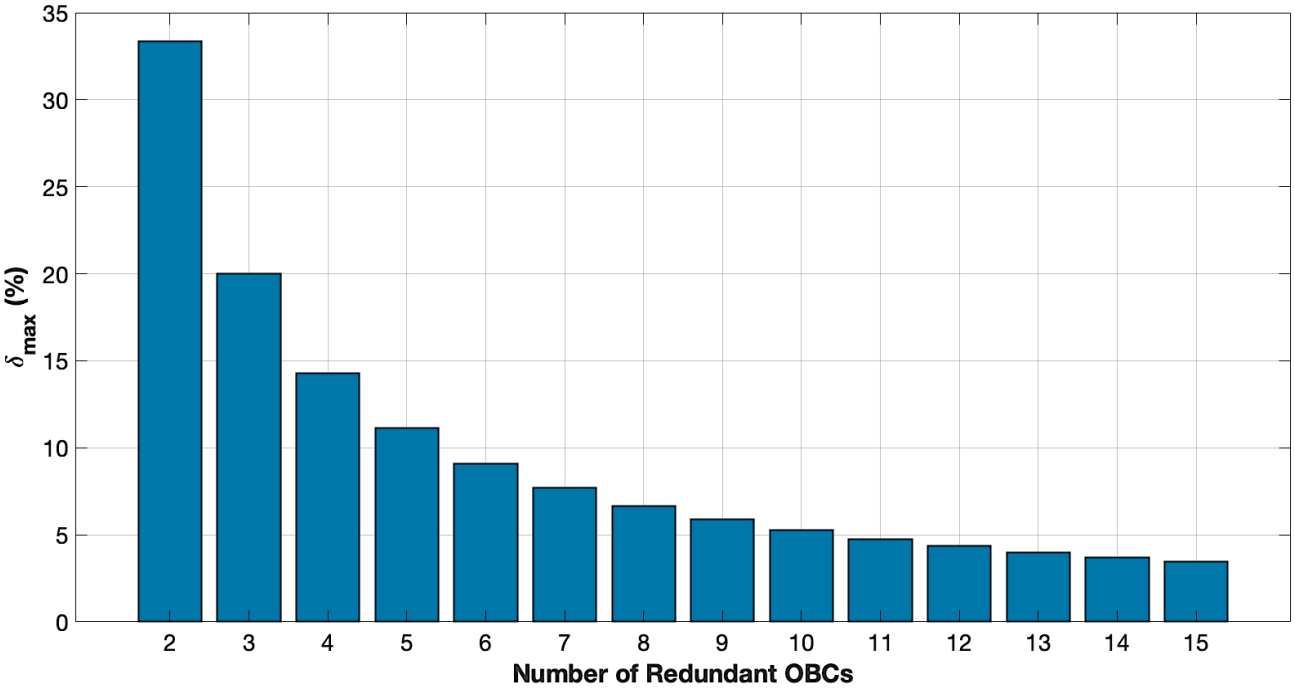

Figure 4-11: Maximum Allowable Clock Rate Deviation

4.7 SUMMARY

A novel de-centralized arbitration strategy for OBCs in cold redundancy was presented in this chapter. The proposed architecture does not rely on an external arbiter for arbitrating between the redundant OBCs, thereby eliminating that single point of failure from cold redundant architectures. The architecture places each OBC in parallel redundancy with the rest of the spacecraft platform, as shown in Figure 4-12. Note the direct communication interface between the spacecraft platform and the redundant OBCs show in this figure is only representative, and tri-state buffers maybe required to isolate the OBCs depending on the choice of the physical interface.

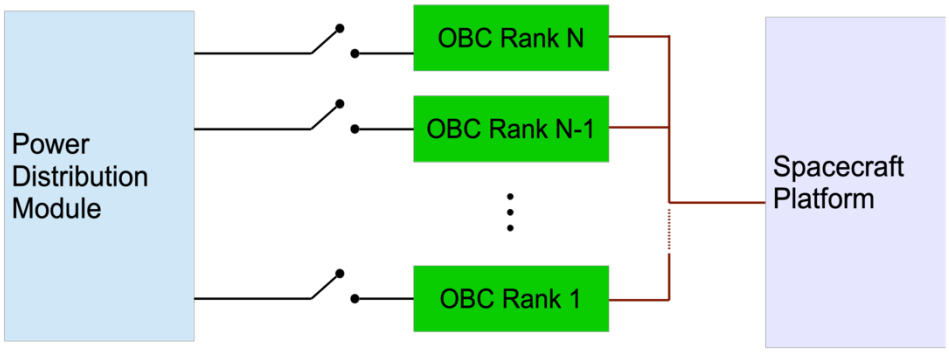

Figure 4-12: Parallel Redundancy of Proposed Architecture

The gain in system reliability of the proposed architecture (in terms of mean time to failure) was compared with a similar N-modular cold redundant architecture with an external arbiter. In this comparison, the system reliability of a redundancy architecture using an external arbiter was demonstrated to only asymptotically approach the system reliability of the proposed redundancy architecture.

CHAPTER 5 DESIGN IMPLEMENTATION AND VALIDATION

Summary: This chapter discusses the ground validation test setup for the proposed arbitration architecture, using three Raspberry Pi Zero W modules to emulate the cold redundant OBC architecture. The results from these validation tests demonstrate the ability of the proposed architecture to resolve the arbitration process during nominal and off-nominal OBC operations. Furthermore, the extensibility of this architecture is demonstrated by adding another Raspberry Pi Zero W module. Lastly, the implementation of the proposed architecture onboard the DESCENT CubeSat mission is discussed.

5.1 GROUND VALIDATION

5.1.1 Experimental Setup

The test setup emulates the core sub-systems of a spacecraft bus required for the ground validation of the proposed architecture. The Onboard Computer (OBC) for the initial set of tests is comprised of three redundant Raspberry Pi Zero W modules. The Power Distribution Module (PDM) is emulated using a 4-channel solid-state relay which is commanded by an independent Raspberry Pi module, known as the Process Observer. The Process Observer also provides visibility on the arbitration cycle by gathering events from each of the three redundant OBCs. Finally, a dummy payload is emulated using a Real Time Clock (DS1307) and will used to demonstrate arbitrated access to the spacecraft bus. The entire test setup is powered via a 5V 15W wall adapter, which emulates the power generation and storage systems. Figure 5-1 shows a block diagram of the hardware setup used for the ground validation tests.

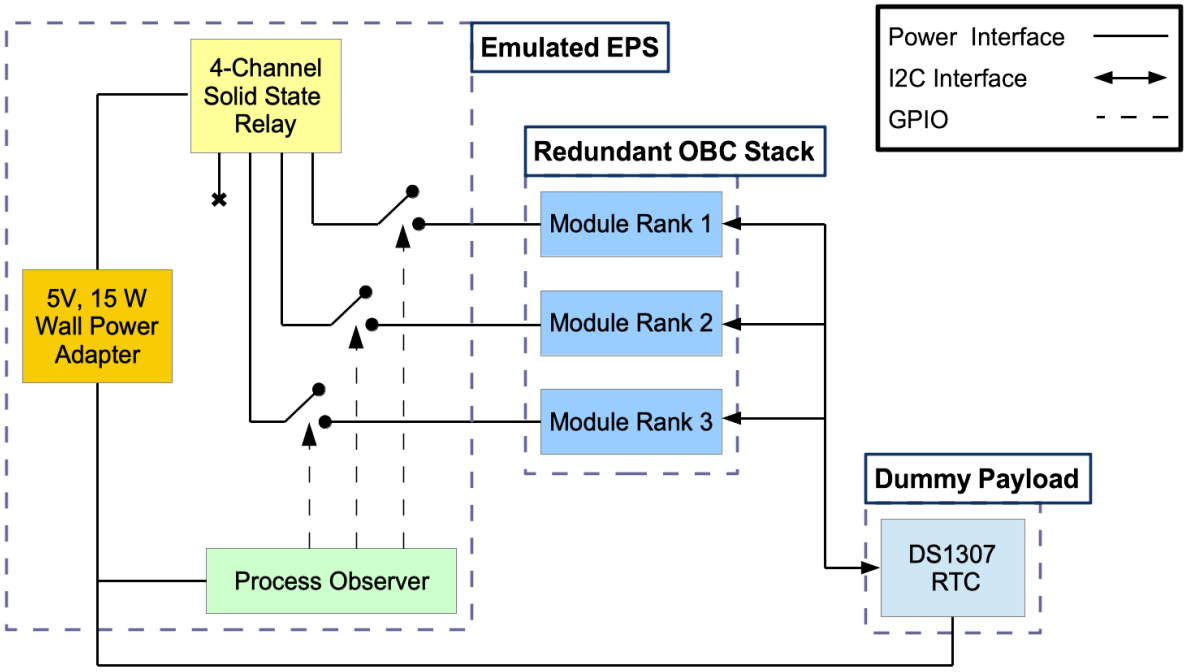

Figure 5-1: Block Diagram for Ground Test Setup

Each Raspberry Pi Zero W module is powered using a 5V line connected to one of the four channels of the solid-state relay. The state of each channel (ON or OFF) is controlled independently by the Process Observer, however, the commands to change the state of the channel can only be sent by one of the three Raspberry Pi modules. Each time a redundant OBC sends a bus reset command, the Process Observer power cycles the state of all channels. The Process Observer also stores the default state of each channel as a software parameter. If the one of the Raspberry Pi modules is permanently disabled during the arbitration process, this software parameter is updated, and Process Observer does not power cycle that particular channel during subsequent bus reset commands. This combination of the solid-state relay controlled by the Process Observer emulates a Power Distribution Module whose default switch states can be commanded by the OBCs.

The DS1307 Real Time Clock communicates over I^2C with the Raspberry Pi Modules. The DS1307 has a 1-Byte long control register which determines the configuration of the RTC. This register is set to a known value before the arbitration cycle and will not be changed. Each Raspberry Pi module will query this parameter and compared it to the preset configuration byte. This comparison indicates the bit error rate on the I^2C line and will be used as one of the health metrics for the OBC Health Quotient. The setup emulates a shared spacecraft bus, where all redundant OBCs connect to the payload via the same physical interface. The process supervisor will arbitrate access to this shared bus during the Health Assessment phase to sequentially allow each OBC to query this parameter.

A multi-drop communication channel between the redundant OBCs is emulated through wireless communications. Each of the redundant OBCs communicate with each other using TCP/IP protocol over Wi-Fi. All communications are routed through the Process Observer, which establishes a TCP server on the network. Each redundant OBC joins as a TCP client on the network and uses the Process Observer as a router for sending handshake requests, acknowledgements, and Health Quotient values to other OBCs. To support this, all communication packet headers include the identification for the source and the destination of the message. For inter-OBC communication, the module ranks of OBCs are used as source and destination identification. A message may also be broadcasted by using a broadcast destination ID (set in software), which routes the message to all OBCs.

Lastly, the OBCs can also send messages directly the Process Observer. This allows each redundant OBC to broadcast all its state transitions during the arbitration process. These transition events are time-tagged & stored on the Process Observer. Once the arbitration cycle has finished, these transition events can be retrieved to recreate a detailed timeline of the entire arbitration

process from the perspective of each OBC. Direct messages to the Process Observer are also used to power cycle the solid-state relay channels or permanently disable one of the OBCs in response to any failure modes detected by the arbitration architecture.

5.1.2 Architecture Implementation

The implementation of the arbitration architecture requires definition of health metrics to compute the Health Quotient, as well as the definition of arbitration parameters to compute the timeout values of the software watchdog timers. To support extensibility of the platform scale and the OBC Health Quotient, details of the individual health metrics, along with some of the arbitration parameters, are abstracted away in software from the underlying abstraction logic.

The abstraction of health metrics includes the rank index, original bounds, and step size of the un-normalized health metrics. The information regarding the scale of the platform is contained within health metric with rank index of 0. The bounds of this metric indicate the total number of redundant OBCs, while the value of this metric identifies the module rank of the OBC. Furthermore, the health metric abstraction also contains pointers to functions that compute the current value of the health metrics. During the arbitration process, the underlying logic iterates through the list of health metrics and calls the respective functions to compute the value of the health metric. The health metrics are normalized according to their rank index & original step size and are then aggregated into the OBCs Health Quotient. The health metrics used in the ground validation tests are discussed in section 5.1.2.1.

The arbitration parameters that require modification depending on the platform chosen for the redundancy architecture are called platform-based arbitration parameters. These parameters include, but are not limited to, the number of handshake & health assessment requests, the time required to broadcast each message and the interval between successive retransmission of

messages. Platform-based arbitration parameters are tuned for the choice of the underlying OBC platform, the inter-OBC communications architecture and the requirements of the mission. The watchdog timeouts can be subsequently derived from platform-based arbitration parameters based on rules defined in Chapter 4 These platform-based arbitration parameters are abstracted away in software, but the rules for deriving the timeout values are integrated within the arbitration logic. The complete list of platform-based arbitration parameters for any redundant architecture, along with their selected values for the ground validation tests, are presented in section 5.1.2.2.

Figure 5-2 highlights the abstraction of the health metrics and platform-based arbitration parameters from the underlying arbitration logic.

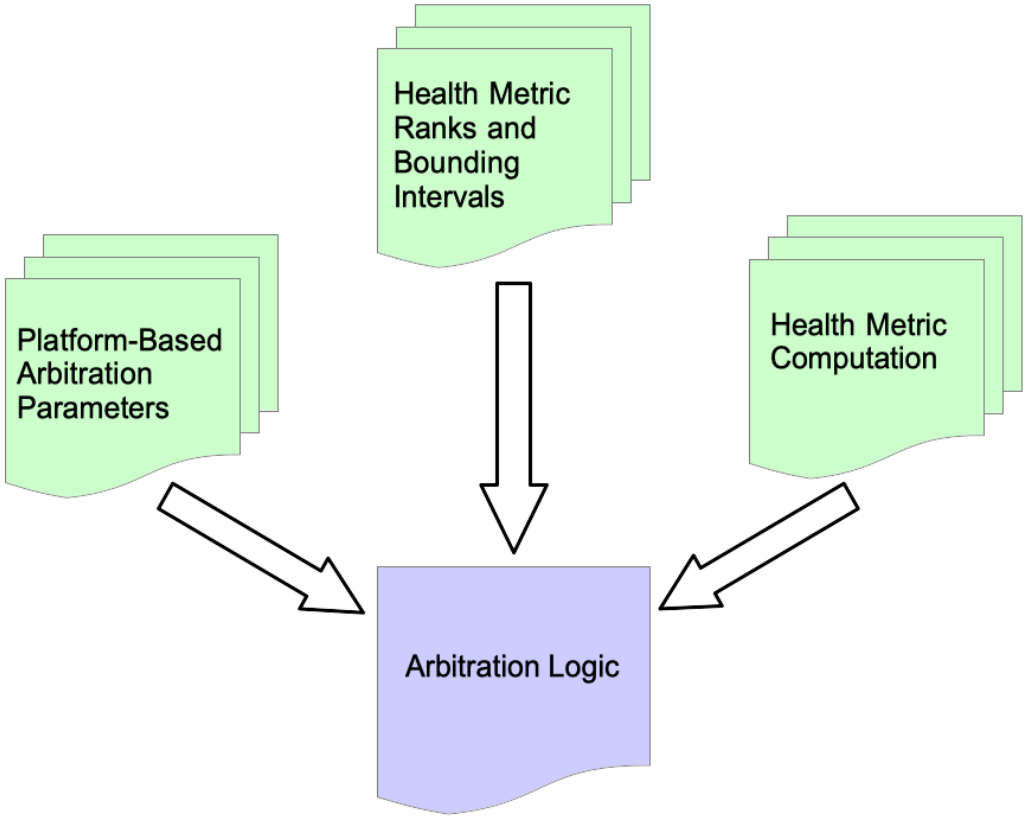

Figure 5-2: Abstraction of the Arbitration Logic

5.1.2.1 Health Metrics

Table 5-1 lists the health metrics used during the ground validation tests. The table also shows the original bounds of the metric as well as the normalized bounds computed based on the metric rank index. Note that the normalized bounds grow quickly as the number of steps in the original bounds increases. This can be driven by either a large interval or a small step size of the original metric value.

Table 5-1: Health Metrics Used in Ground Validation Tests

Health Metric	Rank Index	Original Bounds	Original Step Size	Normalized Bounds	Normalized Step Size
I^2C Functionality	5	[0, 1]	1	[1, 711011095]	711011094
I^2C Data Fidelity	4	[0, 1]	0.025	[1, 693669361]	17341734
Data Read/Write Fidelity	3	[0, 1]	1×10^{-4}	[1, 17340001]	1734
Temperature Check	2	[0, 1]	1	[1, 868]	867
Persistence Metric	1	[0, 288]	1	[1, 865]	3
Module Rank	0	[1, 3]	1	[1, 3]	1

5.1.2.1.1 I^2C Metrics

The I^2C health metrics assess the ability of an OBC to communicate with the emulated payload. The assessment constitutes the I^2C Functionality metric and the I^2C Data Fidelity metric. The I^2C Functionality metric is a Boolean metric that indicates if the attempt to communicate with

the payload was successful or if it timed out. The I^2C Data Fidelity metric is an indicator of the Bit Error Rate (BER) on the communication line. The configuration byte (8 bits) on the payload is sampled 5 times, and sample is compared to a known value of the byte. The number of correct bits is counted across all samples, and the ratio of the correct bits to the total bits sampled (40 bits) is returned as the value of the health metric. Both these metrics are subsequently normalized based on their rank indices.

5.1.2.1.2 Data Read/Write and Temperature Metrics

Similar to I^2C Data Fidelity metric, the Data Read/Write metric performs a write to file and reads the data back. A preset array of 1250 bytes is used for the read/write functions. The metric returns the ratio of the current number of bits read back out of the total number of bits written to file. The Temperature Check metric is another Boolean metric that indicates if the OBC is within its predefined operating temperature range. Both these metrics are subsequently normalized based on their rank indices.

The purpose of the ground tests is not to carry out the actual metric evaluations, but to observe their effects on the arbitration process. Hence, the value of these metrics can be manually adjusted to emulated varying degrees of performance degradation.

5.1.2.1.3 Persistence Metric

The persistence metric helps minimize the switchovers of the commanding OBC in response to transient events. The value of this health metric is computed using the following expression:

$$h_{\text{Persistence}} = \left\lfloor \frac{T_{\text{System}} - T_{\text{Epoch}}}{T_{\text{Step}}} \right\rfloor \tag{5.1}$$

where T_{System} represents the onboard system time on an OBC, while T_{Epoch} is the mission epoch time and is identical for all OBCs. Since the mission epoch time is identical for all OBCs, the

difference between these two terms represents the uptime for each OBC. T_{Step} is set to 5 minutes for the ground test and limits the bounding interval for this metric. As a result, the persistence metric increments by one every 5 minutes. The floor function limits the jitter in the persistence metric. An OBC must stay in command of the spacecraft bus for at least T_{Step} to increment the persistence metric. The upper bound of this metric is bound to an emulated 1-day mission lifetime. This equates to 288 intervals of 5 minutes each. Once this metric reaches that value, it will stop incrementing and will always resolve to its maximum value. The utility of this metric is discussed in greater detail in section 5.2.5.

5.1.2.1.4 Module Rank Metric

The module rank metric indicates the scale of the redundant platform. The ground test uses 3 redundant modules, as indicated by the bounds of the health metric. The value of this metric is immutable throughout the course of the mission i.e., the OBC does not need to compute this value. Instead, the module rank is hard coded for each individual OBC prior to launch. Therefore, the normalized value of the metric is the same as its original value.

5.1.2.2 Platform-Based Arbitration Parameters

Table 5-2 lists the platform-based parameters of the arbitration architecture along with the values chosen for the ground validation tests. The description of the parameter also provides selection criteria for the parameter values. Additionally, the time parameters have been inflated to slow down the arbitration process for the purpose of visualizing the validation test results. These parameters are used to derive the software watchdog timeout values, using the rules presented in Chapter 4 . The abbreviation '*s*' in the **Value** column corresponds to seconds, while '*ms*' corresponds to milliseconds.

Table 5-2: Platform-Based Arbitration Parameters

Platform-Based Parameter	Value	Description
$T_{HDSKRequestInterval}$	1.5 s	Handshake request interval based on the observed deviation in OBC bootup times.
$T_{HDSKRequest}$, $T_{HQRequest}$, $T_{EOPMessage\_S}$, $T_{SyncMessage\_S}$, T_{EOPAck_R}, $T_{SyncMessage\_R}$	250 ms	Time to transmit message based on message length and communication baud rate.
$T_{BroadcastInterval}$, $T_{EOPInterval\_R}$, $T_{SyncInterval\_R}$	250 ms	Interval between successive message re-transmission based on task scheduling and processing loads.
T_{PDM}	2 s	Maximum time to command power system dependent on platform.
$T_{Resolve}$	2 s	Maximum time to resolve arbitration requires all other modules to be powered down (directly tied to T_{PDM})
$N_{MaxCycleCount}$	2	Maximum number of incomplete arbitration cycles to allow recovery from SEEs.
$N_{HQRequests}$, $N_{EOPMessages\_S}$, $N_{SyncMessages\_S}$, $N_{EOPAcks\_R}$, $N_{SyncMessages\_R}$, $N_{BroadcastMessages}$	3	Maximum number of re-transmissions of request or acknowledgement messages

$N_{HDSKRequests}$	5	The number of handshake requests equates to the safety factor for the observed deviation in OBC bootup time.
$T_{EOPInterval\_S}$	1.5 s	Interval between successive Handshake Phase EOP message re-transmissions. As discussed in Chapter 4 , section 4.3.3.3, $T_{EOPInterval\_S} > T_{HDSKToHLTH\_R}$
$T_{SyncInterval\_S}$	1.5 s	Interval between successive Health Phase sync message re-transmissions. As discussed in Chapter 4 , section 4.4.2.2, $T_{SyncInterval\_S} > T_{HLTHToRESL\_R}$

5.2 TEST CASES

The results presented in this section demonstrate the arbitration timeline for each OBC and present the associated Health Quotient values computed during the arbitration cycle. Nominal functioning OBCs evaluate all health metrics to their maximum values indicated in Table 5-1. Any changes in the evaluations of these health metric during test cases that emulate off-nominal scenarios are also highlighted. This arbitration timeline is represented as color-coded plot that indicate the OBC transitions between various arbitration phases and states. Table 5-3 provides a legend of the various color codes used in the arbitration timeline, along with the associated OBC states.

Table 5-3: Arbitration Timeline Color Codes

Timeline Color	Module Role	State
	All	Spacecraft Bus Reset
	All	Module Initialization
	Process Supervisor	Handshake Request State
	Redundant Module	Handshake Sentinel State
	All	Health Quotient Computation
	All	Handshake and Health Assessment Sync
	All	Arbitration Winner
	Redundant Module	Handshake and Health Assessment Reset
	All	Emulated Operations Period

The block diagram for the ground test setup was shown in Figure 5-1. Figure 5-3 shows the physical implementation of this test setup and highlights its constituents.

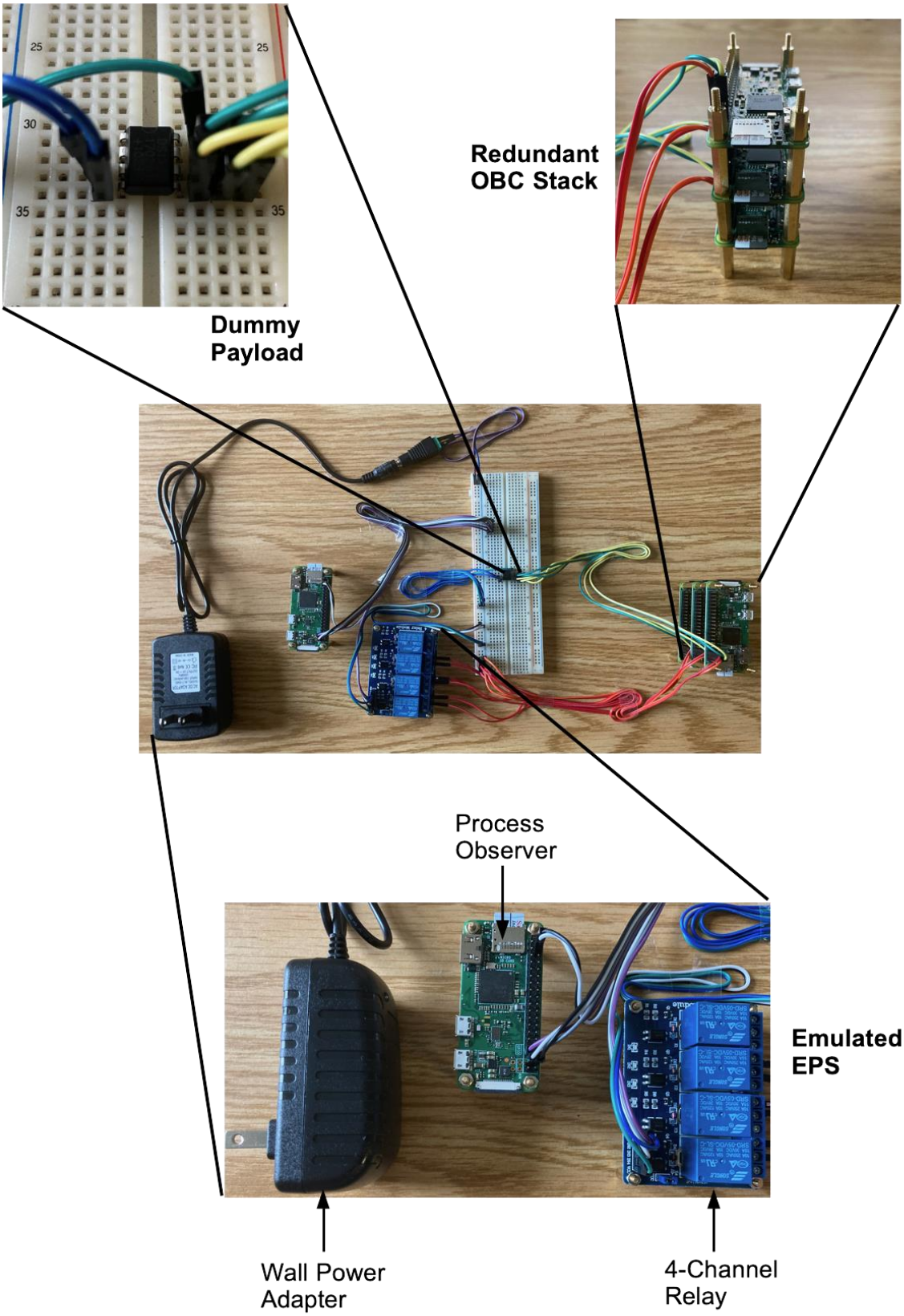

Figure 5-3: Ground Test Setup

5.2.1 Nominal Operations

This case represents nominal functioning of all OBCs and the commanding OBC of the spacecraft bus is established based on the module rank of the OBC. All health metrics evaluate to their maximum values. Table 5-4 lists the Health Quotient values computed for each OBC during the arbitration cycle for this test case. Since all OBCs have equivalent evaluations of the health metrics, the Health Quotient's uniqueness is driven by the module ranks. This can be observed in Table 5-4 where the Health Quotient values of consecutive OBC module ranks differ by one in magnitude from each other. This is the step size of the health metric for the OBC module rank.

Table 5-4: OBC Health Quotients for Nominal Test Case

Module Rank	Health Quotient
1	1,422,022,191
2	1,422,022,192
3	1,422,022,193

Figure 5-4 shows the arbitration timeline for each OBC as they transition through the different phases and states of the arbitration mechanism. The phase transition boundaries are highlighted for the OBC with module rank 3 and are identical for the other two OBCs. The figure also highlights the key events of the arbitration cycle.

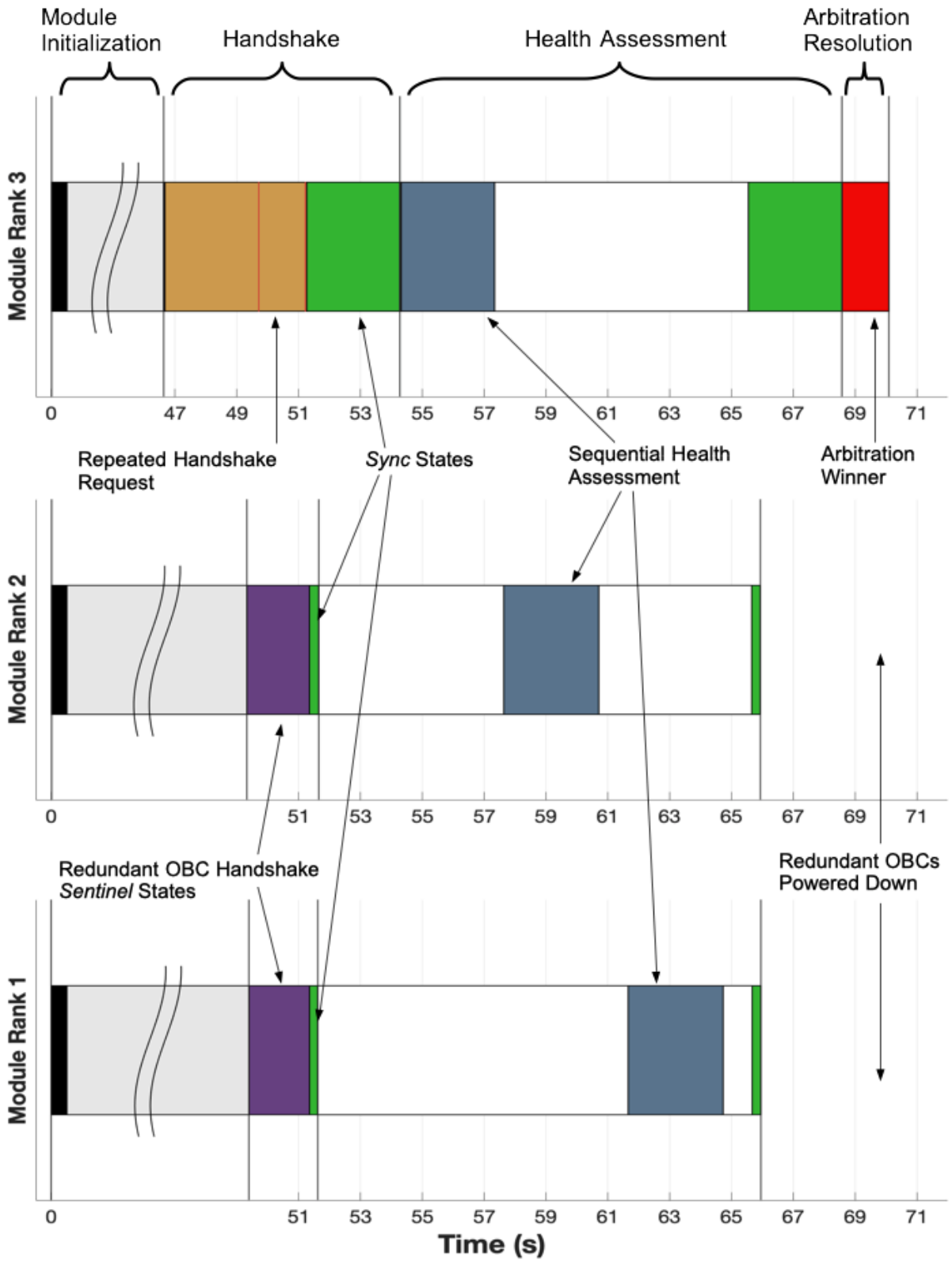

Figure 5-4: Nominal Arbitration Cycle

The time stamps for the arbitration events are assigned by the Process Observer, enabling the observation of each OBC's arbitration timeline using a single clock domain. The timeline starts at the instant the emulated bus is reset, that is, when all relay channels are enabled by the Process Observer. Since the boot up time for the Raspberry Pi Modules is more than 40 seconds, the Module initialization phase shown in the plot is not to scale. This is represented by the sinusoidal overhang in the Module Initialization phase and helps draw emphasis on the latter phases of the arbitration cycle. As noted in Chapter 4 , internal memory latencies can cause variable boot up times on each OBC. The variation in boot-up times is exacerbated by the 4-channel relay whose channels may not activate at the same moment. Figure 5-4 shows that OBCs 1 and 2 (Redundant OBCs) took a bit longer to boot up than OBC 3 (Process Supervisor). Multiple periodic handshake requests made by the Process Supervisor account for this variability in boot time, and the arbitration timeline for each OBC is synchronized at the end of the Handshake phase. The transition windows from the one phase to the next is highlighted by the OBC *Sync* states. As discussed in Chapter 4 section 4.3.3.3, the Process Supervisor is always the last OBC to transition from one phase of the arbitration cycle to the next. This transition window is accounted in the watchdog timer value for the redundant modules to prevent pre-mature timeout of the watchdog.

During the Health Assessment phase, the arbitration timelines demonstrate sequential assessment of the OBC Health Quotient. This is mediated by the Process Supervisor during its *Assessment* state and provides arbitrated access to the spacecraft bus to each OBC. The computation of the Health Assessment has been purposely slowed down in this arbitration cycle (~2.5 seconds) to visualize this phase.

Finally, the OBC with the largest Health Quotient (module rank 3 in this case) wins the arbitration cycle during the arbitration resolution phase and powers down the redundant OBCs.

Once again, the watchdog timers on the redundant OBCs take into account the phase transition window, providing sufficient time for the Process Supervisor to transition in the Arbitration Resolution phase and execute power down sequence of the redundant modules.

5.2.2 Unresponsive Redundant Module

This case represents a failed redundant module. Since, the redundant module is never powered down permanently, the failure mode emulated in this test case covers both transient and permanent failures. To emulate this scenario, the power interface for the redundant OBC with module rank 2 is physically disconnected, such that the OBC cannot power up during the arbitration process. Table 5-5 lists the Health Quotient values computed for each OBC during the arbitration cycle for this test case. Since no performance degradation was introduced to either Process Supervisor or the functioning redundant OBC, their Health Quotient are identical to those computed during the nominal test case.

Table 5-5: OBC Health Quotients for Unresponsive Redundant Module Test Case

Module Rank	Health Quotient
1	1,422,022,191
2	-
3	1,422,022,193

Figure 5-5 shows the arbitration timeline for each OBC for this test case. The arbitration timeline demonstrates repeated attempts by the Process Supervisor to establish a handshake with the unresponsive OBC. The Process Supervisor would observe the same lack of handshake

acknowledge even if the redundant OBC was operational, but its communication interface had failed. In either case, the Process Supervisor correctly concludes that the redundant OBC cannot participate in the current arbitration cycle, and powers it down. The redundant module with rank index 1 waits for the Process Supervisor to complete the handshake attempts, and only transitions into the Health Assessment phase upon receiving the End-of-Phase messages from the Process Supervisor.

The repeated handshake attempts result in a longer execution time of the Handshake phase. However, the execution time of the subsequent Health Assessment Phase is reduced since the Process Supervisor need not request the Health Quotient of the OBC with module rank 2. This is represented in the sequential Assessment state execution of the arbitration timeline, where the health assessment of the Process Supervisor is immediately followed by that of the next available redundant OBC (module rank 1). The list of active modules is updated by the Process Supervisor during its *Sync* state in the Handshake Phase, allowing it to keep track of OBCs that are no longer participating in the arbitration cycle. The arbitration cycle is again won by the Process Supervisor since it still has the largest Health Quotient value.

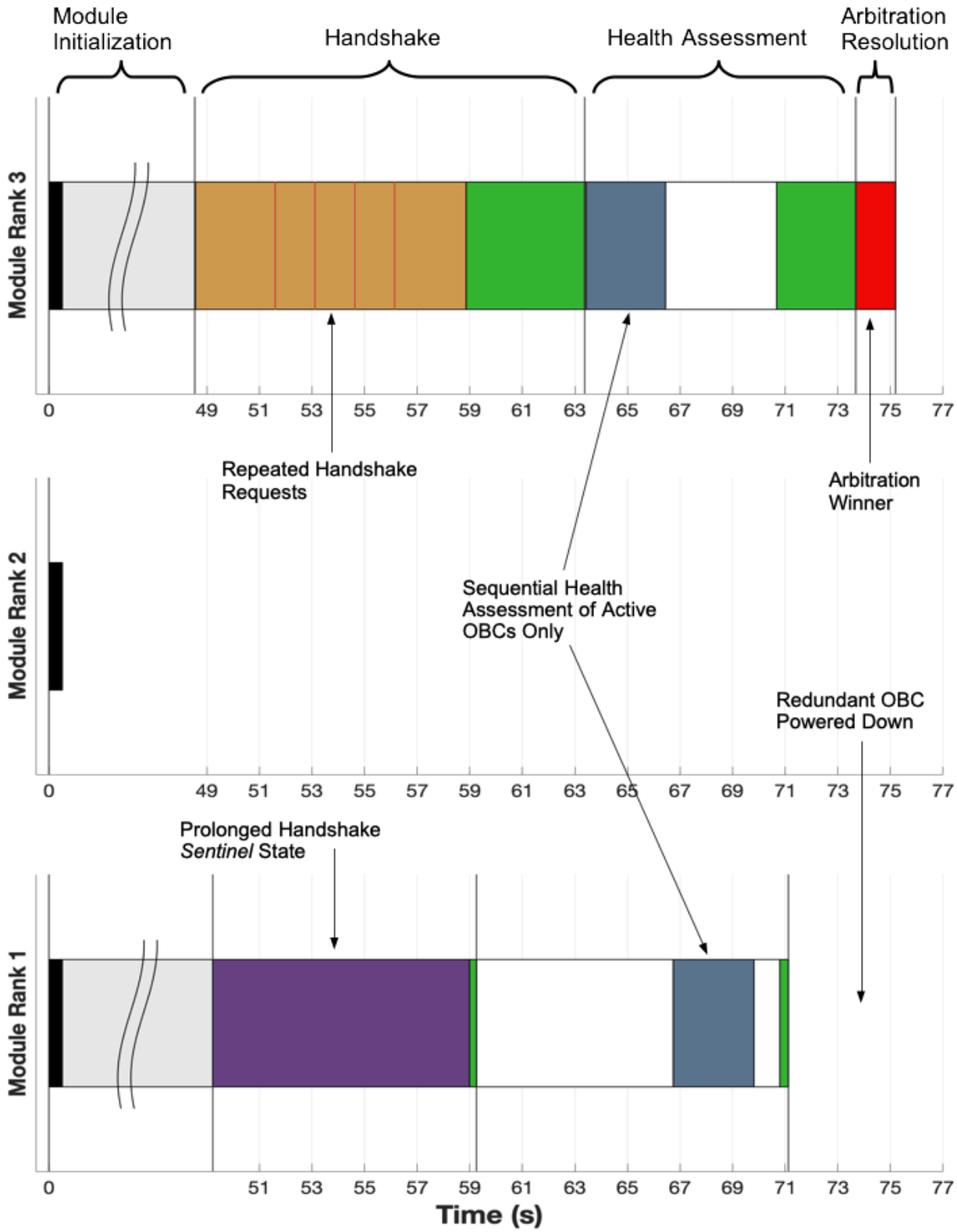

Figure 5-5: Arbitration Cycle for Unresponsive Redundant OBC

5.2.3 Deterioration of the Health Quotient

This test case demonstrates the precedence of individual health metrics that constitute the OBC's Health Quotient. In this test case, the communication interface between the Process Supervisor and the dummy payload is physically disconnected. As a result, the I^2C interface functionality and data fidelity metrics, which have the largest rank indices, cannot be evaluated on Process Supervisor. Table 5-6 shows represents these individual health metric evaluations on each OBC for this test case.

Table 5-6: Health Metric Evaluations for Deteriorated Health Quotient Test Case

Health Metric	Rank Index	Original Bounds	Module 1 Evaluation	Module 2 Evaluation	Module 3 Evaluation
I^2C Functionality	5	[0, 1]	1	1	0
I^2C Data Fidelity	4	[0, 1]	1	1	0
Data Read/Write Fidelity	3	[0, 1]	0	0	1
Temperature Check	2	[0, 1]	0	0	1
Persistence Metric	1	[0, 288]	0	0	288
Module Rank	0	[1, 3]	1	2	3

The health metric evaluations for all metrics, except the I^2C Functionality and the I^2C Data Fidelity, are forced to their maximum values on the Process Supervisor, and to their minimum values on the redundant OBCs. In other words, the Process Supervisor performs better with respect to all health metrics except for the two with the largest rank indices. The rules for the aggregation of the Health Quotient ensure that the OBC with the best performance for the most valued health metrics will gain precedence in the arbitration cycle.

Table 5-7 lists the Health Quotient values computed for each OBC during the arbitration cycle for this test case. Compared to nominal case, the redundant OBCs experience a small decrease in their Health Quotient values as a result of minimizing the lower ranked health metrics. Even though the Process Supervisor performs much better on the lower ranked metrics, its Health Quotient experiences a much more substantial decrease which is driven by its inability to communicate with the dummy payload over the I^2C bus (highest ranked health metrics).

Table 5-7: OBC Health Quotients for Degraded Performance of Process Supervisor

Module Rank	Health Quotient
1	1,404,680,460
2	1,404,680,461
3	17,341,739

Figure 5-6 shows the arbitration timeline for each OBC during this test case. The OBC with module rank 2, which has the largest Health Quotient value, wins the arbitration cycle.

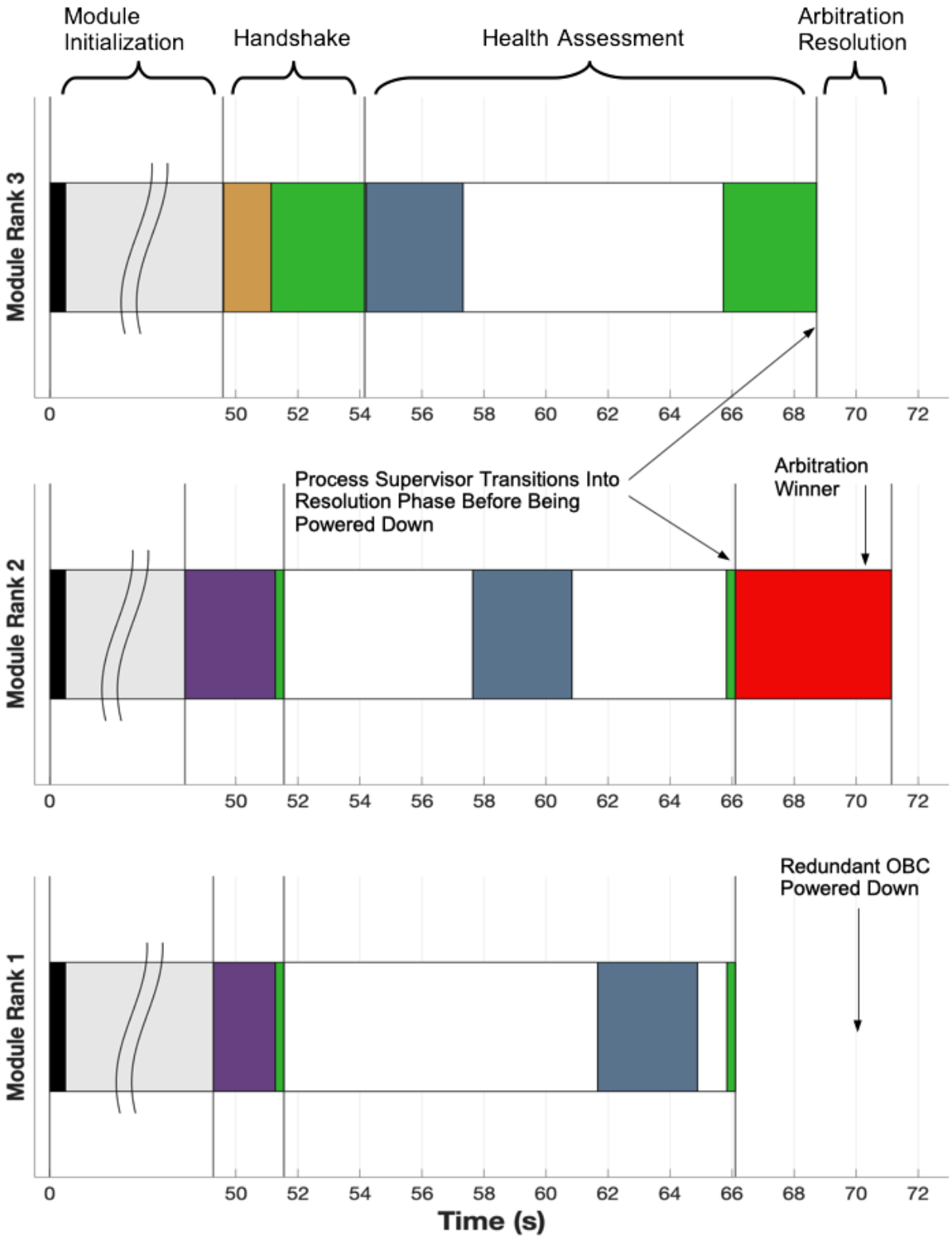

Figure 5-6: Arbitration Cycle for Deteriorated Health Quotient of the Process Supervisor

A key aspect highlighted in the arbitration timeline is that the redundant OBC that won the arbitration cycle does not immediately power down the other OBCs. It accounts for the transition window between arbitration phases and ensures that the Process Supervisor transitions into the Arbitration Resolution phase where it can reset its persistent configuration parameters before it is powered down (as discussed in Chapter 4 , section 4.5). The role of the Process Supervisor does not switch over since the current Process Supervisor is still capable of conducting out the arbitration cycle.

5.2.4 Unresponsive Process Supervisor

This test case emulates a failed Process Supervisor. Similar to the test case with the unresponsive redundant OBC, this scenario is emulated by physically disconnecting the power interface to the initial Process Supervisor (module rank 3). The remaining functional OBCs are assumed to be functioning nominally and their respective health metrics evaluate to the maximum values. Table 5-8 lists the Health Quotient values computed for each OBC. Figure 5-7 shows the arbitration timeline for each OBC during this test case. The OBC with module rank 2, which has the largest Health Quotient value, wins the arbitration cycle.

Table 5-8: OBC Health Quotients for Unresponsive Process Supervisor Test Case

Module Rank	Health Quotient
1	1,422,022,191
2	1,422,022,192
3	-

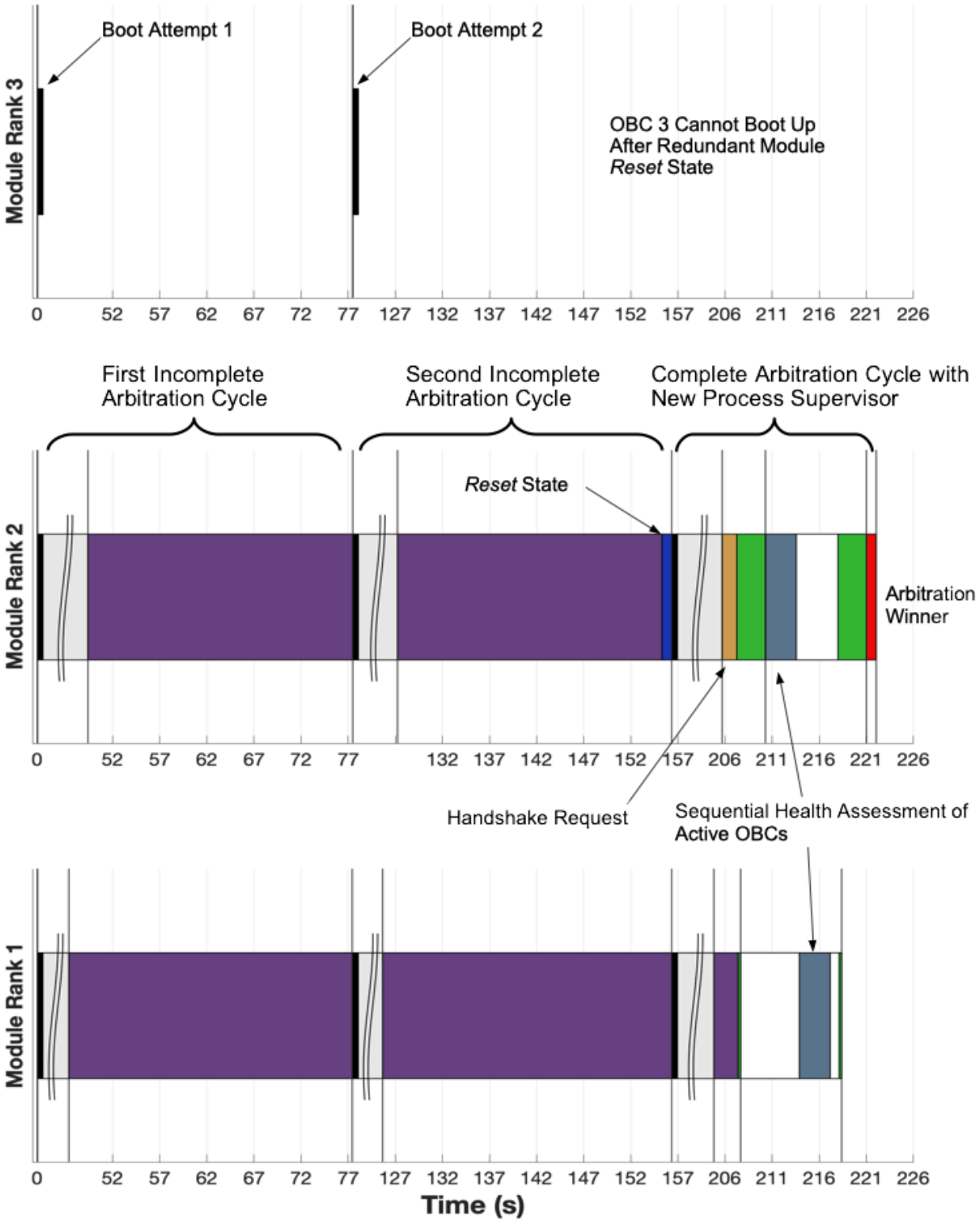

Figure 5-7: Arbitration Cycle with Unresponsive Process Supervisor

As noted in Table 5-2, the max number of allowed incomplete arbitration cycles is two. During the first arbitration cycle, the watchdog timer expires on the OBC with module rank 2 and it resets the spacecraft bus to allow the current Process Supervisor (OBC module rank 3) to recover from a possible Single Event Effect. Since the test case is emulating a failed Process Supervisor, the handshake request is not initialed during the second arbitration cycle as well. After the watchdog timer expires for a second time on OBC 2, the redundant module transitions into its Reset state and permanently disables the Process Supervisor. The spacecraft bus is reset again and OBC 2 takes over as the Process Supervisor. Note that the spacecraft bus reset by OBC 2 in both cases. If OBC 2 had failed as well, the watchdog timer on OBC 1 would have expired and that OBC would have attempted to takeover command of the spacecraft bus.

After a second bus reset, a nominal arbitration cycle is conducted between the two functional OBCs. Since the Process Supervisor always has the largest module rank, it only waits for a handshake acknowledgement from OBC 1. The list of active OBCs is updated by the new Process Supervisor during the handshake phase, and the sequential health assessment is executed amongst the active OBCs only. As noted in Table 5-8, the OBC with module rank 2 has a larger Health Quotient value and it therefore wins the arbitration cycle.

5.2.5 Minimizing Module Switchover

Consider the previous test case, where the OBC with module rank 3 has failed and the OBC with module rank 2 has assumed the role of the Process Supervisor. All future arbitration cycles are now conducted between OBC 2 and OBC 1. If OBC 2 experiences a transient degradation of its health metrics during one of these arbitration cycles, it will lose the arbitration cycle to the next available OBC (module rank 1) which will assume command of the spacecraft bus. Note that in this scenario OBC 2 can still successfully conduct the arbitration cycle and does not relinquish the

role of Process Supervisor. If the transient upset causes OBC 2 to hang up, the watchdog timer on OBC 1 will expire and result in a spacecraft bus reset. The transient error will likely clear up during the next arbitration cycle and OBC 2 will be capable of assuming command of the spacecraft bus, even if the OBC 1 is still functioning nominally. This frequent switchover of the commanding OBC can lead to loss of mission data and roll back of configuration state.

The persistence metric, discussed in section 5.1.2.1.3, minimizes this switchover and provides precedence to the last OBC to command the spacecraft bus. In the ground test setup, this metric is evaluated using Equation (5.1), and the OBC precedence is based on the system time which only increments if the OBC is powered up and operational. Alternative metrics such as current mission state or stored mission data can also be used to provide this precedence.

The setup for this test case continues from the previous test case where OBC 3 has been permanently disabled and OBC 2 has assumed the role of the Process Supervisor. The OBCs will undergo two arbitration cycles. During the first arbitration cycle of this test case, a transient error is emulated on OBC 2 by physically disconnecting its I^2C data interface, causing degradation of its Health Quotient.

The persistence metric requires setup of the mission epoch time T_{Epoch} and the metric scaling factor T_{Step}. Both these parameters have the same value across all OBCs and are immutable over the mission lifetime. During the ground test, T_{Epoch} was set to January 1, 2021, 00:00:00 UT, and T_{Step} was set to 5 minutes. At the instance when the first arbitration cycle commences, it is assumed that OBC 2 has been in command of the spacecraft bus for the past 12 hours and that OBC 1 has never assumed command of the spacecraft bus. Due to the emulated degradation of health metrics on OBC 2, OBC 1 is expected to win the first arbitration cycle and will assume command of the spacecraft bus. This OBC will be allowed to operate for 6 minutes after the end

of the first arbitration cycle. This emulates an accelerated time frame of nominal mission operations before a system wide upset induces another arbitration cycle.

During this emulated operations period, the commanding OBC's (module rank 1) system time is updated. This emulates the ground updating system time on the OBC that has recently assumed command of the spacecraft bus. As discussed in section 5.1.2.1.3, the value of the persistence metric increases by one after every T_{Step} time step. The length of the emulated operations period (6 minutes) is set to be slight larger than T_{Step} to allow the persistence metric to increment during the next arbitration cycle. The value of T_{Step} is kept small during the ground tests to accelerate the testing timeline, however in orbit, this parameter should be kept large enough to allow the spacecraft to make contact with the ground station before the persistence metric increments. Table 5-9 lists the OBC system times at the start of the two arbitration cycles conducted during this test case. The dates on both OBCs are still the same as the emulated epoch, that is, January 1, 2021. The system times do not account for the OBC uptime for the duration of the arbitration cycle itself since it is smaller than T_{Step} and will not result in an increment in the evaluation of the persistence metric. The I^2C data interface of OBC 2 is re-connected during the emulated operations period. At the end of the emulated operation period, the spacecraft bus will be reset to emulate a power upset that initiates the second arbitration cycle of this test case. Table 5-10 lists the evaluated values of the health metrics on the OBCs during both arbitration cycles. The arbitration timelines for both arbitration cycles as well as the intermediary emulated operations period is shown in Figure 5-8.

Table 5-9: OBC System Time at Start of Arbitration Cycles

Module Rank	System Time at Start of First Arbitration Cycle	System Time at Start of Second Arbitration Cycle
1	00:00:00	12:06:00
2	12:00:00	12:00:00

Table 5-10: Health Metric Evaluations During Persistence Metric Test

Health Metric	Rank Index	Original Bounds	First Arbitration Cycle		Second Arbitration Cycle	
			Module 2 Evaluation	Module 1 Evaluation	Module 2 Evaluation	Module 1 Evaluation
I^2C Functionality	5	[0, 1]	0	1	1	1
I^2C Data Fidelity	4	[0, 1]	0	1	1	1
Data Read/Write Fidelity	3	[0, 1]	1	1	1	1
Temperature Check	2	[0, 1]	1	1	1	1
Persistence Metric	1	[0, 288]	144	0	144	145
Module Rank	0	[1, 3]	2	1	2	1

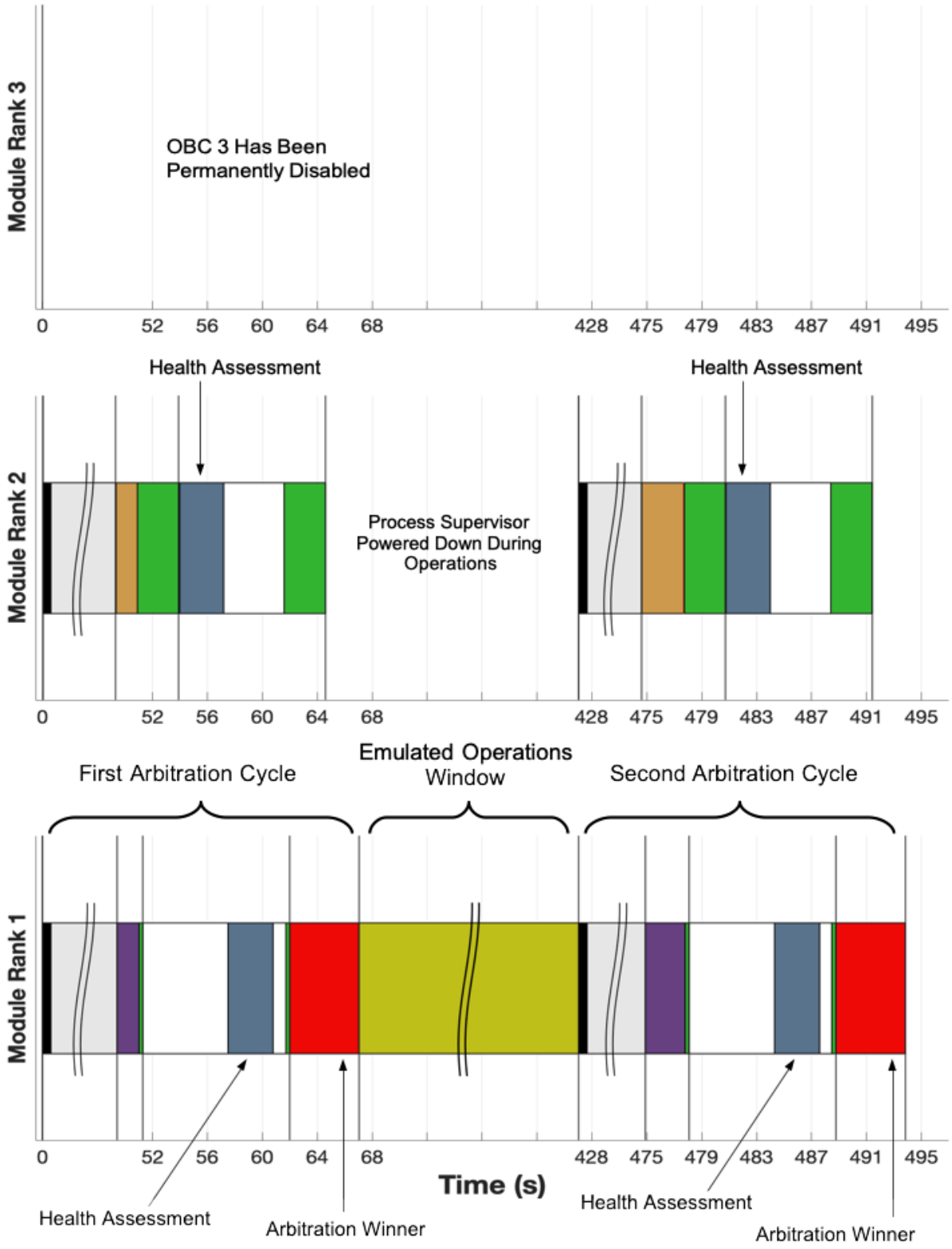

Figure 5-8: Arbitration Cycle Utilizing Persistence Metric

Table 5-11 lists the Health Quotient values computed for the OBCs during each arbitration cycle of this test case. It shows that OBC 1 has the larger Health Quotient value even after the bus reset resolves the transient error on OBC 2.

Table 5-11: Using Persistence Metric in OBC Health Quotient Computation

Module Rank	Health Quotient	
	First Arbitration Cycle	Second Arbitration Cycle
1	1,422,021,327	1,422,021,762
2	17,341,306	1,422,021,760
3	-	-

5.2.6 Extensibility

To demonstrate the extensibility of the arbitration mechanism, an additional Raspberry Pi Zero W module is added to the redundant OBC stack with a module rank of 4. The only modification required in software is updating the total number of OBCs that are available to participate in the arbitration cycle. As discussed in section 5.1.2, this change is made in the abstracted definition of health metrics, and the underlying arbitration mechanism is not modified in anyway. Figure 5-9 shows a block diagram of the updated test setup with the additional OBC.

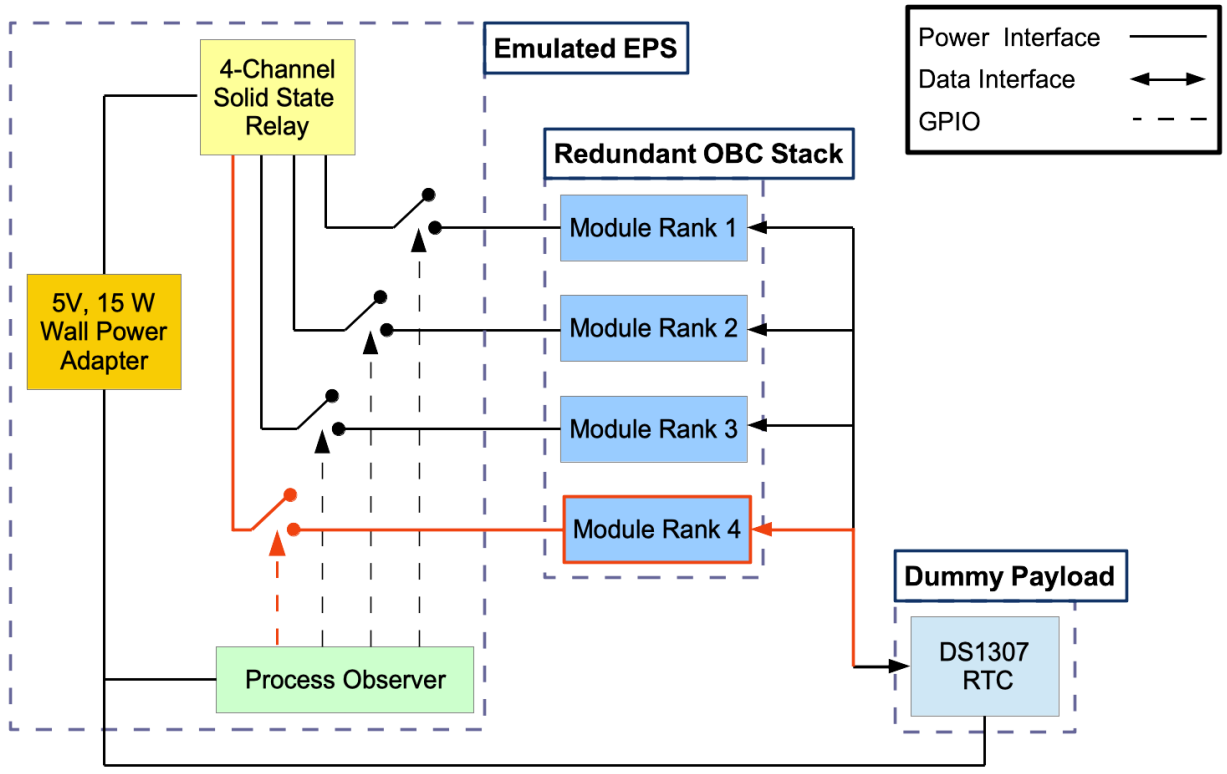

Figure 5-9: Updated Block Diagram for Ground Test Setup

As expressed by Equation (3.10), the step size of any health metric, except for the OBC's module rank, is dependent on the normalized interval of all health metrics that have a lower rank index. Since the introduction of a new OBC changes the normalized bounds of the module rank metric, the step size and normalized bounds of all other health metrics changes as well. Table 5-12 lists the updated normalized bounds and normalized step size for the health metrics used in the ground validation tests. This test case emulates nominal operating conditions for each OBC. Table 5-13 lists the resulting Health Quotient values computed for each OBC during the arbitration cycle of this test case. The arbitration timeline for this test case is shown in Figure 5-10. It mimics the nominal arbitration cycle discussed in section 5.2.1, and demonstrates that the OBC with module rank 4 wins the arbitration cycle, since it now has the largest Health Quotient.

Table 5-12: Updated Normalized Bounds for the Health Metrics

Health Metric	Rank Index	Original Bounds	Original Step Size	Normalized Bounds	Normalized Step Size
I^2C Functionality	5	[0, 1]	1	[1, 948014793]	948014792
I^2C Data Fidelity	4	[0, 1]	0.025	[1, 924892481]	23122312
Data Read/Write Fidelity	3	[0, 1]	1×10^{-4}	[1, 23120001]	2312
Temperature Check	2	[0, 1]	1	[1, 1157]	1156
Persistence Metric	1	[0, 288]	1	[1, 1153]	4
Module Rank	0	[1, 4]	1	[1, 4]	1

Table 5-13: OBC Health Quotient Values for the Extensibility Test Case

Module Rank	Health Quotient
1	1,896,029,586
2	1,896,029,587
3	1,896,029,588
4	1,896,029,589

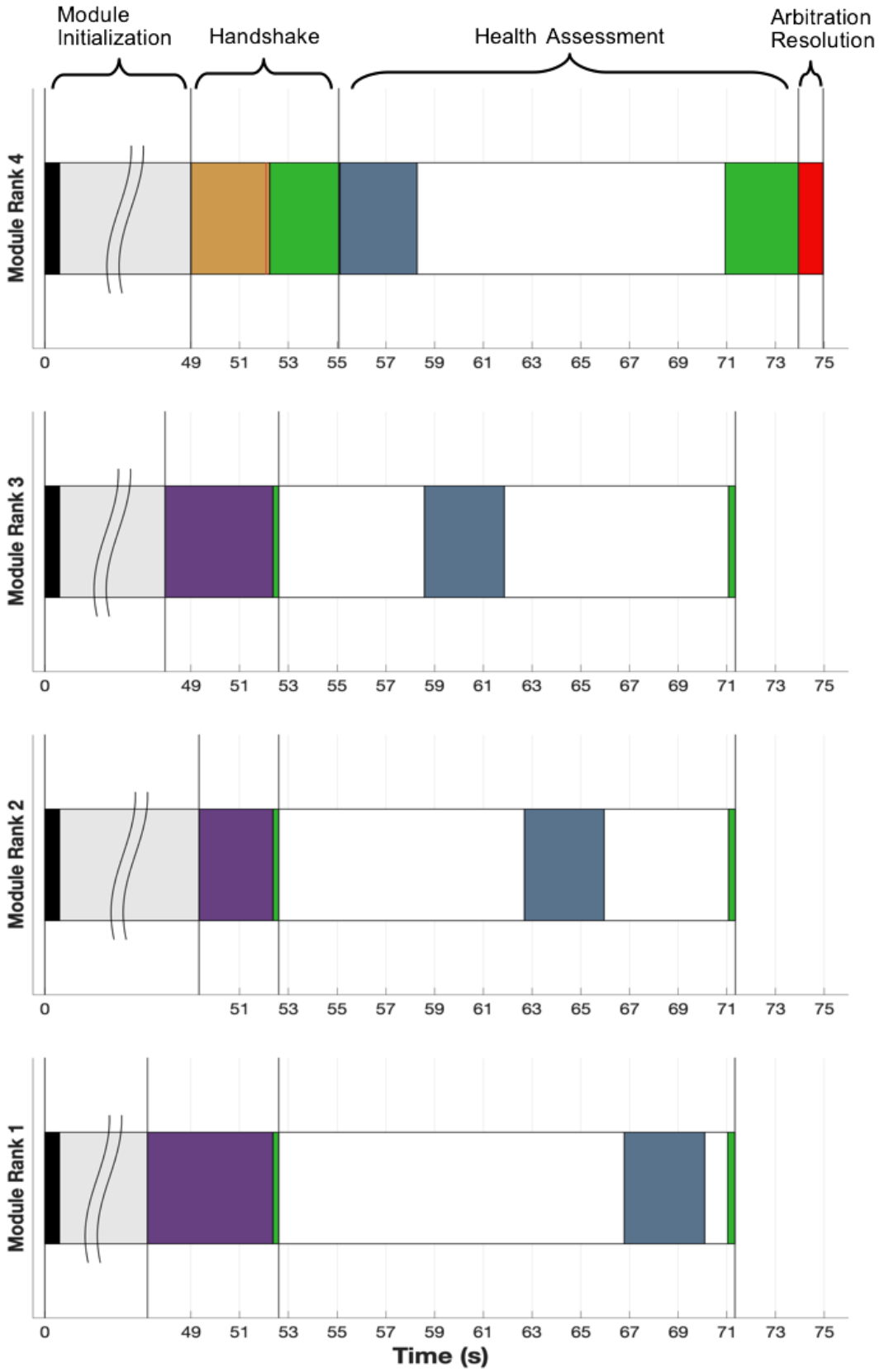

Figure 5-10: Nominal Arbitration Cycle Using Four Redundant OBCs

5.3 THE DESCENT MISSION

5.3.1 Mission Overview

DESCENT (De-orbiting Spacecraft using Electrodynamic Tethers) is 2U CubeSat mission developed at York University. The primary objective of DESCENT is to demonstrate the viability of using a bare electrodynamic tether (EDT) as an effective deorbiting device. The mission consists of a 100 m long conducting tether attached to a 1U CubeSats at each end. One of the ends of the tether is tied to a type of cathode field emitter called Spindt Array [70] [71] and DESCENT will be the first mission to demonstrate the viability of these cathode field emitters in space. The mission will exhibit the ability of EDTs coupled with a Spindt Array to generate current though the tether. This will result in a Lorentz force acting to slow down the orbital velocity of the satellite. Although the DESCENT spacecraft is only a 2.6 Kg CubeSat orbiting at a relatively low altitude of 400 km, a successful demonstration of this technology will provide a proof of concept, allowing future missions to scale up the device for heavier satellites orbiting at higher altitudes. Figure 5-11 shows the EDT concept art for the DESCENT mission (background image of the Earth sourced from [75]).

As a secondary objective, the mission also builds upon the idea of low-cost space technologies. The system architecture of the DESCENT spacecraft was designed such that the data gathered from the daughter satellite (CubeSat orbiting at lower altitude) would be sufficient to demonstrate the deorbit capability of EDTs. Figure 5-12 and Figure 5-13 show the spacecraft mechanical layout of the mother and the daughter satellites respectively. Each CubeSat has a highly limited mass and volume budget, however the small form factor of COTS OBCs allows integration of redundant modules without consuming additional mechanical budgets.

The daughter satellite features all sub-systems required to deploy the electrodynamic tether and enable the Spindt Array. This design allows the mother satellite (CubeSat orbiting at higher altitude) to serve as an experimental platform for the proposed arbitration architecture. The mother satellite on DESCENT integrates two Raspberry Pi Zero W modules in cold redundancy and utilizes the proposed architecture to arbitrate between the two modules.

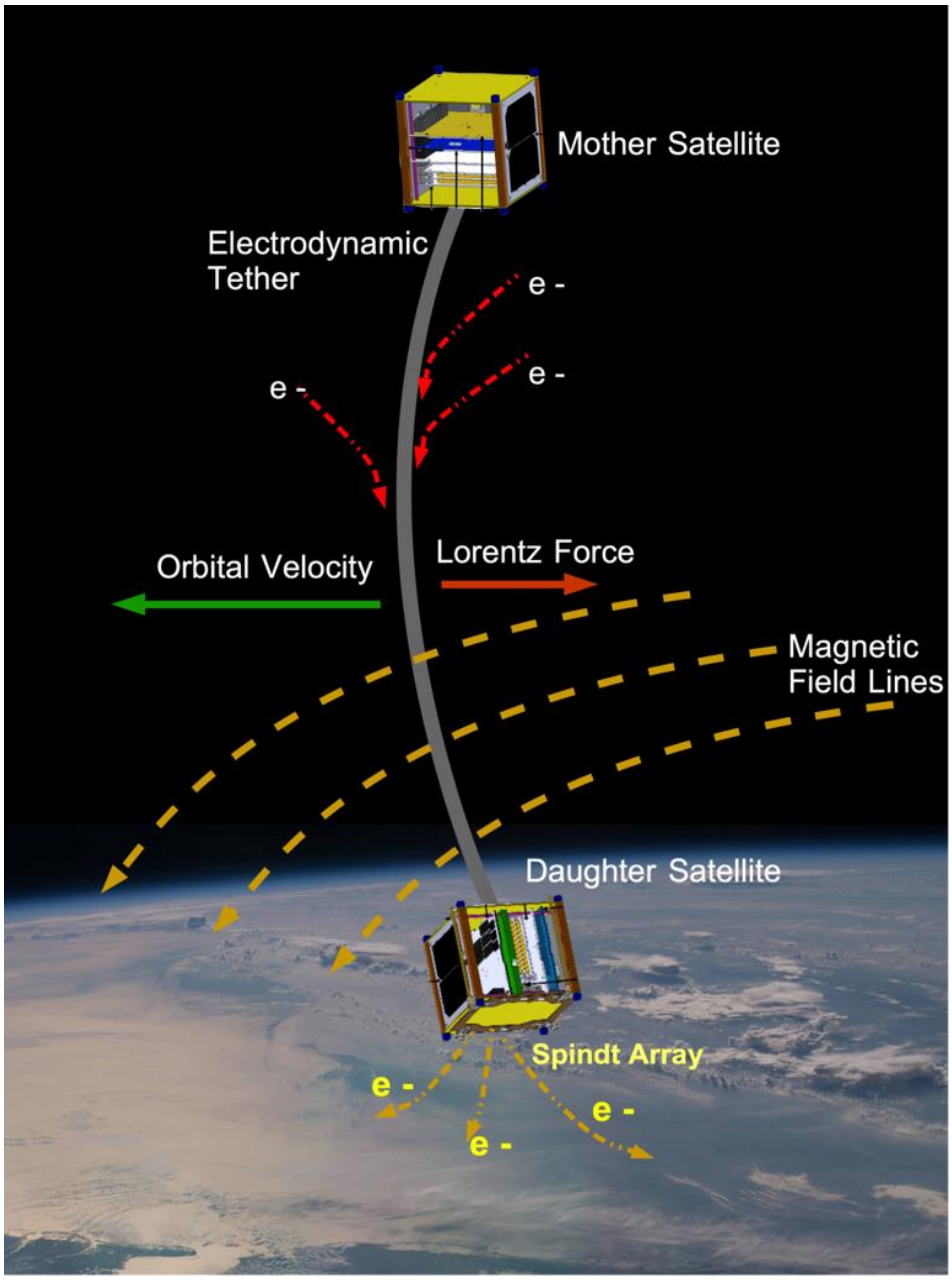

Figure 5-11: Concept Art for DESCENT

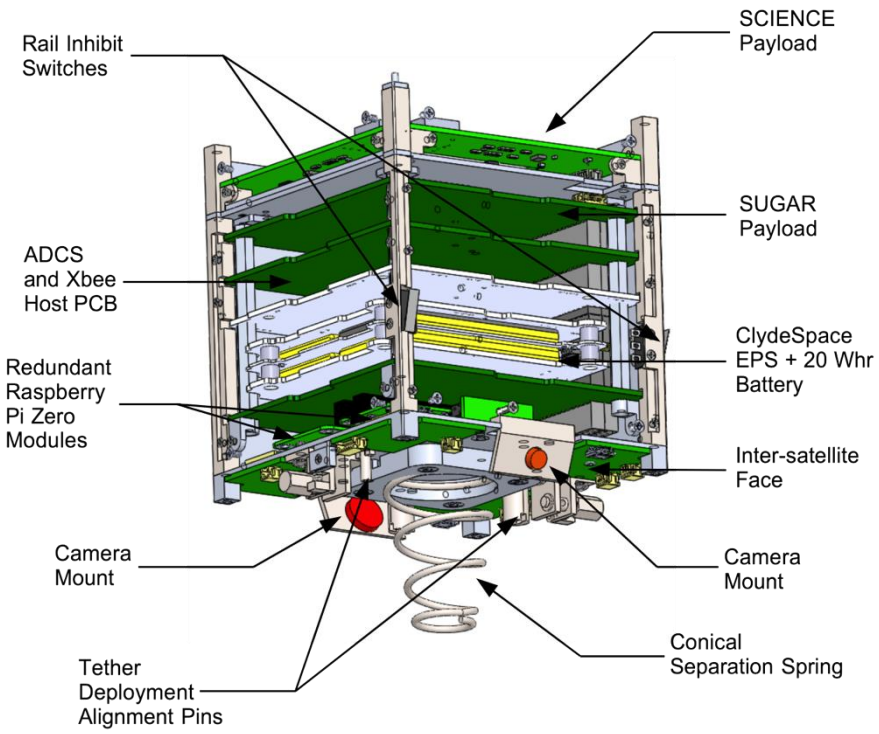

Figure 5-12: Mechanical Layout of the Mother Satellite

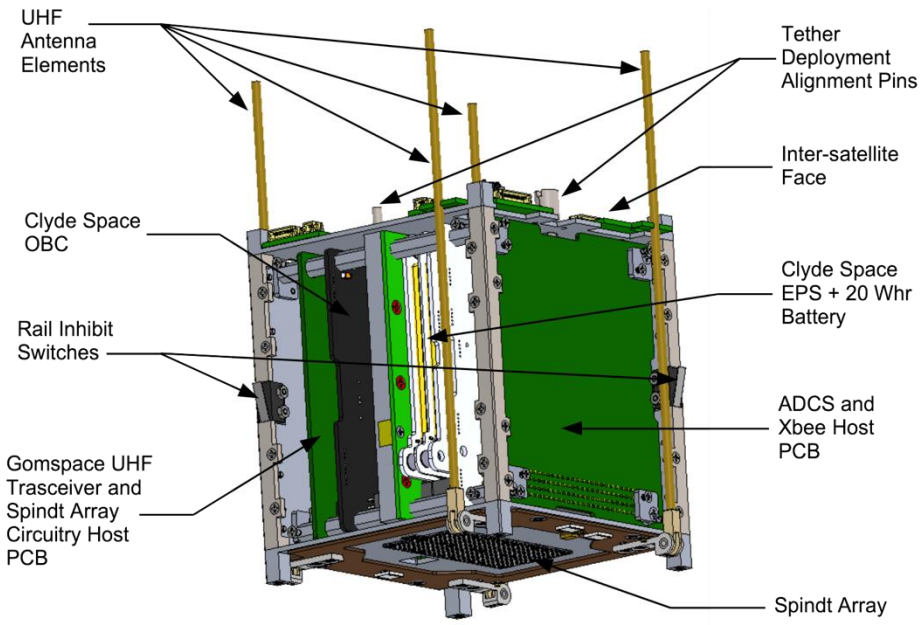

Figure 5-13: Mechanical Layout of the Daughter Satellite

5.3.2 Dual Redundant OBC Architecture

5.3.2.1 OBC Layout

Figure 5-14 shows PCB that hosts the two Raspberry Pi modules onboard the DESCENT mission. The footprint of the redundant OBCs is equivalent to a single CubeSat standard PC-104 form factor PCB i.e., the redundant architecture does not consume addition space within the CubeSat.

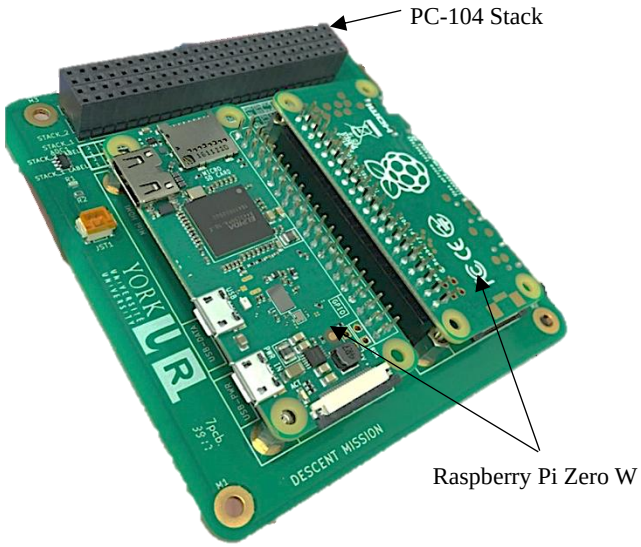

Figure 5-14: Dual Redundant OBCs on PC-104 Form Factor Host PCB

The two OBCs are powered through a Clyde Space EPS and 20Whr battery. Each OBC connected to an independent 5V switched line. The Clyde Space EPS feature a watchdog timer that resets the power bus if it times out. This watchdog timer serves as the external memory-less watchdog to initiates the arbitration process if the OBC hangs up. Both OBCs share the spacecraft bus, which they access through the common PC-104 stack header. Communication between the OBCs is established using UART emulated over the USB port.

The arbitration process for the dual redundant architecture in DESCENT was modified to prevent either module from being permanently disabled. This modification only affects the redundant branch since the process supervisor never permanently disables any OBC. If a handshake between the two modules is established or if the process supervisor does not receive a handshake acknowledge, the arbitration process follows identical mode transitions as discussed in Chapter 4 . However, if the redundant module fails to receive a handshake request, it simply takes command of the spacecraft bus instead of transitioning into the *Reset* mode. Under this modification, the role of the process supervisor never transfers from one OBC to the next. Instead, both OBC always power up after a bus reset and initiate the arbitration process. If the process supervisor experiences a permanent failure, the redundant module will always take over the spacecraft bus. Although this reduces the availability of the system since the redundant module must always wait for the timer in the Handshake phase to timeout, neither of the OBCs are removed from the arbitration process which provides immunity against intermittent faults in the OBC or in the power sub-system. A persistence metric in the OBC's Health Quotient prevents frequent switchovers from the redundant module back to the process supervisor.

5.3.2.2 Health Metrics

The health assessment for the OBCs onboard DESCENT was primarily focused on their ability to communicate with various sub-systems on the spacecraft bus. Table 5-14 lists the health metrics used on DESCENT to arbitrate between the two Raspberry Pi modules. The justification for the relative ranks assigned to the various metrics is also briefly discussed in this section.

Table 5-14: Health Metrics Used Onboard DESCENT

Health Metric	Rank Index	Original Bounds	Original Step Size
I^2C Functionality	5	[0, 1]	1
SPI CE0 (Xbee) Functionality	4	[0, 1]	1
Inter-satellite UART Functionality	2	[0, 1]	1
SPI CE1 (Payload) Functionality	3	[0, 1]	1
Persistence Metric	1	[0, 17280]	1
Module Rank	0	[1, 2]	1

Health metrics for all communication busses are represented by a single Boolean metric representing its functionality. The fidelity of data on each bus is encompassed within this metric, which only evaluates to 1 if there are no bit errors for the queried parameter. The I^2C metric is given the highest precedence since it provides the data interface for the Electrical Power System (EPS). To test this metric, the firmware version of the EPS is queried. There are two systems on the platform that communicate over SPI, an Xbee transceiver used for inter-satellite communications and a secondary payload. The inter-satellite communications channel is given precedence, and it is also tested by requesting the firmware version of the Xbee. A wired UART link acts as a backup communication interface between the two satellites, which is also given higher precedence over the secondary payload. This is because all communications between the

mother satellite and the ground are routed through the daughter satellite, that is, the mother satellite does not have a dedicated space-to-ground link. The wired inter-satellite communication link is test by pinging the OBC on the daughter satellite. Lastly, the persistence metric provides similar functionality as shown in the ground validation tests and minimizes the switchover of modules. The value of the persistence metric increments by one every 15 minutes, and the metric is bounded by a nominal mission lifetime of 6 months.

5.3.2.3 Platform-Based Arbitration Parameters

Table 5-15 lists the configuration of the Platform-Based arbitration parameters for the redundant Raspberry Pi modules onboard DESCENT.

Table 5-15: Configuration of Platform-Based Arbitration Parameters On DESCENT

Platform-Based Parameter	Value
$T_{HDSKRequestInterval}$	3 seconds
$T_{HDSKRequest}$, $T_{HQRequest}$, $T_{EOPMessage\_S}$, $T_{SyncMessage\_S}$, T_{EOPAck_R}, $T_{SyncMessage\_R}$	~6.7 milliseconds
$T_{BroadcastInterval}$, $T_{EOPInterval\_R}$, $T_{SyncInterval\_R}$	10 milliseconds
T_{PDM}	500 milliseconds
$T_{Resolve}$	500 milliseconds
$N_{MaxCycleCount}$	1
$N_{HQRequests}$, $N_{EOPMessages\_S}$, $N_{SyncMessages\_S}$, $N_{EOPAcks\_R}$, $N_{SyncMessages\_R}$, $N_{BroadcastMessages}$	1
$N_{HDSKRequests}$	10

$T_{EOPInterval\_S}$	10 milliseconds
$T_{SyncInterval\_S}$	10 milliseconds

The request and acknowledgement message broadcast times are based on 9600 baud communication rate and a fixed length 8-byte message for both requests and acknowledgements. The maximum number of incomplete arbitration cycles allowed is only 1. This follows from the modification for DESCENT discussed in section 5.3.2.1. If the process supervisor is unresponsive, the redundant module directly takes command of the spacecraft bus. Since the frequency of processor upsets for the Raspberry Pi module was not known, this design choice eliminates forced bus resets during the arbitration process, thereby increasing the availability of the system.

5.3.2.4 Periodic Backups

To increase the robustness of the dual redundant architecture, a periodic backup is performed by the process supervisor to the redundant module. This minimizes loss of mission telemetry and payload data, as well as roll back of the mission configuration state. If the process supervisor experiences a failure mode and the redundant module takes command of the spacecraft, the redundant module recovers the last backup performed by the process supervisor and continues mission operations from this stored checkpoint.

Since the process supervisor must power up the redundant module to perform the backup, another modification is made to the Handshake Phase of the redundant module that halts the arbitration cycle. This is necessary since the redundant module has no knowledge if it was powered up due to a power reset or because the process supervisor is performing a periodic backup. The modification enables the process supervisor to send a handshake request specifically for the backup. If the redundant module acknowledges this handshake, it will halt its arbitration cycle and

the process supervisor will perform the backup. However, if the redundant module fails to acknowledge the handshake, the process supervisor will shut down the redundant module before its watchdog timer in the Handshake Phase expires. This ensures that the redundant module does not force an arbitration cycle during the periodic backup.

The process supervisor on DESCENT performs these periodic backups every 12 hours. The USB ports used for the inter-OBC communication also emulate an ethernet port, which established a local network connection between the OBCs. This network connection is used to copy files between the two Raspberry Pi modules using the secure copy command line tool.

5.3.3 Current Status

Figure 5-15 shows the fully integrated DESCENT spacecraft, comprising of both the mother and daughter CubeSats. The spacecraft was launched to the International Space Station (ISS) on the 2nd of October 2020. The spacecraft was subsequently inserted into orbit on the 5th of November 2020. At the time of preparing this thesis, the ground operations team has been able to communicate with the daughter satellite (integrates space grade OBC), but not with the mother satellite (integrates the dual redundant OBCs). Several hypotheses have been formulated to establish a root cause for this issue.

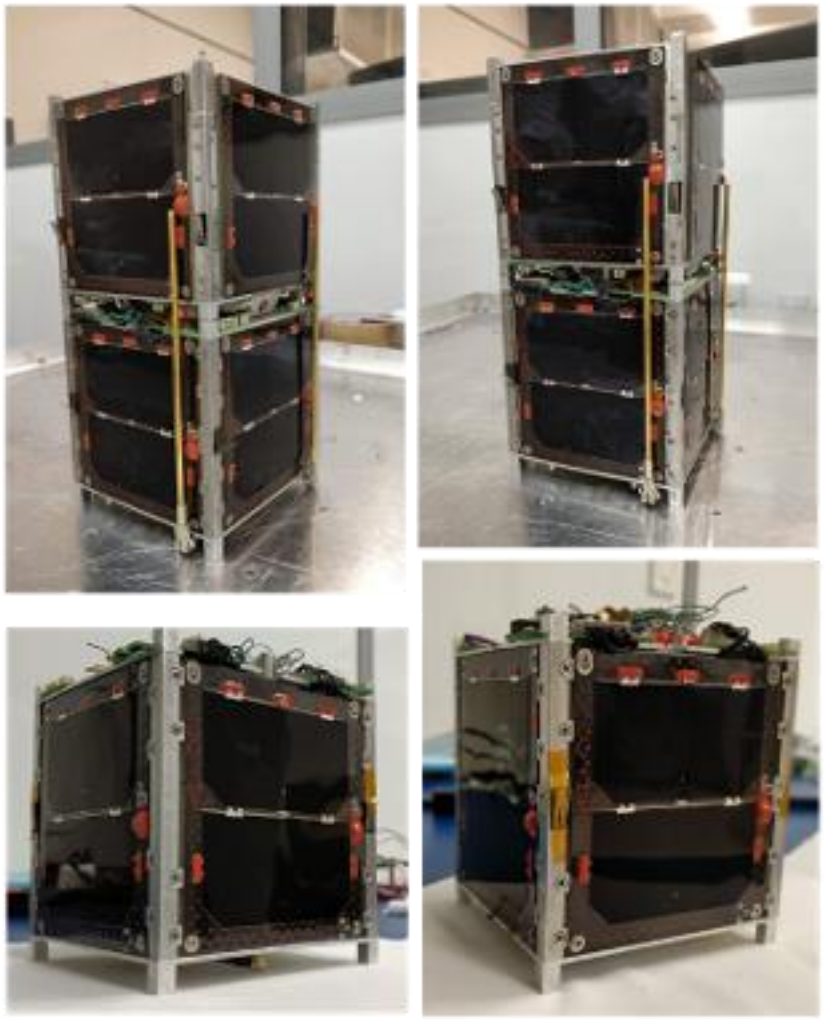

Figure 5-15: The DESCENT Spacecraft

Telemetry from the daughter satellite shows extremely low power generation profiles leading to frequent resets of the OBC. The low power generation stems from a bug in the ADCS software that is causing an uncontrolled tumble of the spacecraft. Additionally, spacecraft re-work performed after solar panel integration is suspected to have degraded panel performance. It is known from the spacecraft systems tests conducted prior to launch, that the mother satellite consumes greater power than the daughter satellite, and it is possible that the low power generation

is inhibiting the mother satellite from powering up for long enough to establish communication with the ground. If this is the case, reducing the spacecraft body rates could improve the power generation profile.

Another hypothesis relates to the communications architecture of the satellite. The mother satellite does not have its independent space-to-ground transceiver. Instead, communications between the mother satellite and the ground are routed via the daughter satellite. There are two communication channels between the CubeSats, a wired UART link intended for use prior to tether deployment and a wireless XBee transceiver intended for use once the tether has been deployed and the CubeSats are 100m apart. The flight software on the mother satellite has an FDIR mechanism that autonomously switches communication to the Xbee if the ground contact has not been establish for 3 weeks. It is possible that the mother satellite may be trying to communicate over the Xbee, however, the proximity of the two CubeSats is causing the link to saturate. If this is the case, the inter-satellite link should improve once the tether has been deployed.

The DESCENT ground operations team is currently executing in-orbit tests to validate (or disprove) these hypotheses.

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

Summary: This chapter summarizes the novel contributions of this research thesis and provides concluding remarks for the proposed redundancy architecture. This chapter also summarizes the limitations of the current architecture and proposes a path forward that builds upon the current work.

6.1 SUMMARY OF CONTRIBUTIONS

A novel cold redundancy architecture was proposed in this research thesis that de-centralizes the arbitration logic in software. This de-centralized redundancy architecture eliminates the single point of failure found in existing cold-redundancy architectures i.e., the external arbiter. Furthermore, the redundancy architecture enables extensibility. This generalized extensible architecture allows the number of redundant OBCs, as well their individual health metrics, to be tuned to specific mission requirements.

6.1.1 Self-Arbitrating Cold Redundancy Architecture

Redundant OBC architectures typically rely on external Watchdog Processors (WDP) to execute arbitration logic and assign control of the spacecraft bus to one of the redundant OBCs. The watchdog processors, in addition to the watchdog timers required to initiate the arbitration process, act as single points of failure in the redundancy architecture. The arbitration process proposed in this thesis eliminates this external arbiter by decentralizing the arbitration logic amongst the OBCs themselves and does not require an external arbiter to execute the arbitration logic. Moreover, none of the OBCs themselves act as a single point of failure, that is, the proposed arbitration process does not rely on nominal functionality of any particular OBC. This is achieved

by dynamically re-assigning arbitration process roles in orbit to remove failed OBCs from the arbitration process. The external watchdog processor is reduced to a memory-less watchdog timer, which is inherently more reliable due to its reduced complexity.

6.1.2 Precedence-Based Unique Health Quotients

Typical redundancy architectures rely on an "all or nothing" strategy of characterizing OBC health in which the OBC resets the external watchdog timer only if it passes all its internal health checks. However, this prohibits a comparative health analysis in larger networks of redundant OBCs, especially if several of the redundant OBCs have experienced varying degrees of degradation. A novel strategy to aggregate various health metrics into a unified health metric for each OBC, denoted as its Health Quotient, was presented in this thesis. The methodology allows individual health metrics to be ranked and weighted relative to all other health metrics selected for the particular OBC architecture, thereby providing precedence to the most valued health metrics. Furthermore, the normalization and aggregation of individual health metrics ensures a unique Health Quotient is generated for each redundant OBC on the spacecraft. The proposed characterization of OBC health enables a comparative capability assessment amongst all OBCs to determine which one can execute the most mission critical functions.

6.1.3 Extensible Redundancy Architecture

The arbitration architecture abstracts away the number of redundant OBCs deployed on the from spacecraft from the underlying arbitration logic. The arbitration logic itself is executed based on one of two roles that an OBC can execute during the arbitration process, namely the process supervisor or a redundant module. Each module is assigned a default role at mission epoch, which is dynamically adjusted during flight in response to failure modes. This enables extensibility of

the arbitration process and additional OBCs can be added to the redundant architecture without requiring any modifications to the underlying arbitration logic. This property of extensibility holds true for the Health Quotient itself as well. The proposed methodology to normalize and aggregate the health metrics is agnostic of the actual health metrics themselves. The computation of the health metric is similarly abstracted away from the arbitration logic, enabling the health metrics to be updated according to the requirements of the mission.

6.2 CONCLUSION

The proposed architecture was implemented in the lab using Raspberry Pi Zero W modules in triple modular redundancy. A spacecraft EPS was emulated using an independent Raspberry Pi module that could be commanded to the power cycle individual or all redundant OBCs. The rest of the spacecraft bus was emulated using a Real Time Clock (DS1307) as a dummy payload. Health metrics defined for the OBC computed internal performance as well as ability to access the emulated spacecraft bus. The Health Quotient generated by aggregating these health metrics was used to perform a comparative health assessment between all OBCs and evaluate their functional capabilities. The arbitration architecture was validated by emulating faults in the OBCs executing both roles, that is, the process supervisor and the redundant modules. These ground tests demonstrated the ability of the architecture to dynamically re-assign the arbitration roles, and not rely on any particular OBC to function nominally. An additional Raspberry Pi Zero W module was added as a fourth redundant OBC to demonstrate the extensibility of the proposed architecture.

Finally, the proposed architecture was implemented on the DESCENT mission using dual redundant Raspberry Pi Zero W modules. System validation tests performed on the integrated satellite demonstrated that the proposed architecture can be implemented using existing the CubeSat bus design and COTS OBCs. The DESCENT spacecraft is current in orbit and work is

ongoing to establish communications with the CubeSat on which this architecture was implemented.

6.3 FUTURE WORK

6.3.1 Potential Areas of Application

The discussion of the proposed redundancy architecture presented in this research thesis was focused on enabling the use of low cost commercially available OBCs in small form factor spacecraft. However, the application of the proposed redundancy architecture can be expanded to other areas as well, both in space as well as ground-based applications.

6.3.1.1 Distributed Computing in Satellites

To enable the use of COTS OBCs in satellites, the cold redundant architecture proposed in this thesis powers down all redundant modules after the arbitration is resolved. This helps reduce the Total Ionizing Dose (TID) damage to the OBCs over the course of the mission, but this also implies that the redundant COTS OBCs are not utilized to offload any computationally intensive tasks. The proposed redundancy architecture is, however, not limited to COTS OBCs, and can be implemented using space grade OBCs as well. Since these OBCs are much more resilient against ionizing radiation, future implementations of this architecture can explore the use of distributed computing. The concept of dynamic re-assignment of the Process Supervisor role can be extended to the computation task role of each OBC within a distributed network. In such a scenario, the most critical functionality could be carried out by the OBC in command of the spacecraft bus, while execution of lower priority tasks can be assigned based on module ranks. This would enable graceful degradation of a distributed system, and the loss of an OBC would trigger the lowest priority task to be transferred to the next available OBC.

Since the architecture does not have any dependence on the underlying OBC, it can also be implemented using different OBC architectures. Utilizing different OBC architectures can help eliminate common-mode errors. The long-term analysis of the OBC state-of-health while performing distributed computing tasks, can be used to increment specialized health metrics which can provide precedence during arbitration to the OBC platforms that perform better.

6.3.1.2 Ground Based Applications

Since the proposed redundancy architecture is not application dependent, it can be extended to ground based applications to improve system robustness. The use of the proposed architecture in Internet of Things (IoT) compatible devices can be explored. The platform agnostic nature of the proposed architecture can be leveraged to integrate it in numerous end-user applications of IoT devices such as home automation and retail. The proposed architecture can also help introduce redundancy in applications with low power profiles, since all redundant OBCs can be powered down after the arbitration cycle and system doesn't need to power an external arbiter to support the redundant architecture. This can be especially useful in remote applications, such as remote weather monitoring systems, where power generation and maintenance opportunities might be limited.

6.3.2 Current Limitations and Future Improvements

6.3.2.1 Improving System Availability

Although the proposed architecture improves upon existing cold redundancy architectures, it imposes a limitation on system availability. As the redundancy architecture scales up, the total time required to resolve the arbitration scales up as well. The time complexity of the arbitration phases is driven by the number of OBCs in the redundancy architecture. From the expressions for the software watchdog timers discussed in Chapter 4 the time required to resolve the arbitration

process increases linearly as the number of redundant OBCs increase. As a result, the availability of the system diminishes, therefore forcing a trade-off between system lifetime and system availability.

Future work on the architecture might explore improvements in system availability by selectively reducing the set of OBCs participating in any given arbitration cycle. Other OBCs might be kept dormant during this time, and partial or complete failures of any OBC within the subset of active OBCs can trigger substitution from the set of dormant OBCs. An effective strategy would need to be developed to determine which of the OBCs participate in any given arbitration cycle.

6.3.2.2 Expanding Scope of Test and Validation

The scope of ground tests performed to validate the proposed architecture, while integrated with the rest of mission flight software, was limited to the ground system tests performed on the DESCENT mission. The ground tests validated the proposed architecture's ability to arbitrate between redundant OBCs in an idealized lab setting. However, characterization of software EDAC mechanisms that protect the execution of the arbitration instructions themselves was beyond the scope of this research. This also includes characterizing the processing overhead of such EDAC mechanisms. Moreover, the interaction between these software EDAC mechanisms and the arbitration logic, and their effect on system availability, must also be characterized.

The scope of the ground validation tests could be expanded to include these performance characterization tests. Additional ground test equipment would be required to inject errors in program instructions, volatile memory registers, and persistent configuration to emulate the effects of ionizing radiation during mission operations, sporadic bus resets and subsequent arbitration cycles.

6.3.2.3 Adaptive Health Metrics

The proposed architecture utilizes a fixed set of health metrics against which the arbitration mechanism characterizes an OBC's state-of-health. However, over the course of the mission, monitoring of additional health metrics may become necessary due to SEEs and sub-system degradation. This degradation might occur on sub-systems other than the OBC but could still affect OBC performance based on the physical interaction between the two sub-systems.

Future implementations of this architecture could expand on the strategy used to generate the OBC Health Quotient by enabling updates to the list of health metrics as well as dynamic re-assignment of the health priorities. Existing machine learning techniques used for root cause analysis based on telemetry violations can be used in conjunction with the proposed architecture to enable autonomous updates to the list of health metrics and their relative priorities. These techniques would not only determine which health metrics need to be observed, but also determine how relative priorities of existing metrics might be updated to accurately characterize the OBC's capability of tackling system degradation.

REFERENCES

[1] K. L. Bedingfield, R. D. Leach and M. B. Alexander, "Spacecraft System Failures and Anomalies Attributed to the Natural Space Environment," NASA, Alabama, 1996.

[2] R. Gubby and J. Evans, "Space environment effects and satellite design," *Journal of Atmospheric and Solar-Terrestrial Physics,* vol. 64, no. 16, pp. 1723-1733, 2002.

[3] Texas Instruments, "Radiation Environments," in *Radiation Handbook for Electronics*, Dallas, pp. 4-24.

[4] H. Bokil, "COTS Semiconductor Components for the New Space Industry," in *2020 4th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, Penang, 2020.

[5] M. Nikulainen, "Usage of COTS EEE Components in ESA Space Programs," in *European Space Components Conference*, Noordwijk, 2019.

[6] J. R. Letaw, "Single Event Effects Rate Predictions in Space," *Nuclear Instruments and Methods in Physics Research,* Vols. 56-57, Part 2, pp. 1260-1262, 1991.

[7] H. B. Garret, "Space Environments and Survivability," in *The International Handbook of Space Technology*, Berlin, Springer, 2014, pp. 37-59.

[8] T. R. Oldham and F. B. McLean, "Total Ionizing Dose Effects in MOS Oxides and Devices," *IEEE Transactions on Nuclear Science,* vol. 50, no. 3, pp. 483-499, 2003.

[9] Texas Instruments, "Radiation effects in electronics - dose effects," in *Radiation Handbook for Electronics*, Dallas, pp. 38-46.

[10] J. D. Cressler, "Radiation Effects in SiGe Technology," *IEEE Transactions on Nuclear Science,* vol. 60, no. 3, pp. 1992-2014, 2013.

[11] G. L. Hash, M. R. Shaneyfelt, F. W. Sexton and P. S. Winokur, "Radiation Hardness Assurance Categories for COTS Technologies," in *1997 IEEE Radiation Effects Data Workshop NSREC Snowmass*, Snowmass Village, 1997.

[12] D. Sinclair and J. Dyer, "Radiation Effects and COTS Parts in SmallSats," in *27th Annual AIAA/USU Conference on Small Satellites*, 2013.

[13] D. McKnight, "Examination of spacecraft anomalies provides insight into complex space environment," *Acta Astronautica,* vol. 158, pp. 172-177, 2017.

[14] H. B. Garrett and S. Close, "Impact-Induced ESD and EMI/EMP Effects on Spacecraft - A Review," *IEEE Transacitons on Plasma Science,* vol. 41, no. 12, pp. 3545-3557, 2013.

[15] R. L. Pease and D. R. Alexander, "Hardness Assurance for Space System Microelectronics," *Radiation Physics and Chemistry,* vol. 43, no. 1-2, pp. 191-204, 1994.

[16] Texas Instruments, "Understanding quality levels for high reliability-rated components," 2018. [Online]. Available: https://www.ti.com/lit/ml/sszb156a/sszb156a.pdf?&ts=1589838914912. [Accessed 15 May 2020].

[17] F.-x. Yu, J.-R. Liu, Z.-L. Huang, H. Luo and Z.-M. Lu, "Overview of Radiation Hardening Techniques for IC Design," *Information Technology Journal,* vol. 9, pp. 1068-1080, 2010.

[18] CubeSat Shop, "Command & data handling," [Online]. Available: https://www.cubesatshop.com/product-category/command-and-data-handling/. [Accessed 17 August 2020].

[19] Endurosat, "ONBOARD COMPUTER (OBC)," [Online]. Available: https://www.endurosat.com/cubesat-store/cubesat-obc/onboard-computer-obc/. [Accessed 17 August 2020].

[20] Pumpkin Space Systems, "Motherboard Module 2," [Online]. Available: https://www.pumpkinspace.com/store/p208/mbm2.html. [Accessed 17 August 2020].

[21] Raspberry Pi, "Raspberry Pi 4," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Accessed 28 March 2021].

[22] Raspberry Pi , "Compute Module 4," [Online]. Available: https://www.raspberrypi.org/products/compute-module-4/?variant=raspberry-pi-cm4001000. [Accessed March 2021].

[23] Raspberry Pi, "Raspberry Pi Zero W," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-zero-w/. [Accessed 28 March 2021].

[24] BeagleBoard, "BeagleBone Black," [Online]. Available: https://beagleboard.org/black. [Accessed March 2021].

[25] Beagle Board, "Pocket Beagle," [Online]. Available: https://beagleboard.org/pocket. [Accessed March 2021].

[26] Gomspace, "Nano Mind A3200," [Online]. Available: https://gomspace.com/shop/subsystems/command-and-data-handling/nanomind-a3200.aspx. [Accessed 28 March 2021].

[27] PC/104 Embedded Consortium, "PC/104 Specification Version 2.6," 13 October 2008. [Online]. Available: https://pc104.org/wp-content/uploads/2015/02/PC104_Spec_v2_6.pdf. [Accessed 18 January 2021].

[28] C. Underwood, S. Pellegrino, V. J. Lappas, C. P. Bridges and J. Baker, "Using CubeSat/micro-satellite technology to demonstrate the Autonomous Assembly of a Reconfigurable Space Telescope (AAReST)," *Acta Astronautica,* vol. 114, pp. 112-122, 2015.

[29] M. Fayyaz, T. Vladimirova and J.-M. Caujolle, "Adaptive Middleware Design for Satellite Fault-Tolerant Distributed Computing," in *2012 NASA/ESA Conference on Adaptive Hardware and Systems,* Erlangen, 2012.

[30] D. Mohan, K. Arichandran and I. McLoughlin, "Fault Tolerant Computer for Low Earth Orbit Micro satellites," in *Small Satellites Systems and Services Services,* 2004.

[31] S. de Jong, G. T. Aalbers and J. Bouwmeester, "IMPROVED COMMAND AND DATA HANDLING SYSTEM FOR THE DELFI-N3XT NANOSATELLITE," in *59th International Astronautical Congress*, Glasgow, 2008.

[32] K. Weiherer, L. Osinski and J. Mottok, "Software-Based Triple Modular Redundancy with Fault-Tolerant Replicated Voters," in *32nd International Conference on Archtiecture of Computing Systems,* Copenhagen, 2019.

[33] ST, "Software techniques for improving microcontrollers EMC performance," June 2014. [Online]. Available: https://www.st.com/resource/en/application_note/cd00004037-software-techniques-for-improving-microcontrollers-emc-performance-stmicroelectronics.pdf. [Accessed November 2020].

[34] Freescale Semiconductor, "Improving the Transient Immunity Performance of Microcontroller-Based Applications," June 2005. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN2764.pdf. [Accessed January 2021].

[35] B. D. Spieth, K. S. Qassim, R. N. Pittman and D. A. Russell, "Shielding Electronics Behind Composite Structures," *IEEE TRANSACTIONS ON NUCLEAR SCIENCE,* vol. 45, no. 6, pp. 2752-2757, 1998.

[36] B. Klamm, "Passive Space Radiation Shielding: Mass and Volume Optimization of Tungsten-Doped PolyPhenolic and Polyethylene Resins," in *29th Annual AIAA/USU Conference on Small Satellites,* Logan, 2015.

[37] T. A. Woodrow and E. A. Ledbury, "EVALUATION OF CONFORMAL COATINGS AS A TIN WHISKER MITIGATION STRATEGY, PART II," in *SMTA International Conference,* Rosemont, IL, 2006.

[38] NASA, "Tin Whisker (and other metal whiskers)," 24 January 2011. [Online]. Available: https://nepp.nasa.gov/WHISKER/index.html. [Accessed 25 March 2021].

[39] European Space Agency, "Tin Whisker," 01 November 2017. [Online]. Available: https://www.esa.int/ESA_Multimedia/Images/2017/11/Tin_whisker. [Accessed 27 March 2021].

[40] L. Zhu and L. Yu, "A design of decentralized dual mode redundant hot standby arbitration switch-over logic and architecture," in *2018 International Conference on Electronics Technology (ICET)*, Chengdu, 2018.

[41] W. Johnston, "Increasing system reliability-a survey of redundant control methods," in *Fourth Annual Canadian Conference Proceedings., Programmable Control and Automation Technology Conference and Exhibition*, Toronto, 1988.

[42] E. Gracic, A. Hayek and J. Borcsok, "Implementation of a fault-tolerant system using safety-related Xilinx tools conforming to the standard IEC 61508," in *2016 International Conference on System Reliability and Science (ICSRS)*, Paris, 2016.

[43] L. Bo, X. Feng and Y. Yunhong, "A Co-processing Method Based on Warm Standby Systems," in *2014 IEEE Symposium on Computer Applications and Communications*, Weihai, 2014.

[44] M. Duríček, M. Pohronská and T. Krajčovič, "Functional Prototype of Multiple Watchdog System Implemented in FPGA," in *2012 International Conference on Applied Electronics*, Pilsen, 2012.

[45] A. M. El-Attar and G. Fahmy, "A Study of Fault Coverage of Standard and Windowed Watchdog Timers," in *2007 IEEE International Conference on Signal Processing and Communications*, Dubai, 2007.

[46] PuWeihua, "A kind of COTS onboard computer fault-tolerant design," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, Harbin, 2017.

[47] M. Fayyaz and T. Vladimirova, "Fault-Tolerant Distributed Approach to Satellite On Board Computer Design," in *2014 IEEE Aerospace Conference*, Big Sky, 2014.

[48] B. F. Straka, "Implementing a Microcontroller Watchdog with a Field- Programmable Gate Array (FPGA)," NASA, Florida, 2013.

[49] J. Javanainen, "Reliability Evaluation of Aalto-1 Nanosatellite Software Architecture," Aalto University, 2016.

[50] R. A. Austin, N. Mahadevan, B. D. Sierawski, G. Karsai, A. F. Witulski and J. Evans, "A CubeSat-Payload Radiation-Reliability Assurance Case using Goal Structuring Notation," in *2017 Annual Reliability and Maintainability Symposium (RAMS)*, Orlando, 2017.

[51] P. Cuenot, P. Bouche, R. de Simone, J. Deantoni and A. Oueslati, "Early validation of satellite COTS-on-board computing systems," in *10th European Congress on Embedded Real-Time Software and Systems*, Toulouse, 2020.

[52] M. Yang, H. Wang, C. Wu, C. Wang, L. Ding, Y. Zheng and Z. Jin, "Space Flight Validation of Design and Engineering of the ZDPS-1A Pico-satellite," *Chinese Journal of Aeronautics,* vol. 25, no. 5, pp. 725-738, 2012.

[53] S. Busch, "Robust, Flexible and Efficient Design for Miniature Satellite Systems," University of Würzburg, 2016.

[54] S. Busch, P. Bangert, S. Dombrovski and K. Schilling, "UWE-3, in-orbit performance and lessons learned of a modular and flexible satellite bus for future pico-satellite formations," *Acta Astronautica,* vol. 117, pp. 73-89, 2015.

[55] K. Laizans, I. Sunter, K. Zalite, H. Kuuste, M. Valgur, K. Tarbe, V. Allik, G. Olentsenko, P. Laes, S. Latt and M. Noorma, "Design of the fault tolerant command and data handling subsystem for ESTCube-1," *Proceedings of the Estonian Academy of Sciences,* vol. 63, no. 2S, pp. 222-231, 2014.

[56] A. Slavinskis, M. Pajusalu, H. Kuuste, E. Ilbis, T. Eenmäe, I. Sünter, K. Laizans, H. Ehrpais, P. Liias, E. Kulu, J. Viru, J. Kalde, U. Kvell, J. Kütt, K. Zalite, K. Kahn and S. Lätt, "ESTCube-1 in-orbit experience and lessons learned," *IEEE Aerospace and Electronic Systems Magazine,* vol. 30, no. 8, pp. 12-22, 2015.

[57] S. Singh, J. Bellardo and J. Puig-Suari, "A Data-Driven Approach to CubeSat Health Monitoring," in *31st Annual AIAA/USU Conference on Small Satellites*, Utah, 2017.

[58] " Telemetry Fault-Detection Algorithms: Applications for Spacecraft Monitoring and Space Environment Sensing," *Journal Of Aerospace Information systems,* vol. 15, no. 5, pp. 239-252, 2018.

[59] S. Lindsay and D. M. Woodbridge, "Spacecraft State-of-health (SOH) Analysis via Data Mining," in *SpaceOps 2014 Conference*, Pasadena, 2014.

[60] ESA eoPortal, "BeeSat-1 (Berlin Experimental Educational Satellite-1)," [Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/b/beesat-1#footback6%29. [Accessed 11 April 2021].

[61] W. J. Ubbels, A. R. Bonnema, E. D. van Breukelen, J. H. Doorn, R. van den Eikhoff, E. Van der Linden, G. T. Aalbers, J. Rotteveel, R. J. Hamann and C. J. M. Verhoeven, "Delfi-C3: a student nanosatellite as a test-bed for thin film solar cells and wireless onboard communication," in *Proceedings of 2nd International Conference on Recent Advances in Space Technologies, 2005. RAST 2005.*, Istanbul, 2005.

[62] C. Underwood, G. Richardson and J. Savignol, "SNAP-1: A Low Cost Modular COTS-Based Nano-Satellite – Design, Construction, Launch and Early Operations Phase," in *15th AIAA / USU Conference on Small Satellites*, Utah, 2001.

[63] M. N. Sweeting, "Modern Small Satellites— Changing the Economics of Space," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 343-361, 2018.

[64] IEEE Standards Association, "754-2019 - IEEE Standard for Floating-Point Arithmetic," IEEE, 2019.

[65] M. J. Crawley, Statistics: An Introduction Using R, Wiley, 2005.

[66] K. Mowery, M. Wei, D. Kohlbrenner, H. Shacham and S. Swanson, "Welcome to the Entropics: Boot-Time Entropy in Embedded Devices," in *2013 IEEE Symposium on Security and Privacy*, Berkeley, 2013.

[67] S. H. Zoalfakar and R. E. El Badawi, "Reliability Prediction of on-board subsystem satellite by Monte Carlo Methodology," in *IOP Conference Series: Materials Science and Engineering*, 2019.

[68] M. M. El Genidy and E. A. Hebeshy, "Estimation the Failure Rate and Reliability of the Triple Modular Redundancy System Using Weibull Distribution," *Asian Journal of Scientific Research*, vol. 10, pp. 128-138, 2017.

[69] S. N. Srihari, "Reliability analysis of majority vote systems," *Information Sciences*, vol. 26, no. 3, pp. 243-256, 1982.

[70] V. M. Agüero and R. C. Adamo, "Space Application of Spindt Cathode Field Emission Arrays," in *6th Spacecraft Charging Technology Conference*, 2000.

[71] I. C. Bell, B. E. Gilchrist, J. K. McTernan and S. G. Bilen, "Investigating Miniaturized Electrodynamic Tethers for Picosatellites and Femtosatellites," *Journal of Spacecraft and Rockets*, vol. 54, no. 1, pp. 55-66, 2017.

[72] D. M. Harland and R. D. Lorenz, "Environmental Failures," in *Space Systems Failures*, Berlin, Springer-Praxis, 2005, pp. 265-284.

[73] R. A. Kimble, P. Goudfrooij and R. L. Gilliland, "Radiation Damage Effects on the CCD Detector of the Space Telescope Imaging Spectrograph," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4013, pp. 532-545, 2000.

[74] M. J. Gadlage, R. D. Schrimpf, J. M. Benedetto, P. H. Eaton, D. G. Mavis, M. Sibley, K. Avery and T. L. Turflinger, "Single Event Transient Pulsewidths in Digital Microcircuits," *IEEE Transactions on Nuclear Science*, vol. 51, no. 6, pp. 3285-3290, 2004.

[75] D. Binder, E. C. Smith and A. B. Holman, "Satellite Anomalies from Galactic Cosmic Rays," *IEEE Transactions on Nuclear Science*, Vols. NS-22, no. 6, 1975.

[76] C. Poivey, G. Gee, K. A. LaBel and J. L. Barth, "In-flight Observations of Long-Term Single Event Effect (SEE) Performance on X-ray Timing Explorer (XTE) Solid-State Recorders (SSRs)," in *IEEE Radiation Effects Data Workshop*, Atlanta, 2004.

[77] N. V. Kuznetsov, "The Rate of Single Event Upsets in Electronic Circuits onboard Spacecraft," *Cosmic Research*, vol. 43, no. 6, pp. 423-431, 2005.

[78] C. Poivey, J. L. Barth, K. A. LaBel, G. Gee and H. Safren, "In-flight observations of long-term single-event effect (SEE) performance on Orbview-2 solid state recorders (SSR)," in *IEEE Radiation Effects Data Workshop*, Monterey, 2003.

[79] A. M. Albadri, R. D. Schrimpf, K. F. Galloway and D. G. Walker, "Single Event Burnout in Power Diodes: Mechanisms and Models," *Microelectronics Reliability*, vol. 26, no. 2-4, pp. 317-325, 2006.

[80] M. Alenspach, J. R. Brews, K. F. Galloway, G. H. Johnson, R. D. Schrimpf, R. L. Pease, J. L. Titus and C. F. Wheatley, "SEGR: A Unique Failure Mode for Power MOSFETs in Spacecraft," *Microelectronics Reliability,* vol. 36, no. 11/12, pp. 1871-1874, 1996.

[81] D. K. Nichols, J. R. Coss and K. P. McCarty, "Single Event Gate Rupture in Commercial Power MOSFETs," in *Second European Conference on Radiation and its Effects on Components and Systems,* Saint Malo, 1993.

[82] J. L. Barth, J. W. Adolphsen and G. B. Gee, "Single event effects on commercial SRAMs and power MOSFETs: final results of the CRUX flight experiment on APEX," in *IEEE Radiation Effects Data Workshop,* Newport Beach, 1998.

[83] R. Perez, "Analysis and Effects of Space Radiation Induced Single Event Transients," in *2016 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, Ottawa, 2016.

[84] H. L. Lam, D. H. Boteler, B. Burlton and J. Evans, "Anik-E1 and E2 satellite failuresof January 1994 revisited," *Space Weather,* vol. 10, no. 10, 2012.

[85] D. C. Wilkinson, S. C. Daughtridge, J. L. Stone, H. H. Sauer and P. Darling, "TDRS-1 Single Event Upset and Effects of the Space Environment," *IEEE Transactions on Nuclear Science,* vol. 38, no. 6, pp. 1708-1712, 1991.

[86] G. M. Swift and S. M. Guertin, "In-Flight Observations of Multiple-Bit Upset in DRAMs," *IEEE Transactions on Nuclear Science,* vol. 47, no. 6, pp. 2386-2391, 2000.

[87] Jet Propulsion Laboratory, "NASA Lessons Learned: MSL Sol-200 Anomaly," 29 April 2014. [Online]. Available: https://llis.nasa.gov/lesson/11201. [Accessed 7 June 2020].

[88] C. Z. Chena and D. Y. Hu, "Geometry Effect with Respect to ESD and Radiative Charged Particles in SoC," in *China Semiconductor Technology International Conference (CSTIC)*, Shanghai, 2017.

[89] J. S. Mayo, H. Mann, F. J. Witt, D. S. Peck, H. K. Gummel and W. L. Brown, "The Command System Malfunction of the Telstar Satellite," *The Bell System Technical Journal*, vol. 42, no. 4, pp. 1631-1657, 1963.

[90] PC/104 Embedded Consortium, "PC/104-Plus Specification Version 2.3," 13 October 2008. [Online]. Available: https://pc104.org/wp-content/uploads/2015/02/PC104_Plus_v2_32.pdf. [Accessed 30 January 2021].

[91] R. Pillay, S. Punnekkat and S. Dasgupta, "An Improved Redundancy Scheme for the Optimal Utilization of Onboard Computers," in *2009 Annual IEEE India Conference*, Ahmedabad, 2009.

[92] Z. Dong, Y. Guo, Y. Gong and C. Li, "A High Reliability Radiation Hardened On-Board Computer System for Space Application," in *2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, Harbin, 2016.

APPENDIX A: THE EFFECTS OF IONIZING RADIATION

Total Ionizing Dose (TID) and Displacement Damage Dose (DDD) are the two mechanisms that fall under dose damage by which ionizing radiation can cause irreversible damage to spacecraft electronics. Dose damage manifests gradually, but its effects are permanent. Total Ionizing Dose (TID) is the accumulated radiation that is absorbed by the any electronic device onboard the spacecraft over its mission lifetime. The radiation absorbed by semiconductor material creates electron-hole pairs inside its gate oxide. Depending on the charge separation between the generated pairs, some of these electron-hole pairs simply recombine and have no effect on the semiconductor. The separation of the generated electron-hole pairs usually increases with an increase in the applied electric field across the semi-conductor. When these pairs can't recombine, they move through the semiconductor and try to exit the gate oxide. The higher mobility electrons are removed from the gate oxide much quicker than the positively charged holes, which are left behind are often trapped inside the gate oxide. Over time, the accumulation of these trapped charges causes alterations in carrier concentrations and lifetimes, shifts in voltage threshold, and leads to current leakage [3] [8] [9]. Although the degradation is gradual, the system eventually fails beyond recovery. Displacement Damage Dose (DDD) has similar effects to TID although the underlying mechanism is slightly different. Displacement damage is caused by protons, neutrons and high energy electrons that knock atoms out of the semiconductor lattice. The displaced atoms are trapped in interstitial spaces in the lattice increasing the lattice imperfections. As the lattice imperfections grow, the performance of the semiconductor degrades resulting in current leakage [3] [9] [10].

The effects of either type of dose damage are most commonly observed in optical instruments. Japan's Engineering Test Satellite-6 failed to achieve geostationary orbit due to a failed apogee kick motor. As a result of the lower orbit, the spacecraft experienced drastically higher doses of radiation from the outer Van Allen belt. The degradation of the solar panels was quantified to be more than an 8% reduction in power generation over the first ten days of solar panel deployment [1]. A similar degradation of solar panel performance was observed on the GOES-7 satellite due to an intense solar storm that occurred in March 1995. The resulting solar panel degradation reduced the mission life expectancy of the satellite by 2-3 years [1]. The joint mission by NASA and CNES called TOPEX/Poseidon experienced a gradual increase in the amount of dark current produced by one of its optical attitude determination sensors [72]. Similar effects were noted in the Space Telescope Imaging Spectrograph on board the Hubble Space Telescope, where the degradation in the CCD's charge transfer efficiency become notable after 3 years in orbit [73].

Unlike dose damage, Single Event Effects (SEE) cause instantaneous change in the functionality of an electrical circuit of any spacecraft system. A Single Event Effect is a broad umbrella term used for a variety of soft errors and hard errors caused by a high energy particle travelling through the semiconductor. The particle leaves an ionized trail in the body of the semiconductor which causes sporadic off-nominal effects that maybe transient or permanent [6, 7, 10, 12].

Single Event Upsets (SEU) are temporary changes in the state of a bi-stable circuit caused by the interaction of the circuit with a particle. The most significant effect of this state change are bitflips experienced by memory cells or processor registers onboard the spacecraft including volatile & non-volatile memory, registers on the Microcontroller Unit (MCU) or Floating-Point

Unit (FPU), and Field Programmable Gate Arrays (FPGA) cells. In the best-case scenario, the effect of an SEU will be limited to data corruption. However, an SEU might cause an unpredictable change in software state of the system resulting in off-nominal mission operations [6, 74]. Single Event Upsets were first reported in 1975 where unexpected triggering of JK flip-flop circuits was observed in communication satellites [75]. Since then, numerous space missions have reported memory or logical upsets resulting from increased exposure to radiation. A survey of these SEUs can be found in these papers [76] [77]. While most records for SEUs onboard spacecraft are tagged to sporadic radiation strikes, NASA's X-Ray Timing Explorer mission compiled all SEU occurrences on its onboard electronics over an 8-year period. A similar survey of SEUs was conducted onboard NASA's Orbview-2 mission, where SEU rates were observed over a 4.3-year period. These missions provided statistically significant data to correlate the changes in space weather with the frequency of SEEs [76] [78]. For instance, when satellites pass over the South Atlantic Anomaly (SAA) in Low Earth Orbit, they experience an increase in the flux of ionizing radiation due to the locally weaker magnetic field. The Hubble Space Telescope recorded several bitflips in the Random-Access Memory (RAM) module for its guidance and navigation system while crossing the South Atlantic Anomaly [1].

While SEUs are usually not destructive, there exist Single Event Effects that can cause irreversible damage to the spacecraft electronics. The most common of these effects are Single Event Burnouts (SEB) and Single Event Gate Rupture (SEGR). It is worth noting here that these effects are almost exclusively observed in power circuitry using high power Metal Oxide Semiconductor Field Effect Transistors (MOSFET) and Bipolar Junction Transistors (BJT). Single Event Burnouts occurs when the incident ionizing radiation that creates charge carrier pairs activates parasitic structure within the transistor which act like thyristors. The resulting feedback

mechanisms can cause an avalanche effect, resulting in high amounts of current and voltage passing through the device. This causes the device to heat up and eventually fail [79] [80]. In a Single Event Gate Rupture, the energy deposited by an incoming radiation strike causes a significant separation of the generated charge carrier pairs. The electric field generated by the separation of these charges adds to the electric field applied across the device terminals. If the overall electric field exceeds the dielectric breakdown strength of the gate oxide in the semiconductor, it causes permanent device failure [80] [81].

In orbit, it is difficult to distinguish between the failures due to the different destructive SEEs mentioned here. To obtain orbital data, satellites are launched with experiments that can measure specific SEEs. One such satellite was the Advanced Photovoltaics and Electronics Experiments (APEX). The Cosmic Ray Upset Experiment (CRUX) onboard that satellite measured the effects of SEEs on commercial SRAMs and power MOSFETs. In terms of destructive SEEs, the CRUX experiment focused on proton induced SEBs and showed that the frequency of SEBs increases with increasing drain-to-source voltage [82]. In general, destructive SEEs usually manifest as a sudden loss of system functionality.

There are certain transient Single Effect Events where permanent device damage is not guaranteed but can lead to irreversible damage if these effects persist. The most common of these are Single Event Latchups (SEL) and Single Event Transients (SET). A Single Event Latchup causes the activation of parasitic structures in semiconductor devices, much like those in SEBs. However, SELs can occur in low power logic circuits, where the latched state doesn't necessarily damage the transistor but can render it non-functional. Usually, resetting the power to the device causes the latchup to clear. However, depending on the sensitivity of the semiconductor device and the duration before which the latchup is cleared, an SEL could result in permanent loss of

functionality. A Single Event Transient causes a transient signal spike in part of a logical circuit. The transient travels through the circuit and clears up on its own without the need for external intervention. The width or time period of the transient signal depends on the energy deposited by the ionizing radiation. As the transient propagates through the circuit it can lead to data noise [74] [83]. The Solar Maximum Mission experienced failure of one its gyroscopes. The most likely cause identified for the failure was the susceptibility of CMOS devices to transient radiation. The satellite was able to regain control of the satellite by deploying a redundant gyroscope [1]. In combinatorial logic circuits including those used at the nodes of communication busses between onboard computers and spacecraft sub-systems, the transient can lead to noisy communication if the transient signal coincides with the latching clock signal of the communication bus. This gives rise to an interest property of SETs. Their frequency of occurrence is not just dictated by the flux of the ionizing radiation, but also the operating frequency of the logic circuit. Higher frequency combinatorial logic circuits such as high baud rate communication busses are more susceptible to SETs [74].

The final category of ionizing radiation effects is spacecraft charging. Electrostatic Discharge (ESD) is a result of differential surface charging, internal charging of insulating material or high charge build up on the spacecraft relative to the ambient plasma. While radiation contributes to the build-up of this charge, other mechanisms such as photoelectric emission can also result in differential charging. The build-up of charge culminates in a sudden discharge arc across spacecraft surfaces or from the spacecraft to the ambient plasma. This sudden discharge dissipates large amounts of energy which can damage any electrical circuits that may be electrically coupled to the surfaces experiencing the discharge arc [13]. Both Anik E1 and Anik E2 experienced failure of their Guidance and Navigation system due to an ESD. The resulting loss

of attitude control resulted in the communication loss and caused disruption of television and telephone services throughout Canada [1] [84]. A similar incident of IntelSat-K disabled the momentum wheels on the satellite. Redundant attitude actuators were deployed to restore spacecraft attitude. Both Anik satellite and the IntelSat-K experienced these events during the same solar storm [1].

While charge buildup is the primary and most prevalent mechanism for ESDs, there is evidence showing high velocity impacts can induce electrostatic discharge as well [13, 14]. This makes spacecraft electronics another potential victim to the ever-growing threat of space debris, which is not only capable of causing structural damage, but can also affect the electronics protected within the spacecraft chassis.

As mentioned earlier, onboard computers are not immune to these effects of ionizing radiation. These mechanisms affect the OBC's memory interfaces, processor registers and communication nodes. Single Event Effects, especially SEUs, that are most prominent in memory modules cause corruption of mission data, configuration and states. Since its impossible to predict which blocks of memory will be affected by the upset, the corrupted data could have mission critical consequences. An SEU in the non-volatile memory module for the Attitude Control System of the Tracking and Data Relay Satellite -1 (TDRS-1) required the ground crew to actively monitor the spacecraft state to prevent the satellite from tumbling [85]. Certain memory modules like Dynamic Random-Access Memory (DRAM) are significantly more prone to SEEs like latchups. The Cassini spacecraft integrated a DRAM architecture in its Solid-State Recorder (SSR) and recorded a baseline error rate of roughly 280 SEUs per day. This error rate spiked significantly during a solar event, recording as many as 130 SEUs per hour [86]. SEUs in processor registers like the instruction address register or the program return address in non-volatile memory can

cause unexpected code jumps and instruction execution. Similar effects might also be observed if the corrupted mission configuration raises an unexpected functional interrupt. For instance, an SEU on the Onboard Computer of the Solar Maximum Mission forced the spacecraft to switch into its safe hold condition. The change in the satellite's operating mode resulted in an interruption of data collection for 8 days [1]. Another such an incident occurred onboard the Curiosity Rover on Mars, where uncorrected errors in flash memory caused several flight software tasks to hang-up, preventing the OBC from shutting down for the rover's scheduled battery recharge cycle [87].

Latchups can result in hung up communication interfaces. Transients can cause the same interfaces to be noisy, especially those operating at high frequencies. Transients can also lead to latchups in communication nodes resulting in a constant error over that interface until it is reset [74] [83]. These effects are not limited to external communication interfaces that allow the OBC to communicate with the rest of the spacecraft bus. Internal data interfaces between the memory modules and the processor are similarly affected. Unlike SEUs where the corrupted data can be re-written if identified, latchups can damage memory cells leading to bad memory blocks with "stuck bits". It can also cause the OBC to unexpectedly halt, requiring a power reset to the processor.

Events such as SEBs or SEGRs in the power distribution systems can lead to a sudden inrush of current that may permanently damage the OBC. Power distribution systems can fail due to other reasons as well. Although the discussion of those mechanisms is beyond the scope of this thesis, the failure of the power distribution system due to any reason that causes a spike in voltage or an inrush current can damage the OBC. ESDs can have similar affects if the discharge arc strikes the OBC [88].

Even though the effects of dose damage are most evident in optical devices, the degradation experienced by semiconductors in every other sub-system including the onboard computer is similar. The effect of TID on the spacecraft command system was most notably observed after a US Air Force test named Starfish, conducted in 1962, detonated a nuclear warhead at an altitude of 325 Km above the Pacific Ocean. The blast ionized a significantly large region in the upper atmosphere which released electrons that were trapped in Earth's magnetosphere. This newly created artificial radiation belt lasted for about a decade and accounted for an increase of about two orders of magnitude in the cumulative dose experienced by the satellites. Telstar, the world's first communication satellite, was one of the satellites affected by this intense increase in radiation and experienced a failure of its command system. Even though functionality was temporarily recovered, the irreversible damage to the system only allowed the satellite to function for another two months. In the months following the Starfish test, 6 other satellites experienced some form of complete or partial failure due to the increased radiation dosage [72, 89].