Design Methodology for High Reliable System

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

Abstract

This paper deals with design methodology of high reliable digital system based on modified duplex system. Our modified duplex system is based on two FPGAs, where every FPGA can be reconfigured when a fault is detected. Design implemented in each FPGAs is based on one of several methods how to ensure the self checking properties. Our design methodology reflects requirements on area overhead and value of dependability parameters. Our modified duplex system is compared with standard duplex system. Our dependability model and dependability calculations are used to quantify proposed solutions. Availability parameters have been calculated by dependability Markov models. The final reliable system is fault tolerant. Combinational circuit benchmarks have been considered in this work to compute the quality of the final adapted duplex system. The benchmarks are represented by two level networks (truth table). All of our experimental results are obtained by XILINX FPGA implementation by EDA tools and our design tools.

Keywords: adopted duplex system, on-line testing, self checking circuits, FPGA, dependability parameters.

1. 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].

The FPGAs are based on SRAM memories sensitive to Single Event Upsets (SEUs), therefore simple usage of FPGA circuits in mission critical applications without any method of SEUs detection is impossible.
One change of a bit in the configuration memory by SEUs leads to a change of a circuit function, even drastically. The Concurrent Error Detection (CED) techniques allow a faster detection of soft errors (errors which can be corrected by the reconfiguration) caused by 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 some 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 great area and/or time overhead. Our proposed structure increases dependability parameters together with ensuring a relatively low area overhead 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. Availability is a function of time, *A(t)*, defined as the probability that a system is operating correctly and is available to perform its function its functions at the instant of time [7].
Our previous research shows the relation between the area overhead and the SEUs fault coverage [8]. Due to a need for 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%. Therefore an additional method of fault detection has to be used to ensure complete

SEUs fault coverage and to increase dependability parameters.

2. Basic On-Line Testing Criteria

There are three basic quantitative criteria in a field of CED: fault security (FS), self-testing (ST) and totally self-checking (TSC) properties [7]. These three aspects have to be 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, the number of detectable faults belonging to one of four classes A, B, C and D [9] have to be calculated.
This fault classification can be used to calculate how much the circuit satisfies the FS or ST property and then calculate TSC properties.
The parity predictor is used to generate the proper output code of the circuit in our research, Figure 1. These techniques ensure a small area overhead and a higher SEUs fault coverage but the SEUs fault coverage reached is not 100% [10, 11, 12].
The circuit area overhead significantly depends on parity codes used. If we use a strong error detecting code, like a Hamming code or Berger code, the FS parameter is almost 100% but the area overhead is high [8, 13]. The logic synthesis method of the area reduction for circuit described by multilevel network is described in [14].
The following structures are vulnerable to SEUs: multiplexer select lines, programmable interconnect point states, buffer enables, LUT values, and control bit values.

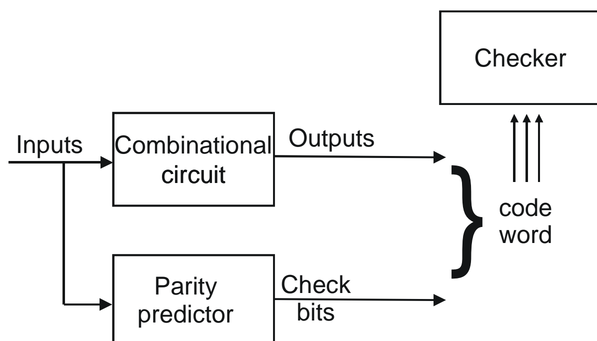

Figure 1. Basic structure of TSC circuit

Any changes of a mux select lines, programmable interconnect point states or buffers lead to a significant circuit function change but the function change is hardly detected for SEUs impacted in LUTs [15]. The probability of SEUs impacting routing resources (mux select lines, programmable interconnect point states and buffers) is about 78% and only about 15-21% for LUTs. It means many SEUs leads to significant circuit function change. But any change in LUTs is hardly detected because of its small impact to the realized function. In some cases these faults may be undetected.
We have used the LUT upset failure in our calculations. The only LUT upset assumption giving to us the worst case for availability values obtained for our benchmarks. It means that the final results are worst in comparison with method assuming all faults in FPGA. The most faults belong to the routing resources group. In this case we have assumed faults occurring in routing resources, the dependability parameters are higher then in a case where only LUTs are calculated.
We want to obtain the worst case of dependability parameters and due to this fact our fault model accepts only changes in LUTs memory. The FS property depends on the class B (Detectable faults)[9]. A low number of faults belong to class B leads to low FS property. The FS values for MCNC and ISCAS [15] benchmarks used to validate our modified duplex system are shown in Table 1. Here ”C” is benchmark circuit, ”IN” is number of inputs, ”OUT” is number of outputs, ”AO” is the area overhead, ”FS” is a probability, that a fault is detected by code word and ”Ass” is a steady-state availability.
We have used our simulator described in [16] to obtain A, B, C, D classes and FS property. This simulator has these features:

- The simulation is performed for circuits described by a netlist format (EDIF).
- The stuck-at-1 and stuck-at-0 faults on inputs and outputs of components are considered.
- Combinational and sequential circuits are supported.

This simulator supports circuits where inputs, outputs and internal states (in the case of a

sequential circuit) are coded by even parity, multiple parity and 1 out of *N* code. Multiple code groups can be used to ensure TSC. The simulator also supports Hamming like codes and *M* out of *N* code.

Table 1. Single even parity – PLA

C	IN	OUT	ORIG [LUT]	AO [%]	FS [%]
alu1	12	8	8	688	100
apla	10	12	45	53	83
b11	8	31	38	8	75
br1	12	8	50	20	63
al2	16	47	52	12	94
alu2	10	8	30	140	92
alu3	10	8	28	121	90
s1488	14	25	312	13	86
s1494	14	25	317	13	86
s2081	18	9	24	125	96
s27	7	4	4	75	72
s298	17	20	39	49	91
s386	13	13	51	39	71

The FS property expresses the probability that an existing fault is detected on a primary output of the circuit. If the FS is fully satisfied (to 100%) a fault occurring in a circuit is always detected.

3. Proposed Structure

Our previous results show that to fully satisfy TSC property (100%) is difficult, so we have proposed a new structure based on two FPGAs, see Figure 2.

Each FPGA has one primary input, one primary output and two pairs of checking signals OK/FAIL. 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. The probability of the information correctness depends on the FS property and in many cases is higher than 75%. In a case when outputs are different and one of the TSC circuits signalizes fail function, the wrong FPGA is correctly recognized. Correct outputs are processed by the next circuit.

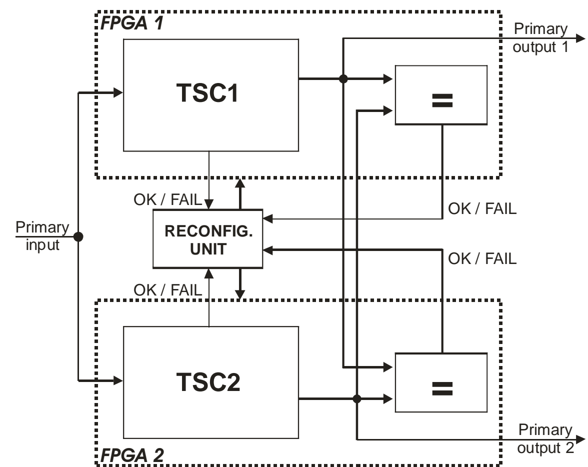

Figure 2. Reconfigurable duplex system

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. We only select the faulty FPGA and we reconfigure it in our solution. It means that we do not localize the faulty block inside the compound design. The time to localize a fault and to reconfigure the faulty part can be similar to the time to reconfigure the whole FPGA. The whole FPGA reconfiguration also repairs the faults which occurred in an unused logic. The reconfiguration process can be initiated also when one of the two FPGAs signalize the “FAIL” signal. This situation occurs when a fault is detected by one of the small TSC blocks inside the compound design. The fault propagation to the primary outputs may take a long time.

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. When 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 reset synchronization is possible.
Each FPGA contains a TSC circuit and a comparator. The TSC circuit is composed of small blocks where each block satisfies the TSC property. The structure of the compound design satisfying the TSC property is described in [17].

4. Design methodology

The design methodology of TSC circuit creation is described in Figure 3. To generate the output parity bits, all the output values have to be defined for each particular input vector. Unfortunately, it is not so in the benchmark definition files. Only several output values are specified for each multi-dimensional input vector, the rest are assigned as don't cares; they are left to be specified by another term. Thus, to be able to compute the parity bits, we have to split the intersecting terms, so that all the terms in the truth table are disjoint.
In the next step the original primary outputs are replaced by parity bits. Two different error codes were used to calculate output parity bits (single even parity code and multiple parity code), but our design methodology allows use also Hamming like code or standard duplication. Another tool was used in the case where the original circuit was modified in multilevel logic. This tool is described in [18]. Two circuits generated in the first step (original circuit and parity circuit) are processed separately to avoid sharing any part of circuit. Every part can be minimized by the BOOM [19] or Espresso tool [20]. The final area overhead depends on the software that was used in this step. Many tools were used to reach a small area of the parity bits generator. BOOM was used to minimize the final area. In this step the area overhead is known, but we can decide if the fault coverage is sufficient.
In the next step the "pla" format is converted into the "bench" format. The "bench" format was used due to the fact that the tool, which generates the exhaustive test set uses this format. An exhaustive test set has 2n patterns and we used it to evaluate TSC goals.

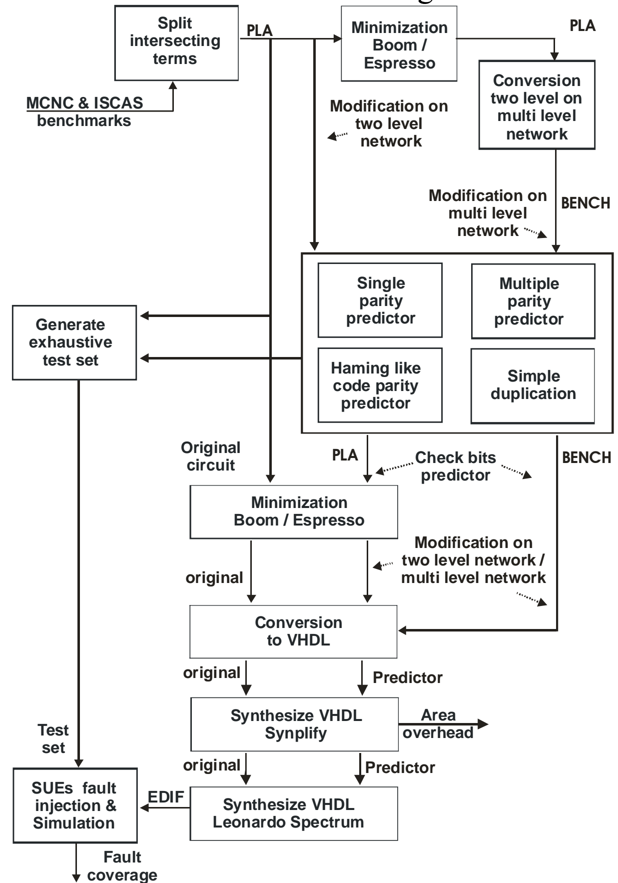

Figure 3. Design methodology flow

Another conversion tool is used to generate two VHDL codes and the top level. Top level is used for incorporating original and parity circuit generator. In the next step the synthesis process is performed by Synplicity Synplify Pro tool. The constraints properties set during the synthesis process express an area overhead and a SEU fault coverage. If the maximal frequency is too high, hidden faults occur during the fault simulation. The hidden faults are caused by the circuit duplication. The size of the area overhead is obtained from the synthesis process. The final netlist is generated by the Leonardo Spectrum software. The fault coverage was obtained by simulation using our software [16].
To evaluate the area overhead and fault coverage special tools had to be developed. In addition to some commercial tools such as Leonardo Spectrum and Synplify we have used format converting tools, parity circuit generator tools and simulation tools.
At first, the area minimization and term splitting is preformed for original circuit by

BOOM [19]. Hamming code generator (or single parity generator) is generated by the second software. These two circuits are minimized again with BOOM. Next two tools convert the two-level format into a multi-level format. The first one converts a “pla” file to “bench” and the second one “bench” to VHDL. The second software is used for generating the final circuit in the ”bench” format due to its further usage in exhaustive test set generator. The format converting software and parity generator software was written in Microsoft Visual C++. The netlist fault simulator was written in Java. The parser source code was used to parse the netlist that is generated by two commercial tools described above.

6. Results

The results obtained by our design methodology for highly reliable system was validated on MCNC and ISCAS [19] benchmarks. Results are shown in table 2.
Here ”Circuit” is benchmark circuit, ”AO” is the area overhead, ”PN” is the number of parity nets, ”C” and ”D” is the number of undetected faults, that are not detected by code word and ”Ass” is the steady-state availability

Table 2. Availability parameters

Circuit	AO [LUT]	PN	AO [%]	C	D	Ass [%]
alu1	55	1	687	0	0	1
alu1	16	2	200	0	0	1
apla	24	1	53	1	109	0.9999912
apla	22	2	49	1	92	0.9999928
b11	3	1	8	42	59	0.9999938
b11	7	2	18	38	52	0.9999937
br1	10	1	20	47	154	0.9999883
br1	23	2	46	41	142	0.9999871
alu2	42	1	140	0	58	0.9999906
alu2	40	2	133	0	52	0.9999910
alu3	34	1	121	0	63	0.9999897
alu3	33	2	118	0	63	0.9999888
s1488	41	1	13	94	321	0.9999962
s1488	94	2	30	80	267	0.9999961
s386	20	1	39	15	176	0.9999878
s386	25	2	39	8	149	0.9999892

7. Conclusion and future work

Our modified duplex system based on two FPGAs with high reliable system design methodology has been presented. The design methodology allows select proper code with respect to the system requirements.
We can use four methods for the totally self checking circuit design. The selected method depends on the final area overhead and the SEUs fault coverage. In a case, when the high reliable system is required and area overhead can be high, the duplication or Hamming like code is better to use. These two methods ensure that the fault security is fulfilled on hundred percent and Ass parameter is equal to hundred percent too.
In a case, when the low area overhead and a high reliable system is required, the simple or multiple parity predictor is better to use.
Our high reliable structure ensures that the final system is better than a standard duplex system with 0,999978248 of Ass [6].
Our future work will be dedicated to some practical case studies (e.g., railway applications). We will use a hardware fault simulator and practical experiments based on the ATMEL FPSLIC circuit.

8. Acknowledgement

This research has been supported in part by the GA102/04/0737 grant and MSM6840770014 research program.

References

[1] Dobiáš, R., Kubátova, H.:“FPGA Based Design of Raiway's Interlocking Equipment”, DSD2004, In Proceedings of EUROMICRO Symposium on Digital System Design, Piscataway: IEEE, 2004, pp. 467-473.
[2] 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, 2005, pp. 436 – 444.
[3] QuickLogic Corporation.: “Single Event Upsets in FPGAs”, 2003, www.quicklogic.com

[4] 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, 2004, pp. 584-589.
[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", ICECS2005, In Proceedings of the 12th International Conference 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] Kubalík, P., Fiser, P., Kubátová, H.: "Minimization of the Hamming Code Generator in Self Checking Circuits", DESDes'04, Proceedings of the International Workshop on Discrete-Event System Design, Zielona Gora: University of Zielona Gora, 2004, pp. 161-166.
[9] Kafka L., Kubalík P., Kubátová H., Novák O.: "Fault Classification for Self-checking Circuits Implemented in FPGA", DDECS2005, Proceedings of IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop, Sopron University of Western Hungary, 2005, pp. 228-231.
[10] Drineas, P., Makris, Y.: "Concurrent Fault Detection in Random Combinational Logic", ISQED2003, Proceedings of the IEEE International Symposium on Quality Electronic Design, 2003, pp. 425-430.
[11] Mitra, S., McCluskey, E., J.: "Which Concurrent Error Detection Scheme To Choose?", Proc. International Test Conf. 2000, pp. 985-994.
[12] Mohanram, K., Sogomonyan, E. S., Gössel, M., Touba, N. A.: "Synthesis of Low-Cost Parity-Based Partially Self-Cheking Circuits", IOLTS2003, Proceeding of the 9th IEEE International On-Line Testing Symposium 2003, pp. 35.
[13] Bolchini, C., Salice, F., and Sciuto, D.: "Designing Self-Checking FPGAs through Error Detection Codes", 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'02), Canada, pp. 60.
[14] Touba, N. A., McCluskey, E. J.:"Logic Synthesis Techniques for Reduced Area Implementation of Multilevel Circuits with Concurrent Error Detection", Proc. of ACM/IEEE International Conference on Computer-Aided Design (ICCAD), 1994, pp. 651-654.
[15] Graham, P., Caffrey, M., Zimmerman, J., Sundararajan, P., Johnson, E., Patterson, C.:"Consequences and Categories of SRAM FPGA Configuration SEUs", MAPLD2003, Military and Aerospace Programmable Logic Devices International Conference, Washington DC, Paper C6.
[16] Kafka, L.: "Design of TSC circuits implemented in FPGA", CTU FEE, 2004.
[17] Kubalik, P., Kubatova, H.: "High Reliable FPGA Based System Design Methodology", DSD 2004, Work in Progress Session of 30th, Universitat Linz, 2004, pp. 30-31.
[18] Kubalík, P., Kubátová, H.:"Design of Self Checking Circuits Based on FPGA", In: Proc. of 15th International Conf. on Microelectronics, Cairo, Cairo University, 2003, pp. 378-381.
[19] Hlavička, J., Fišer, P.: "BOOM - a Heuristic Boolean Minimizer", ICCAD2001, Proc. International Conference on Computer-Aided Design, San Jose, California (USA), 2001, pp. 439-442.
[20] Brayton, R. K. et al. :"Logic minimization algorithms for VLSI synthesis", Kluwer Academic Publishers, Boston, MA, 1984, pp. 192.