Software Detection of Hardware Plateform Failures due to Electromagnetic Fields

Gilles Motet, Louise Tysk

Software Detection of Hardware Plateform Failures due to Electromagnetic Fields

MOTET Gilles[1], TYSK Louise[2]
1. LESIA / INSA, 135 avenue de Rangueil, 31077 Toulouse cedex 4, France
Gilles.Motet@insa-tlse.fr
2. Linköping Universitet, SE-58183 Linköping, Sweden
louise@student.liu.se

Abstract - The reduction of embedded electronic systems susceptibility to electromagnetic fields is often obtained by hardware devices whose cost must be added to each product. On the contrary, we consider the use of conventional hardware systems (COTS) handling their misfunctioning by software techniques. This paper introduces software detection mechanisms whose development cost is low as the mechanisms are generic, that is, independent on the specific functionalities of the software applications processed by the electronic systems.

1. REQUIREMENTS

Electronic systems are embedded in numerous products to provide new functionalities and also to improve the performance of the implementation done using other technologies (mechanic, hydrolic, etc.). The electronic systems embedded in cars provide numerous examples of such situation. The improved performance concerns the reaction durations as well as the reliability. However, the hardware technology sometimes decreases the system safety as this technology is sensitive to the stresses of the environment and, in particular, to electromagnetic fields.
Unfortunately, the study of the evolution of the electronic systems shows that more and more responsibilities are transfered from humans to these systems: ABS and EBV are two examples of automotive systems. These increasing responsibilities imply that the designer must guarantee the safety of the electronic system, to assur its quality (for instance, automotive or space systems), or to obtain a certification (for instance, avionic, medical, nuclear, transportation systems) [6].
The safety of these systems may be obtained by prevention means or tolerance techniques [7]. Thus, to reduce the susceptibility of hardware systems to the aggressions of electromagnetic fields, prevention techniques such as circuit design rules are used or protection barriers are added. To tolerate the effects of these agressions, redundant mechanisms (replication) or detection and handling features are implemented [2]. However, the overcost of these means is not neglictable and increases the cost of each electronic product.

2. GENERIC SOFTWARE HANDLING

On the contrary, no extra cost exists when a software application is duplicated. The use of software mechanisms to obtain robust electronic systems seems to be a solution which has to be investigated. However, two remarks must be signaled immediately:

1. The adding of software elements induces an increasing of the cost of the hardware platform components: additional memory and higher processor speed. To be acceptable, the proposed solution must only add a small quantity of software code. In particular, redundant techniques such as N-Version programming [5] cannot be used. Moreover, these techniques may be unefficent due to the common modes of the hardware platform failures. Thus, the software mechanisms have to detect the hardware misfunctioning and to handle it.
2. The design of the added software leads to expenditure. This amount is distributed on the price of each produced electronic system. To be acceptable, the design cost must be as low as possible. Consequently, the development of the mechanisms used to detect and to handle the hardware failures must be simple.

To take these requirements into account, we propose to add to software applications executed by the hardware platforms, software mechanisms which are generic, that is, not dependent on each specific application. Due to this property, the detection and handling can be implemented automatically. So their development cost is null.
At the moment, our research work only concerns the detection means. The recovery of the errors will be study later.

3. PRINCIPLES

The phenomena due to the electromagnetic fields being transient, on-line detection techniques have to be proposed. Several points of view must be considered to select the suitable solution.

Detection associated with the microprocessor instructions

Research activities were done to propose mechanisms to detect and to correct damaged data before their processing: *Software-Implemented Error Detection and Correction* techniques or *Error Detection by Duplicated Instructions* (cf., for instance, Stanford Argos project [9]). Our project aims at handling hardware failures occurring due to electronic fields, detecting their effects on the data and control flows of the software application executed by the hardware platform. The instructions offered by the microprocessors may be considered as services called by the executable program. These services act on the program data as well as on the program control flow (jump instruction, for example). The implementation of additional instructions checking that each application instruction was correctly executed cannot be envisaged, being untractable.

The techniques such as *Enhanced Control Flow Checking using Assertions* propose to include instructions checking assertions for each block of instructions [1]. Other proposals aim at detecting the Instruction Pointer corruption in particular due to electromagnetic interference [8] or to use techniques based on signatures [9].

These solutions have a main drawback: they are specific to the used microprocessor. In fact, the misfunctioning of the processor instructions due to the agressions causes a wrong behaviour of a program. Therfore, the detection means can be studied on the source program instead of the executable program.

Detection associated with the software application

A source program is written with a given programming language. A first approach consists in adding statements checking that the software application functionalities behave as expected. It is implemented using defensive programming techniques based on pre-conditions, post-conditions and invariants on variables or subprogram calls [4].

However, this method needs again a specific design of the program instrumentation for each specific application. This solution is therefore expensive. Moreover, the actual efficiency of the detection depends on the competence of the engineers who instrumented the program. This efficiency is assessed with difficulties, as we do not have precise models of the errors due to electromagnetic fields on program behaviors.

Detection associated with the programming language

Our proposal does not depend on the specific functionalities of the software applications as it is based on the programming language characteristics. So, our results can be reused for any application implemented with this programming language.

A programming language offers features (declaration of variables, assignment statements, etc.) which can be considered as services of an “abstract machine” defined by the semantics of the language. For instance, the statement “J=I+1” uses a machine which can evaluate the arithmetic expressions and can assign the result to a variable. This abstract machine is implemented by hardware and software components. For instance, “printf(...)” assumes that a service exists to display a text; this service is partially implemented by an operating system.

This “abstract machine” implements the *operational semantics* of the programming language which also provides a *checking semantics* often implicitly expressed. For example, let T be an array. T(I) expresses an access to the I-th value of the array (operational semantics). However, a pre-condition is associated with the execution of this service: the value of I must be in the range specified when the array was declared (between 0 and N-1 for an array of size N declared in C language). The violation of this property detects an error. Let us remark that this property is not specific to a particular application. We don't know what application concept is implemented by this array. We are only sure that an error occurs when this property is false. Thus, the checking of the property “the value of an index used to access to an array must be in the range specified at the array declaration time” is therefore a generic error detection means.

Our studies lead to propose a checking semantics of the C language. We previously provided a first example concerning the arrays. Hereafter, we give two other examples: the first property deals with the control flow whereas the second is associated with the data flow.

1. “The execution of a block of statements must start at the first statement and be completed at the last one”. The negation of this property detects an abnormal branching due to a misfunctioning of the hardware platform, if the “go to” statement cannot be used in the program.
2. “A variable must be assigned before its use”. The negation of this assertion is an error which may be due to a design fault detectable before execution [2] or due

to an hardware misfunctioning which must be detected at run-time.

4. PROPOSAL ANALYSIS AND FUTURE WORK

Several issues must be considered now to conclude the project.

At first, the started experimentations must completed to assure that the detection of the hardware failures due to electromagnetic fields, by these mechanisms, is effective.

Then, it is necessary to assess the efficiency of each checked property. Indeed, to preserve an acceptable size of the program, it will not be possible to instrument all the C programming statements of a program.

These experimentations must also study the intrinsic safety of the proposed solution. The detection statements can be executed during the hardware misfunctioning. We must examine the effects. The risk is reduced when the number of checking statements is low. However, the decreasing of this number may reduce the detection performance and so the system safety. Therefore, efficiency of the properties must be studied to include the actually useful checks, and no more.

The experimental studies whose first results will be exposed during the conference, will also allow the understanding of the effects of the electromagnetic aggressions on the program behaviour to be improved. An error model associated with the electromagnetic agressions will be studied for a given microprocessor. These models will concern the microprocessor instructions level and then the programming language level. The obtaining of such error models presents three main interests:

1. It allows the detection mechanisms to be chosen in an efficient way and this selection to be justified. Efficiency is required by economical constraints; the justification is essential if the system safety has to be guaranteed by certification organisms [6].
2. Thanks to error models, the efficiency of the detection mechanisms can be studied without any hardware platform nor aggressions means, using fault simulation tools such as Ferrari [3].
3. The knowledge of the sensitivity of the microprocessor instructions to electromagnetic fields allows a compiler generating executable code using the robust instructions to be chosen

To conclude, we must highlight the fact that the proposed solution does not defeat the purpose of the hardware techniques. For instance, our solution does not allow electromagnetic aggressions provoking the processor shut down to be handled. Indeed, in such a situation, the processor cannot execute the software detection mechanisms.

Our proposal is complementary: it increases the safety of high-critical systems adding checks at software application level; moreover, it provides a low-cost solution for conventional applications.

5. REFERENCES

[1] Alkhalifa Z., Nair V.S., “Design and Implementation of a Portable Control-Flow Checking Technique”, Proceedings of the High-Assurance Systems Engineering Workshop, IEEE Publishers, pp 120-123, 1997.

[2] Geffroy J.-C., Motet G., “Design of Dependable Computing Systems”, Kluwer Academic Publishers, 2002.

[3] Kanawati G., Kanawati N., Abraham J., “Ferrari: A Flexible Software-Based Fault and Error Injection System”, IEEE Transactions on Computers, vol. 44, n° 2, IEEE Publishers, pp 248-260, 1995.

[4] Luckham D., “Programming with Specification”, Springer-Verlag, 1990.

[5] Motet G., Marpinard A., Geffroy J.-C., “Design of Dependable Ada Software”, Prentice Hall, 1995.

[6] Motet G., “Certification of Real-Time Systems: Consequences on Modeling Tools and Modeling Processes”, Lectures Notes in Computer Sciences, Springer, 2002 (to be published in november).

[7] Motet G., Geffroy J.-C., “Dependable Computing: An Overview”, Theoretical Computer Sciences, vol. 291, n° 2, Elsevier Publishers, 2003 (to be published).

[8] Ong R. H., Pont M.J., “Empirical Comparison of Software Error Detection and Correction Techniques for Embedded Systems”, International Symposium on Hardware / Software Codesign, ACM Publishers, pp 230-235, 2001.

[9] Shirvani P. P., Oh N., McCluskey E. J.,, Wood D. L., Lovelette M. N., Wood K. S., “Software-Implemented Hardware Fault Tolerance Experiment COTS in Space”, International Conference on Dependable Systems and Network (FTCS-30 and DCCA-8), IEEE Publishers, pp B56-7, 2000.