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

Systems realized by FPGAs are more and more popular due to several properties and advantages:

- High flexibility in achieving multiple requirements such as cost, performance, turnaround time.
- Possible reconfiguration and later changes of the implemented circuit e.g. only via radio net connections.
- Mission critical applications such as aviation, medicine, space missions or also in railway applications [1].

FPGAs are based on SRAM memories sensitive to Single Even Upsets [2, 3] (SEUs), therefore simple usage of FPGA circuits in mission critical applications without any method of SEUs detection is practically impossible.

One change of a bit in the configuration memory by SEUs leads to a change of a circuit function, even drastically. The CED techniques allow a faster detection of soft errors (errors which can be corrected by the reconfiguration) caused by Single Event Upsets (SEU) [2, 3, 4]. 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 [5].

The possibilities how to keep proper system functions are based always on some redundancy. Redundancy obviously means a great area and/or time overhead. Our proposed structure increases dependability parameters together with ensuring a relatively low area overhead as compared with classical methods such as duplication or triplication [6].

The term dependability is used to encapsulate the concepts of reliability, availability, safety, maintainability, performability, and testability [7]. Availability is a function of time, A(t), defined as the probability in time that a system is operating correctly and is available to perform its function [7]. Availability computation to compare modified duplex system with standard duplex system is described in [8].

Our solution combines on-line testing design methods with the classical duplex design [6, 8]. It assumes the dynamic reconfiguration of the faulty part of the system after on-line fault detection. The most important criterion is the speed of the fault detection and the safety of the whole circuit with respect to the application requirement.

Our previous research shows the relation between the area overhead and the SEUs fault coverage [9]. To ensure a small area overhead, the SEUs fault coverage for most circuits is less than 100%. The SEUs fault coverage varies typically from 75% to 95%.

This paper comprises partial research results based on software and hardware simulation experiments presented in [10, 11, 12]. Some techniques of software simulation and hardware emulation are described in [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 clasification

There are three basic quantitative criteria in a CED field: Fault Security (FS), Self Testing (ST) and Totally Self-Checking (TSC) properties. These three aspects are used in an on-line testing field to evaluate the level of safety of the designed or modeled system.

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 [10].

- Class A – hidden faults. These are faults that do not affect the circuit output for any allowed input vector. Faults belonging to this class have no impact to the FS property, but if this fault can occur, a circuit cannot be self-testing (ST).
- Class B – faults detectable by at least one input vector and they do not produce an incorrect codeword (valid code word, but incorrect one) for other input vectors. These faults have no negative impact to the fault-secure (FS) and ST property.
- Class C – faults that cause an incorrect codeword for at least one input vector and they are not detectable by any other input vector. Faults from this class cause undetectable errors. If any fault in the circuit belongs to this class, the circuit is neither FS, nor ST.
- Class D – faults that cause an undetectable error for at least one vector and a detectable error for at least one another vector. Although these faults are detectable, they do not satisfy the FS property and so they are also undesirable.

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. Duplex system structure

Our previous results [9] 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 Figure 1.

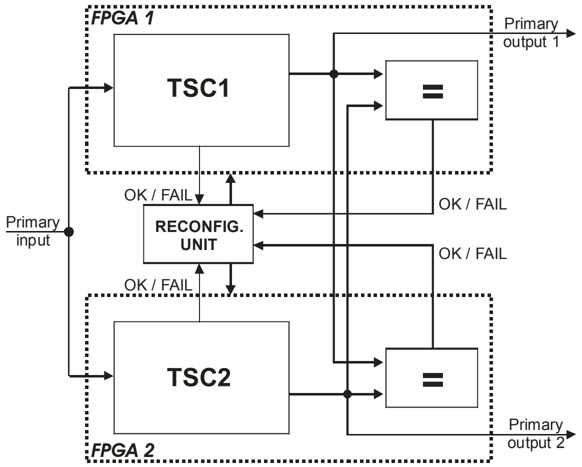


Figure 1: Duplex system architecture

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”).

To increase the dependability parameters we must add two comparators, 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. The reconfiguration solves two problems: localization and correction of the faulty part. The time needed to localize the faulty part is not negligible and must be included in the calculation of dependability parameters.

When the outputs are different, and both circuits signalize a correct function, we must stop the circuit function and the reconfiguration process is initiated for both FPGA circuits.

After the reconfiguration process is performed, states of both FPGAs are synchronized. It means that our modified duplex system can be used in an application where the system synchronization by reset is possible or another method of the system synchronization is implemented.

Each FPGA contains a TSC circuit and a comparator. The TSC circuit is composed of small blocks where each block satisfies the TSC property.

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.

The hardware emulator has great advantages in comparison with the software simulation process mainly as concern the speed of:

- the test vector generation process,
- a fault injection time,
- the process testing,
- respond results of test vector application.

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 biggest advantage of testing in hardware is speed of emulation. Every faulty circuit reaction on every input vector and on each possible fault has to be calculated.

This gives us number of iteration cycles

$$I = v \cdot f ,$$

where v is the number of vectors and f is the number of faults. This can lead to a very large number of iteration.

This iteration step takes only 1 clock pulse in hardware in comparison with software simulation, where time of single iteration depends on the number of gates. Therefore the advantage of speedup grows with the size of the circuit.

To utilize the advantage of speedup, we need to be able to switch from faulty state of the circuit to a proper state and vice versa in a short time. The granularity of the reconfiguration unit (the needed amount of transferred data) is a very important factor with respect to the speedup.

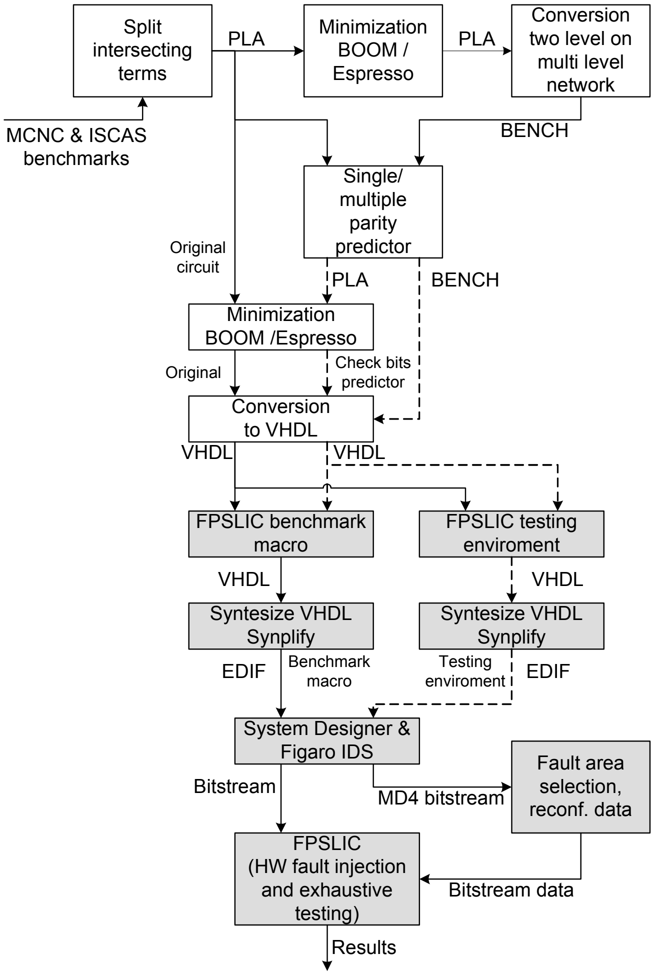


Figure 2: Hardware emulation design flow

The granularity of dynamic reconfiguration makes ATMEL's company FPGA family AT40K (especially AT94K, which embed AVR) suitable hardware for the proposed hardware fault emulator.

The hardware emulator presents a collection of the software tools and Atmel AT40K testing board. The design flow of hardware emulator of circuit is shown on Figure 2. 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 a 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

The schematic diagram of the hardware emulator (shown in Figures 3 and 4) is like as the schematic diagram of the fault classification presented in [10].

The hardware part of emulator is based on Atmel FPSLIC development board. It is divided into AVR part and FPGA part, see Figure 3.

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).

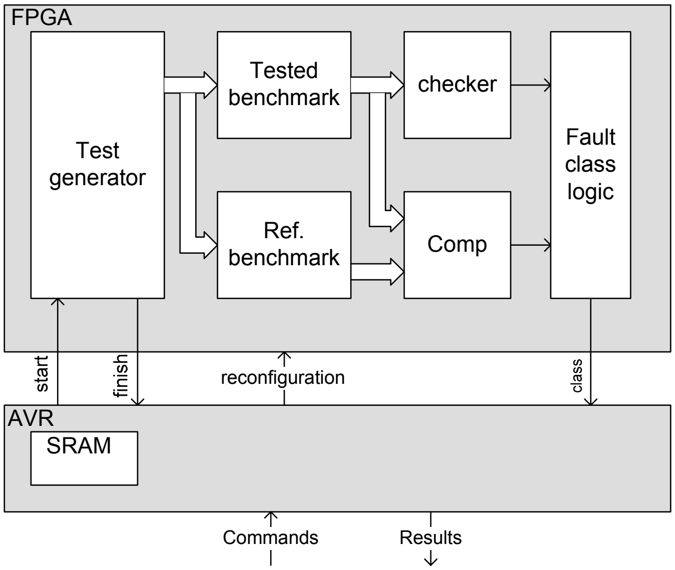


Figure 3: Basic hardware emulation diagram

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.

The FPGA part consists of one benchmark, which is under the test, and one reference copy of benchmark, which is present as a fast source of an error-free output, which is further compared with the tested copy. The result of comparison together with the result of a correct codeword checking is used to classify the fault.

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 emulator in FPGA (see Figure 4) 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.

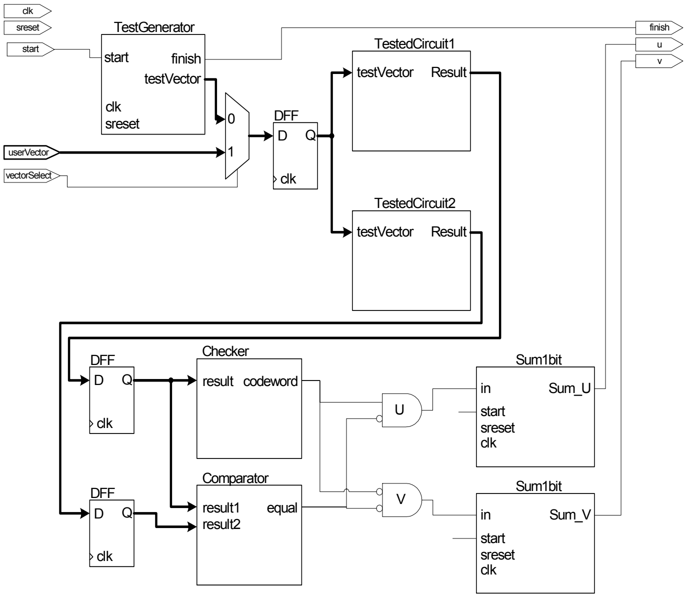


Figure 4: Structure of benchmark test

Parity checkers and comparators are used both in the hardware emulator (non-TSC versions) and in a dependable system based on duplex system [6]. Therefore special attention to them will be paid.

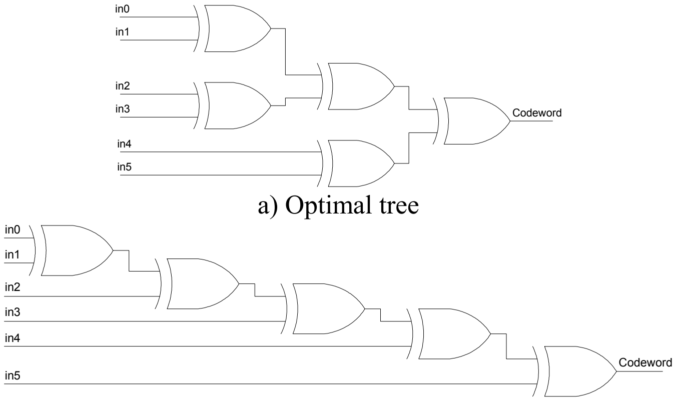
The even and odd parity can by calculated by the following equations:

a) Even parity

$$codeword = \overline{in_0 \oplus in_1 \oplus in_2 \oplus \ldots \oplus in_{n-1}}$$

b) Odd parity

$$codeword = n_0 \oplus in_1 \oplus in_2 \oplus \ldots \oplus in_{n-1}$$



a) Optimal tree

b) Unbalance tree

Figure 5: Optimal and unbalanced tree

The occupied area of even parity checker does not depend on the realized structure and it is equal to both optimal and unbalance variant. Only the final delay depends on realized structure. The optimal tree structure keeps smaller delay than the delay for unbalance tree structure. The difference between these two variants is shown in Figure 5. The example is based on 2-input XOR.

The area occupied by *n*-long vector in FPGA with *p*-input can be expressed by the formula:

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

, 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 area of the even and odd parity is equivalent. The number of LUTs *M* used in AT40K for even parity checker is:

$$M = \left\lceil \frac{n-1}{3} \right\rceil$$

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

The output of the comparator can be calculated by this equation:

$$equal = \overline{r_0 \oplus s_0} \cdot \overline{r_1 \oplus s_1} \cdot \overline{r_2 \oplus s_2} \cdot \ldots \cdot \overline{r_{n-1} \oplus s_{n-1}}\ .$$

The structure of a comparator is shown in Figure 6. 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.

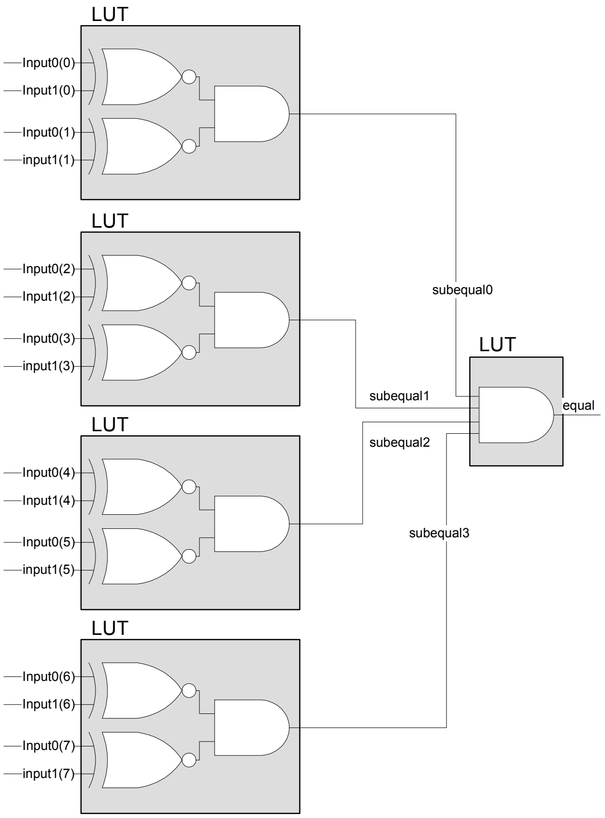


Figure 6: Comparator

Resulting size of TSC versions of the comparator and the checker is described in section 6. 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:

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

, 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 shown in Figure 2 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 1. 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 1: 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
alu3	10	8	45	45	1044	130	877	8	29
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 2., 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 2: 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
alu3	45	45	100	6	14	86.78	96.5	90
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 3. 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 3: 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
alu3	10	6.9	0.41	16.8
apla	10	6.0	0.34	17.6
B11	8	0.8	0.06	13.3
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 modified duplex system based on two FPGAs has been presented. In this emulator we are able to calculate FS, ST and TSC parameters of tested circuit more exactly than in the software simulator.

Experimental results of several benchmarks show consistency between the software fault simulation results and hardware fault emulation results, see Table 2.

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.

VIII. Acknowledgments

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

IX. References

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

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

[3] 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.

[4] 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.

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

[6] Dobiáš, R., Kubalík, P., Kubátová, H.: “Dependability Computations for Fault-Tolerant System Based on FPGA”, In Proceedings of the 12th International Conferrence on Electronics, Circuits and Systems, IEEE Circuits and Systems Society, 2005, vol. 1, pp. 377-380.

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

[8] 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.

[9] 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.

[10] 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.

[11] Kafka, L.: “Design of TSC circuits implemented in FPGA”, CTU FEE, 2004, (in Czech).

[12] Kvasnicka, J.: “Highly Reliable Design Based on FPGA circuits”, CTU FEE, 2006, (in Czech).

[13] 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 2006, vol. 1, pp. 274-278.