# Towards Certification-aware Fault Injection Methodologies Using Virtual Prototypes

Jaime Espinosa[†], David de Andrés[†], Juan-Carlos Ruiz[†], Carles Hernandez[‡], Jaume Abella[‡]
[†]Universitat Politècnica de València
[‡]Barcelona Supercomputing Center (BSC-CNS)

***Abstract*—Safety-critical applications are required today to meet more and more stringent standards than ever. In the need of reducing the costs associated with the certification step, early robustness evaluation can provide valuable information, as long as it is fast and accurate enough. Microarchitectural simulators have been employed for testing reliability properties in several domains in the past, but their use in the process of robustness verification of safety critical systems has not been validated yet, as opposed to RTL or gate-level simulations. In the present work, we propose a methodology to improve the accuracy of fault-injection results when targeting robustness verification, by using microarchitectural simulators and virtual prototypes for an early estimation of deviations with respect to the certification standards.**

## I. Introduction

To cope with the increasing functionality demands coming from the safety-critical systems industry processor designs offering more computation power are required. However, the growing complexity together with the requirement for adhering to certification standards causes processor designs targeting safety critical markets rarely achieve reduced time to market. In that context, new verification and test methodologies and tools have to be devised for a quick and cost-effective way to check whether robustness properties are achieved throughout the whole design flow on top of complex processor designs.

Simulation-based fault injection is regarded as a suitable methodology for the robustness verification process, as quick and cheap corrections on misbehavior can be made. Unfortunately, fault injection experiments are often carried out at gate level, and so the testing process becomes excruciatingly slow. This problem is alleviated when fault-injection is applied at a higher level of abstraction like the Register Transfer Level (RTL). Verification at the RTL level reduces the burden but still results can get too slow for repeated use in the context of complex designs.

Software-based fault injection techniques and in particular fault-injection experiments using virtual prototypes (or microarchitectural simulators) are a potential candidate to reduce the costs associated with the robustness verification process [16][2]. The main benefits of these approaches with respect to RTL-based fault injections are the simulation speed and the possibility to anticipate deviations w.r.t. the safety requirements before having an actual implementation of the system. Note that the effort required to have a virtual prototype [22][14] of the system is much lower than the one required to have an RTL processor description[1]. However, for these approaches to be employed in the robustness verification process of safety critical processors their accuracy must be proven. It is important to mention that fault-injections using microarchitectural simulators are typically restricted to the register file [10][21] and the different memory structures [20][1]. The reason is that the majority of the potential injection nodes that are present at more detailed abstraction levels like RTL or gate-level are missing at this level of abstraction.

[1]In fact virtual prototypes are already needed to enable software developers to test their software before the actual processor is shipped.

In this context, estimating accurate failure rate metrics using virtual prototypes is challenging. On one hand, implementation details of virtual prototypes provided with commercial tools [22] are typically protected. On the other hand, even if implementation details were available, the existing information in virtual prototypes implementation is reduced in comparison to other detailed implementations like RTL. Taking these facts into account, obtaining failure metrics as the ones required by safety critical systems certification standards [7][19] using virtual prototypes is a complex task. Our hypothesis is that for virtual prototypes to be considered for the robustness verification process, results obtained at this level must be correlated with the ones obtained at lower levels of abstraction like the RTL or gate-level as fault-injection at these lower abstraction levels has already been shown to provide accurate enough results [18].

In this paper we propose a methodology that increases the confidence on fault injection experiments into virtual prototypes by using some key information extracted from more detailed levels of abstraction (say for instance RTL or Gate Level). In particular, we are interested in knowing the likelihood that a fault in any [22] possible processor net, gate, or flip/flop, propagates to the register file, system registers or the different memory structures existing in the CPU virtual prototype. Note that this architectural information represents the minimum implementation details that need to be available in any virtual prototype and visible to the user.

The rest of the paper is organized as follows. Section II reviews the state-of-the art in fault injection related to the certification of safety-critical systems. Section III presents the problem in detail and the proposed methodology towards enabling the use of Virtual Prototypes for injection. Section V shows the preliminary evaluation data and finally, in Section VI some conclusions are drawn.

## II. Related Work

Several techniques exist to perform RTL level fault injection. A widely-used method is the injection in the HDL through simulator commands [12], which works well for most of the fault models described in the literature. In fact, some unconventional fault models such as those involving several injection points –short-circuit, multi-bit injection– can be applied if the more intrusive technique of saboteurs is used [3], but an instrumentation of the model –and the consequent decrease in simulation speed– is entailed.

Fault models representativeness has been validated for logic/RTL levels [8]. On the contrary, for higher abstraction levels like the microarchitectural one, some works have pointed

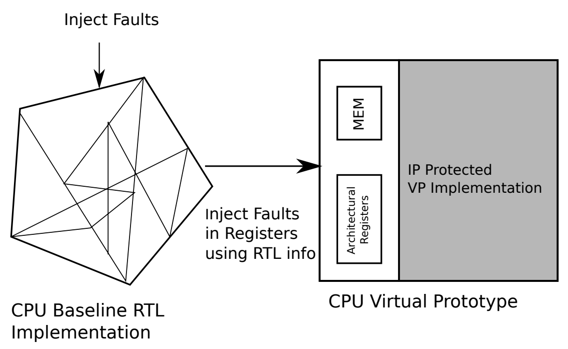

Fig. 1. Proposed methodology.

out the difficulties of correlating these results with the ones obtained at the physical level [13]. In [4], a quantitative analysis on the divergences was presented, though restricted to bit-flip models injected in flip-flops. Further analysis on the impact at instruction level of low-level (RTL and gate level) faults was presented in [15]. In that work, stuck-at and bit-flip models are injected to profile higher-level implication, but injections are limited to the control logic and a reduced set of it in the case of gate level.

## III. Certification-Aware Fault Injection in Virtual prototypes

In this paper we present a methodology that targets performing meaningful fault-injection experiments using virtual prototypes. The proposed methodology consists of two separate steps: (1) a characterization of the fault propagation in an RTL description of the processor, and (2) the actual injection in the virtual prototypes.

### A. Characterizing Fault behaviour at RTL level

We are interested in analyzing the influence of faults in the system towards the incorrect delivery of results, i.e. the appearance of failures, and the moment at which those failures may appear. To characterize fault propagation we propose a methodology that consists of injecting faults in all possible nets of an RTL processor description, using the tool FALLES described in Section IV.

From the injectable nets we have excluded the register file and cache memory structures due to the following reason: errors occurring within these structures are effectively detected and/or corrected by employing redundancy mechanisms (e.g., error correction codes) and this is the case in most of the processors targeting safety-critical applications [25][11]. Moreover, available nets in these structures do not realistically represent their area. Memory structures are typically implemented using SRAM cells to miminimize area and power and the RTL includes only an instantiation of these components as a black box and/or its behavioral description.

For every fault injection where a net has been forced to a given value, we compare the outputs of architectural registers (general purpose and control registers) and the data and address buses of the core at the on-chip boundaries to the ones obtained with a fault-free simulation. Furthermore, we account for the time elapsed between a mismatch appearing in any of those architectural registers and the subsequent mismatch in the buses which appears in some cases. Gathering this information enables, for instance, detecting the most sensitive registers, which may be ideal candidates for mitigation, or the average time it takes to show erroneous state to the buses, which can be used to determine the maximum detection timespan in lockstep systems [10]. More importantly, we can match it with the information available at the architectural level to increase the accuracy of the injections at such level [6]. In depth, if the propagation information –the number of injected faulty nodes which reflect the wrong value in any architectural register or memory position– is used to inject precisely those nodes of the virtual prototype (see Figure 1), an increase in the accuracy of the high-level injection should be attained.

### B. Fault injection at Virtual prototypes

Typically, a CPU virtual prototype consists of two differentiated parts: the functional emulator and the timing simulator. Depending on the tool and the targeted CPU, the details of the implementation for both the timing simulator and the functional emulator might be not accessible. A functional emulator is able to run application code that has been compiled for a particular architecture and to perform its execution in such a way that the memory data and architectural registers contain an exact representation of the real processor state. Precisely, the information of the processor state has to be disclosed to allow compilers and system software to use the modeled CPU. In that respect, we propose a fault-injection approach that uses only architectural registers and memory contents to minimize the intrusiveness of the fault injection in the virtual prototype. Additionally, as mentioned before, to accurately capture the behavior of the faults affecting the different processor nets, fault-injections at the virtual prototype must be enriched with meaningful information from the RTL. For example, in line with the results shown in [5], if fault injections target architectural registers only it is important to know the percentage of total processor faults that are represented by such fault-injection strategy. We let as future work the definition of the suitable injection strategies in the virtual prototype.

## IV. FALLES: Fault injection and Analysis for Low Level Evaluation Suite

The tool FALLES is an injection and analysis tool comprising a set of TCL and AWK scripts. It is designed to operate with low level descriptions of a system, mainly RTL and Gate Levels, by using the technique of simulation commands [12]. To do so, it currently supports Modelsim [26] tool to perform the simulation.The reason of its existence is accelerating the costly process of simulating a complex system running a realistic workload. The methods to achieve the improvement are quite straightforward. First of all, it trims all the simulation generated outputs to the minimum required amount of data, offering the possibility to apply any type of post-processing to them. Second it exploits massive parallelism of multi-core or grid computers to run up to several thousand concurrent threads. Third, it uses highly optimized text parser AWK to process the results.

When using a tool like FALLES, maintainability, extendability, ease of use and versatility are paramount. It can perform cycle-accurate simulations or, for an implemented design in Gate level, optionally accept Standard Delay Files (SDF) as inputs to study process corners. If such type of simulation is performed clusters or grids are the preferred option to exploit a huge amount of cores. Up to now Oracle Sun Grid Engine (SGE) management system [23] is supported, but it is very easy to add support for other grid managers.

Regarding the injectable fault models, stuck at 1 or 0, open line, indetermination, pulse or bit-flip are supported, in the

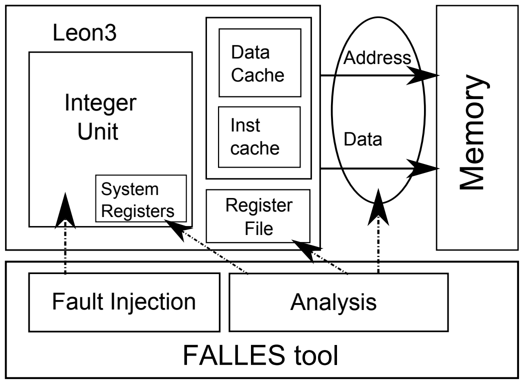

Fig. 2. RTL robustness verification framework

variations of transient, intermittent or permanent duration. New fault models can be added at will just editing some TCL code.

## V. Experimental Results

In this section we present some preliminary results of the characterization of how faults in any of the available nets in the processor propagate to the architectural registers. The results presented in this section correspond with fault injections performed in a CPU RTL description and focus on the propagation of faults in every processor net to the processor architectural registers. In fact, this information is the one that will be used in the future to feed fault injections in the virtual prototype.

### A. Experimental Setup

For the experimentation a 32-bit LEON3 SparcV8 microcontroller is selected, mainly because an RTL model is available along a microarchitectural description, and it is a processor used in safety-critical systems [24]. The LEON3 comprises mainly a 7-stage pipeline for integer operations (integer unit, *IU*) plus data and instruction caches. Since a minimal configuration of the processor has been chosen to limit the cost of RTL simulations, all instructions use all pipeline stages, and there is no floating point unit. The style of RTL description is homogeneous in the integer unit, which is described in a structural synthesizable VHDL. Such *IU* unit has been chosen as the target of injections, in a test framework as described by Figure 2. Injection and analysis points have been selected according to Section III. To enable analysis of register faults in a timely manner LEON3 has been configured to use a flat register file[2] as this reduces the number of total registers that need to be studied in every simulation.

The workload chosen for investigation includes programs from 2 different benchmark suites: the Mälardalen WCET group suite [9], suitable to test real-time system properties and the EEMBC Autobench suite [17], which reflects realistic tasks of some automotive safety-critical systems. The selected programs are: a finite impulse response filter over a 700 items long sample (*fir*), a matrix multiplication of 4x4 size (*matmult*), a road speed calculator (*rspeed*), a CAN bus reader (*canrdr*) and a tooth-to-spark task, which locates the engine's cog when the spark is ignited (*ttsprk*).

Regarding the faultload, several permanent hardware fault models have been chosen, specifically single stuck-at-1, stuck-at-0 and open line. These have been injected using simulator commands as in [12]. The campaign for each fault model and workload has consisted of one experiment per injection node (since permanent faults are applied), totaling 5,246 nodes. As the focus of the experiments is to classify fault propagation, each experiment applies a single injection in a fixed instant: just before the execution of the main procedure, after the initialization.

[2]Note that a typical SPARC configuration uses a windowed register file configuration with around 144 32-bit registers. Tracking the contents of 144 32-bit registers even for relatively small benchmarks is currently unfeasible.

### B. Results

After injections, Figure 3 shows the distribution of different errors in the register file ('user registers') and control registers ('system registers') of the integer unit. The axis shows the percentage of total injected faults which propagated to the registers to become errors, for the specified fault models. Only one of the different benchmarks, ttsprk, is shown for space reasons. As observed, with the FALLES tool we have determined the information regarding how are faults propagated throughout the circuits to reach when applicable the analyzed registers, where the critical positions of the register file appear to be numbers 15 and 129 for such benchmark.

In addition, Figure 4 shows the histogram of the distribution of latencies for the same ttsprk benchmark for the 3 fault models. The shown latencies account for the time spent since the first moment an architectural register is altered until a mismatch is detected at any of the peripheral buses (data or address). Taking into account that the clock cycle of the considered system was 10 ns, we can tell the number of cycles it takes to propagate an error to a failure (considering the buses as outputs) is mainly under 138, with some sparse cases taking longer time. This latency information is helpful when assessing the behavior of real time systems such as those meant to be certified.

## VI. Conclusions

The use of virtual prototypes has recently arised as a promising approach to reduce the costs associated with the robustness verification of safety critical processors. However, for this low-cost simulation approach to be adopted its accuracy must be validated. In this paper we presented a methodology to increase the confidence on the fault injection experiments using virtual prototypes. The proposed methodology uses some meaningful imformation that is extracted from fault injection experiments at the RTL to enrich the fault injections at the virtual prototype. In the proposed methodology fault injections in the virtual prototype target only memory and architectural registers of the CPU to minimize its intrusiveness. The actual definition of the fault injection strategies in the virtual prototypes is let as future work.

## Acknowledgements

The research leading to these results has received funding from the Ministry of Science and Technology of Spain under contract TIN2012-34557 and HiPEAC. Likewise, Jaume Abella is partially supported by the Ministry of Economy and Competitiveness under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717.

## References

[1] J. Abella, E. Quiones, F.J. Cazorla, M. Valero, and Y. Sazeides. Rvc-based time-predictable faulty caches for safety-critical systems. In *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*, pages 25–30, July 2011.

[2] ARTEMIS Joint Undertaking. *VeTeSS project: www.vetess.eu.*

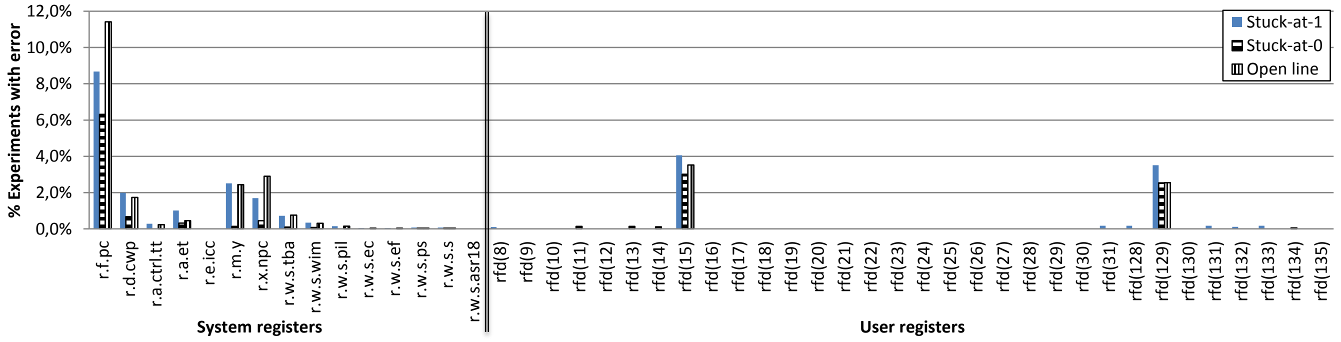

Fig. 3. Errors distribution in system and user registers, ttsprk

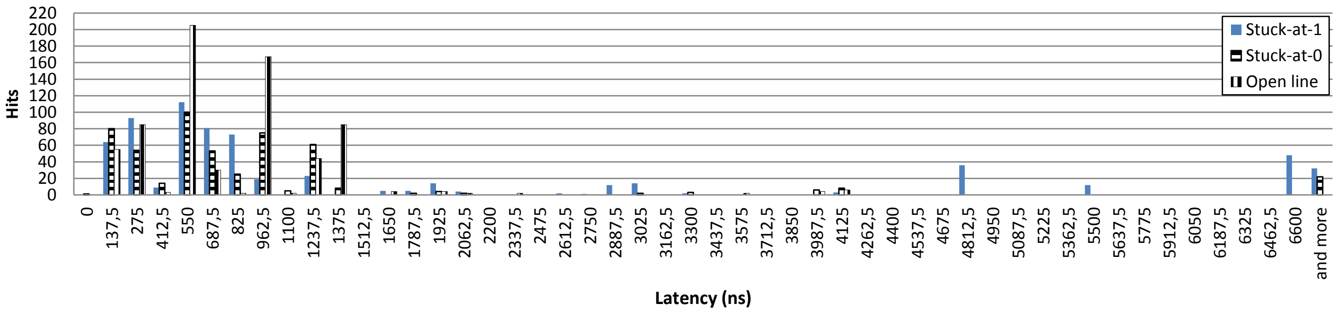

Fig. 4. Histogram of propagation latencies from error to failure, ttsprk

[3] J.-C. Baraza, J. Gracia, S. Blanc, D. Gil, and P.-J. Gil. Enhancement of fault injection techniques based on the modification of vhdl code. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(6):693–706, June 2008.

[4] Hyungmin Cho, S. Mirkhani, Chen-Yong Cher, J.A. Abraham, and S. Mitra. Quantitative evaluation of soft error injection techniques for robust system design. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–10, May 2013.

[5] Jaime Espinosa, David de Andrés, Juan-Carlos Ruiz, Carles Hernandez, and Jaume Abella. Towards certification-aware fault injection methodologies using virtual prototypes. In *Proceedings of the 21st International Online Testing Symposium (IOLTS15) Elia, Halkidiki, Greece. 6-8 July 2015*, 2015.

[6] Jaime Espinosa, Carles Hernandez, Jaume Abella, David de Andres, and Juan Carlos Ruiz. Analysis and rtl correlation of instruction set simulators for automotive microcontroller robustness verification. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 40:1–40:6, New York, NY, USA, 2015. ACM.

[7] International Organization for Standardization. *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.

[8] Pedro Gil, Jean Arlat, Henrique Madeira, Yves Crouzet, Tahar Jarboui, Karama Kanoun, Thomas Marteau, Joo Dures, Marco Vieira, Daniel Gil, Juan Carlos Baraza, and Joaqun Gracia. Fault representativeness. Technical report, DBench project, IST 2000-25425 [Online]. Available: http://www.laas.fr/DBench, 2002.

[9] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper. The Mälardalen WCET benchmarks – past, present and future. In Björn Lisper, editor, *WCET2010*, pages 137–147, Brussels, Belgium, jul 2010. OCG.

[10] C. Hernandez and J. Abella. Live: Timely error detection in light-lockstep safety critical systems. In *DAC*, 2014.

[11] Infineon. AURIX - TriCore datasheet. highly integrated and performance optimized 32-bit microcontrollers for automotive and industrial applications, 2012. https://www.infineon.com/dgdl?folderId=db3a304412b407950112b409ae660342&fileId=db3a30431f848401011fc664882a7648.

[12] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson. Fault injection into VHDL models: the mefisto tool. In *FTCS*, 1994.

[13] Man-Lap Li, P. Ramachandran, U.R. Karpuzcu, S.K.S. Hari, and S.V. Adve. Accurate microarchitecture-level fault modeling for studying hardware faults. In *HPCA*, 2009.

[14] LiP6. Soclib, 2003-2012. http://www.soclib.fr/trac/dev.

[15] M. Maniatakos, N. Karimi, C. Tirumurti, A. Jas, and Y. Makris. Instruction-level impact analysis of low-level faults in a modern microprocessor controller. *Computers, IEEE Transactions on*, 60(9):1260–1273, Sept 2011.

[16] J.-H. Oetjens, N. Bannow, M. Becker, O. Bringmann, A. Burger, M. Chaari, S. Chakraborty, R. Drechsler, W. Ecker, K. Grüttner, Th. Kruse, C. Kuznik, H. M. Le, A. Mauderer, W. Müller, D. Müller-Gritschneder, F. Poppen, H. Post, S. Reiter, W. Rosenstiel, S. Roth, U. Schlichtmann, A. von Schwerin, B.-A. Tabacaru, and A. Viehl. Safety evaluation of automotive electronics using virtual prototypes: State of the art and research challenges. In *DAC*, 2014.

[17] J. Poovey. *Characterization of the EEMBC Benchmark Suite*. North Carolina State University, 2007.

[18] P. Ramachandran, P. Kudva, J. Kellington, J. Schumann, and P. Sanda. Statistical fault injection. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 122–127, June 2008.

[19] RTCA and EUROCAE. *DO-178B / ED-12B, Software Considerations in Airborne Systems and Equipment Certification*, 1992.

[20] Daniel Sánchez, Yiannakis Sazeides, Juan M. Cebrián, José M. García, and Juan L. Aragón. Modeling the impact of permanent faults in caches. *ACM Trans. Archit. Code Optim.*, 10(4):29:1–29:23, dec 2013.

[21] B. Sangchoolie, F. Ayatolahi, R. Johansson, and J. Karlsson. A study of the impact of bit-flip errors on programs compiled with different optimization levels. In *EDCC*, 2014.

[22] Synopsys. *Platform Architect. http://www.synopsys.com/Prototyping/ArchitectureDesign/Pages/platform-architect.aspx.*

[23] http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html. *Sun Grid Engine*. Oracle.

[24] http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53. *Leon3 Processor*. Aeroflex Gaisler.

[25] http://www.gaisler.com/index.php/products/processors/leon3ft. *Leon3 fault-tolerant Processor*. Aeroflex Gaisler.

[26] http://www.mentor.com/products/fv/modelsim. *ModelSim Simulator*. Mentor Graphics.