Non-intrusive Software-Implemented Fault Injection in Embedded Systems

Pedro Yuste, Juan Carlos Ruiz, Lenin Lemus, and Pedro Gil

Fault Tolerant Systems Group (GSTF)
Technical University of Valencia
46022 Valencia, Spain
{pyuste, jcruizg, lemus, pgil}@disca.upv.es

Abstract. Critical embedded systems, like those used in avionics or automotive, have strong dependability requirements and most of them must face with fault tolerance. One of the methods typically used to validate fault tolerance mechanisms is fault injection. The idea is to study the behavior of the system in presence of faults in order to determine whether the system behaves properly or not. Software-implemented fault injection (SWIFI) techniques enable fault injection to be performed by software. Although interesting, major drawbacks of existing SWIFI techniques are the temporal and the spatial overheads they induced in the systems under study. The reduction of these overheads is thus crucial, in order to be confident on the results and conclusions of a SWIFI experiment. This paper focuses on this problem. It proposes a new non-intrusive SWIFI technique for injecting faults in embedded (system-on-chip) applications. The technique exploits the features of a standard debugging interface for embedded systems, called Nexus, in order to inject faults without temporal overhead. Then, Nexus features are also exploited in order to observe, without spatial intrusion, the behavior of the target system in presence of the injected faults. In other words, the embedded system under study can be controlled (for injecting faults) and observed (for tracing its behavior) without customizing its original structure or altering its normal execution. Since based on Nexus, the technique has also the benefit of being applicable to any Nexus-compliant system. In order to illustrate the potentials of the approach, we use an automotive embedded control unit application as a case study. Some preliminary results obtained from the experiments performed are also discussed.

1 Introduction

Nowadays, embedded systems are widely used in a large spectrum of application domains, varying from automotive systems to avionics. In these domains, a failure of

---

1 This work has been supported by the European IST project DBench (Dependability Benchmarking, IST-2000-25425).
the system could have disastrous consequences from both economical and human viewpoints. For this reason, these applications need a high degree of dependability.

Fault injection (FI) is nowadays a well-known technique for characterizing the behavior of (critical) systems in presence of faults [1]. Basically, the goal is to study whether or not computing systems are able to tolerate (or to handle in a safety manner) the occurrence of a fault while providing their service. As stated in [2], these faults can be (i) injected using specialized hardware tools (hardware-implemented fault injection or HWIFI); (ii) emulated from programs (software-implemented fault Injection or SWIFI) or (iii) simulated using models (simulation-based fault injection). In this paper, our interest focuses on SWIFI techniques.

From a high level viewpoint, the term SWIFI groups a set of techniques used to inject faults on computer systems using software. One of the goals of these techniques is to emulate, at a logical level, the consequences of hardware faults in a system [3]. As stated in [4], SWIFI techniques also enables the emulation of software (design) faults. However, this second issue goes beyond the scope of this paper.

Most of the existing SWIFI techniques suffer from two major problems. The first regards their portability, as they usually require deep modifications in the system in which they are applied [5]. The second is the introduction of a temporal overhead that slows down the execution of the application [6]. This overhead may prevent certain of these SWIFI techniques from being applicable on systems with real-time constraints. Observability is an additional problem to consider when applying SWIFI techniques on embedded (System-on-Chip or SoC) systems. In this case, complexity and scale of integration makes difficult the task of observing and tracing the behavior of the system in presence of faults.

This paper focuses on the definition of a new SWIFI technique that handles the above problems. Exploiting the standard features provided by the embedded processor debugging interface Nexus [7], our technique is able to inject faults in SoC applications on-the-fly, while avoiding any temporal intrusion in their normal execution. Then, the approach also provides solutions for observe and study the behavior of the target application in presence of the injected faults. This observation is performed without any spatial intrusion, i.e., without customizing the considered application.

Section 2 provides a high-level description of the main features of Nexus that are useful for injecting faults in embedded (SoC) applications. Taking into account these features, Section 3 presents the SWIFI technique that we have defined. Section 4 illustrates this technique using an embedded diesel engine control unit (ECU) as a case study. Section 5 comments the relation of this work with others of the domain. Section 6 provides the conclusions and depicts the future of this research.

2 Nexus-Based Fault Injection

Nexus is a standard for embedded system debugging [7]. It has been defined by the Nexus 5001 Forum™, a consortium including important processor manufacturers (such as Hitachi, Infineon, Motorola, National Semiconductor and ST). The Nexus