Error simulation based on the SystemC design description language

Francesco Bruschi, Fabrizio Ferrandi, Michele Chiamenti, Donatella Sciuto
Politecnico di Milano

1. Abstract

The combined effects of devices increased complexity and reduced design cycle time creates a testing problem: an increasing larger portion of the design time is devoted to testing and verification. Today EDA tools, moving towards higher levels of abstraction, promise greater designer productivity, resulting in increased design complexity and size.
In order to reduce the testing and verification time, different high-level approaches have been proposed in literature [2]. Most of these approaches are based on the definition of an error or fault model, applicable at a higher level of abstraction of the description of the system to be implemented.
In this paper we concentrate our attention on the evaluation of error models, used in test generation and in functional verification. Evaluation of error models is also an important aspect when fault injection methodologies are used to evaluate the dependability of complex system.
The ideas proposed by this work try to solve this evaluation and analysis problem starting from the following requirements:

- the error simulation task should be based only on the original hardware description language primitives;
- the flow from the given specification to the fault simulation should be as automatic as possible;

With these two rules satisfied we could perform error simulation just using a standard simulator, with no significant extra effort.
In our work all the considerations and examples are based on the SystemC hardware description language, but the same considerations can be applied to VHDL, Verilog or SpecC languages.
Our approach is articulated in two phases. The translation phase gets the SystemC description as input, and automatically modifies it in order to allow the fault simulation. This work is done by means of a code parsing phase, followed by a series of transformations of the intermediate structure, and by the translation back to the code form. The result is an *instrumented* version of the design, in which a set of ad-hoc functions has been inserted. The behavior of these functions can be controlled by a module external to the design under test: each function can behave as a simple identity (absence of error) or can return a value that is modified according to the selected error model. After that, a *testbench environment* is automatically built; in this environment the modified model is istantiated together with a set of other modules; during the simulation these modules feed the model with the input vectors, read the list of errors to be simulated and activate the proper error functions. Moreover, for each simulated error they check if the input sequence detect the considered error, that is, if there exists an output response of the error-free description that differs from the outut value produced by the description with the error injected.
The testbench built is a SystemC model, so the simulation can be achieved with a standard simulator, without any additional feature.
Main benefit of this approach is the possibility of performing analysis and comparison of different error models, disregarding the simulation problem. This allows, for example, the study of new and more sophisticated error models, able to catch more design errors, without having to implement a new error simulator each time the error model is modified. Changing the error model would mean changing only the error functions behavior and not rewriting a new error simulator. Coverage obtained by this process can be related to gate or defect-level fault coverage thus allowing comparison between error models based on their relation with fault or defect models. Moreover, the use of a standard HDL simulator allows the exploitation of all the possible optimizations developed for standard simulation.
Another main benefit coming from this approach is that the instrumentation and testbench creation steps are completely automated, so that almost no extra effort is needed to perform the error simluation.
The experiments we conducted show that the computational overhead implied by the execution of the error injection functions is negligible in all the cases examined, thus proving the effectiveness of the proposed approach.

2. References

[1] E. Jenn, J.Arlat, M.Rimén, J.Ohisson, J.Karlsson,"Fault injection into VHDL models: the MEFISTO tool", *Digest of Papers of 24th International Symposium on Fault-Tolerant Computing*, pp.66-75, 1994.

[2] S.Tasiran, K. Keutzer, "Coverage metrics for functional validation of hardware designs", *IEEE Design & Test of Computers* , Vol. 18 I. 4, July-Aug. 2001, pp. 36 – 45.

[3] B. Parrotta, M. Rebaudengo, M. Sonza Reorda, M. Violante, "New techniques for accelerating fault injection in VHDL descriptions", *IOLTW2000: International On-Line Test Workshop,* pp. 61-66, July 2000.