Faults Coverage Improvement based on Fault Simulation and Partial Duplication

Jaroslav Borecký, Martin Kohlík, Hana Kubátová, Pavel Kubalík
Dept. of Digital Design
Czech Technical University in Prague
Prague, Czech Republic
{borecjar; kohlimar; hana.kubatova; pavel.kubalik}@fit.cvut.cz

Abstract— **A method how to improve the coverage of single faults in combinational circuits is proposed. The method is based on Concurrent Error Detection, but uses a fault simulation to find Critical points – the places, where faults are difficult to detect. The partial duplication of the design with regard to these critical points is able to increase the faults coverage with a low area overhead cost. Due to higher fault coverage we can increase the dependability parameters. The proposed modification is tested on the railway station safety devices designs implemented in the FPGA.**

Keywords: Fault Tolerant, FPGA, Secure Device, Railway Station, Fault Simulation, Concurrent Error Detection, Partial Duplication

I. INTRODUCTION

We propose a method how to improve the observability and coverage of single faults in combinational circuits using Concurrent Error Detection (CED) techniques. Combinational circuits are combinational parts of finite state machine (FSM) in this paper. FSM can be modified as well by separating both combinational parts (next-state logic and output logic) from flip-flops, modifying these logics separately and put it all back together. Due to the fact, that states are also coded with a self-checking code, the whole FSM is the self-checking circuit.

CED techniques have been proposed many times in the past. Almost all of them have focused on the objective of being able to detect all faults.

Selective Partial Replication (SPaRe) technique presented in [12] has very similar aim as our modification. SPaRe is able to detect all faults in finite state machines (FSM). SPaRe uses a prediction logic that creates independent predicted signals. The optimization objective of SPaRe is to minimize the number of outputs of the prediction logic. Based on the observation that a subset of output bits per state transition is typically sufficient to detect all faults, SPaRe aims at identifying a minimal such set.

The main difference between SPaRe and our modification is that we combine two independent circuits that predict the values of the outputs of the original logic. The first predictor uses a regular parity bit, the second predictor is created by the partial duplication of original logic and the first predictor. We use a fault simulation to determine parts that have to be duplicated. Single faults similar to Single Event Upset (SEU) are inserted into the design.

Another technique presented in [13] is one of the few techniques, which is not focused on detecting all faults. It uses a partial parity function in concurrent error detection scheme. This methodology allows increasing the faults coverage with low area overhead. Our modification uses a regular parity in the first step, thus both techniques could have been used together.

Our modification is aimed to the railway station safety devices application implemented in the FPGA, but its principles can be generalized. Railway station safety devices are composed of cooperating FSM blocks. Each block is designed as a self-checking circuit based on Modified Duplex System (MDS) architecture principles [1], [2] and [3]. The self-checking circuit quality is determined by an area overhead and Fault Security (FS) property. The area overhead of both circuits that are the main parts of an MDS has to be minimized. On the other hand, it is useful to FS property of each circuit as high as possible. Higher FS property allows fault localization that is needed for an efficient use of the partial reconfiguration ability of the FPGA. Moreover, we have been planning to use the dynamic reconfiguration of the FPGA when a permanent fault is detected. The dynamic reconfiguration process is able to relocate the part of the design from the damaged place to the unused resources. We must be able to localize the fault as good as possible to speed up the reconfiguration process and to spare resources for the relocation.

Our modification is able to detect all single SEU-like faults that can affect the tested parts of the design, but our main goal is to find a trade-off between the fault coverage and the overhead of the design. The faults coverage is determined by our modification tool. It uses fault simulation to find the places, where faults are difficult to detect and then connects these places to the test outputs. It is assumed that connecting these places to the test outputs will increase FS property and its checking will not increase overhead too much. These outputs increase also the fault observability. It means that self-testing parameter is also increased.

Our modification is supplemental method to a regular duplex system. The regular duplex system will be used in a case in which 100% overhead is acceptable. It is important to say that the regular duplex system does not allow fault localization so the reconfiguration quite loses its effectiveness.

The paper is structured as follows: A short introduction to Fault Security calculations and Concurrent Error Detection methods and the description of Railway station safety devices are in Section II. Section III contains the description of the proposed modification. Tables, the settings of the parameters of our simulation and modification tool and results are presented in section IV. Section V concludes the paper.

II. BACKGROUND

A. Fault Security calculations

There are three basic quantitative criteria in a field of CED: [4]

- Fault Security (FS).
- Self-Testing (ST).
- Totally Self-Checking (TSC).

These three aspects have to be used in the on-line testing field to evaluate the level of safety of the designed or modelled system.

To determine whether the circuit satisfies the TSC property (both FS and ST properties are satisfied), the possible faults should be classified and separate into four classes, A, B, C and D [5].

- 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 ST.
- Class B - faults detectable by at least one input vector. 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 FS and ST property.
- Class C - faults that cause an incorrect codeword for at least one input vector. They are not detectable by any other input vector. Faults from this class cause undetectable errors. If any fault in a 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.

This fault classification should be used to calculate the level of satisfaction of FS or ST properties of the designed circuit and then calculate TSC properties.

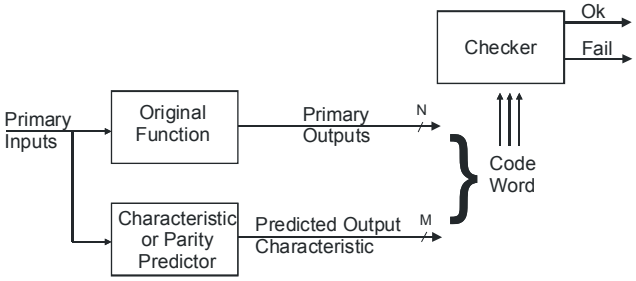

Figure 1. Basic Concurrent Error Detection diagram

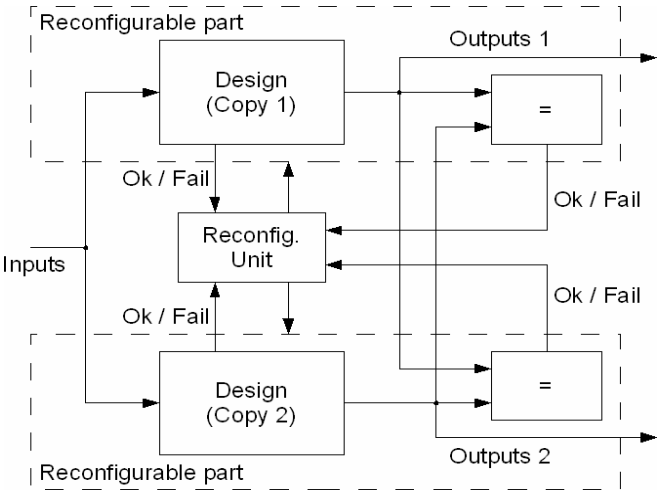

Figure 2. Block diagram of Modified Duplex System (MDS)

B. Concurrent Error Detection

CED is a common way to detect faults in the system design [6], [7]. The basic diagram of CED is shown in Figure 1.

Almost all CED techniques are based on adding a checking circuit to the original one. The added circuit independently calculates some special characteristic of output (e.g. parity bit). A checker circuit checks whether the special characteristic of the output actually produced by the system in response to the input sequence is the same as the predicted response, and produces an error signal when a mismatch occurs.

Both predictor and checker circuits are the sources of the area overhead of the design. The area overhead must be low to keep the effectiveness of this method. Any added logic does not only consume FPGA resources, it may be also affected by a fault.

MDS architecture uses two instances of design with CED that may be not fault tolerant. The purpose of MDS architecture is to achieve the whole circuit including all checkers and comparators to be fault tolerant. The block diagram of MDS is shown in Figure 2. Each improvement of the FS property of the design increases the probability of detecting error inside the block. Detecting an error inside the block initiates reconfiguration of the damaged block, but the second block may still be operational. If the error is not detected inside the block, it is detected by comparators.

The error detected by comparators initiates the reconfiguration of both blocks (outputs from blocks are different, but the source of the error cannot be determined).

C. *Railway station safety devices*

The proposed methodology is demonstrated on the practical experiment for safety railway station device design. Nowadays the safety device of a railway station is in many cases realized by several functional blocks based on relays. These functional blocks occupied high area. It means, that for the railway station with 10ths number of rails it can occupy whole building. This solution also leads to high current consumption and high power supply. The devices based on relays have been very popular due to their high safety factor ensured by a structure corresponding with a railway scheme. Current research deals also with systems based on two or more parallel working processor (instead of relays). The safety property of this system depends more on a human factor due to properties based and given by software. The safety device based on processors is described in [11].

In our practical experiment the safety device of the railway station is based on a five blocks realized by an FSM. These basic five blocks are based on basic meaning of the safety railway station systems containing relay unit. Security property is given by possible hardware combination. It means that one train path cannot be set with using occupied block by another train path. This approach is different from PC based system, where possibility of setting of a free path is defined by rules. The rules are set by programmer and his mistake could lead to an accident.

The scheme of a simple railway station based on the relay blocks is shown in Figure 3. The definition of these blocks and their function is in [8].

The new proposed system investigated in our research team [8] uses the same structure as a relay-based system but the function implemented inside the blocks and communications between these blocks are completely different. Some blocks from an old system were joined together. Each new block is based on an FSM. The simple railway station based on the new proposed blocks is shown in Figure 4.

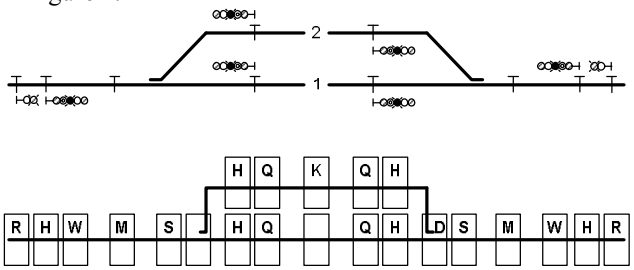

Figure 3. Simple railway station with relay-based blocks

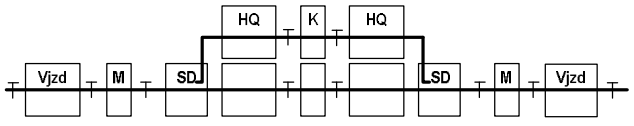

Figure 4. Simple railway station composed of the new FSM-based blocks

These blocks are defined as follow:

- *VJZD block* represents a home signal. This block is a start point of a train path. This train path is built directly from this block. This block can be also the end of a train path. VJZD block solves a track before the home signal and signalizes whether this track is free or occupied.
- *M block* controls correctness of a train position. In a case, when the train path is divided into three parts, the train coming from left to right must firstly occupy the left segment following by the middle one and at last the right segment. In any other case, an error is signalized.
- *SD block* represents a rail switch and also controls the right position of a train.
- *HQ block* represents an exit signal. This block serves as a start point of the train path and the train path is built to the right from this block. This block can be also the end of the train path.
- *K block* represents a station track and controls the correctness of a train position.

The complex railway station safety device can be generated from these basic blocks. The inputs and the outputs of each block are divided into groups according theirs functionality. There are three types of inputs and three types of outputs.

Inputs are divided as follow:

- *I name* – an input from track
- *IB name* – an input from others blocks
- *IO name* – an input from control device

Outputs are divided as follow:

- *V name* – an output to track
- *VB name* – an output to others blocks
- *VO name* – an output to control device

All FSM blocks are specified in KISS format. KISS format is presented as the subset of BLIF format [9]. There are two blocks of combinational logic. The first block generates next state and the second block generates outputs form current state. A set of flip-flops is assumed. The current state is stored in flip-flops and we assume its representation only as a data path in our approach. The current state is encoded by the selected code (Binary or One Hot) and forms the code word. The code word is generated by one of combinational logic used to obtain the next state.

III. FAULT SIMULATION AND FAULT COVERAGE IMPROVEMENT MODIFICATION FLOW

The proposed modification can be divided into two separate parts. “Top” designs and corresponding “.tst” files are prepared in the first part. “Top” design contains next-state or output logic extracted from a chosen safety device FSM and the corresponding parity predictor. “.tst”

files contain the test vectors, the correct output responses and the definitions of the check codes. The second part contains the fault simulation to find Critical points, the modification in these points, the partial duplication and the final fault test.

A. *First part – file preparation*

The first part of the proposed modification is the preparation of input files ("Top" design and ".tst" file) for the second part. The process flow of this part is shown in Figure 5. The input to this part is chosen FSM of safety device of the railway station described in KISS format.

The short description of the first part:

- *States encoding, logics extraction, test vectors generation* – Both combinational logics from the FSM have to be extracted from the KISS file and saved in the PLA format. The encoding of the states of the FSM has to be chosen before extraction. Binary and 1-out-of-n (One Hot) encoding is selected in this paper. Predictors "Predictor1" are created as the parity bit generators from each of combinational logics and are also saved in the PLA format. The last outputs in this step are the ".tst" files containing the test vectors to check the parity. Each ".tst" file corresponds to one "Top" design. Further steps are equivalent for both combinational logics.

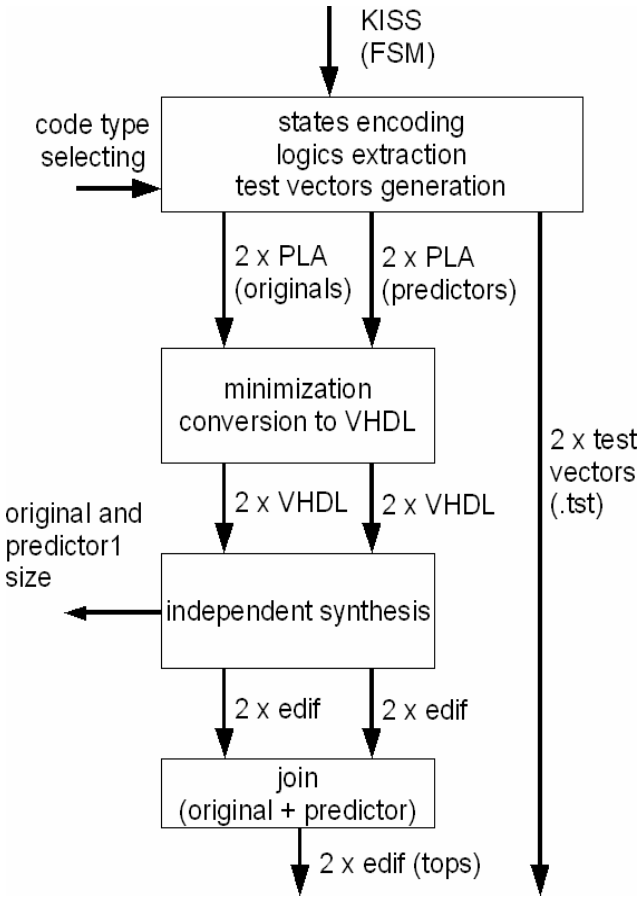

Figure 5. The process flow of file preparation

- *Minimization, conversion to VHDL, independent synthesis* – Both original combinational logics and both predictors are minimized, converted into VHDL and synthesized separately to disable resource sharing.
- *Join* – Each "Original" (synthesized combinational logic) and corresponding "Predictor1" (synthesized parity predictor) are joined to "Top" design at Electronic Design Interchange Format (EDIF) level to ensure the independence of both parts. This step is performed by our simulation and EDIF manipulation utility.

The outputs of this part are two "Top" EDIF files and two ".tst" files. The first EDIF contains next-state combinational logic and its parity predictor, the second EDIF contains output logic and its parity predictor. ".tst" files contain the corresponding test vectors and respective correct responses. The sizes (used LUTs in the FPGA implementation) of both "Original" and both "Predictor1" designs are obtained from synthesis results.

B. *Second part – simulation and partial duplication*

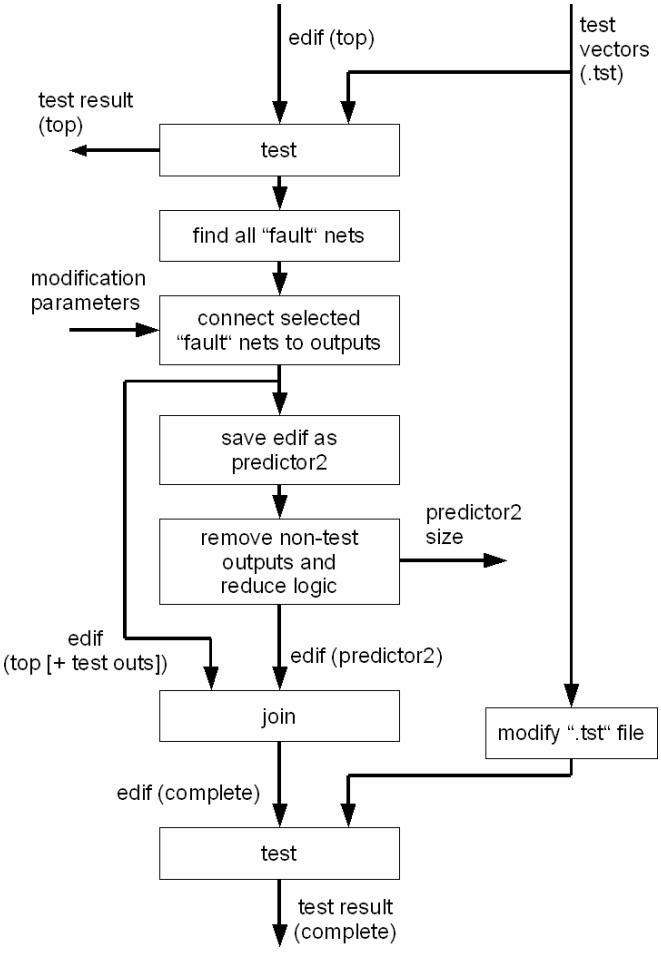

Figure 6. The process flow of modification and test

The second part of the proposed modification contains the fault simulation and the EDIF modification (partial duplication). All process steps in the second part are using a simulator and EDIF manipulation utility developed in our research group. One pair "Top" EDIF – ".tst" file enters this part of the process. The flow of this part is shown in Figure 6. All steps are made at the EDIF level. The EDIF parser is based on BYU Edif Tools [10].

This part is based on the results from the fault simulation. The main idea is to add the second predictor that contains part(s) of the "Top" design. The block diagram of "Complete" design, that contains both "Top" and "Predictor2" designs, is shown in Figure 7.

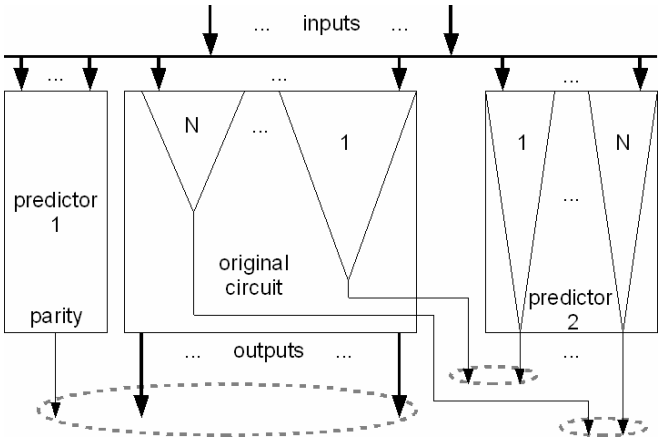

Figure 7. Block diagram of "Complete" design

The description of the second part follows:

- *Test "Top" design* – The fault test of the design is the key part of proposed modification and deserves detailed description. The correct output response to the input word is calculated in the first step and set of all applicable faults are found. The faults may be inserted into the Lookup Table (LUT) as flip of one bit in the LUT's memory or may be inserted at the connection point between the part instance and the net. Both methods provide similar results so the results in this paper are calculated by using only the second method.

- One fault from the faults set is inserted into the design then and if any change is detected, the design is
re-simulated. Output word code check (if the code is set in the ".tst" file) and the comparison with the correct output word are made in the next step. If the code check fails, the input word is added to the "test" group. If the comparison of the outputs fails, but the code check passes, the input word is added to the "error" group. When all input words are tested, the fault class is determined. The classification is based on the size of "test" and "error" groups and is described in Table I.

TABLE I. FAULT CLASSIFICATION

Fault class	"test" group	"error" group
A	Empty	Empty
B	Non-empty	Empty
C	Empty	Non-empty
D	Non-empty	Non-empty

- Fault is removed from the design and all values, which have been changed after fault insertion and re-simulation, are restored. Fault insertion, re-simulation, output checking, fault classification and values restoration are preformed for all faults from the faults set. Statistics for each fault and numbers of "A", "B", "C" and "D" class faults are generated. FS property is calculated from numbers of faults.

- *Find all "fault" nets* – All nets connected to places with class "C" or "D" faults and not connected to primary inputs are marked as "fault" nets. Nets connected to primary inputs are taken off the "fault" nets list because the output added to such net could detect only the fault that is on the input pin, but such fault will be propagated to second predictor "Predictor2" as well and it would not be detected.

- *Connect selected "fault" nets to outputs* – A list with "fault" nets sorted by number of input words, that produces incorrect output codeword, descending is created. Selected nets are connected to the newly-formed outputs. Modification parameters designate number of newly-formed outputs and "fault" nets that are to be connected to them.

- *Remove non-test outputs and reduce logic* – The EDIF created in previous step is duplicated and saved as "Predictor2". Only newly-formed outputs are kept and useless instances and nets are removed. Statistics with used LUT resources in "Predictor2" is created.

- *Join* – EDIF designs from two previous steps are joined to the "Complete" design.

- *Modify ".tst" file* – ".tst" file is modified to include newly-formed test outputs from "Top", outputs from the second predictor and their check codes. All pairs "test_output_X" – "predicted_output_X" must have equivalent values, because they are generated form the same (but duplicated) logic. These check rules (codes) must be specified in the ".tst" file in the case of simulation or would have to be checked by checker circuit during the standard operation.

- *Test "Complete" design* – The fault test of the "Complete" design is the final step of the proposed modification. Statistics for each fault and numbers of "A", "B", "C" and "D" class faults are generated. FS property is calculated from numbers of faults.

The outputs from the second part are statistics of numbers of "A", "B", "C" and "D" class faults and the size of "Predictor2". Another output is "Complete" design including original logic, parity predictor "Predictor1" and predictor "Predictor2" that contains duplicated parts from original logic and "Predictor1".

The numbers of "A", "B", "C" and "D" class faults are required to calculate the FS property of the design. The size of "Predictor2" is used to calculate relative overhead.

IV. Experiments results

Experimental results are taken from multiple runs of the second part of the process (simulation and partial duplication) with different parameters. The modification of each "Top" design is launched with parameters as follows:

- Added outputs: All; From position: 0
- Added outputs: 1; From position: 0, 1, …, 18, 19
- Added outputs: 5; From position: 0, 5, 10, 15
- Added outputs: 20; From position: 0

"Added outputs" parameter specifies the number of test outputs that are added during modification. "All" value means that the test output is created for each "fault" net.

The position is designated from the list of all "fault" nets that is sorted by the size of the "error" group descending. Position "0" matches the net with the largest "error" group. It is assumed, that connecting this net to the test output and checking its value with the value of the corresponding output from "Predictor2" will cause the increase of FS property.

The best achieved results for each settings and design are in the following tables. The results of next-state logic designs from all blocks are in Table II. The results of output logic designs from all blocks are in Table III.

Both tables have the same structure. A design name including the name of the original block of the safety railway station system and the code, which is used to encode its FSM states, is in the first column. K1 indicates the next-state logic part and K2 indicates the output part of the original FSM. The second column contains FS property of design before "Predictor2" is added. FS is calculated as the probability that the fault is in class "B". All faults in class "A" are ignored in calculations. The third column contains FS of design with both predictors. It is divided into four sub-columns according to the count of added test outputs. The overhead of the "Predictor1" relative to corresponding "Original" is in the fourth column. The overhead value is calculated from the number of LUTs used in the design. There is no overhead of "Predictor1" of all next-state logics using One Hot code because logic function of such "Predictor1" is always logic "1". The last column contains the total overhead (the overhead of both predictors). It is divided into four subcolumns as in the case of the third column.

It is obvious that the addition of one test output causes the small increase of both FS and overhead values. The more outputs are added, the bigger increase of both FS and overhead values is achieved. Next-state logic designs are larger (ca. 100-300 LUTs) so 20 test outputs can be added without problems in the most cases. Output logic designs are small (ca. 10-30 LUTs) so 20 test outputs cannot be added in some cases and therefore some values in the table may be equal for "All" and other counts of added outputs.

Both tables show significant improvement of FS. Adding of 20 test outputs increases FS to 90% and more in most cases. FS increase by 1% costs ca. 1% relative overhead size of "Predictor2".

V. Conclusion and future work

The method how to increase Fault Security property has been presented. Results indicate that Fault Security increase by 1% costs ca. 1% relative overhead size of "Predictor2". The method allows finding a balance between a Fault Security increment and a design overhead. Big overhead is unwelcome in the case of the safety railway station system design therefore the count of added test outputs is limited to maintain the overhead in acceptable boundaries. However, Fault Security can be increased from 60%-85% to 90%-99% at the cost of total overhead from 10% to 50% in applications, in which a bigger overhead can be tolerated. The fact, that 100% of Fault Security parameter has not been reached means that another method of concurrent error detection such as modified duplex system should be used. Then the system is failsafe and even for many faults fault tolerant. Our preliminary results show that the only remaining non-detectable faults in "All" setting are located on the primary inputs. When insertion of such faults is disabled, Fault Security is able to reach 100%.

The presented method is not limited for safety railway station system design only. It allows increasing Fault Security property in general applications or System on a Chip (SoC) designs. The second part (the fault simulation and the EDIF modification) of the presented method does not require any commercial tools to operate.

Simulation and modification tools are fully automated and are able to generate results without any human intervention. This allows to explore the state space of the combinations of added test outputs and to find the combinations that allows a bigger Fault Security improvement for the lower overhead cost. A state-space exploration is in development.

Acknowledgment

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

References

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

[2] P. Kubalík and H. Kubátová, "Dependable design technique for system-on-chip", Journal of Systems Architecture. 2008, vol. 2008, no. 54, pp. 452-464. ISSN 1383-7621.

[3] R. Dobiáš, P. Kubalík, H. Kubátová: "Dependability computations for fault-tolerant system based on FPGA", Proc. of 12th IEEE International Conference on Electronics, Circuits and Systems, 2005, pp. 1-4

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

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

[6] S. Mitra and E. J. McCluskey: "Which Concurrent Error Detection Scheme To Choose?", Proc. of International Test Conf. 2000, pp. 985-994.

[7] S. Mitra and E. J. McCluskey: "Diversity Techniques for Concurrent Error Detection", Proc. of IEEE 2nd International Symposium on Quality Electronic Design, 2001, pp. 249-250.

[8] J. Borecký, P. Kubalík and H. Kubátová: "Reliable Railway Station System based on Regular Structure implemented in FPGA", Proc. of 12th EUROMICRO Conference on Digital System Design, Los Alamitos, IEEE Computer Society, 2009, pp. 348-354.

[9] Berkeley Logic Interchange Format (BLIF), University of California Berkeley, 2005.

[10] BYU Edif Tools: Available from: http://reliability.ee.byu.edu/.

[11] V. Chandra, M.R. Verma: "A Fail-Safe Interlocking System for Railways", IEEE Design & Test of Computers, 1991, pp. 58-66.

[12] P. Drineas, Y. Makris, "SPaRe: Selective partial replication for concurrent fault-detection in FSMs", IEEE Transactions on Instrumentation and Measurement, 2003, vol. 52, pp. 1729-1737.

[13] K. Mohanram et al. "Synthesis of Low-Cost Parity-Based Partially Self-Checking Circuits" In 9th IEEE International On-Line Testing Symposium, Kos Island, Greece, 2003

TABLE II. BEST VALUES – NEXT-STATE LOGIC

Design name	[Original + Predictor1] FS (%)	[Complete] FS (%)				[Original + Predictor1] Overhead (%)	[Complete] Overhead (%)			
		Number of added test outputs					*Number of added test outputs*			
		All	*1*	*5*	*20*		*All*	*1*	*5*	*20*
Code_Binary-fsm_HQ_K1	85.17	97.84	85.98	88.44	96.46	22.28	33.49	22.70	24.34	31.58
Code_Binary-fsm_K_K1	86.05	97.71	87.00	88.77	96.68	26.11	37.63	26.58	28.40	35.91
Code_Binary-fsm_M_K1	87.22	97.92	88.03	90.27	97.20	29.38	37.81	29.78	31.32	36.55
Code_Binary-fsm_SD_K1	85.31	98.57	85.65	86.76	90.77	22.22	34.80	22.38	23.82	28.65
Code_Binary-fsm_VJZD_K1	84.41	97.90	85.32	89.70	95.96	29.33	41.11	29.80	33.33	39.77
Code_OneHot-fsm_HQ_K1	36.42	97.02	37.29	48.33	59.79	0.00[a]	52.63	5.82	11.80	20.98
Code_OneHot-fsm_K_K1	81.29	92.77	83.09	87.58	92.77	0.00[a]	14.89	1.23	5.88	14.89
Code_OneHot-fsm_M_K1	82.92	94.48	84.35	87.71	94.48	0.00[a]	16.80	0.95	4.59	16.80
Code_OneHot-fsm_SD_K1	89.84	97.20	90.25	90.78	93.44	0.00[a]	12.38	0.56	2.03	6.35
Code_OneHot-fsm_VJZD_K1	86.08	96.27	86.64	87.96	93.22	0.00[a]	16.85	0.65	3.16	13.56

a. "Predictor1" is only VCC – LUTs are not required

TABLE III. BEST VALUES – OUTPUT LOGIC

Design name	[Original + Predictor1] FS (%)	[Complete] FS (%)				[Original + Predictor1] Overhead (%)	[Complete] Overhead (%)			
		Number of added test outputs					*Number of added test outputs*			
		All	*1*	*5*	*20*		*All*	*1*	*5*	*20*
Code_Binary-fsm_HQ_K2	61.28	98.51	73.40	82.63	98.51	5.13	36.21	24.49	26.00	36.21
Code_Binary-fsm_K_K2	84.75	92.31	92.31	92.31	92.31	10.00	18.18	18.18	18.18	18.18
Code_Binary-fsm_M_K2	87.56	95.71	92.83	95.71	95.71	10.00	18.18	14.29	18.18	18.18
Code_Binary-fsm_SD_K2	71.01	98.76	72.95	85.88	96.48	8.96	34.41	10.29	21.79	31.46
Code_Binary-fsm_VJZD_K2	86.50	97.37	90.00	97.37	97.37	9.68	22.22	12.50	22.22	22.22
Code_OneHot-fsm_HQ_K2	65.38	95.81	77.60	86.46	95.81	11.43	40.38	34.04	39.22	40.38
Code_OneHot-fsm_K_K2	62.21	94.04	77.78	89.90	94.04	26.32	48.15	48.15	48.15	48.15
Code_OneHot-fsm_M_K2	67.04	94.20	72.24	84.50	94.20	26.09	46.88	29.17	39.29	46.88
Code_OneHot-fsm_SD_K2	52.30	90.39	70.86	80.21	87.30	5.00	39.68	39.68	39.68	39.68
Code_OneHot-fsm_VJZD_K2	68.28	90.73	79.65	85.95	90.73	13.33	35.00	31.58	33.33	35.00