Fault Injection and Simulation for Fault Tolerant Reconfigurable Duplex System

Pavel Kubalík, Jiří Kvasnička, Hana Kubátová
Department of Computer Science and Engineering
Czech Technical University in Prague
Karlovo nam. 13, 121 35 Prague 2
e-mail: (xkubalik, kvasnj1, kubatova)@fel.cvut.cz

Abstract – **The implementation and the fault simulation technique for the highly reliable digital design using two FPGAs under a processor control is presented. Two FPGAs are used for duplex system design, each including the combination of totally self-checking blocks based on parity predictors to obtain better dependability parameters. Combinatorial circuit benchmarks have been considered in all our experiments and computations. A Totally Self-Checking analysis of duplex system is supported by experimental results from our proposed FPGA fault simulator, where SEU-fault resistance is observed. Our proposed hardware fault simulator is compared also with the software simulation. An area overhead of individual parts implemented in each FPGA is also discussed.**

I. Introduction

FPGAs are based on SRAM memories sensitive to Single Even Upsets [1, 2, 3] (SEUs), therefore simple usage of FPGA circuits in mission critical applications without any method of SEUs detection is practically impossible. SEUs can change the content of the embedded memory or Look-up Tables (LUTs) used in the design. These changes are not detectable by off-line tests; therefore appropriate CED techniques have to be used. The probability of a SEU occurrence in the random access memory (RAM) is described in [4].

The term dependability is used to encapsulate the concepts of reliability, availability, safety, maintainability, performability, and testability [5]. Availability computation to compare modified duplex system (MDS) with standard duplex system is described in [6].

This paper comprises partial research results based on software and hardware simulation experiments presented in [8]. Some techniques of software simulation and hardware emulation are described in [9].

The TMR structure is unsuitable when a high area overhead is unacceptable. Some hybrid architecture must be used. TMR architecture and a hybrid system, e.g., the modified duplex system with a comparator and some CED technique are compared in [10, 12]. A technique based on Duplication With Comparison (DWC) and the Concurrent Error Detection (CED) technique are described in [11, 13].

The structure of the paper is following: the basic classification of faults is presented in Section 2. A brief overview of our duplex system is shown in Section 3. Our proposed hardware emulator structure is presented Section 4. An implementation of this emulator is described in Section 5. Section 6 summarizes results obtained by both hardware and software fault simulations and Section 7 concludes the paper.

II. Faults Classification

There are three basic quantitative criteria in a CED field: Fault Security (FS), Self Testing (ST) and Totally Self-Checking (TSC) properties. To determine whether the circuit satisfies the TSC property, faults have to be selected to one of four classes A, B, C or D according [8]. Classification of each fault into one of previous 4 classes can be used to decide, whether or how much the circuit satisfies the FS, ST and TSC properties.

III. MDS Architecture

Our previous results [7] show, that satisfying fully TSC property with low area overhead is difficult, so we proposed a new structure based on two FPGAs, as shown on Fig. 1.

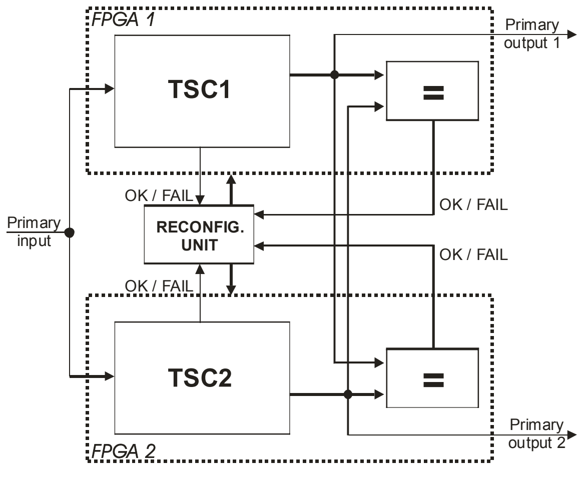

Fig. 1. MDS architecture

The probability of the information correctness depends on the FS property. When the FS property is satisfied only to 75%, the correctness of the checking information is also 75%. It means that the signal "OK" give a correct information for

75% of occurred errors (the same probabilities for both signals “OK” and “FAIL”).

Two Comaparators have to be added to increase the dependability parameters, one for each FPGA. The comparator compares outputs of both FPGAs. The fail signal is generated when the output values are different. This information is not sufficient to determine, which TSC circuit is wrong. Additional information to mark out the wrong circuit is generated by the original TSC circuit. In a case when outputs are different and one of the TSC circuits signalizes fail function, the wrong FPGA is correctly recognized. The reconfiguration process is initiated after a fault is detected. MDS architecture is described in greater detail in [6].

IV. OUR HARDWARE EMULATOR

The hardware emulator was used to emulate SEU faults in FPGAs. These faults are converted into bitstream faults in the tested circuit. Moreover, the fault emulation in the FPGA gives us possibility to test bridging faults and opens in the interconnection network. More accurate TSC parameters can be obtained from fault injection into implemented design.

The hardware emulator presents a collection of the software tools and Atmel AT40K testing board. The software tools are used to convert benchmarks from “.pla” format to “.vhdl” format. These software tools also allow the self-checking modification of the original circuit. The synthesis and the mapping process are fulfilled manually by Atmel FPSLIC design tool. The final bitstream is put into Atmel FPGA. In our software tool the complete bitstream is analyzed and areas concerning the fault injection selected. The selection has to be done in software due to limited capacity of SRAM in AVR, which can not hold the whole bitstream.

V. IMPLEMENTATION OF BENCHMARKS TESTING IN FPGA

Our hardware emulator (see Fig. 2) is structurally based on the schematic diagram of the fault classification presented in [8].

The hardware part of emulator is based on Atmel FPSLIC development board. It is divided into AVR part and FPGA part, see Fig. 2. The AVR controls the test process and reconfiguration. The partial bitstream concerning faulty areas is loaded into the SRAM, which is shared between AVR and FPGA. This bitstream is further analyzed with respect to the real occupation of LUTs in AVR for its reconfiguration use in FPGA. A correct bitmask for the test injection is obtained by the analysis of bitstream/LUT use. This analysis is performed in AVR, because it needs less computation power than the transfer of analysis results (considering the time of testing).

The fault classification result can be obtained separately for each fault or distribution of all faults into 4 categories (see section 2) can be obtained as a result. The exhaustive test must be processed for an on-line test. The testing vectors are generated automatically and only a set of tested faults are loaded.

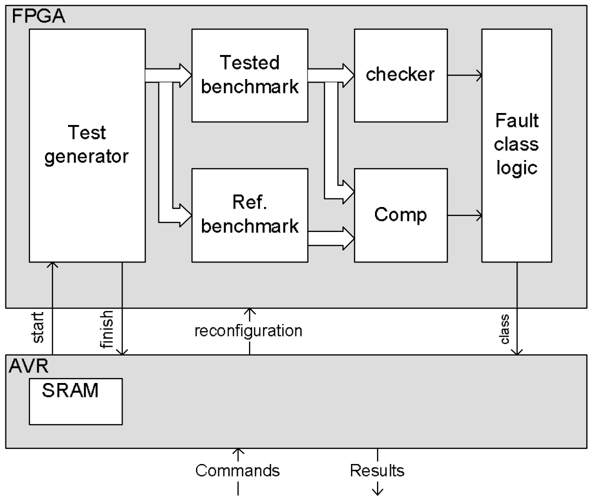

Fig. 2. Basic hardware emulation diagram

The reconfiguration process is performed during the idle phase of testing (when the previous exhaustive test is finished and the new one hasn’t begin) to ensure no testing during the reconfiguration process. The reconfiguration process begins by putting the FPGA into the correct state (restoring the bitstream from the previous fault) and than continues by uploading a new fault into the bitstream.

The structure of our emulator in FPGA (see Fig. 3) needs the additional registers to ensure a maximal clock frequency.

Only one-bit counters are necessary to decide the category which the injected fault belongs into, therefore they occupy only one logic cell in the FPGA structure.

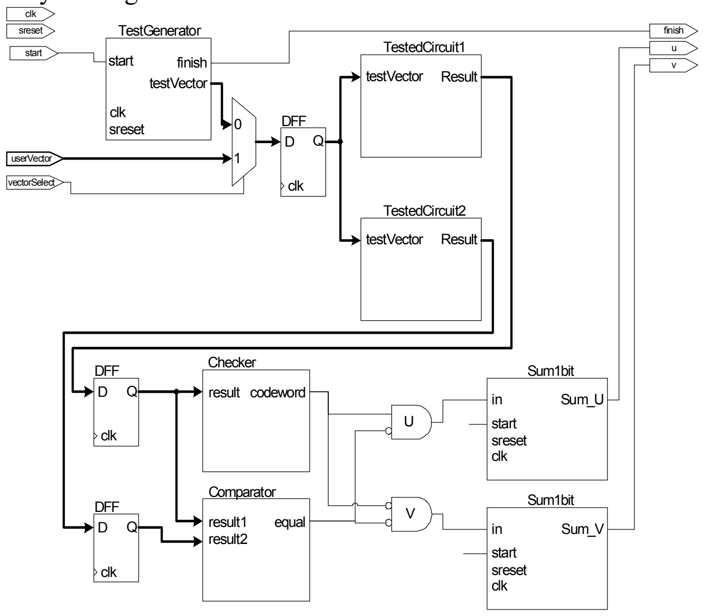

Fig. 3. Structure of benchmark test

Parity checkers and comparators are used both in the hardware emulator (non-TSC versions) and in a dependable system based on MDS architecture. Therefore special attention to them will be paid. The area occupied by n-long vector in FPGA with p-input can be expressed by the formula:

$$M = \left\lceil \frac{n-1}{p-1} \right\rceil \qquad (1)$$

, where M is the number of LUTs used.

The solution presented above is fulfilled for even parity checker realizing only “CheckOK” signal. The odd parity checker must be used to generate “CheckFAIL” signal. These signals together form the TSC version of a checker, under the condition that they don’t share any logic between them. The occupied area of the even and odd parity is equivalent. The number of LUTs M used in AT40K for even parity checker is:

$$M = \left\lceil \frac{n-1}{3} \right\rceil \qquad (2)$$

, where M is the number of LUTs and n is the number of inputs.

The comparator is used to analyze whether the primary output is correct. In the final solution, the comparator is used only for outputs leaving an FPGA. The comparator can be divided into 2 phases: a) comparing pairs of input (producing sub-equals $se_i = \overline{r_i \oplus s_i}$) and b) final collecting of sub-equals (into equal $eq = \prod_i se_i$).

The calculation of the area occupied by a comparator must consider the architecture of a logic cell, especially the number of inputs to LUT. It is hard to map sub-equals to a LUT with the odd number of inputs. Only even-input LUTs are optimally used for sub-equal functions. The area occupied by the checker and the comparator in our hardware emulator is only half-size, because non-TSC versions are present. The problem of AND-tree of sub-equal size enumerating is similar to a parity tree, therefore calculation will be omitted.

The number of LUTs M used for the comparator in worse case is:

$$M = \left\lceil \frac{n}{2} \right\rceil + \left\lceil \frac{\left\lceil \frac{n}{2} \right\rceil - 1}{3} \right\rceil \qquad (3)$$

, where M is the number of 4-input LUTs and n is the number of inputs. The size of checker and comparator is not dramatically high.

The test is divided into two parts. These parts are composed from a safe test set and a risk test set. Our hardware simulator can test only a part of the safe test set composed mainly from look-up-table tests. This test set covers only 11 percents of the AT40K bitstream size. The risk test set is composed of interconnection tests and can cause shorts. To the risk test set consist of the interconnection between cells (23 percents), the interconnection inside cells (36 percents) and 30 percent belong to others configuration bits (for example clock distribution, input/output cell and RAMs).

VI. Simulation results

The simulation design methodology was used to generate bitstream for many MCNC benchmarks. The benchmark’s bitstream was loaded into Atmel FPGA. The faults were injected only into used parts of LUTs. The fault injection into unused logic would increase only number of undetected faults. The unused logic is generated in a case when less than 4-inputs LUTs are used. The hidden faults can obviously occur in the unused logic, therefore it is not possible to detect them by any way. Results of hardware fault emulation are shown in Table I. Here “Circuit” is benchmark name, “Inputs” and “Outputs” are numbers of primary inputs and primary outputs, “Original circuit” means a number of used LUTs for original circuit, “Parity generator” means a number of used LUTs for the parity generator, “Number of all faults” are all tested faults and “A, B, C, D” are classes derived by our fault classification.

TABLE I
Result of fault simulation - classes

Circuit	Inputs	Outputs	Original circuit [LUTs]	Parity generator [LUTs]	Number of all faults	A (hidden faults)	B (detected faults)	C (undetected faults)	D (temporary detected)
alu1	12	8	8	47	656	0	656	0	0
alu2	10	8	44	47	1072	109	935	0	28
Apla	10	12	48	25	900	141	625	5	129
br1	12	8	50	15	810	141	456	69	144
s1488	14	25	310	50	4286	638	3060	85	503
s1494	14	25	276	53	3938	645	2785	67	441
s2081	18	9	22	25	536	22	494	0	20
s386	13	13	57	18	976	170	646	25	135

Results of ST, FS properties and area occupation are shown in Table II., Here “Circuit” is benchmark name, “Original circuit” is number of used LUTs for original circuit, “Parity generator” is number of used LUTs for parity generator, “Area overhead” is area needed for parity generator in percentage, “TSC checker” and “TSC comparator” are numbers of used LUTs for TSC checker and output TSC comparator, “ST coverage, FS coverage” are self-testing and fault secure properties in hardware emulation, “FS coverage in SW” is fault secure property in software simulation and “Test time” is time needed for full test execution in hardware emulation.

The resulting FS property of tested circuit is higher than 70% for all benchmarks, even better on average. The area overhead depends on tested benchmarks. For 50% of benchmarks the area overhead is less than 50%. The benchmarks “alu” are typical problem for single parity generator.

TABLE II
ST AND FS PROPERTIES RESULTS

Circuit	Original circuit [LUTs]	Parity generator [LUTs]	Area overhead [%]	TSC checker[LUT]	TSC comparator[LUTs]	ST coverage[%]	FS coverage [%]	FS coverage in SW [%]
alu1	8	47	588	6	14	100.0	100.0	100
alu2	44	47	107	6	14	89.83	97.4	92
apla	48	25	52.1	8	16	83.78	85.1	83
br1	50	15	30.0	6	10	74.07	73.7	63
s1488	310	50	16.1	16	34	83.13	86.3	86
s1494	276	53	19.2	16	34	81.92	87.1	86
s2081	22	25	114	6	16	95.90	96.3	96
s386	57	18	31.6	8	18	80.02	83.6	71

The software simulator is slower than the hardware emulator. Results of consumed time are described in Table III. Here "Circuit" is benchmark name, "Inputs" are numbers of primary inputs and primary outputs, "Software simulation" is time needed for full test execution in software simulation, "Hardware emulation" is time needed for a full test execution in the hardware emulation, "SW/HW time rate" is the speedup factor between the hardware and software simulation time.

TABLE III
SOFTWARE/HARDWARE SIMULATION TIME

Circuit	Inputs	SW simulation [s]	HW emulation [s]	SW/HW time rate
alu1	12	34.0	0.92	37.0
alu2	10	9.0	0.41	22.0
apla	10	6.0	0.34	17.6
br1	12	18.0	1.10	16.4
s1488	14	2406.3	23.82	101.0
s1494	14	2518.9	21.84	115.3
s2081	18	1217.9	49.30	24.7
S27	7	0.1	0.03	3.3
s386	13	677.7	2.49	272.2

The time for reconfiguration is negligible in comparison with exhaustive test of circuit. All hardware simulation time cover testing time, reconfiguration time, reconfiguration preparation time and AVR overhead. The synthesis time is not included in results.

VII. CONCLUSION

The hardware fault emulator for our MDS architecture based on two FPGAs has been presented. In this emulator we are able to calculate FS, ST and TSC parameters of tested circuit physically mapped in the FPGA

Experimental results of several benchmarks show consistency between the software fault simulation results and hardware fault emulation results, see Table II.

Our future work will be dedicated to several practical case studies (e.g., railway applications). We will focus on the fault list creation, which would make shorts and opens testing possible, either based on the fault injection into the interconnection of the FPGA (the risk test set) or based on the transformation of the risk test set into the safe LUT testing.

ACKNOWLEDGMENTS

This research has been supported in part by the MSM6840770014 research program.

REFERENCES

[1] QuickLogic Corporation.: Single Event Upsets in FPGAs, 2003, www.quicklogic.com

[2] Bellato, M., Bernardi, P., Bortalato, D., Candelaro, A., Ceschia, M., Paccagnella, A., Rebaudego, M., Sonza Reorda, M., Violante, M., Zambolin, P.: "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.

[3] Sterpone, L., Violante, M.: "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.

[4] Normand, E.: "Single Event Upset at Ground Level," IEEE Transactions on Nuclear Science, vol. 43, 1996, pp. 2742-2750.

[5] Pradhan, D. K., Fault-Tolerant Computer System Design, Prentice-Hall, Inc., New Jersey, 1996.

[6] Kubalik, P., Dobias, R., Kubatova, H.: "Dependable Design for FPGA based on Duplex System and Reconfiguration", In Proceedings of 9th Euromicro Conference on Digital System Design, Los Alamitos: IEEE Computer Society, 2006, pp. 139-145.

[7] Kubalík, P., Fiser, P., Kubátová, H.: "Minimization of the Hamming Code Generator in Self Checking Circuits", Proceedings of the International Workshop on Discrete-Event System Design - DESDes'04. Zielona Gora: University of Zielona Gora, 2004, pp. 161-166.

[8] Kafka L., Kubalík P., Kubátová H., Novák O.: "Fault Classification for Self-checking Circuits Implemented in FPGA", Proceedings of IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop. Sopron University of Western Hungary, 2005, pp. 228-231.

[9] Kafka, L., Novak, O.: "FPGA-based fault simulator", In Proceedings of the 2006 IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems DDECS2006, CTU Prague , vol. 1, pp. 274-278.

[10] Kastensmidt, de L., G., F., Neuberger, G., Hentschke, F., R., Carro, L., Reis, R.: "Designing Fault-Tolerant Techniques for SRAM-Based FPGAs", IEEE Design and Test of Computers ,vol. 21, no. 6, pp. 552-562, November/December, 2004.

[11] Lima, F., Carro, L., Reis, R.: "Designing Fault Tolerant Systems into SRAM-based FPGAs" In Proceedings of the 40th Design Automation Conference, DAC'03, pp. 650, June 2003.

[12] Yu, S.-Y., McCluskey, E., J.: "Permanent Fault Repair for FPGAs with Limited Redundant Area", In Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT01, pp. 125, 2001.

[13] Mitra, S., Huang, W.-J., Saxena, R., N., Yu, S.-Y., McCluskey, J., E.: "Reconfigurable Architecture for Autonomous Self-Repair", IEEE Design and Test of Computers, pp. 228-240, May 2004.