In-field test of safety-critical systems: is functional test a feasible solution?

Matteo Sonza Reorda
Dipartimento di Automatica e Informatica
Politecnico di Torino, Italy
{matteo.sonzareorda@polito.it}

***Abstract*[1]—**The growing usage of electronic systems in safety- and mission-critical applications, together with the increased susceptibility of electronic devices to faults arising during the operational phase mandate for the availability of effective solutions able to face the effects of these faults. When the target system includes a processor, one possible solution is based on running suitable test programs able to detect the occurrence of faults. This solution provides several advantages (e.g., in terms of flexibility, IP protection, and defect coverage), although it is limited by the cost for developing the test programs. This paper overviews the state of the art in the area, and discusses the trends in the area.

I. INTRODUCTION

In several domains (e.g., automotive, biomedical, space, aircrafts), electronic systems are used within mission- or safety-critical applications. In these scenarios, a misbehavior may cause catastrophic effects (e.g., hurting humans, or provoking huge economical losses). Hence, solutions must be devised and adopted to face possible faults. Regulations (e.g., ISO 26262 for automotive applications) also play a key role in this scenario. One possible solution is based on running suitable tests (involving the whole system or each single component) able to detect the occurrence of possible faults, thus triggering suitable actions able to prevent the occurrence of catastrophic effects. A possible solution, especially when the system include a processor, is based on running suitable test programs, which leverage time periods left idle by the applications (e.g., in a periodic manner, or at start-up). These test programs are supposed to be able to excite any possibly existing fault, and propagate its effects to some observable location (e.g., some memory area). This approach is sometimes referred as *Software-Based Self-Test* (or SBST) [2], and generically labeled as "functional".

When speaking about Functional test, different definitions may apply

- In some case, a functional test is meant as a test, which does not rely on any Design for Testability (DfT) structure: hence, this test only acts on the system functional inputs, and only observes the system functional outputs.
- In other cases, a functional test is intended as a test, which has been generated by only exploiting functional information about the target system (i.e., without knowing its structure). As a consequence, this test does not rely on any structural fault model, leading to possible limitations in its defect coverage capabilities.

The two definitions may be adopted in a complementary manner: for example, a test can be generated working on the functional information about the target system, and the test is only applied resorting to functional inputs and outputs.

Functional test is currently adopted in different test scenarios (both for end-of-manufacturing and in-field test) and may be motivated by different reasons, sometimes to complement other solutions, in other cases as an alternative to them.

When considering the in-field test of a system, it may happen that the DfT structures of the composing devices are not accessible any more (e.g., because they have been destroyed or made inaccessible to better protect the system safety), or are not documented by the device providers. Hence, the only feasible solution for the system company in charge of developing the in-field test is to adopt the functional approach.

It must also be underlined that, since the functional approach by definition tests the system in the operational mode (without moving to a test mode and without reconfiguring in any manner the system), it is guaranteed not to produce any form of overtesting, and may provide advantages in terms of defect coverage with respect to some DfT solutions.

II. ADOPTING IN-FIELD FUNCTIONAL TEST

When functional test is the selected solution for in-field test, two major issues have to be faced:

- How to apply the functional test, i.e., where to store the test program, how to trigger the processor to execute it, how to retrieve and check the produced results;
- How to generate the test program.

Solutions to the first issue are typically dependent on the targeted system and to the existing constraints, coming from the application environment. Moreover, when in-field test is addressed, most of the test tasks are commonly orchestrated by the Operating System.

On the other side, generating suitable test programs for processors or for processor-based systems has been the subject of numerous research efforts, starting from [1]. Clearly, this task is highly computationally intensive, being a special case of the well known sequential ATPG problem. At the moment, no commercial tools are available to automate it, and test program generation, when required, is performed in a manual manner, or with the support of some in-house developed tool.

[1] Part of this work has been supported by the European Commission through the FP7 STREP Project no. 619871 (BASTION).

However, in the last decade a number of researchers proposed guidelines able to reduce the cost for generating suitable test programs for some commonly found modules within a processor, knowing only their general architecture. For example, the work in [6] allows the generation of effective test programs for testing caches, when their architecture is known. Similarly, the work in [9] addresses Branch Prediction Units, and the one in [10] the Register Forwarding and Pipeline Interlocking Unit existing in most pipelined processors.

Interestingly, some of the faults affecting these modules do not produce wrong results, but rather force the processor to behave in a temporarily different way, typically requiring a longer time to complete the test program execution (performance faults). The detection of these faults may be particularly challenging, since it requires some techniques to observe the time behavior of the processor in a precise manner [3].

The above methods mainly correspond to algorithms, allowing a skilled engineer to manually write a test program targeting a given module or a whole processor. However, the effort and time for achieving this result may be significant, and represents a major drawback of the functional approach.

Previous efforts to automate the process were based on constrained random generation [13] or on evolutionary techniques [4]. These techniques were often limited by the huge computational effort they require.

Recently, it was shown in [5] that formal techniques can be successfully exploited to automatically generate functional test programs for a pipelined processor, taking also into account the specific constraints of in-field test [11].

The test of communication peripheral components +is also important, and can be addressed in a similar manner. In this case, the functional approach requires the combined action of the processor, programming the component and exercising/observing it on one side, and that of an external body (e.g., an ATE), exercising/observing the component on the other side [8]. For in-field solutions, where the ATE can hardly be exploited, a loop-back connection is often adopted. A similar approach can be adopted for system peripherals, such as Interrupt and DMA controllers [12], although in this case the complexity of the test grows, since it requires the combined action of several peripherals.

In the last decade methods to generate functional test programs running under specific constraints (e.g., in terms of power [14]) or providing diagnostic information [15] were developed.

Finally, it is worth mentioning that when more regular processor architectures are targeted, such as those of VLIW processors, it is possible to adopt a hierarchical approach, in which the global test program can be built, once the test program for each composing unit is known [7]. In this way the major drawback of the functional approach, corresponding to the huge cost for manually generating the test (as a consequence of the lack of automated tools), can be successfully faced.

III. Conclusions and future trends

The feasibility of an in-field test based on a functional approach, although promising, is currently limited by the cost for test program development. Despite this obstacle, which may shortly be overcame by the identification of new solutions, functional test is already practically used. Its success does not only depend on technical issues, but also on its ability in taking into account other issues, such as the ease of integration into the existing scenario, where commercial and IP protection issues are important. From this point of view, the availability of suitable test libraries that can be used by a system company when developing the in-field test for a processor is crucial. Some semiconductor companies already made some steps into this direction.

Further research activities are currently on-going, leading to a better understanding of how to identify those faults that are functionally untestable in an in-field scenario [16], or how to compact the required test programs [17].

References

[1] S. Thatte and J. Abraham, “Test Generation for Microprocessors”, IEEE Transactions on Computers, vol. 29, no. 6, pp. 429–441, June 1980.

[2] M. Psarakis et al., “Microprocessor Software-Based Self-Testing”, IEEE Design & Test of Computers, vol. 27, no. 3. May-June 2010, pp. 4-19.

[3] M. Hatzimihail et al., “A methodology for detecting performance faults in microprocessors via performance monitoring hardware”, Proc. of IEEE International Test Conference, 2007.

[4] F. Corno et al., “Automatic Test Program Generation: a Case Study”, IEEE Design & Test of Computers, vol. 21, pp. 102-109, 2004.

[5] A. Riefert et al., “An effective approach to automatic functional processor test generation for small-delay faults”, Proc. of the Conference on Design, Automation and Test in Europe (DATE), 2014.

[6] S. Di Carlo et al., "Software-Based Self-Test of Set Associative Cache Memories", IEEE Trans. on Computers, vol. 60, issue 7, pp. 1030-1044, July 2011.

[7] D. Sabena et al., “On the Automatic Generation of Optimized Software-Based Self-Test Programs for VLIW Processors”, IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 22, n. 4, pp. 813-823, 2014.

[8] A. Apostolakis et al., “Test Program Generation for Communication Peripherals in Processor-Based Systems-on-Chip”, IEEE Design & Test of Computers, vol. 26 n. 2, pp. 52-63, 2009.

[9] E. Sanchez, M. Sonza Reorda, “On the Functional Test of Branch Prediction Units”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2014

[10] P. Bernardi et al., “A functional test algorithm for the register forwarding and pipeline interlocking unit in pipelined microprocessors”, Proc. of 8th International Design and Test Symposium (IDT), 2013

[11] A. Riefert et al., “On the Automatic Generation of SBST Test Programs for In-Field Test”, Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015

[12] M. Grosso et al., “Software-Based Testing for System Peripherals”, Journal of Electronic Testing (JETTA), vol. 28 n. 2, pp. 189-200, 2012.

[13] P. Parvathala et al., “FRITS - a microprocessor functional BIST method”, Proc. of IEEE International Test Conference, 2002, pp. 590-598

[14] J. Zhou and H.-J. Wunderlich, “Software-Based Self-Test of Processors under Power Constraints”, Proc. of Conference on Design, Automation and Test in Europe (DATE), pp. 430-436, 2006.

[15] P. Bernardi et al., “An Effective technique for the Automatic Generation of Diagnosis-oriented Programs for Processor Cores”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, pp. 570-574, 2008.

[16] P. Bernardi et al., “On-line functionally untestable fault identification in embedded processor cores”, Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013, pp. 1462-1467

[17] M. Gaudesi et al., “On test program compaction”, Proc. of IEEE European Test Symposium, 2015