Fault Models Usability Study for On-line Tested FPGA

Jaroslav Borecký, Martin Kohlík, Pavel Kubalík and Hana Kubátová
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
Email: {borecjar; kohlimar; pavel.kubalik; kubatova}@fit.cvut.cz

***Abstract*—Field Programmable Gate Arrays (FPGAs) are susceptible to many environment effects that can cause soft errors (errors which can be corrected by the reconfiguration ability of the FPGA). Two different fault models are discussed and compared in this paper. The first one – Stuck-at model – is widely used in many applications and it is not limited to the FPGAs. The second one – Bit-flip model – can affect SRAM cells that are used to configure the internal routing of the FPGA and to set up the behavior of the Look-Up Tables (LUTs). The change of the LUT behavior is the only Bit-flip effect considered in this paper. A fault model analysis has been performed on small example designs in order to find the differences between the fault models. This paper discusses the relevance of using two types of models Stuck-at and Bit-flip with respect to the dependability characteristics Fault Security (FS) and Self-Testing (ST). The fault simulation using both fault models has been performed to verify the analysis results.**

***Index Terms*—FPGA, fault model, Bit-flip, Stuck-at, fault simulation**

I. Introduction

Systems realized by Field Programmable Gate Arrays (FPGAs) are more and more popular and widely used, even in mission critical applications such as aviation, medicine, space missions, and railway applications as well [1], [2], [3]. It is even possible to use FPGAs based on SRAMs, which are sensitive to Single Event Upsets (SEUs), but a good method of the error detection should be included into the design methodology. The Concurrent Error Detection (CED) techniques allow a faster detection of soft errors (errors which can be corrected by reconfiguration) caused by SEUs [4], [5], [6]. SEUs can change also the content of embedded memory, Look-up Tables (LUTs) and other configuration bits. These changes are not detectable by off-line tests, therefore CED techniques have to be used. The probability of a SEU occurrence in the SRAM is described in [7].

Our research results in this field were presented in [8]. A Self-Checking (SC) circuit (Figure 1) is one possible realization of the CED scheme. The SC circuit is typically composed of the original circuit, parity predictor and checker. Many papers have been published on this topic [9], [10]. In many publications the quality of the SC circuit is characterized by the number of detected faults.

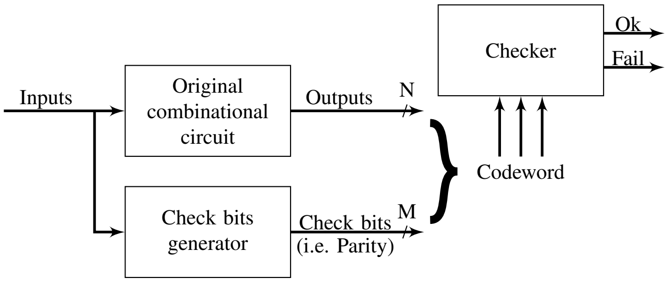

Fig. 1: Basic structure of Self-Checking circuit

But here could be a problem. What faults? Some fault model is always used, mostly the Stuck-at fault model. But it is possible to use it also in on-line testing methods? Can the type of a model affect the dependability parameters?

A fault model is an important thing to calculate the dependability parameters accurately. Wrong selection of an appropriated fault model can lead to unusable results. But even selection of the proper fault model is not enough to reach accurate results. We need tools written to obtain dependability parameters using selected fault model. In many publications the author used tools dedicated to an off-line testing. These tools are based on a Stuck-at fault model in a case, when a Bit-flip fault model must be used. Typical situations, where this mistake is performed are the FPGA based systems using same kind of an on-line testing method. The accurate description of the fault manifestation is needed to compare both fault models and their effect on a FPGA memory based structure.

The paper is organized as follows: basic terms concerning the classification of faults and the used fault models are presented in Section II. Fault model analysis is described in Section II-C. Experiments and their results is presented in Section III and Section IV concludes the paper.

II. Theoretical Background

A. Fault Security Calculation

Some redundancy has to be incorporated into the circuit design to improve dependability parameters. We have performed experiments with online testing to obtain CED [11].

There are three basic quantitative criteria in a field of CED used in this paper: Fault Security (FS), Self-Testing (ST) and Totally Self-Checking (TSC).

The following four fault classes can be used to calculate FS and ST parameters and to determine whether the circuit satisfies the TSC property. The possible faults are classified and separated into four classes, A, B, C and D according to their impact on the tested circuit design in the FPGA. The detailed description of the classes can be found in [12].

Class A – Hidden faults,
Class B – Detectable faults,
Class C – Undetectable faults,
Class D – Partially detectable faults.

B. Basic description and analysis

Here the fault manifestation is described for both fault models. In a Stuck-at fault mode, the fault can manifest either at primary inputs or at a primary output as the Stuck-at 0 or the Stuck-at 1. On the other hand, for the Bit-flip fault model, the fault manifests as a Bit-flip in LUT memory (see Figure 2). The main reason why we are interesting for this type of comparison at this design level is to obtain real dependability parameters easily and faster than method using hardware emulation described in [13], [14].

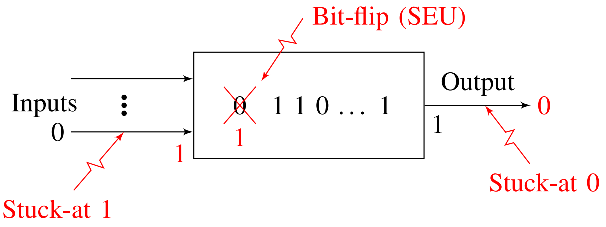

Fig. 2: Stuck-at and Bit-flip faults in LUT

The Stuck-at fault model describes an input or an output value change caused by a short with another wire with a constant logic value. The typical value for the Stuck-at fault model is ”0” for a low voltage level and ”1” for a high voltage level. The Bit-flip fault model represents a change in a memory caused by a SEU. The typical value for the Bit-flip fault model is opposite value of one memory bit represent as the logic zero (”0”) or the logic one (”1”).

The following section compares these two models. Firstly we try to say, whether both models are convertible to each other. We analyze both models and we determine the number of testing places and the number of tests needed to calculate dependability parameters. In our calculation, we use n-input LUT with one primary output, so 2^n bits are required to define the behavior of the LUT.

One test vector is needed to test one memory bit in the case of the Bit-flip fault model, so 2^n test vectors must be used to test the whole LUT memory.

For the Stuck-at 0 at the primary output we need one test vector. Also for Stuck-at 1 at the primary output we need only one test vector. For the primary inputs we need n test vectors for Stuck-at 1 and n test vectors for Stuck-at 0. Finally, for one n-input LUT $2 * n$ test vectors have to be performed to cover and test $2 * (n + 1)$ possible faults.

It can be summarized that the information obtained from this basic analysis is not sufficient to decide, if these two models are convertible to each other. Each fault model has different number of faults and also different number of test vectors needed. The Bit-flip fault model has more possible places where a fault can be manifested. According this analysis we can say, that the Stuck-at fault model can be used to simulate a single event transient (SET) manifesting as a Bit-flip in data-path captured by registers. The analysis has to be performed in more detail to say if and when the both fault models are convertible to each other.

C. Accurate fault model analysis

Stuck-at fault at the output of the LUT has the same effect as setting all memory bits of the LUT to the stuck value. Stuck-at 0 covers all memory bits containing logic 1 (they can be flipped to logic 0) and vice versa. Both fault models has the same coverage in this regard, but the results of Bit-flips have to be grouped in order to get the result of the Stuck-at. This grouping causes the main difference between fault models results.

The difference between fault models results is formed when the fault classes of all faults are not equal. The example containing LUT configured as logic AND can be used to show this difference. The example can be designed to produce the fault class distribution shown in Table I. The results in the lower part of the table show that the FS parameters obtained from the example may differ widely. A similar example using logic OR can be constructed. Such example has FS parameter equal to 25% when Bit-flip model is used.

TABLE I: Results showing fault models results differences.

			Bit-flip – position				**Stuck-at – net X**	
a	**b**	**No fault**	**0**	**1**	**2**	**3**	**0**	**1**
0	0	0	1	0	0	0	0	1
0	1	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	1
1	1	1	1	1	1	0	0	1
Fault class			B	B	B	C	C	B
FS			$\frac{3}{3+1} = 75\%$				$\frac{1}{1+1} = 50\%$	

TABLE II: Fault Classification

Fault class	**Detectable**	**Incorrect**
A	false	false
B	true	false
C	false	true
D	true	true

The fault classes of grouped Bit-flips may be different, too. The fault class of Stuck-at corresponding to these

TABLE III: Results showing combining multiple Bit-flips into single output Stuck-at.

		No fault				**Bit-flip – position 0**				**Bit-flip – position 2**				**Stuck-at 0 – net X**			
a	**b**	**X**	**Out 0**	**Out 1**	**Parity**	**X**	**Out 0**	**Out 1**	**Parity**	**X**	**Out 0**	**Out 1**	**Parity**	**X**	**Out 0**	**Out 1**	**Parity**
0	0	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fault class						B				C				D			

grouped Bit-flips can be determined from the ”Detectable” and ”Incorrect” parameters of the fault classes (see Table II). Logic OR operator is applied on the attributes of each Bit-flip.

The example design shown in Figure 3 is used to illustrate the differences between fault models. A real synthesis will not provide such design; it is created manually for this experiment only. The design contains an additional LUT that forms the Parity output. The faults are injected inside the marked LUT or at its inputs and the output.

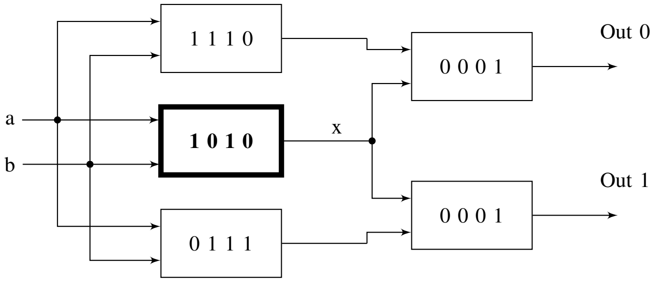

Fig. 3: Example showing joining multiple Bit-flips into single output Stuck-at. (Parity LUT not displayed)

The behavior of the example design is shown in Table III. The table contains values of all outputs and the internal net X (the output of marked LUT) for a non-faulty state and three states with faults injected. The behavior of the design with the Bit-flip fault injected at the position 0 in a marked LUT is shown in the third column. Only one output is faulty, therefore parity check fails and the fault is classified as the B class fault. The behavior of the design with the Bit-flip fault injected at position 2 is shown in the fourth column. Two outputs are faulty, therefore parity check passes, but the output word is incorrect. The fault is classified as the C class fault. The last column shows the behavior of the design with the Stuck-at fault injected at the net X. This fault can be detected by a parity check in the case of input word ”00”, but the output word is incorrect and the parity check passes in the case of input word ”10”. This fault is classified as the D class fault.

The combination from the example design (B-C) can be made as follows:

- Detectable – true or false = true
- Incorrect – false or true = true

Result is true-true, which corresponds to the class D.

The behavior of a Stuck-at fault at the input of a LUT can be obtained similarly. Another example containing LUT3 configured as shown in table IV can be used. The table also contains the behavior of the output of this LUT with the Stuck-at faults injected at the inputs.

TABLE IV: Results showing incomplete coverage of Bit-flip faults by Stuck-at faults. (Marked lines are not covered by any Stuck-at fault.)

				Stuck-at					
				a		**b**		**c**	
a	**b**	**c**	**No fault**	**0**	**1**	**0**	**1**	**0**	**1**
0	0	0	1	1	0	1	1	1	0
0	0	1	0	0	1	0	1	1	0
0	1	0	1	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1
1	0	0	0	1	0	0	1	0	1
1	0	1	1	0	1	1	1	0	1
1	1	0	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1

As you can see, Stuck-at 0 at input ”a” flips the output value in the cases of input word ”100” and ”101”, therefore it can be combined from the Bit-flip faults at positions 4 and 5. The other Stuck-at faults can be combined similarly. As you can see, the cases of input word ”010” and ”111” are not covered by any Stuck-at fault.

III. Experimental Results

A. Description of all processes

This subsection describes the whole simulation process. The standard MCNC benchmarks described in PLA format were used in our experiments. A predictor is created as the parity bit generator from the benchmark logic and it is also saved in the PLA format. The next outputs in this step are two ”.tst” files containing the test vectors to check the parity. Each ”.tst” file corresponds to one fault insertion method. The original combinational logic and the predictor are minimized separately by ESPRESSO [15] and then BOOM [16] is used to translate them to VHDL. Next the Synplicity Synplify [17] synthesis is performed for both VHDL files and is executed separately to disable resource sharing. Synthesized logics are stored in the Electronic Design Interchange Format (EDIF) format. The original and the predictor circuits are joined to ”Top” design at EDIF level to ensure the independence of both parts. This step is performed by our simulation and EDIF manipulation utility.

TABLE V: Fault Security and Self-Testing parameters results for two different fault models

		FS(%)		ST(%)	
Name	Size(LUTs)	SA[1]	BF[2]	SA[1]	BF[2]
alu1	53	97.43	100.0	100.0	100.0
apla	59	73.98	73.64	96.43	95.96
br1	66	62.46	58.74	87.54	82.7
br2	38	57.39	52.72	81.77	75.74
b12	41	90.16	96.58	100.0	100.0
dk17	43	80.28	84.0	100.0	100.0
dk27	19	94.34	97.87	100.0	100.0
dk48	59	82.78	87.3	99.34	99.65
ex1010	1413	88.24	90.31	100.0	100.0
f51m	39	93.95	98.42	100.0	100.0
gary	227	83.4	86.55	99.03	98.92
mp2d	51	85.14	87.99	94.02	93.65
m1	27	88.08	93.67	97.68	98.1
newapla	24	73.02	81.03	98.92	97.41
newbyte	10	95.16	100.0	95.16	100.0
newcpla1	63	73.31	77.46	92.92	92.02
newcpla2	34	72.22	74.35	88.06	84.29
p82	34	84.55	88.89	95.79	95.16
sex	27	86.34	91.3	98.14	98.76
sqr6	49	86.64	92.13	98.58	98.46

[1] Stuck-at
[2] Bit-flip

After joining, the ”Top” design is tested with two test vectors files. Statistics containing numbers of faults belonging to classes ”A”, ”B”, ”C” and ”D” are generated. FS and ST properties are calculated for each fault model.

B. *Experiments Results*

Experimental results are taken from two Test runs with two different test vectors files. Each standard benchmark is designed as a self-checking circuit and tested with two fault models. All measured parameters are shown in Table V.

The table has the following structure: The first column contains the name of the used benchmark. The second column contains the FS property and third column contains the ST property of the design. These two columns are divided into two sub-columns according to each fault model. The fault models are as follows:

SA – Stuck-at
BF – Bit-flip

The table shows significant differences of FS and ST parameters among different fault models.

IV. Conclusion

Theoretical analysis presented in this paper shows that to count exactly the difference between Stuck-at fault model and Bit-flip fault model with respect to dependability parameters values is impossible. It means that any conversions between both fault models cannot be done even in a case, when some estimation of dependability parameters is useful when we need to compare two methods of a design of self-checking circuits. The Stuck-at fault model covers all faults in Bit-flip fault model and also faults at primary inputs and primary outputs of the design but the Stuck-at fault model cannot be used to obtain real dependability parameters for a Bit-flip fault model. The experimental results obtained by applying both fault models to the same benchmarks show, that for some benchmarks the FS parameter is higher for Stuck-at fault model then for Bit-flip fault model and vice versa. This should be a proof, that these two fault models are not convertible to each other, but the difference is smaller than 5% for most of the tested benchmarks. Therefore, the final conclusion should be the following: both models have the same coverage, but the Fault Security and Self Testing parameters may differ.

Acknowledgment

This research has been partially supported by the projects MSM6840770014, GA102/09/1668 and SGS10/118/OHK3/1T/18.

References

[1] R. Dobiáš and H. Kubátová, “FPGA Based Design of Railway's Interlocking Equipment,” *In Proceedings of EUROMICRO Symposium on Digital System Design. Piscataway: IEEE*, pp. 467–473, 2004.

[2] D. Ratter, “FPGAs on Mars.” www.xilinx.com, 2004.

[3] A. Corporation.
http://www.actel.com/documents/FirmErrorPIB.pdf, 2007.

[4] L. Sterpone and M. Violante, “A design flow for protecting FPGA-based systems against single event upsets,” *DFT2005, 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 436–444, 2005.

[5] Q. Corporation., “Single Event Upsets in FPGAs.” www.quicklogic.com, 2003.

[6] M. Bellato, P. Bernardi, D. Bortalato, A. Candelaro, M. Ceschia, A. Paccagnella, M. Rebaudego, M. S. Reorda, M. Violante, and P. Zambolin, “Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA.,” *Design Automation Event for Electronic System in Europe 2004*, pp. 584–589, 2004.

[7] E. Normand, “Single Event Upset at Ground Level,” *IEEE Transactions on Nuclear Science*, vol. 43, pp. 2742–2750, 1996.

[8] P. Kubalík and H. Kubátová, “Dependable Design Technique for System-on-Chip,” *Journal of Systems Architecture. 2008*, vol. 2008, no. 54, pp. 452–464, 2008. ISSN 1383-7621.

[9] S. J. Piestrak, *Design of Self-Testing Checkers for m-out-of-n Codes Using Parallel Counters*. London: Kluwer Academic Publisher, 1998.

[10] D. Nikolos, *Self-Testing Embedded Two-Rail Checkers*. London: Kluwer Academic Publisher, 1998.

[11] D. K. Pradhan, “Fault-Tolerant Computer System Design,” *Prentice-Hall, Inc.*, 1996.

[12] P. Kubalík, P. Fišer, and H. Kubátová, “Fault Tolerant System Design Method Based on Self-Checking Circuits,” *Proc. 12th International On-Line Testing Symposium 2006 (IOLTS'06), Lake of Como, Italy*, 2006.

[13] J. Kvasnička, P. Kubalík, and H. Kubátová, “Experimental SEU Impact on Digital Design Implemented in FPGAs,” *In Proceedings of 11th Euromicro Conference on Digital System Design.*, pp. 100–103, 2008.

[14] M. Bellato, P. Bernardi, D. Bortalato, A. Candelori, M. Ceschia, A. Paccagnella, M. Rebaudego, M. S. Reorda, M. Violante, and P. Zambolin, “Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs,” *IEEE Transactions on Nuclear Science*, vol. 50, pp. 2088–2094, 2003.

[15] R. K. Brayton and et al., *Logic Minimization Algorithms for VLSI Synthesis*. Boston: Kluwer Academic Publisher, 1984.

[16] P. Fišer and J. Hlavička, “BOOM - A Heuristic Boolean Minimizer,” *Computers and Informatics*, vol. 22, no. 1, pp. 19–51, 2003.

[17] Synopsys, Inc.
http://www.synopsys.com/, 2007.