Designing Self-Checking FPGAs through Error Detection Codes

Cristiana Bolchini, Fabio Salice, Donatella Sciuto
Politecnico di Milano, Dip. Elettronica e Informazione
P.zza L. da Vinci, 32, I20133 Milano, Italy
{bolchini, salice, sciuto}@elet.polimi.it

Abstract

The paper presents an approach for adapting sound techniques for designing Totally Self-Checking (TSC) combinational circuits based on error detection codes developed in the past for ASIC platforms to FPGA architectures. The attention is focused on the constrained synthesis of the logic functions that must guarantee that each possible fault produces only observable errors with respect to the adopted error detection code. An experimental framework has been setup to evaluate the effectiveness of Concurrent Error Detection (CED) design methodologies to a set of benchmark circuits. Results show that the physical implementation of the functionally TSC network onto an FPGA does not maintain the desired properties, requiring a post synthesis modification of the obtained network in order to achieve the pursued fault coverage. As a consequence, specific FPGA techniques for both fault analysis and CED design methodologies will have to be investigated.

1 Introduction and Motivation

Today Field Programmable Gate Arrays (FPGAs) are not only considered for fast system prototyping but also as flexible platforms for hw/sw systems, with their low manufacturing cost and the possibility to be configured (and reprogrammed dynamically) in short periods of time. Due to our research interests, Xilinx FPGAs ([1]) are the target platform for the realization of reliable devices; Xilinx's devices are Look-Up Table (LUT) FPGAs, and our work targets this class of building blocks for the final implemtation.

Mission critical applications require that any failure that may lead to erroneous behavior and computation is detected and signaled as soon as possible in order not to jeopardize the entire system. Totally Self-Checking (TSC) systems are designed to be able to autonomously detect faults when they occur during normal circuit operation. Hardware redundancy is a commonly adopted strategy to achieve such on-line fault detection capability. This issue has been thoroughly studied, and still is, referring to ASIC platforms.

In the past, FPGA testing and reliability issues have been investigated with regard to testing [2], interconnections [3, 4] and fault-tolerance [5, 6, 8, 9]; a few studies have been carried out dealing with on-line fault detection [7, 10]. We are currently investigating the possibility to use part of the internal resources of the FPGA to implement an on-line fault detection, pursuing in the future the exploitation of the dynamic programmability feature to achieve also fault tolerance properties.

Our interest relates to the traditional issues concerning the gap between the functional assessment of self-checking properties through error detecting codes and the final logic implementation. As it has been seen in a different environment (e.g. ASIC), the synthesis cannot be straightforward: ad-hoc techniques are required to enforce fault/error observability constraint [11]. Moving to a different platform, such as the FPGA is, the previously investigated and defined methodologies are not working anymore, and further exploitation of the "traditional" methodologies needs to be investigated.

The paper is organized as follows. Section 2 introduces the adopted fault model and the proposes the general methodology for designing the TSC FPGA through the adoption of Error Detection Codes. Section 3 introduces the proposed approach for FPGA platforms, discussing how to guarantee and verify the TSC properties for a combinational network. Section 4 presents the experimental setup, together with the simulation architecture used evaluate the achieved fault coverage, and to allow the local modification of the logic in order to achieve the complete coverage. Concluding remarks and future developments are presented in the final Section 5.

2 Background and basics

This section introduces the adopted fault model and presents the design methodology leading us to the design of FPGAs with on-line fault detection properties.

2.1 The reference platform

The target platform introduces significant constraints as far as the technology is concerned, in terms of the available basic elements used for implementing the circuit functionality. We refer to the Xilinx's Virtex^{TM}-II Family of FPGA platforms, characterized by the following basic elements:

- Input/Output Blocks
- Combinational Logic Blocks (CLBs)
- Switch Matrix constituting the interconnections among the CLBs

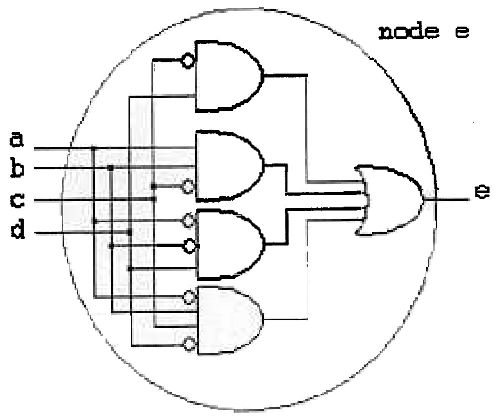
The Virtex-II configurable logic blocks (CLB) are organized in an array and are used to build combinatorial and synchronous logic designs [12]. Each CLB element is tied to a switch matrix to access the general routing matrix. A CLB element comprises 4 slices, each one including two 4-input function generators, carry logic, arithmetic logic gates, wide function multiplexers and two storage elements. Each 4-input function generator is programmable as a 4-input Look-Up Table memory (LUT), 16 bits of distributed SelectRAM memory, or a 16-bit variable-tap shift register element. Four independent inputs are provided to each of the two function generators in a slice (F and G). These function generators can implement any arbitrarily defined boolean function of four (or less) inputs. In addition to the basic LUTs, the Virtex-II slice contains logic (MUXF5 and MUXFX multiplexers) that combines function generators to provide any function of five, six, seven, or eight inputs (MUXFX can be either MUXF6, MUXF7 or MUXF8 according to the slice considered in the CLB).

The LUT desired functionality is "stored" into the LUT at configuration time, and the inputs work as address lines selecting the stored value to be put on the LUT output. Fig. 1 shows a 4 inputs LUT example, starting from (a) the truth table, (b) the implementation with logic gates, and (c) the LUT implementation. In the LUT it is possible to see the internal configuration consisting of a register to store the output values and a multiplexer to select the value based on the input signals.

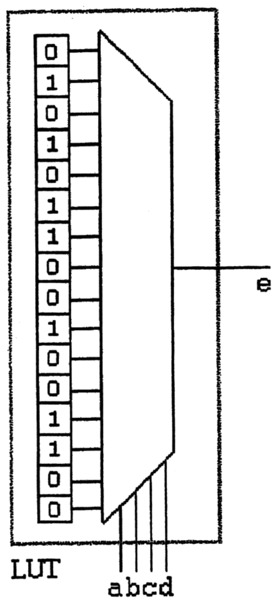
abcd	e
0000	0
0001	1
0010	0
0011	1
0100	0
0101	1
0110	1
0111	0
1000	0
1001	1
1010	0
1011	0
1100	1
1101	1
1110	0
1111	0

e = !cd + ab!c + !a!bd + !abc!d

(a)



(b)



(c)

Figure 1. (a) Truth table and boolean function of the desired functionality, (b) the netlist at gate level, and (c) the LUT implementing it.

The switch matrix is programmed to obtain the necessary connections between the CLBs, whereas Input/Output Blocks are programmed to create the FPGA interface with the environment. The next subsection will introduce the elements covered by the proposed investigation.

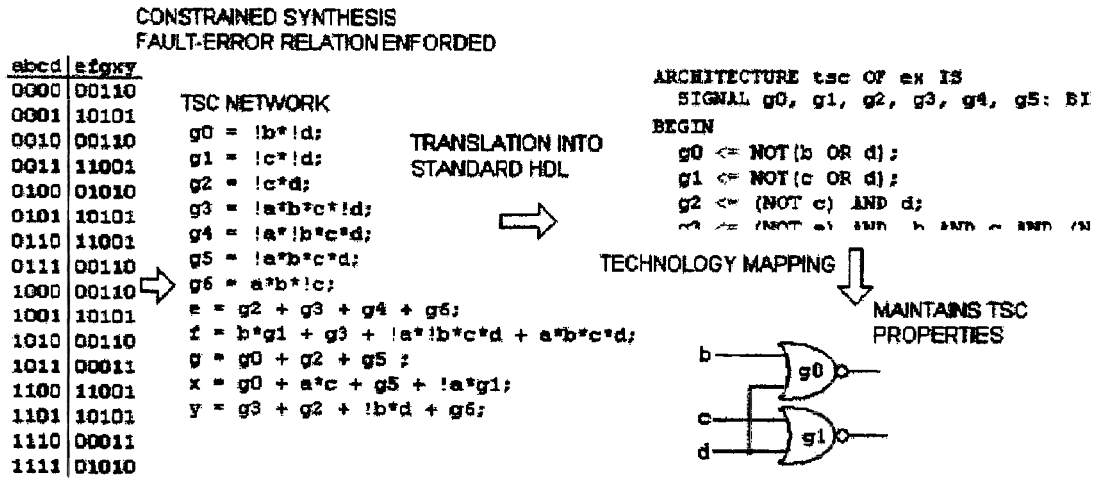
2.2 Adopted fault model

The goal of the proposed investigation is to explore the suitability of Concurrent Error Detection (CED) techniques based on Error Detection Codes on the FPGA platform. Given this premise, the attention has been initially devoted to the stuck-at faults and to single upset events (SEU) that may corrupt the internal memory of the LUTs, a fault model adopted in other studies also [7]. As far as the interconnection switch matrix is concerned, other approaches [3, 4] can be used to provide the desired fault coverage for such elements.

2.3 On-line fault detection through error detection codes

Previous efforts had been devoted to the exploitation of the fault-error relation determined by the adopted error detection code. The optimized and synthesized circuit was analyzed with a fault simulator for Self-Checking circuits detecting all those faults that caused undetectable errors with respect to the adopted code. The next step in the design methodology provided the local modification of the obtained network in order to guarantee observability also to all faults within the circuit. The Parity bit code, Berger and the m-out-of-n codes had been take into account for the design of self-checking circuits [13]. Such a methodology produces an optimal logic description with TSC properties that are preserved during a conventional library based technology mapping.

Consider as an example the simple example reported in Fig. 2, where a 4 inputs, 3 outputs network has been encoded with Berger code in order to achieve a complete fault coverage for any single stuck-at fault, except those on the primary inputs. Some traditional methodologies required a unate synthesis of the network to achieve the desired fault coverage, others imposed a less stringent constraint named "weighted observability" [13]. Similarly, when the Parity bit code is adopted, independent synthesis or odd-observability constraints are imposed in the synthesis process [13, 14, 15], to enforce a fault-error relation that guarantees that any fault, when causing an error, would produce only out-of-code words, detectable by a Totally Self-Checking Checker. All these methodologies provided, as a final output, a combinational network with a defined structural logic description of the device functionality to be implemented by commercial tools without any further modification that would modify the provided properties.



abcd	efgxy
0000	00110
0001	10101
0010	00110
0011	11001
0100	01010
0101	10101
0110	11001
0111	00110
1000	00110
1001	10101
1010	00110
1011	00011
1100	11001
1101	10101
1110	00011
1111	01010

```
g0 = !b*!d;
g1 = !c*!d;
g2 = !c*d;
g3 = !a*b*c*!d;
g4 = !a*!b*c*d;
g5 = !a*b*c*d;
g6 = a*b*!c;
e = g2 + g3 + g4 + g6;
f = b*g1 + g3 + !a*!b*c*d + a*b*c*d;
g = g0 + g2 + g5 ;
x = g0 + a*c + g5 + !a*g1;
y = g3 + g2 + !b*d + g6;
```

```
ARCHITECTURE tsc OF ex IS
  SIGNAL g0, g1, g2, g3, g4, g5: BI
BEGIN
  g0 <= NOT(b OR d);
  g1 <= NOT(c OR d);
  g2 <= (NOT c) AND d;
  g3 <= (NOT a) AND b AND c AND (N
```

Figure 2. Enforcement of CED techniques through Berger code for ASIC platforms.

This investigation explored the effects of the mapping of the TSC network specified onto the FPGA, more specifically the possibility to maintain the fault-error relation when CLBs are configured and LUT memory programmed.

3 The proposed methodology

The main issue in the application of a methodology targeted for ASIC platforms to FPGAs relates the basic blocks constituting the FPGA itself, i.e. the combinational logic blocks and the LUTs inside them. The success of the defined methodologies relates to the fact that the mapping of the HDL specification of the TSC network would maintain the same nodes specified by the description (data-flow style in the VHDL case) independently of the library gates used to implement them. Such a result could be achieved without additional area costs.

Let us analyze the effects of the faults belonging to the adopted model.

3.1 Faults effects

The adopted fault model takes into account the single stuck-at fault that may affect either one of the inputs or the output of the LUT. Consider the example node and LUT presented earlier in Fig. 1. Any fault on the LUT' inputs or on its output (Fig. 1c) has the same effect as a fault on any one of the inputs/output of the gates of the logic network (Fig. 1b). In fact, if the input pattern applied to lines a, b, c and d is such that the fault is exercised and an erroneous value is set on the LUT's output, there is no functional difference between the gate implementation and the LUT. If the logic equations describing the network functionality are TSC, the LUT implementation has the same property.

When considering *bit-flip* faults, a similar consideration can be drawn; a SEU causes the LUT to implement a different functionality with respect to the designed one. To better explore this situation, consider a bit-flip in the LUT memory, corresponding to the input configuration 0100, where instead of a 0 a 1 is stored.

Two cases may occur: 0100 is a test vector for at least one of the possible faults in the node, or not. Let analyse the two cases. 0100 *is* a test vector for at least one of the faults in the node (corresponding to the Boolean equation). The production of a 1 value on output e instead of the correct value 0 has the same effect of a stuck-at 1 on the node output, thus in a TSC network, this will lead to an out-of-code word, detected by the checker.

0100 *is not* a test vector; this means that for this configuration of lines a, b, c and d, the output is not observable on the circuit primary outputs, thus not modifying the fault-free output, i.e., producing the same correct codeword.

To summarize, both single stuck-at faults and single bit-flips faults are covered by the TSC properties guaranteed through an appropriate fault-error relation enforced at the logical level, where by means of customized tools it is possible to analyze the effects of each fault on the primary encoded outputs with respect to the adopted code [13].

3.2 The analysis and design methodology

Previous approaches benefit from the fact that it was possible to work at functional/logical level, by providing the necessary fault observability properties to each node constituting the functional description of the device under consideration. With ASIC, even if mapping with different technological libraries, commercial tools are able to maintain the functional description of each node constituting the network, thus producing the TSC device even if the used gates are not exactly the ones identified by the Boolean equations. With FPGA, nodes constituting the network are collapsed and merged to better suit the basic CLB elements constituting the FPGA resources in order to minimize the used area. The operation modifies each fault observability, thus potentially not fulfilling the required and previously provided fault-error relation, hence no assumptions can be made on the observability of each fault on the primary outputs, so that a subsequent TSC fault analysis and re-design steps are necessary.

Fig. 3 presents the proposed design flow adapted to cope with the different platform, providing means to carry out the fault analysis w.r.t. the adopted fault model, and a preliminary modification technique to achieve the complete fault coverage.

The circuit under consideration is initially designed to be Totally Self-Checking independently of the final platform used to implement it, according to the design methodologies presented in [11, 13]. The output is a dataflow stlye VHDL description of the resulting network characterized by a complete fault coverage with respect to the single stuck-at fault model (let us recall that this will cover also the SEU faults in the FPGA implementation).

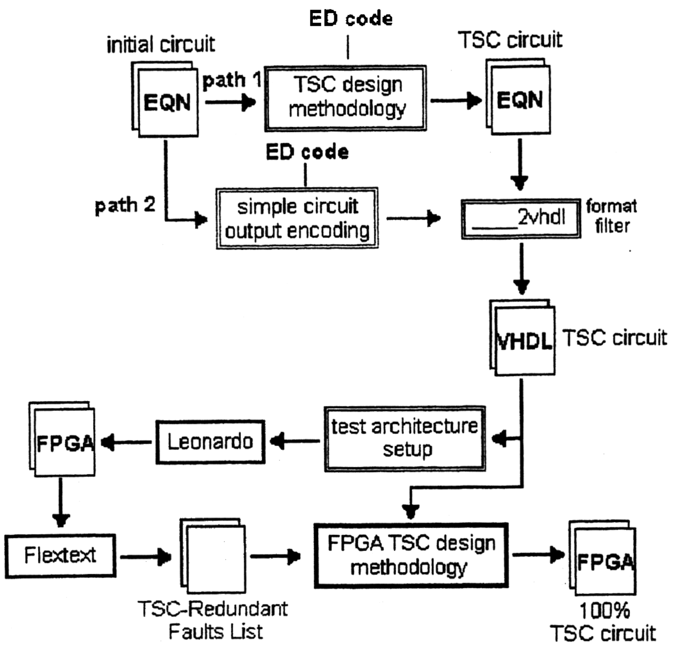


Figure 3. The design methodology for achieving TSC combinational networks with Error Detection Codes for FPGA platforms.

The description is then processed through a commercial design flow to produce the physical model for the FPGA. Fault simulation w.r.t. the TSC properties is necessary at this point, to identify those faults that do not produce detectable errors, due to the synthesis process.

The TSC fault analysis has been carried out with a traditional ATPG (Mentor Graphics' Flextest), provided a specific architecture is used to study the errors (detectable or not) produced by the faults in the fault set, with respect to the adopted ED code. Such a specific TSC-Test architecture is defined in VHDL and is reported in Fig. 4.

Once the fault analysis has been carried out identifying all faults that produce valid but erroneous codewords, for each fault a local modification of the circuit is performed to add another observability point for that fault so that the error is detected by the checker.

4 The experimental environment

Our design flow uses the Virtex^{TM}-II Family for the final platform for the implementation of the TSC network. A set of MCNC91 [16] has been taken into consideration for evaluating feasibility, efficiency and costs of the proposed approach when considering FPGA platforms. In order to analyze the fault-error relation affecting each possible fault in the designed network (expected to be Totally Self-Checking) a fault simulation architectural setup has been defined, so that a traditional ATPG, such as Mentor's Flextest, could be used even with TSC circuits. In fact, in the present situation, the task carried out by the ATPG is the identification of input vectors, if any, able to produce on the encoded circuit outputs errors not detectable by the applied checker circuit for the adopted error detecting code. Such errors are characterized by the fact that the produced output belongs to the adopted code but is not the one produced in a fault-free situation. A hierarchical architecture has been defined in VHDL, structurally instantiating two copies of the circuit under test (CUT), a checker for the selected ED code to monitor the outputs of the CUT, a simple comparator to monitor if the two copies of the CUT produce the same outputs. A final element summarizes the output of the checker (codeword or not) and the output of the comparator (erroneous or correct output) and allows the identification of those faults that are not covered by the TSC methodology.

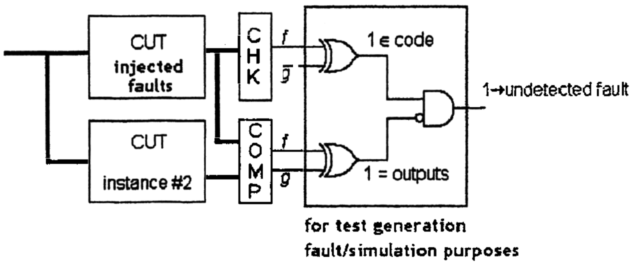


Figure 4. The architectural environment setup to use a standard VHDL tool set (FPGA synthesizer and test generator) to identify TSC redundant faults in networks encoded with Error Detecting Codes.

As a first step, all networks have been encoded with the Parity and Berger codes, directly synthesized with Mentor's Leonardo and mapped with the Xilinx' Design Project onto the Virtex^{TM}-II. The resulting area and achieved fault coverage achieved without applying any specific constrained synthesis, have been evaluated: on average all faults cause undetectable errors with respect to the applied code and are reported in Table 1.

Circuit	Code	I/O ports	LUTs	MUXFX	TSC Fault coverage
5xp1	Parity	7/11	18	6	100.00%
5xp1	Berger	7/14	76	32	62.50%
rd85	Parity	8/5	64	5	67.86%
rd85	Berger	8/7	45	6	100%
cu	Parity	14/12	23	4	60.24%
cu	Berger	14/15	32	5	89.58%
x2	Parity	10/8	20	4	97.22%
x2	Berger	10/10	27	14	95.44%

Table 1. MCNC91 benchmark circuits synthesized without any specific constraint to achieve the necessary fault-error relation to guarantee a complete fault coverage.

The second set of experiments has been carried out on the same encoded circuits, synthesized and fault analyzed at gate level to identify the faults not satisfying the TSC properties. A local modification is then carried out according to the methodology presented in [11] to achieve the complete fault coverage for an ASIC platform implementation. The obtained circuit, specified in a dataflow style VHDL is mapped onto an FPGA following the previous standard flow. Table 2 reports the result of synthesis and fault simulation.

The second set of experiments has been carried out on the modified and TSC network, used for the FPGA synthesis flow, using no optimization for the CUT so that only minor modifications are carried out while producing the final implementation. The proposed approach aims at locally re-designing those parts of the circuit which, following the tool synthesis, are not TSC. The synthesized circuit is fault simulated using the architecture shown in Fig. 4 to identify those points of the circuit that are not TSC.

Circuit	Code	LUTs	MUXFX	TSC Fault coverage
5xpl	Parity	35	9	100.00%
5xpl	Berger	141	55	60.00%
rd85	Parity	62	7	100.00%
rd85	Berger	55	4	91.17%
cu	Parity	27	4	61.11%
cu	Berger	36	4	100.00%
x2	Parity	27	3	100.00%
x2	Berger	27	14	100.00%

Table 2. Results of the circuits after mapping on the FPGA. The initial VHDL has a 100% fault coverage.

As it can be seen the achieved fault coverage is usually higher, yet not always complete. The reasons can be found in the bigger area (and consequently higher number of possible faults) and on the good performance, even with the FPGA optimization of those techniques that provide the desired fault-error relation by an independent synthesis of the circuit outputs. Eventually, the advantages of a preliminary TSC design of the network can be exploited by a minimal circuit re-design, with a reduced computational complexity.

The partial fault coverage is also caused by the FPGA nature; whereas in ASIC the only network logic is the one for implementing the desired functionality, in FPGA the unused LUTs and CLBs can affect the observability of faults, thus biasing the initial fault-error relation analyzed on the device functionality. This aspect will be further investagated in future research.

The local re-design methodology is an enhanced partial duplication approach which provides the duplication of the element affected by an undetected fault (LUT or multiplexter). If the element has a critical fanout w.r.t. the adopted code (for instance, an even fanout-degree in the case of the Parity bit code), the duplicated element is used to modify the fanout and achieve the necessary fault-error relation. If it is not possible to fix the fault observability, the duplicated element as a redundant element to be compared with the nominal one to detect mismatching output values caused by a fault. In this latter case a two-rail checker will also be used to merge the checker for the adopted code, and the additional observation points introduced by this post-synthesis partial duplication into a unified error detection signal.

5 Concluding remarks

This paper addresses the issue of Self-Checking FPGA design based on the adoption error detection codes (e.g., Berger code, Parity code, ...) as an evolution of the traditional approaches developed in the past years for the ASIC platform. We investigated the applicability of design techniques defined for introducing hardware fault detection properties in a combinational network through information redundancy at functional/gate level.

This approach is the starting point for the definition of a more complete methodology to dynamically reconfigure FPGAs in response to a fault, once it has been detected. Furthermore, we are presently adapting the original fault-error analysis tool to work on the circuit description produced by the Leonardo, so that the fault-error relation enforcement can be directly suited for the FPGA thus better controlling the effects of commercial tools' manipulations and the presence of unused logic.

References

[1] Xilinx Inc. "Virtex Series Configuration Architecture User Guide". version 1.5, Sept. 27, 2000.

[2] X. T. Chen, W. K. Huang, F. Lombardi, X. Sun. "A Row-Based FPGA For Single and Multiple Stuck-at Fault Detection". Proc. IEEE Int. Workshop on Defect and Fault Tolerance in VLSI Systems, DFT, 1995, pp.225-233.

[3] C. Stroud, M. Abramovici. "BIST-Based Test and Diagnosis of FPGA Logic Blocks". IEEE Transaction on VLSI Systems, Vol. 9, No. 1, Feb 2001, pp.159-172.

[4] I. Harris, R. Tessier. "Interconnect testing in cluster-based FPGA architectures ". Proc. IEEE/ACM 37th Conference on Design Automation, DAC, 2000, pp.49-54.

[5] J. Lach, W. H. Mangione-Smith, M. Potkonjak "Efficient Supporting Fault-Tolerance in FPGAs". Proc. ACM/SIGDA 6th Int. Symp. on Field Programmable Gate Arrays, 1998, pp.105-115.

[6] V. Lakamraju, R. Tessier. "Tolerating operational faults in cluster-based FPGAs". Proc. ACM/SIGDA international symposium on Field Programmable Gate Arrays, 2000, 187-194.

[7] N.R. Shnidman, W.H. Mangione-Smith, M. Potkonjak. "On-line fault detection for bus-based field programmable gate arrays". IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 6, no. 4 , 1998, pp.656-666.

[8] T. Bartzick, M. Henze, J. Kickler, and K. Woska. "Design of a Fault Tolerant FPGA". Proc. Field-Programmable Logic and Applications, The Roadmap to Reconfigurable Computing, 10th International Workshop, FPL 2000, Villach, Austria, 2000. Lecture Notes in Computer Science 1896 Springer 2000, pp.151-156.

[9] J. M. Emmert, D. K. Bhatia. "A fault tolerant technique for FPGAs". Journal of Electronic Testing: Theory and Applications, Kluwer Academics Publishers, 2000, Vol. 16, no. 6, pp.591-606.

[10] P. K. Lala, A. L. Burress. "A Technique for Designing Self-Checking Logic for FPGAs". Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS '99, Vol. 1, 1999, pp.94-96.

[11] C. Bolchini, F. Salice, D. Sciuto. "A synthesis methodology aimed at improving the quality of TSC devices". Proc. Int. Symp. on Defect and Fault Tolerance in VLSI Systems (1999), pp.247-255.

[12] Xilinx Inc. "Virtex^{TM}-II Platform FPGA Handbook". November 2001.

[13] C. Bolchini, F. Salice, D. Sciuto. "Fault analysis for networks with concurrent error detection". IEEE Design & Test of Computers , Volume: 15 Issue: 4 , Oct.-Dec. 1998, pp.66 -74.

[14] N. A. Touba, E. J. McCluskey. "Logic Synthesis Techniques for Reduced Area Implementation of Multilevel Circuits with Concurrent Error Detection". Proc. of Int. Conf. on Computer-Aided Design (ICCAD), pp. 651-654, 1994

[15] S. J. Piestrak. "Self-Checking Design in Eastern Europe" IEEE Design & Test of Computers, Spring 1996, pp. 16-25, Spring 1996

[16] S. Yang "Logic Synthesis and Optimization Benchmarks User Guide, Version 3.0". Technical Report, Microelectronics CEnter of North Carolina, 1991.