Evaluation of the MARS Architecture by means of Three Physical Fault Injection Techniques

Jean Arlat[1] Yves Crouzet[1] Johan Karlsson[2] Peter Folkesson[2] Günther Leber[3]

[1] LAAS-CNRS, Toulouse, France
[2] Chalmers University of Technology, Gothenburg, Sweden
[3] Technical University of Vienna, Austria

Extended Abstract

Introduction

Among the large number of experiments reported concerning physical fault injection, all used widely different techniques and/or were applied to distinct target systems. This significantly hampers the possibility to identify the difficulties/benefits associated to each fault injection technique and to analyze the results obtained.

The conducted study (see [1] for more details) relies on two major objectives. The first one is to get a better understanding of the impact and features of the three physical fault injection techniques that are considered and in which the sites have gained expertise in developing and applying dedicated experimental tools or in using standard support environments, respectively: heavy-ion radiation, pin-level injection and electromagnetic interferences (EMI). The distributed fault-tolerant system architecture MARS [2] developed by TU-Vienna is being used as the target system to carry out these experiments. Thus, the other driving objective is to evaluate the coverage of the built-in fault-tolerance features of the MARS system. A distributed testbed architecture featuring five MARS nodes and a common test scenario have been implemented at all three sites to perform a coherent set of experiments.

The fault injection techniques

Heavy-ion radiation: A Californium-252 source can be used to inject *single event upsets*, i.e., bit flips at internal locations in integrated circuits. The heavy-ion method has been used to evaluate several hardware- and software-implemented error detection mechanisms for the MC6809E microprocessor [3]. The irradiation of the target circuit must be performed in a vacuum as heavy-ions are attenuated by air molecules and other materials. Consequently, the packaging material that cover the target chip must also be removed. In these experiments, a miniature vacuum chamber containing the target circuit and the Cf-252 source was used. A comprehensive description of the heavy-ion fault injection technique and of the supporting tools is given in [4].

A major feature of the heavy-ion fault injection technique is that faults can be injected into VLSI circuits at locations which are impossible to reach by other techniques such as pin-level and software-implemented fault injection. The faults are also reasonably well spread within a circuit, as there are many sensitive memory elements in most VLSI circuits. Thereby, the injected faults generate a variety of error patterns which allows a thorough testing of fault handling mechanisms.

Pin-level fault injection: The injection of faults directly on the pins of the ICs of a prototype was until now the most widely applied physical fault injection technique. It has been used for (i) the evaluation of the coverage of specific mechanisms (in particular for error detection by means of signature analysis [5], and (ii) the validation of fault-tolerant distributed systems (e.g., [6, 7]). Flexible tools supporting some general features have been developed (e.g., the test facility used on the FTMP [8], MESSALINE at LAAS-CNRS [9] or RIFLE [10] at the University of Coimbra).

The tool MESSALINE that was used in these experiments is capable of adapting easily to various target systems and to different measures [9]. The forcing technique was used for the fault injection experiments carried out on the MARS system. The injected faults were temporary stuck-at (0 or 1) faults. As the pin-forcing technique is being used, it can be confidently considered that all pins of the ICs connected to an injected pin are tested as well. Accordingly, in the set of experiments conducted to date, to simplify the accessibility to the pins of the microprocessors of the application and communication units, the target ICs were mainly buffer ICs connected to them.

EMI: An important class of computer failures are those caused by electro-magnetic interference (EMI). Such disturbances are common, for example, in motor cars, trains and industrial plants. Consequently, we decided to investigate the use of EMI for the evaluation of the MARS system. The fault injector used in the experiments generates bursts conforming to IEC 801-4 standard (CEI/IEC), i.e., the duration of the bursts is 15 ms, the period is 300 ms, the frequency is 1.25, 2.5, 5, or 10 kHz, and the voltage may be selected from 225 V to 4400 V. These bursts are similar to those, which arise when switching inductive loads with relays or mechanical circuit-breakers.

The faults were injected into the target system, which consisted of a single computer board, in two different ways. In the first way, the computer board was placed between two conducting plates connected to the burst generator. The second way was to use a special probe that could expose a smaller part of the board to the disturbances. In order to direct the faults to specific parts of the computer board, such as the CPU buses, small pieces of wire functioning as antennas were connected to the pins of specific ICs. The antennas were used with both the probe and the plates. In addition, experiments were also conducted using the probe without the antennas.

The Common Experimental Set-up

Figure 1 describes the common set-up used to perform the fault injection campaigns at all sites. The figure identifies the interactions with the fault injector devices. In the case of heavy-ion (HI), the target circuit is inserted in a miniature vacuum chamber containing a Cf-252 source; radiation can be controlled by an electrically manoeuvred shutter [4]. For pin-level injection, the pin-forcing (PF) technique is used; thus, the injection probe is directly connected to the pins of the target IC [9]. For EMI, both the technique using the two plates and the probe was used for the injections. To obtain similar experimental conditions with the three techniques, faults were only injected inside, on the pins, or in the vicinity of either the application CPU or the communication CPU of the tested node.

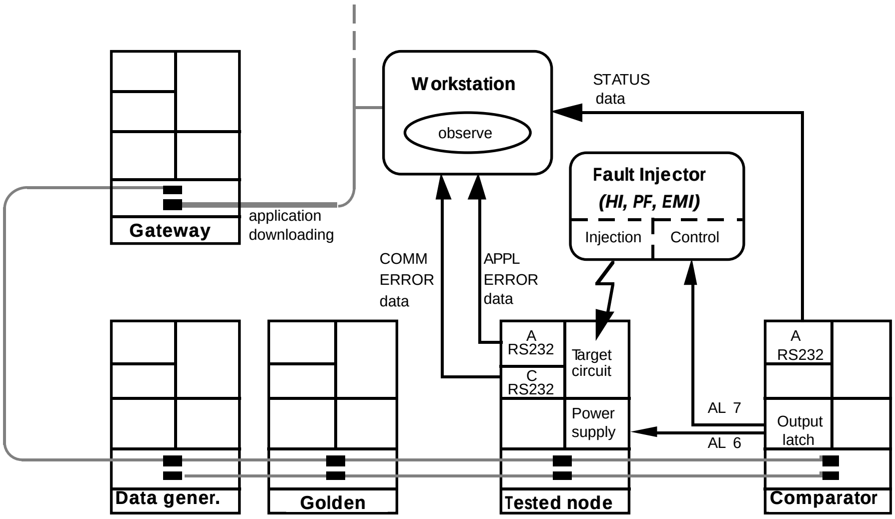

Figure 1: Detailed set-up architecture

Results

The results that will be presented concern the observed efficiency of the error detection mechanisms (EDMs) implemented into one MARS node when submitted to the three fault injection techniques. Three levels of EDMs are implemented : (i) hardware EDMs, (ii) system software EDMs implemented in the operating system [11] and support software (i.e., the Modula/R compiler), and (iii) application-level (signature-check and double execution) EDMs at the highest level. The error detection mechanisms provide the fail-silence property of the MARS nodes.

The results show that the hardware EDMs detect most of the errors, followed by the system software. Although the application-level EDMs detected the smallest amount of errors for all techniques, but when these were disabled, the fail-silence coverage was reduced (particularly for heavy-ion radiation) which shows the necessity of using these mechanisms as well.

The results obtained also helped identify the similarities and differences in the error sets generated by the three techniques. The error sets were observed indirectly via the distribution of error detections among the various EDMs. The observations reveal fairly large differences in the distribution of the error detections among the various EDMs for the three fault injection techniques. This suggests that the techniques are rather complementary, i.e. they generate to large extent different types of errors. The pin-forcing technique exercised the hardware EDMs located outside the CPU more effectively than the other techniques, while the heavy-ion and EMI techniques appear to be more suitable for exercising software and application level EDMs. Heavy-ion radiation showed the largest spread in the detections among the EDMs.

References

[1] J. Karlsson, P. Folkesson, J. Arlat, Y. Crouzet, G. Leber and J. Reisinger, “Application of Three Physical Fault Injection Techniques to the Experimental Assessment of the MARS Architecture”, in *Proc. DCCA-5*, Urbana-Champaign, IL, USA, September 1995.

[2] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft and R. Zainlinger, “Distributed Fault-Tolerant Real-Time Systems: The MARS Approach”, *IEEE Micro*, vol. 9, pp. 25-40, February 1989.

[3] U. Gunneflo, J. Karlsson and J. Torin, “Evaluation of Error Detection Schemes Using Fault Injection by Heavy-ion Radiation”, in *Proc. FTCS-19*, Chicago, IL, USA, 1989, pp. 340-347 (IEEE Computer Society Press).

[4] J. Karlsson, P. Lidén, P. Dahlgren and R. Johansson, “Using Heavy-Ion Radiation to Validate Fault-Handling Mechanisms”, *IEEE Micro*, vol. 14, pp. 8-23, February 1994.

[5] M. A. Schuette, J. P. Shen, D. P. Siewiorek and Y. X. Zhu, “Experimental Evaluation of Two Concurrent Error Detection Schemes”, in *Proc. FTCS-16*, Vienna, Austria, 1986, pp. 138-143 (IEEE Computer Society Press).

[6] A. Damm, “The Effectiveness of Software Error-Detection Mechanisms in Real-Time Operating Systems”, in *Proc. FTCS-16*, Vienna, Austria, 1986, pp. 171-176 (IEEE Computer Society).

[7] C. J. Walter, “Evaluation and Design of an Ultra-Reliable Distributed Architecture for Fault Tolerance”, *IEEE Transactions on Reliability*, vol. 39, pp. 492-499, October 1990.

[8] J. H. Lala, “Fault Detection, Isolation, and Reconfiguration in FTMP: Methods and Experimental Results”, in *Fifth AIAA/IEEE Digital Avionics Sys. Conf.*, 1983, pp. 21.3.1-21.3.9.

[9] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, E. Martins and D. Powell, “Fault Injection for Dependability Validation — A Methodology and Some Applications”, *IEEE Transactions on Software Engineering*, vol. 16, pp. 166-182, February 1990.

[10] H. Madeira, M. Rela, F. Moreira and J. G. Silva, “A General Purpose Pin-level Fault Injector”, in *Proc. EDCC-1*, Berlin, Germany, 1994, pp. 199-216 (Springer-Verlag).

[11] H. Kopetz *et al.*, *The Distributed Fault-Tolerant Real-Time Operating System MARS*, IEEE Operating Systems Newsletter, vol. 6, 1992.