# On-line Testing for FPGA

**Pavel Kubalík, Hana Kubátová**
Department of Computer Science and Engineering,
Czech Technical University in Prague, Karlovo nam. 13, 121 35 Prague 2
E-mail: xkubalik@fel.cvut.cz, kubatova@fel.cvut.cz

## Abstract

This paper focuses on the on-line error detection in circuits implemented in FPGAs. We have used error detection codes to ensure the self-checking property. A fault in a given combinational circuit has to be detected and signalized at the time of its appearance and before the further distribution of errors. Hence a safe operation of the designed system is guaranteed. The check bits generator and the checker were added to the original combinational circuit to detect an error during normal circuit operation called concurrent error detection and to ensure the Totally Self-Checking property. Only combinational circuits are considered. The benchmarks used in this work in order to compute a quality of the used code, are described by equations instead of tables, mainly used. All of our experiments assume their XILINX FPGA implementation.

## 1. Introduction

Nowadays when the circuit integration increases, the importance of radiation impact on integrated circuits grows even at the sea level. The mobile nets are becoming more important because of their radiation. They can affect any circuit used every day. Some machines like the control units in cars can play an important role in places such as tunnels, where a car fault can endanger human lives. Other important areas are aviation, medicine or space missions. All of these applications and many others depend on a correct function of circuits and one wrong result can lead to the huge losses.

The FPGA circuits are more and more often used to realize any function because of their prices and capabilities to upgrade the function when a bug is discovered. Another advantage is their possible dynamic reconfiguration when a fault in the circuit is detected and localized [4].

This paper is organized as follows. Section 2 describes related works in this field, Section 3 introduces the used fault model and proposes general methodology for comparing the quality of used error detection codes. Section 4 presents the experimental issues and solutions of the parity bits generation for all used benchmark tests. Our future work and conclusions are presented in the Section 5.

## 2. Related works

There are many papers focused on concurrent error detection in a random logic circuit. The combinational circuits are used as a basic element for testing of the proposed method that ensures totally self-checking (TSC) properties. There are two basic properties that must be taken into account and which are contradictory:

- Fault coverage must be achieved as high as possible - up to one hundred percent. The error detecting codes are used to ensure TSC properties. The maximal fault coverage must be ensured for the whole design – for the redundant parts too. All these on-line design methods increase the area overhead of the designed circuit.
- Area overhead is a second property forcing designers to reach its minimum while saving the maximal fault coverage. There is a relation between the area overhead and the fault coverage. It can be shown that higher fault coverage does not mean higher area overhead, in some cases [2].

The concurrent error detection (CED) design methodology used to satisfy TSC

property has a deep impact to the fault coverage of circuits implemented in FPGAs. Some results of the fault coverage implemented in FPGAs are presented in [1].

Basic methods used for the fault detection in logic circuits are based on a simple duplication. This methodology allows to determine the final area overhead before the duplication generator is executed. The duplicated part can be modified to remove common-mode failures (CMFs). Another approach can be used in cases when duplicated circuit is modified to decrease the number of outputs of duplicated part (output parity bits are used instead original outputs). The error codes can be used in this case. Both of these techniques are compared in [3, 6].

The area overhead plays an important role in popularity of the method used for ensuring the TSC property. Therefore special schemes were designed for circuits with regular structures, for example adders, multipliers or memories.

There are two main reasons why the CED techniques were not so popular: a very high area overhead and a low disposition to temporary faults due to their large feature sizes. Nowadays when the deep submicron technology is widely used, CED techniques for circuit with an unknown structure are more and more important.

Some of the new design methods try to reach smaller area overhead but achieve low fault detection. For example, only some inputs may be used to ensure the partial self-checking property of a multilevel logic, by using low-cost parity error detecting codes [7].

Next different design methodology ensuring smaller area overhead uses duplication of only some part of the original circuit. This method is based on reduction of the number of selected input combinations [8].

Some techniques describe methods how to detect the faulty part of an FPGA without stopping its function [5]. These methods test unused part of the FPGA. When the test is performed the tested part is exchanged with the used part and the testing process is started again for currently unused area. Here the BIST techniques can be successfully exploited [11].

## 3. TSC circuit design

We have used the structure on Figure 1 as a basic model of the totally self-checking circuit. The final scheme consists of four basic blocks: the original combinational circuit, its duplication, the check bits generator and the checker.

The checking bits are generated from the primary inputs of the original circuit. The primary outputs and the checking bits are used as an encoded output. The checker compares the check bits with the check bits generated directly from the primary outputs.

To satisfy the self-checking property the checker must have at least two outputs [10]. The first output is used for the regular operation and the second for the error indication.

This basic structure ensures that a circuit can be totally self-checking (TSC). Another condition that has to be satisfied is that the basic structure has to be self-testing and fault secure.

The blocks “Duplicated circuit” and “Code generator” can be minimized and the resulted design of both of them can be less then the original tested circuit.

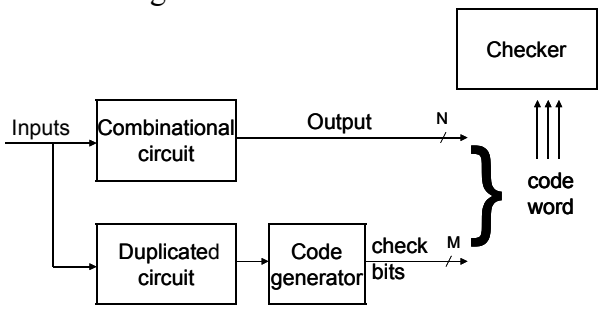


**Figure 1. Approach of TSC circuit**

### 3.1 Adopted fault model

All of our experiments use FPGA circuits. The circuit implemented in an FPGA consists of individual memory elements (LUTs - look up tables). In Figure 2 we can see 3 gates mapped into a LUT.

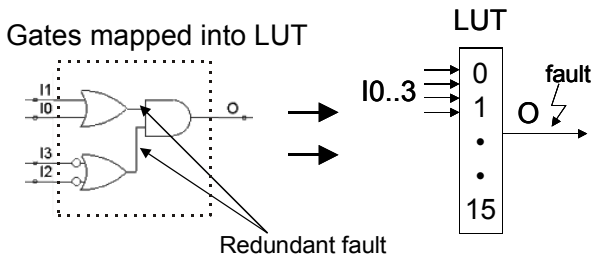


**Figure 2. Fault model**

The original circuit has two inner nets. The original set of the test vectors covers all faults in these inner nets. For the LUT these test vectors are redundant. For circuits realized by LUTs the change (a defect) in the memory leads to a single event upset (SEU) at the primary output of the LUT. Therefore we can use the stuck-at fault model in our experiments to detect SEU – only some from the detected faults will be redundant.

The used fault model is described by a simple example in Figure 3. We have used only one LUT, for the simplicity. The LUT realizes a circuit containing 3 gates. Primary inputs from I0 to I1 are the same as the address inputs for the LUT. When this address is selected its content is propagated to the output.

We assume the following situation: The content of this LUT can be changed, e.g., an electromagnetic interference, cross-talk or alpha particles. The appropriated memory cell is set to one and the wrong value is propagated to the output. It means that the realized function is changed and output behaves as a single event upset. By this example we can say that a change of any LUT cell leads to a stuck-at fault on the output. This fault is observed only if the bad cell is selected. This is the same for circuits based on gates. Some fault can be masked and does not necessarily lead to an erroneous output.

Due to the fact that some faults are masked, the possibility of their appearance may be situated in time when unused logic is used. For example if one bit of a LUT is changed, the erroneous output will appear, while the appropriate bit in a LUT is selected by the address decoder.

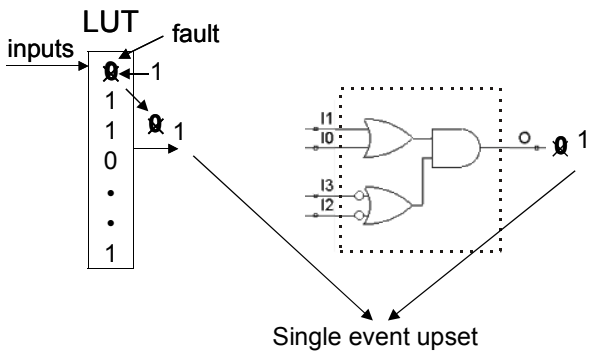


**Figure 3. Fault Model - Example**

### 3.2 Software for experimental evaluation

Figure 4 describes how the test is performed for every detecting code. We have used the ISCAS85 benchmarks [9] in our experiments. Every benchmark is duplicated by renaming the original circuit and loading the same original circuit again. Next we add the code generator behind the duplicated part. Then we generate the modified benchmark.

The Atalanta ATPG tool was used to generate the minimal tests for benchmarks. The obtained test set with the modified benchmark is put into the last part. In this part we inject a fault and we compute the fault coverage by our simulator. The bold rectangles represent our original software.

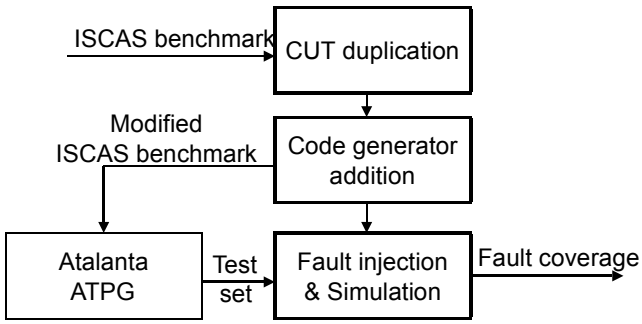


**Figure 4. Design scheduling of self-checking circuit.**

### 3.3 Software solution description

The first task is the modification of the original circuits – in Figure 4 "CUT duplication". It means that this part ensures loading, saving and renaming a circuit. To duplicate circuit we have to read the original

circuit, rename this circuit and load the original circuit again.

The second task ("Code generator addition") is adding the check nets. This could be done by one of these methods: by adding a net or by adding a gate.

The third task is the simulation of the modified circuit and the fault injection ("Fault injection & Simulation"). The last task is the computation of the fault coverage for the whole circuit (the original circuit and the check bits generator).

All the software was written in Microsoft Visual C++. The Atalanta ATPG tool was used to generate minimal test set.

### 3.4 Compound design

The implemented design has to satisfy the condition of modularity. Due to this fact we have proposed a special design procedure suitable for TSC and reconfiguration properties. The TSC property must be fulfilled for every module and of course for the whole design too. We have proposed such structure that satisfies self-checking properties and enable dynamical reconfiguration, see Figure 5.

The number of outer nets and the complexity of every block affect the fault coverage and the final area overhead.

## 4 Experimental results

All our experiments use the ISCAS85 benchmarks [9] where all the circuits are combinational only. These benchmarks are based on real circuits from large designs.

Description of tested benchmarks:

- c432 27-channel interrupt controller
- c499/c1355 32-bit SEC circuit
- c880 8-bit ALU
- c1908 16-bit SEC/DED circuit
- c2670 12-bit ALU and controller

The performed experiments were focused to obtain the hundred percent of a fault coverage.

**Table 1 - Fault coverage**

| Benchmark | Fault coverage [%] | | | |
|---|---|---|---|---|
| | 1 bit | 2 bits | 6 bits | 7 bits |
| **c432** | 83,5 | 92 | 99,5 | 98 |
| **C499** | 96,5 | 98 | 100 | 100 |
| **c880** | 97,5 | 99 | 99,9 | 100 |
| **c1355** | 97 | 98,5 | 100 | 100 |
| **c1908** | 89 | 96 | 99,8 | 99,5 |
| **c2670** | 80 | 95,7 | - | 99,98 |

In a case, when only one parity bit is used, the high fault coverage is reached (more then 80 percent), see Table 1. Last two columns show that more parity bits can mean a less fault coverage. The fault coverage strongly depends on the used system of parities, [2].

The fault coverage depends also on the circuit structure, where the less outputs implicates the worst fault coverage, see Table 2. But it follows from the controllability and observability properties, [11].

**Table 2 - Number of inputs and outputs**

| Benchmark | Inputs | Outputs |
|---|---|---|
| c432 | 36 | 7 |
| C499 | 41 | 32 |
| c880 | 60 | 26 |
| c1355 | 41 | 32 |
| c1908 | 33 | 25 |
| c2670 | 233 | 140 |

These experiments were performed before their implementation into the FPGA. It means that we cannot say that after implementation we obtain more or less fault coverage. In some cases we obtain more but in some cases less one [1].

The final fault coverage depends on the synthesis process. Uncontrolled optimizations and duplications may decrease the fault coverage. It is caused by a modification of the original circuit (duplication and minimization processes). All of these steps must be done before the addition of the parity bits generator. To prevent the TSC circuit against the fault

coverage decreasing, all synthesizes modifications must be disabled.

The fact that the fault coverage increases after mapping, is caused by removing of some inner nets of the original circuit (Figure 2).

## 5. Conclusion and future work

This work is a part of the methodology of the automatic design process for the concurrent error detection (CED) circuits based on FPGAs with the possible dynamical reconfiguration of the faulty part. The reliability characteristics can increase by reconfiguration after the error detection. The most important criterion is the speed of the fault detection and the safety of the whole circuit with respect to the surrounding environment.

We can summarize, that all of our experiments say that 100% fault coverage can be reached for the whole design including checking parts. It is achieved by using more redundancy outputs generated by the special codes.

The Hamming-like code can be used as a suitable code to generate check bits. Its type depends on the number of outputs and on the complexity of the original circuit [2]. More complex circuits need more check bits. We would like to reduce the duplicated circuit and compute the fault coverage again.

We have proposed a new solution of the check bits generator design. Because we want to increase the reliability characteristics of the circuit implemented in FPGAs we have to modify the circuits at the netlist level.

All of our experiments apply the combinational circuits only. But many circuits in real designs are composed from sequential parts, too. However such circuits can be disjoint to the simple combinational parts separated by flip-flops. As an example, the finite state machine can be divided into two parts, where the first part covers combinational logic from inputs to flip-flops (with feedback) and the second one covers the combinational logic from flip-flops to outputs (with the nets that are connected directly from the input to the output). Therefore the restriction to the combinational circuits only does not reduce the quality of our methods and experimental results.

Our future improvement has to discover more closed relations between real FPGA defects and the used fault models. The minimization of the whole TSC design to obtain the less area overhead has been experimented now. Also the appropriate decomposition of the designed circuit is under our intensive research.

## Acknowledgement

This research has been in part supported by the GA102/04/2137 grant, CTU0408913 grant and MSM 212300014 research program.

## References

[1] C. Bolchini, F. Salice and D. Sciuto. “Designing Self-Checking FPGAs through Error Detection Codes.” 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT’02), November 06 - 08, 2002 , Canada, pp. 60.

[2] P. Kubalík and H. Kubátová. “Design of Self Checking Circuits Based on FPGA.” In: Proceedings of the 15th International Conference on Microelectronics. Cairo: Cairo University, 2003, pp. 378-381.

[3] S. Mitra and E. J. McCluskey. “Which Concurrent Error Detection Scheme To Choose?” Proc. International Test Conf., pp. 985-994, 2000.

[4] XAPP 151 (v1.5), “Virtex Series Configuration Architecture User Guide”

[5] M. Abramovici, C. Stroud, S. Wijesuriya, C. Hamilton, and V. Verma, "Using Roving STARs for On-Line Testing and Diagnosis of FPGAs in Fault-Tolerant Applications," Proc. IEEE Intn'l. Test Conf., pp. 973-982, 1999.

[6] S. Mitra and E. J. McCluskey. “Diversity Techniques for Concurrent Error Detection” Center for Reliable Computing, Dept. of Electrical Engineering and Computer Science Stanford University, Technical Report 00-7, June 2000.

[7] K. Mohanram, E. S. Sogomonyan, M. Gössel, N. A. Touba. “Synthesis of Low-Cost Parity-Based Partially Self-Cheking Circuits”, Proceeding of the 9th IEEE International On-Line Testing Symposium (IOLTS'03), pp. 35.
[8] P. Drineas, Y. Makris, "Concurrent Fault Detection in Random Combinational Logic," in Proceedings of the IEEE International Symposium on Quality Electronic Design (ISQED), pp. 425-430, 2003
[9] F. Brglez, H. Fujiwara. „A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan.” Proc. of International Symposium on Circuits and Systems, pp. 663-698, 1985.
[10] M. Nicolaidis and Y. Zorian. “On-Line Testing for VLSI - A Compendium of Approaches.” On-Line Testing for VLSI, Kluwer Academic Publisher, London 1998,
[11] Ch. E. Stroud. “A Designer’s Guide to Built-In Self-Test.” Kluwer Academic Publisher, London 2002.
[12] M. L. Bushnell and V. D. Agrawal. “Essentials of Electronic Testing.” Kluwer Academic Publisher, London 2000.

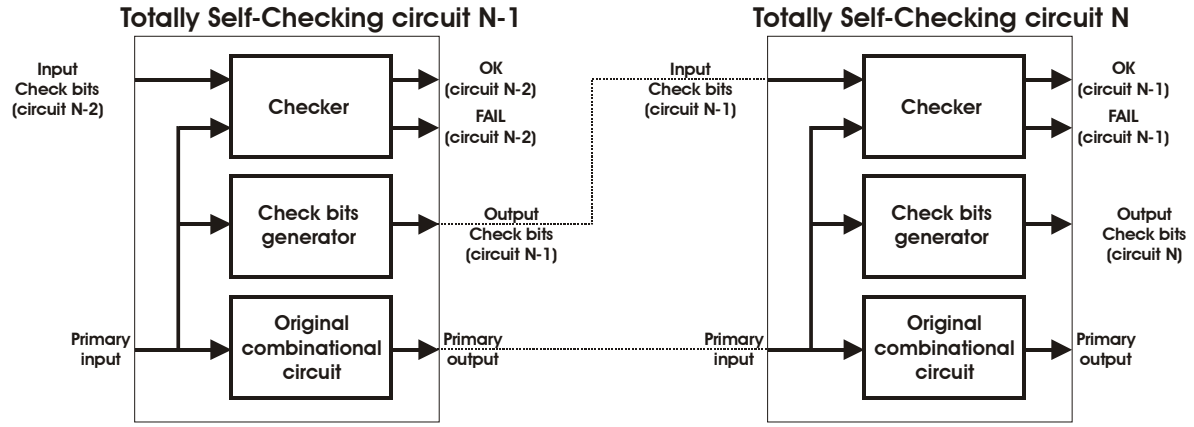


**Figure 5. Proposed structure of TSC circuits implemented in FPGA**