Acta Astronautica 95 (2014) 198–209

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Injecting faults to succeed. Verification of the boot software on-board solar orbiter's energetic particle detector[☆]

Antonio da Silva[*], Sebastián Sánchez, Óscar R. Polo, Pablo Parra

Space Research Group, University of Alcala, Alcalá de Henares, Madrid, Spain

A R T I C L E I N F O

Article history:
Received 25 January 2013
Received in revised form
6 September 2013
Accepted 6 November 2013
Available online 14 November 2013

Keywords:
Fault injection
Verification and validation
Bootstrap testing
Single Event Effect
Virtual platforms

A B S T R A C T

It is said that even the longest journey begins with the first step. This is also true for application software. When the power is switched on, computer systems execute an initial set of operations that usually perform memory tests and load the final runtime environment. This paper describes the Single Event Effects (SEEs) requirements verification of the boot software that will run in the Instrument Control Unit (ICU) of the Energetic Particle Detector (EPD) on-board Solar Orbiter. Since in the booting stage there are no software services at all, it is difficult to achieve a complete software verification on real hardware. To shortcut this issue the Space Research Group (SRG) of the University of Alcalá has developed a LEON2 Virtual Platform (Leon2ViP) based on SystemC with fault injection capabilities. This way it is possible to run the exact same target binary software as if were run on the physical system, but in a controlled and deterministic environment, thus allowing a stricter requirements verification. The use of Leon2ViP has meant a significant improvement, in both time and cost, in the development and verification processes of the ICU's boot software.

1. Introduction

Because of the tough robustness requirements in space software development, it is imperative to carry out verification tasks at a very early development stage to ensure that the implemented exception mechanisms work properly. This also helps to evaluate the possible risks, revealing how the system behaves in the presence of faults. These fault tolerance requirements call for full simulation environments, in which virtual platforms allow software to be developed and tested with a high degree of accuracy at a very early hardware development stage. This is fundamental in evaluating the fault detection and recovery mechanisms implemented in the software design. The verification of software fault tolerance mechanisms implemented in critical systems to recover the system from exceptional situations can be difficult. This is because such situations must be systematically and artificially brought about during the verification phase. Fault tolerance mechanisms are often verified by means of experimental techniques such as fault injection, which comprises a variety of techniques for introducing faults into a system and modifying its behavior to facilitate the reproduction of hidden or unforeseen problems in order to:

- verify exception handling and recovery mechanisms: in classic software testing methodologies, particular exception or error handling procedures, if any, are rarely triggered and even less tested [1];
- provide an experimental assessment of the risks [2]: the system's behavior in the presence of faults can be used as a way to quantify potential risks of the system and to allow the prediction of worst-case scenarios.

Virtual platforms are executable models of complete systems that provide software developers with working

[☆] This work has been supported by the MINECO under the Grant AYA2011-29727-C02-02.

[*] Corresponding author. Tel.: +34 9133 655 11.
E-mail addresses: adasilva@srg.aut.uah.es, adasilva@diatel.upm.es (A. da Silva).

0094-5765/$ - see front matter © 2013 IAA. Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.actaastro.2013.11.004

frameworks a long time before the real hardware is available. Virtual platforms enable the concurrent development of System-on-Chip (SoC) hardware and software, significantly shortening their integration times. For embedded software development and verification some of the advantages of using virtual platforms are to:

- run the same target software binary as if on the physical system, but in a controlled and deterministic environment;
- reduce the dependencies of the software and system tasks on hardware availability;
- provide debugging and fault injection capabilities which are unattainable otherwise;
- offer the capability to connect physical devices through standard communication interfaces, thus allowing a virtual hardware-in-the-loop testing approach.

Verification is a major process in the development of aircraft and spacecraft software. Its purpose is to detect and report errors that may have been introduced during the development process. Verification is typically a combination of reviews, analyses and tests with the aim of assessing, with a high degree of confidence, that errors that could lead to unacceptable failure conditions have been removed.

Embedded software testing has been mainly carried out on dedicated hardware resources. The limitations incurred while using these dedicated resources have been known for a while: cost, availability, the use of intrusive testing techniques and the lack of debugging capabilities, observability and controllability. These limitations are noticeably more acute when dealing with the testing of fault tolerance-related properties. These tests require specific hardware and software setups, which are not always technically achievable, nor practically affordable in a non-intrusive manner. The use of a fully virtual platform is an alternative approach able to yield effective solutions to these limitations. From an embedded software perspective, the use of virtual platforms allows the development and verification processes to be started earlier in the design flow so as to detect and correct errors that would otherwise propagate to the final implementation stages. Moreover, it is easier to access and modify the internal state of the virtual prototypes, so that a comprehensive fault injection campaign and fault tolerance assessment can be carried out.

The remainder of the paper is organized as follows: relevant related works are detailed in the next subsection. Section 2 describes the mission's characteristics. Section 3 the embedded software development and testing challenges. Section 4 describes the adopted solution based on the use of an ad-hoc virtual platform development. Section 5 describes the experimental setup used to verify several robustness software requirements along with the results. Section 6 contains the conclusions.

1.1. *Related work*

The use of virtual platforms gives developers far more visibility and control over system design by the very nature of its virtuality. Any state is within reach and any condition can be triggered. Therefore, virtual platforms have become widely used in design space exploration and early software development in avionics and space software environments, before the hardware becomes available [4,5]. There are several approaches, ranging from symbolic execution to binary compatible instruction-set simulators. Current research focuses on experimental techniques and tools that allow software robustness verification through fault injection.

Several works deal with model-based verification and symbolic execution. Symbolic execution has been used for a wide variety of software testing and maintenance purposes. The main idea behind these techniques is to interpret the program by simulating its execution with symbolic values rather than executing it on real hardware. Although, many symbolic execution techniques assume that the hardware does not experience errors during the execution of the program, the work [3] describes a framework which introduces a formal model to represent programs expressed in a generic assembly language with the ability to bring about faults that could potentially cause program failures.

Unlike symbolic methods, experimental measurement is an attractive option for evaluating an existing system or prototype closely in the field, because it allows the real execution of the system to be observed to obtain measurements (hopefully highly accurate) in its working environment. In this regard, fault injection is an attractive option in verifying the fault tolerance requirements present in critical systems. Using SystemC Transaction Level Models (TLM) it is possible to model mixed hardware/software models in order to simulate the system in the presence of faults. For example, works [6,7] use this methodology for the design and testing of fault tolerant systems implemented in an FPGA platform with different types of diagnostic techniques. The experimental results show the fault coverage and how Single Event Upset (SEU) occurrences cause faulty behaviors in the implemented systems. [8–10] use the same approach to verify the software of networked embedded systems long before the final hardware is available.

The work [11] describes a previous virtual platform from the Space Research Group of the University of Alcalá. The framework integrates a SPARC Instruction Set Simulator (ISS) together with other platform components by means of TLM 2.0 interfaces. It enables early software development and verification of platforms based on LEON3, a 32-bit SPARC CPU-based system used by the European Space Agency. SimSoC [12] is a similar environment for ARM processors.

The framework presented in this work is a specific LEON2 virtual platform with fault injection capabilities for the Instrument Control Unit (ICU) of the Energetic Particle Detector (EPD) on board Solar Orbiter, along with the first test results of the boot software. As far as we know, the platform provides fault injection capabilities that had never been provided so far by any other LEON2 based development platform.

2. Solar Orbiter mission

The Solar Orbiter [13] is a planned Sun-observation satellite, under development by the European Space Agency (ESA),