Error Detection and Correction for Low-Cost Nano Satellites

Kjell Arne Ødegaard, R. Birkeland and A. Skavhaug
Department of Engineering Cybernetics, NTNU

NUTS NTNU TEST SATELLITE
A NORWEGIAN CUBESAT PROJECT

Introduction

The objective of this paper is to develop a low-cost, robust Error Detection And Correction (EDAC) solution for use in applications such as nano satellites, where price is a primary concern. Different methods have been evaluated, with the main result mitigation Single Event Effects causing bit flips in system memory utilizing Bose-Chaudhuri-Hocquenghem (BCH) codes. The general implementation is resource intensive and the algorithm has been optimized for the embedded platform. The codes have been implemented on a low-cost microcontroller with a real time operating system and faults have been injected during run-time to emulate a radiation environment. The performance impact and dynamic behavior of the algorithms is studied with third party trace analysis tools.

NTNU Test Satellite

The Norwegian University of Science and Technology (NTNU) Test Satellite (NUTS) project is aiming to launch a nanosatellite into Low Earth Orbit (LEO) by 2014. The satellite is a double CubeSat, measuring 10 x 10 x 20 cm and weighing less than 2.66 kg, which conforms to the CubeSat Standard. The satellite will carry an IR-camera for atmospheric observations as its main payload.
The NUTS project was started in September 2010, and is a part The Norwegian Student Satellite Program, ANSAT, run by NAROM (Norwegian Centre for Space-related Education). This program involves three educational establishments, namely the University of Oslo (UiO), Narvik University College (HiN) and NTNU.

Error Detection and Correction

In addition to being considerably more expensive, radiation hardened components traditionally lag behind their non-hardened equivalents in performance. At the same time it is usually not as important with a high availability design for CubeSats since the system does not control critical applications, but rather performs data collection tasks of an exploratory nature. It is important to receive correct data and to know if the satellite has suffered a malfunction, but the timeliness is of less importance. By using software methods, combined with some *simple* measures of redundancy for the most important subsystems, we aim to achieve several things: Higher performance and more flexibility. Lower price by not using hot standby redundant components, reducing the satellite complexity. The reason for this software approach is twofold: The most important systems in NUTS has already been realized in hardware, and a redesign at such a late stage is not desired by the project management. We want the proposed solutions to be relevant for projects that does not have the resources to build a conventional high reliability system.
Due to the random nature of the expected faults it is difficult to determine of the stored data is error free and safe to use. To increase the likelihood of error free operation we deploy the BHC error correcting algorithm. This algorithm is capable of detecting and correcting multiple errors at run time.
To protect against an undetected error, or if the EDAC codes are overwhelmed, we use a checkpointing system. Checkpointing is a proven solution in software system redundancy. It stores 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 [2].
The protected data will be stored in flash memory. With a planned flash size of 16 GB, we have a lot of options both regarding the number of stored data variables and the number of restoration points, and for most realistic configurations we are not limited by flash size.

Single Event Phenomena

Single Event Phenomena (SEP) is the collective term for the effects caused by cosmic radiation in electronic components. The radiation in orbit stems mainly from two sources, our Sun and Galactic Cosmic rays (GCR). In brief, these radiation effects can occur when cosmic radiation strikes certain parts of the semiconductor material,

Single Event Phenomena Cont.

as outlined by Figure 1. If the cosmic ray has enough energy it 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.
In spite of their small number, the heavy elements are very important due to their densely ionizing tracks. They are responsible for a large portion of the effects in detectors and microelectronics. Particle flux is also larger over the polar regions where "open" geomagnetic field lines allow easier access [5, p. 1-28]. This shows that heavy ions are more damaging to the components, as they can transfer more charge and often have higher energy than than the protons.

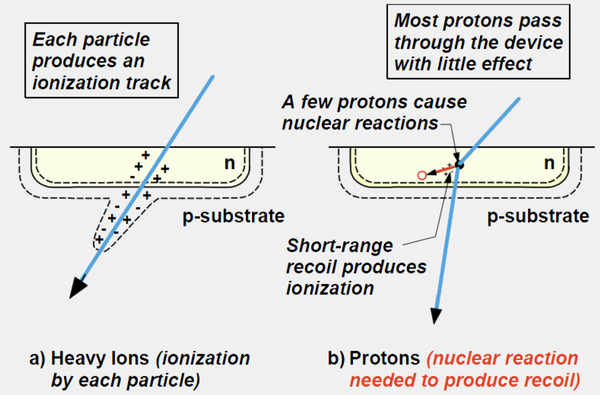
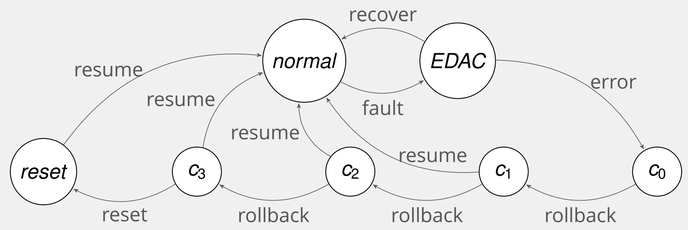


Figure : Mechanisms for Heavy Ion and Proton SEU effects [6]

System

The system starts in a normal state. If a fault is discovered EDAC attempts recovery. If the error can not be recovered the system enters the checkpoint stages. If the rollback is successful the system continues, if not it resets.



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 that 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 cpu boards [4]. This is somewhat better because the efficiency, e.g. of the error correcting code, can be determined directly since the number of inserted faults is known. Arguments against this testing regime is the lack of some of the 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.

Testing Cont.

This is the most economically viable option for us, while at the same time allowing for repeatable test runs and allowing us to focus on specific parts of the system.

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 are costly and very inefficient [3, p. 161], by taking advantage of specific aspects of the BCH codes and using look-up tables, the 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% [3, p. 164], but the precise computational cost may vary with the chosen embedded processor. The typical features that affect performance is word length (32, 16 or 8-bit) and if the processor uses soft float or have floats implemented in hardware. Other factors such as the ability to use specific processor capabilities like assembly instructions for digital signal processing executed in hardware can also increase performance.
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 after the payload and radio systems have been fully integrated and tested. This is due to energy budgeting, and the codes should not be run a significant amount of time since the available battery power is limited.

Conclusion

The main focus of this work has been to provide the satellite with measures to increase its dependability. 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 compression algorithms. Because of this it is not advantageous to provide a finely tuned system at this point, but rather to focus on a useful module for the satellite being built now. An exhaustive fault injection test to determine how the full system performs under stress is planned as the system reaches completion.

References

- Daniel P. Siewiorek and Robert S. Swarz, *Reliable Computer Systems, Design and Evaluation*. Digital Press, Burlington, 2nd Edition, 1992.
- Amund Skavhaug and Odd Pettersen *microFaultTolerant (μFT) - A system for achieving cost effective fault tolerance in microcontroller based equipment*, Real-Time Systems, 1995. Proceedings., Seventh Euromicro Workshop on, Conference Publication, 1995
- Hazarathaiah Malepati, *Digital Media Processing, DSP Algorithms Using C*. Newnes, Burlington, 2010.
- Olof Hannius and Johan Karlsson, *Impact of Soft Errors in a Jet Engine Controller*, Computer Safety, Reliability, and Security Lecture Notes in Computer Science, Volume 7612, Springer, 2012.
- Edward M. Silverman, *Space Environmental Effects on Spacecraft: LOE Materials Selection guide*, NASA Contractor Report 4661 Part 1, 1995.
- Sammy Kayali, *Space Radiation Effects on Microelectronics*, JPL NASA

NTNU
Norwegian University of Science and Technology

kjellaod@stud.ntnu.no