IAA-CU-13-13-01
Survey of correction methods for faults and errors induced by cosmic radiation on operating system level in CubeSats

Kjell Arne Ødegaard*, Amund Skavhaug†

Abstract

The NTNU Test Satellite (NUTS) project aims to build and launch a double CubeSat within 2014. The Cubesat is increasingly used as a low-cost platform for research and even commercial use. By investigating what is possible to accomplished with low cost components in terms of reliability, two important things can be achieved. The cost of using space platforms for research can be reduced while maintaining the required level of reliability and at the same time the number of non-functional satellites (i.e. problematic space junk) can be reduced.

The focus of this work has been to explore the use of consumer components in space and the impact of the harsh vacuum and radiation environment on these components. Although some approximations regarding long term radiation effects are investigated, the main focus have been intermediate radiation faults in processors and memory and how to best mitigate them.

This paper provides an overview of significant problems with consumer electronic components in a space environment and investigates these problems. It also presents a brief summery of the effects that causes these problems and how they are most likely to affect the finished system. Data from NASA is used in order to find an approximation of the expected fault intensity in the satellite's components.

A number of possible low-cost solutions to the presented problems are evaluated according to the assumptions in the error model. Checkpointing provides the system with a restorable safe state in the event of a restart. Error detection and correction in storage in storage systems provides safe storage of critical runtime variables. In addition, two levels of watch dog timers, program memory integrity check and a periodic full reset with power cycle to clear remaining errors.

Our design shall be able to mask the system's operation from even frequent resets while the Error Detection And Correction system prevent the expected hundreds of errors per day from accumulating in memory, at some point resulting in failures. To emulate the space environment for repeatable test runs on specific parts of the system a fault injection rig using JTAG is being constructed.

*Master Student, Department of Engineering Cybernetics, Norwegian University of Technology and Science, Norway, kjellaod@stud.ntnu.no

†Associate Professor, Department of Engineering Cybernetics, Norway, Norwegian University of Technology and Science, amund.skavhaug@itk.ntnu.no

Introduction

The gateway to space for research institutions and commercial actors has traditionally been associated with a very high cost. Recent year's development of small, inexpensive satellites known as pico and nano satellites can change this by considerably lowering both the price point of satellite construction and launch. The small standardized form factor where many satellites can be packed together an piggybacked to other launches keeps the launch costs low, and the satellites themselves often use Consumer Of The Shelf (COTS) electronic components.

The use of consumer solutions allows for fast development with modern tools and enables the designers to get full advantage of the economy of scale with cheap and plentiful components and development tools. Due to the typically shorter lifespan of these satellites compared to traditional endeavors, it is possible to use newer, more innovative and even unproven components and design without running big financial risks. This is interesting as it allows for development and advancement in an otherwise conservative industry.

Due to both cost concerns, availability and simplicity, CubesSats commonly use Consumer Of The Shelf, or COTS, components. A number of different factors that will be detailed further on in this paper make these components vulnerable to the environment in space and in this paper we explore measures to alleviate the impact of these factors to the reliability, availability and survivability of the design.

Problem

One of the main challenges for space applications is the hard radiation operating conditions [3] [4]. To solve these challenges, radiation hardened electronic components and fault tolerant hardware has been used is space systems for a number of years to either ensure error free operation or to mask the errors from the system. In the context of a CubeSat, however, the challenges of high reliability system design shifts. It is still desirable with a high reliability system but the budgetary constraints are much stricter than for commercial or government designs.

In addition to being considerably more expensive, radiation hardened components traditionally lag behind their non-hardened equivalents in performance. This means that one gets a less capable system at a higher price point. At the same time it is not very important with a high availability design since the system does not control critical applications, but rather performs data collection tasks. This means that the on line redundant backup components can be omitted as long as we ensure that the system does not malfunction critically (i.e. fail). By using software methods, combined with redundancy for the most important subsystems, it is therefore possible to get higher performance, more flexibility and lower price, all without hot standby redundant backup components.

This paper aims to investigate low-cost methods to increase mission lifetime of small COTS based satellites. When considering the different reliability measurements it is important not to impact the performance of the rest of the system to an unacceptable degree. By mitigating the effects of Single Event Phenomena (SEP) occurrences in non-hardened components, it is possible to ensure higher up-time and increased mission lifespan. This further promotes safe operation and increases the likelihood of not loosing mission critical or payload

data. Student satellites do not have access to the established solutions because of budget constraints, and have to rely on smart solutions and COTS hardware to have a usable system in extreme conditions.

The problems with COTS components in space is numerous, as detailed by NASA [3]. In brief, radiation effects known as SEP can occurs when cosmic radiation strikes certain parts of the semiconductor material as outlined by figure 1. If the cosmic ray has enough energy this can alter the electrical charge and thereby alter the digital value in the component. This is known as a bit-flip and can corrupt saved data in addition to causing instability in the system. The expected number of errors estimated by NASA [3] is $10^{-5}\ errors/bit-day$. For NUTS this results in hundreds of errors per day in RAM and and up to a thousand errors per day in the flash data-banks. The expected radiation level is 1000-10000 rad(Si)/year with an orbital inclination between 20 and 85 degrees [3]. NUTS will have an even higher inclination and thus even higher radiation levels can be expected. With the total dose failure level of flash memories from 5-15 krad(Si) and microprocessors for 15-70 krad(Si) [3] of radiation, both can during the satellite's mission lifetime.

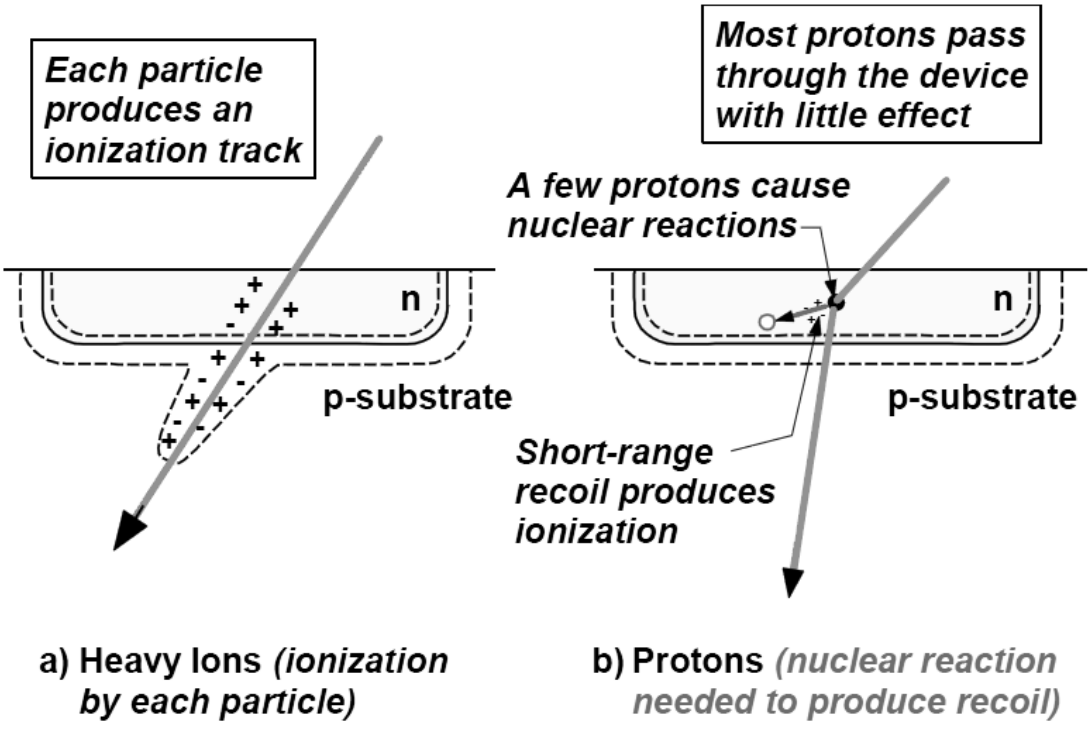

Figure 1: Mechanisms for Heavy Ion and Proton SEU effects [4]

Methods

The majority of the work in this paper has been to study the satellite and its systems and suggest solutions to the problems that are likely to be encountered.

Error Detection and Correction, EDAC

It would be advisable to have a system task that is in charge of secure storage of variables. Due to the random nature of the expected faults it would be difficult to determine if the data variables are safe to operate on. To guard against this it would be advisable to store the variables multiple times in order to be able to do a majority voting on the correctness or have an error correcting algorithm such as Bose-Chaudhuri-Hocquenghem (BHC) [2, p. 155] codes to correct the faults in run-time. This would add a large amount of complexity to the individual tasks running on the satellite. At the same time it is something that would be advisable for most of the tasks, due to the expected number of SEUs that will affect the system memory and accumulate over time.

A specialized secure storage system task could ease the programming burden for the rest of the designers by removing the sometimes complex algorithms from the smaller programs. A module based design is also favorable in programming because of the increased ease of maintaining and ensuring the correctness of smaller modules. This point applies even more for reliable systems [1, p. 202].

Checkpointing

Checkpointing is a proven solution in software system redundancy. It works by storing the system state that is necessary for continued execution and completion of the process, at specific points during process execution [1, p. 214]. This enables the system to roll back in the case of an error or initialize quickly and without losing critical data in the event of a system restart. An important feature to ensure is the ability to roll back multiple instances in the case of some unforeseen fault being present in the restored system.

Power cycling of faulty modules is already implemented in the backplane. The modules of the satellite must therefore tolerate a sudden reset without losing any significant amount of work. There are also some events such as antennae deployment and detumbling that should only be executed once and including these events in the saved system state will provide a simple measure of ensuring operational progress for the satellite.

Testing

The most realistic test would be to expose the system to a radiation environment and measure how the system holds up under real stress. While this might be desirable for the finished system it is not very useful when testing specific algorithms or sub modules in the system. The reason for this is that it is very difficult to control which module is to be tested and next to impossible to replicate the exact error conditions in order to determine the severity of the fault.

Another alternative is to simulate random error occurrence via JTAG in the software running on the board. This is somewhat better because the efficiency of the error correcting code can be determined directly since the number of inserted faults is known. Arguments against this testing regime are the lack of realistic errors. Latchup, for instance, is hard to simulate in software.

With these considerations in mind, the preferred testing method is to simulate errors with

JTAG injection of faults during runtime. This is the most economically viable option while at the same time allowing for repeatable test runs and focus on specific parts of the system.

Other methods

In addition to EDAC and Checkpoint, a number of other features are being implemented. Master-Slave functionality allows for a spare control computer in case the main crashes. The Watch Dog Timer (WDT) ensures that the system does not deadlock while interfacing with other system components. A periodic reset protects against any undetected failures that linger in the system. The ability to disable faulty modules safeguards against a malfunctioning module affecting the rest of the system. Finally, the ability to perform an integrity check on the program memory makes it possible to detect and possibly restore errors.

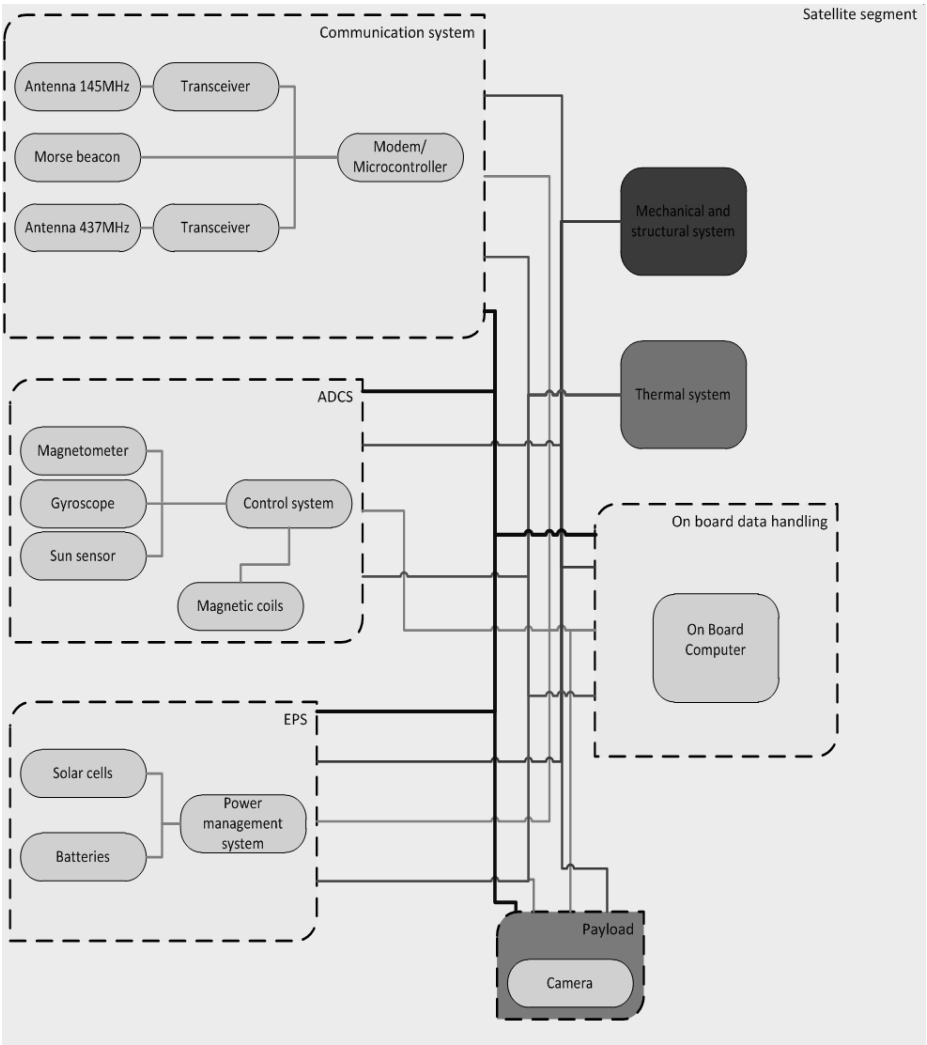

Figure 2: System overview [5]

Results and Observations

The preliminary results are encouraging. The chosen parameters for the $BCH(N, K)$ codes can detect and correct up to two randomly occurring errors per message block.

While the general implementations of $BCH(N, K)$ codes is costly and very inefficient [2, p. 161]. By taking advantage of specific aspects of the BCH codes and using look-up tables, the BCH codes can be optimized for embedded hardware. By choosing constant values for N and K the heavy computation of the generator polynomial coefficients can be done in advance. With the optimization techniques in place, the number of required cycles can be reduced by up to 51% [2, p. 164], but the precise computational cost may vary with the chosen embedded processor.

For testing purposes, optimized $BCH(67, 53)$ codes have been implemented. However, the final parameters have to be adjusted based on how much computational power that is available in the finished system.

Discussion and Conclusion

The main focus of the work has been to study the satellite to date and present possible solutions in order to implement a reliable overall system. The bulk of the work has been to understand the satellite's systems and reason which solutions that are most fitting to solve the expected problems.

The different problems expected to affect the satellite is presented together with suggested solutions. Further, it details how these problems can be solved with the constraint of using the already developed satellite systems.

Some of the strategy for low-cost components can be questioned. Why use a low-cost component when the launch cost is very high. But then again these components have the low complexity required to be included in student designs. Even with these low-cost solutions one should remember that a processor that costs $1 today is more powerful and uses much less power than the expensive processors from 25 years ago. When this is combined with the wide availability of inexpensive sensors the result is that it is possible to collect much more data at a lower cost than before.

The future work will focus on implementation of the solutions discussed in this paper. As more of the subsystems reaches completion they have to be integrated in the scheduling and fault recovery schemes of the satellite. The available processing power will be determined by the system's operating parameters and the load of other tasks such as the compressing algorithms. Because of this it is not advantageous to provide a finely tuned system at this point, but rather focus on a useful module for the future satellite. An exhaustive fault injection test to determine how the full system performs under stress is planned as the system reaches completion.

References

[1] Daniel P. Siewiorek and Robert S. Swarz, *Reliable Computer Systems, Design and Evaluation*. Digital Press, Burlington, 2nd Edition, 1992.

[2] Hazarathaiah Malepati, *Digital Media Processing, DSP Algorithms Using C*. Newnes, Burlington, 2010.

[3] *Space Radiation Effects on Electronic Components in Low-Earth Orbit*, PRACTICE NO. PD-ED-1258, JPL NASA, APRIL 1996.

[4] Sammy Kayali, *Space Radiation Effects on Microelectronics*, JPL NASA

[5] Emma Litzier, *System Overview - Space Segment*, http://nuts.cubesat.no/the-satellite, 2013.