Automatic Design of Fault-Tolerant Systems for VHDL and SRAM-based FPGAs

Jakub Lojda, Richard Panek, Zdenek Kotasek
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Bozetechova 2, 612 66 Brno, Czech Republic
Email: {ilojda, ipanek, kotasek}@fit.vutbr.cz

***Abstract*—This paper presents and evaluates the possibility of automatic design of fault-tolerant systems from unhardened systems. We present an overview of our toolkit with its three main components: 1) fault-tolerant structures insertion (which we call *helpers*); 2) fault-tolerant structures selection (called *guiders*); and 3) automatic testbed generation, incorporating advanced acceleration techniques to accelerate the test and evaluation. Our approach is targeting complete independence on the HW description language and its abstraction level, however, for our case study, we focus on VHDL in combination with fine-grained n-modular redundancy. In the case study part of this paper, we proved that it is undoubtedly beneficial to select a proper fault tolerance method for each partition separately. Three experimental systems were developed with the usage of our method. Two of them achieved better reliability parameter while even lowering their chip area, compared to static allocation of equivalent fault tolerance technique type. In the case study, we target the best median time to failure, the so-called t50, however, our method is not dependent on this parameter and arbitrary optimization target can be selected, as soon as it is measurable.**

Keywords*—Fault-Tolerant System Design, Electronic Design Automation, Redundancy Insertion, Redundancy Allocation, Multiple-choice Knapsack Problem, FPGA, VHDL, t50.

I. Introduction

Certain types of electronic systems must be able to maintain high level of reliability. The various reasons for this exist. For instance a control system of a medical equipment must remain stable otherwise a human health would be endangered. Another group of systems cannot be repaired because it is difficult or even impossible to access them. This group includes, for example, satellites, space research probes or space rovers. The design of all of these systems must, therefore, reflect the demand for the high reliability. Generally, reliable systems must be able to perform their task while delivering correct results in prescribed time. One well-known approach to reliable system design is the so-called *Fault Tolerance* (FT) [1]. This approach is based on FT enhancement of the system, while the components are considered naturally unreliable. The architecture of the system is, however, designed and configured in such way, that a failing component does not influence the correctness of produced results nor their timing requirements.

In our research, we focus primarily on FT of commercially available *Field Programmable Gate Arrays* (FPGAs) that are storing their configuration bitstream in the SRAM memory. These are, especially in the area with increased radiation, prone to the so-called *Single Event Upsets* (SEUs). SEUs have potential to flip a configuration bit, thus, changing the implemented design function and possibly the correctness of results. Primarily, we research the possibilities in the FT design automation. Our previous publications [2], [3], [4] presented the possibilities of automatic incorporation of FT structures into algorithms written in a higher programming language, synthesized using the *High-Level Synthesis* (HLS) Design Flow [5]. In this new paper, we present a new method of incorporating FT structures into VHDL language. Description code modification algorithms are strictly separated from allocation algorithms. This is different from the related work FT design automation tools and allows to operate our FT automation toolkit on various description languages of various levels of abstraction while re-using most of the toolkit. As opposed to behavioral-level C++ design in [3], this paper is primarily focused towards designs described in the structural-level VHDL. This research aims to abstract from the description language and bring the FT system design automation in a comprehensive way, which should also be the contribution of our research.

Tools to insert a particular redundancy method exist. Some of them are available only commercially, such as the Xilinx TMRTool [6], which works as a part of the synthesis process, during which it modifies the synthesized design. Another tool is the *BYU-LANL TMR Tool* (BL-TMR) [7], which is not strictly commercial as the TMRTool. The tool targeting Verilog, called TMRG [8], works on the description-code level. It focuses on *Triple Modular Redundancy* (TMR) exclusively.

Approaches to solve the reliability allocation problem can also be found in literature. For example, the genetic algorithm was used for this purpose in [9] and [10], where the use of *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) was used to find a number of promising solutions. The authors of paper [11] present a combination of previously mentioned BL-TMR insertion tool with design space exploration, while targeting various optimization goals.

After a design is hardened, it must be properly tested to ensure its compliance with its specification. In the papers [12], [13], techniques of fault injection into a real FPGA board are shown. There is no need to modify the original design, which is an important advantage. The paper [13] presents the platform called FLIPPER. This platform utilizes two FPGAs, one running the *Design Under Test* (DUT) and the other acting as a controller. The paper [14] presents evaluation platform, which was previously developed in our research group. It runs on a PC and evaluates data captured from an FPGA. This platform is, however, more suitable for the final testing, not for the massively accelerated evaluations, that are necessary in the process of FT system design automation.

This paper is organized as follows: Section II shows the principles of our FT design automation toolkit and its main concepts. The experiments setup and results are presented in Section III. Finally, the Section IV concludes the paper.

II. Fault-Tolerant Design Automation

The following section presents our FT design automation. Our approach is based on the traditional flow, which incorporates manual iteration-oriented improvement of the system while addressing its weakest points. In the automated flow, there is a description of the original system and the target specification available at the beginning. The specification might include, for example, the percentage of critical bits of FPGA bitstream or a *Time to Failure* (TTF). At first, the system must be partitioned. At the moment, partitions are created based on instances of VHDL entities. For the description language, the so-called *helpers* are built. These allow to incorporate FT into a partition of the system. Subsequently, the so-called *guider* must select the most appropriate FT technique for each partition, following the reliability specifications. The last part of the automated flow is testing. This part is crucial according to our previous experiences. The testing and parameters measurement are usually performed in high quantities, making it very time-consuming part of the design flow. The context of the traditional and the automated flow can be observed in Figure 1.

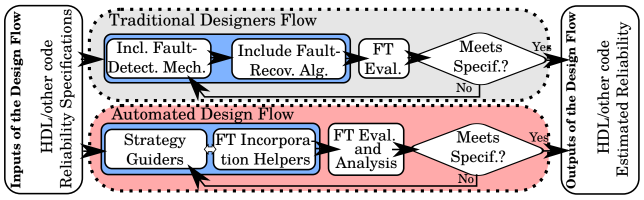

Figure 1: Traditional and Automatic Flows for FT System Design.

A. The Helpers: Fault Tolerance Incorporation

In this paper, we use the newly created helpers for VHDL, which allow us to harden specific entity instances. We consider these as partitions, in the sense of the previously established terminology. Special code comments must be written around the instantiation, which instruct the helpers to make the specific modification. Our VHDL helpers are based on a group of generic templates, which simplifies the addition of new architectures. These are, however, limited by the encapsulation of the entity instance, as these are currently considered as black boxes. The group of templates is supplemented by additional procedures that search for necessary data and use this data to fill a generic template. At first, the VHDL helper divides the original VHDL file into code-block tokens delimited by the special code comments and identifies the tokens (e.g. instantiation block, *don't care* block, etc.). After that, the instances marked for modification are selected and the whole VHDL project is searched for basic pieces of the source description code. These include, for example, entity declarations, signals, etc. These are then parsed to obtain additional information, such as signal directions, bit widths etc. for filling the generic template. Also the clock signal name is detected, in order to route this signal to an optional auxiliary component, such as a scrubbing unit, in the template. After the template is filled, the previous instantiation is modified to refer to this newly filled template. The modification flow is displayed in Figure 2.

B. The Guiders: Fault Tolerance Strategy

It is important to have a strategy to select proper FT techniques for each partition, while meeting the given constraints (e.g. chip area). Such strategy, in our toolkit, is called the *guider*. The guider basically solves the allocation of redundancy techniques. It selects the appropriate FT techniques for the partitions, in order to strengthen FT of the system, while considering one or more constraints. Unachievable constraints cause the design process to stop without a candidate solution.

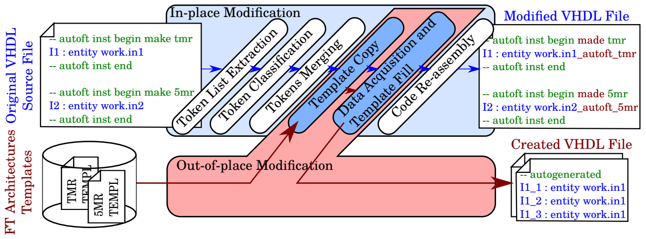

Figure 2: Simplified Code Example with VHDL Modification Flow, Automated using VHDL Helpers.

In our toolkit, we utilize a form of the so-called *Multiple-Choice Knapsack Problem* (MCKP) [15] solver in place of the main guider. The MCKP is a specific variant of the general *Knapsack Problem* (KP). The KP is one of the so-called combinatorial optimization problems [16], the target of which is to maximize the value of items put into a hypothetical knapsack of a given load capacity. The MCKP variant constrains the items that are put into the knapsack. The items are divided to classes and from each class, exactly one item must be selected. As can be seen, the solution to this problem is convertible to our problem of FT technique selection: we have classes of different implementations for each partition. Each implementation has different value (i.e. benefit in the form of increased FT) and different weight (i.e. chip area, power demand, etc.).

C. Fault Tolerance Evaluation

Testing and evaluation of a component or a system is performed relatively often during the design flow. This makes it the most time-consuming part of the complete FT design flow. For this reason, we developed our *Fault Tolerance Estimation* (FT-EST) framework [4], in which we stressed its acceleration possibilities to expedite the evaluation. A test design consists of a test controller and the tested units. We call this complete formation a testbed.

The generated testbed has a fixed structure, although the components are very configurable. The main part of a testbed includes the so-called *Input Generation Unit*, which generates the so-called stimuli for the testing. These can be streams of data (e.g. generated using a counter or a *Linear Feedback Shift Register* (LFSR) unit) or transactions of data. The outputs of tested units are compared against the golden (i.e. reference) unit and the differences are captured. The running tested unit is paused through clock-gating. Fault Injector [17] artificially and permanently changes utilized bits of *Look-Up Tables* (LUTs) in specified times, based on the required fault intensity. Detailed description of our FT-EST testbed generator can be obtained from our previous publication [4].

III. The Case Study and Experimental Results

In the following section, a case study utilizing our FT system design automation toolkit will be presented, based on hardening of an artificially constructed system. Also the parameters of the resulting systems will be discussed alongside with the design flow.

A. Toolkit Setup

In our experiments, we prepared the helpers to include the TMR and *5-Modular Redundancy* (5-MR) techniques. We also utilize the MCKP guider. We focus on the minimization of the median time to failure, also called the *t50* parameter. This parameter defines the time of 50% probability that the system is still fully functioning. The t50 quantification is more useful for our measurement, as it tends to remove extreme values, opposed to the classical *Mean Time To Failure* (MTTF), which utilizes the mathematical average. Also, the MTTF (i.e. the period from system start, for which the fault masking is possible) tends to lower with the added modular redundancy for longer mission times [18]. We chose to precisely evaluate each partition in advance and then estimate the resulting system parameters inside the MCKP solver. After the solver finishes, the best matched system is then evaluated precisely. We use the FT-EST framework to generate our testbeds. As the test stimuli generator, we use various bit-width variants of LFSRs utilizing corresponding maximal-length polynomials to produce a pseudo-random sequence of all the possible combinations. The fault model includes permanent faults of utilized bits of LUTs. Faults are injected into the precisely selected part of the FPGA configuration bitstream. Their intensity is derived from this bitstream size of the tested design, based on the fault injection intensity unit – *injection/s/bit*. To measure the results, testbeds for each partition and each system are synthesized using the Xilinx *Integrated Synthesis Environment* (ISE) 14.7 and prepared using the Xilinx PlanAhead 14.7. Testbeds are run on the ML506 board [19] with the Virtex 5 technology.

B. Toolkit Input: Benchmark System

We prepared a benchmark system composed of four hypothetical partitions: 1) addition, 2) constant addition, 3) *Cyclic Redundancy Check on 8 bits* (CRC-8) computation; and 4) number of high bits detection. Connection of these components including their input and output bit widths can be observed in Figure 3. The system was described in VHDL.

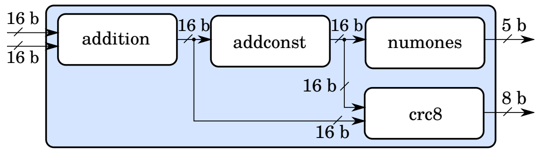

Figure 3: Benchmark System Structure.

C. The Helpers: Variants Generation

With the usage of helpers, we created two hardened variants for each partition of the system. The overview of the partitions, including their real measured parameters, can be seen in Table I. As can be observed, each partition has a different size. After the application of each technique, the FPGA synthesis surely optimized the larger partitions better, as the sizes of hardened partitions do not correspond to the theoretically predicted overheads (i.e. more than triple for TMR and quintuple for 5-MR techniques). The so-called majority voters, which are an integral part for both the TMR and 5-MR architectures, were both included in the timing analysis and subject to fault injection (as other parts of the complete component or system). As can be seen, the t50 parameter improved for each partition except the addconst TMR version, for which it was nearly 20% worse, compared to the simplex t50. This might be caused by the internal structure of the implementation or the nature of the computation itself. The addconst holds the added constant in its implementation. This, if hit, results in a logically functioning design. The results are, however, computed from a different constant values, rendering them incorrect. As can be seen from the results, the effectiveness varies among the partitions, thus supporting the need to methodically select the proper FT method for each partition.

TABLE I: System Partitions with Their Size and Reliability Parameters under Fault Injection Intensity of $2e{-}5\,\mathrm{inj/s/bit}$

Partition Name	FT Technique	Bitstream Area [b]	t50 [ms]	t50 Compared to Simplex [ms]	[%]
addition	simplex	4 288	197 635	+ 0	+ 0.00
	TMR	7 552	208 793	+ 11 158	+ 5.64
	5-MR	9 856	225 042	+ 27 407	+ 13.87
addconst	simplex	3 264	337 843	+ 0	+ 0.00
	TMR	6 656	271 246	- 66 597	- 19.71
	5-MR	9 088	345 745	+ 7 902	+ 2.34
crc8	simplex	4 800	39 484	+ 0	+ 0.00
	TMR	9 792	47 222	+ 7 738	+ 19.6
	5-MR	14 272	60 227	+ 20 743	+ 52.54
numones	simplex	3 072	94 549	+ 0	+ 0.00
	TMR	6 848	102 603	+ 8 054	+ 8.52
	5-MR	10 304	119 195	+ 24 646	+ 26.07

The box plot chart displayed in Figure 4 illustrates the scatter on the measured values for each partition. As can be seen, the benefit of an FT technique is very fluctuating among different circuit types. Also, the simplex minimum time to failure (i.e. the worst measured case) is always better, compared to the TMR and 5-MR versions. This means that the dispersion rates of hardened partitions (at least towards minimum values) are higher. Also, for the crc8 and numones partitions, the middle 50% interquartile range is concentrated nearer the median, indicating lower variability of these results. This indicates that the TMR and 5-MR work better on these partitions. The addconst was the only component visibly deviating in efficiency of FT techniques, specifically for TMR. As can be seen, the 5-MR version has a slightly better median value, although the difference is nearly negligible. Nonetheless, for the 5-MR version of this partition, the variability of the middle 50% is also smaller, similarly but not so obvious as for the numones and crc8 partitions.

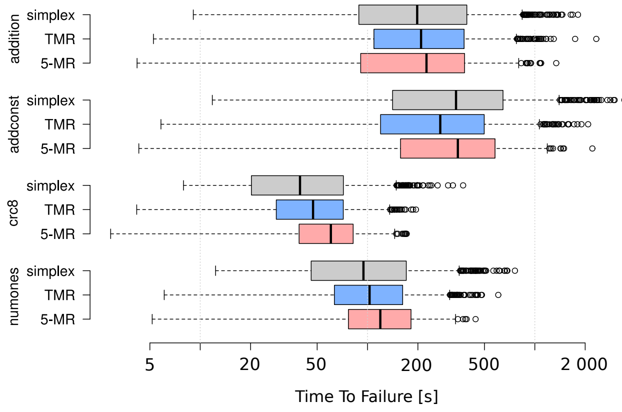

Figure 4: Box Plot Chart of Time to Failure for Each Partition and Their Hardened Variants.

D. The Guiders: Automatic Composition of Systems

Three systems were automatically composed using our methods. Methods were configured to minimize the t50 parameter while not exceeding a given chip area. These area limits were based on the bitstream area that was subject to the fault injection. This was 20 000, 25 000 and 30 000 bits. The overview of synthesized systems, including their parameters, can be seen in Table II. We also created two additional homogeneous reference systems, each of which is utilizing one type of FT technique applied to each partition.

TABLE II: Automatically and Manually Composed Systems (as a Reference) with Reliability Parameters under Fault Injection Intensity of $2e{-}5\,\mathrm{inj/s/bit}$

System Name	FT Techniques				Bitstream Area [b]	t50 [ms]
	addition	addconst	crc8	numones		
auto_20000	simplex	simplex	5-MR	simplex	18 624	43 198
auto_25000	simplex	simplex	5-MR	TMR	22 400	49 935
auto_30000	simplex	simplex	5-MR	5-MR	25 856	49 675
ref_simplex	simplex	simplex	simplex	simplex	9 152	23 559
ref_TMR	TMR	TMR	TMR	TMR	24 704	42 173
ref_5-MR	5-MR	5-MR	5-MR	5-MR	37 376	55 900

As can be observed, the guider based on the MCKP solver targeted the mostly failure-prone partitions: the crc8 and the numones. Incorporation of FT techniques into the remaining two partitions was evaluated as not sufficiently effective. The crc8 partition was the most error-prone, and thus the highest hardening was allocated for this partition in all the three cases. The smallest automatically composed system occupied approximately $18\,\mathrm{kbits}$, that is only 75.34% size of the reference system size for which the TMR was manually assigned to each partition. Despite this, the automatically composed system shows slightly better t50 parameter than for the manually created reference, thus, saving circa 25% of area. The second automatically composed system is still by 9.33% smaller than the manually created TMR one, yet its t50 is more than $7\,\mathrm{s}$ longer. For the last automatically created system, the t50 is nearly equivalent to the previous, second one. The size of the third system is, however, larger. This wrong choice of partitions by the MCKP solver is apparently caused by imprecise estimation of system t50 from the components t50 times, thus, confusing to solver to choose sub-optimal selection of FT techniques. Nevertheless, this third system is still more than 30% smaller compared to the manually created 5-MR system and its t50 is only by 11% worse.

IV. Conclusions

This paper presents a novel approach to FT system design, which is able to work in various abstraction levels with various language description formats. We implemented our solution in the form of a toolkit with each part of the toolkit specializing on a different task of the FT system design automation. New template-based approach to helpers for incorporating FT techniques into VHDL was presented alongside with the usage of MCKP solver as the guider for the redundancy allocation. Our automatic testbed generation framework was also briefly described. We modified it to monitor, detect and report the time of the first failure observation. The experimental evaluation and illustration of our approach was presented in the Section III. In this section the experimentation is performed on our artificial benchmark circuit. During our experiments, we proved that it is undoubtedly beneficial to select FT method for each partition separately. Three automatically generated versions of the experimental system were developed with the usage of our method. Two of them achieved better reliability parameter while even lowering their chip area, compared to static allocation of equivalent FT technique type.

Acknowledgements

This work was supported by the Brno University of Technology under number FIT-S-20-6309 and the JU ECSEL Project SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems), Grant agreement No. 783119.

References

[1] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[2] J. Lojda, J. Podivinsky, Z. Kotasek, and M. Krcma, "Data Types and Operations Modifications: A Practical Approach to Fault Tolerance in HLS," in *2017 IEEE East-West Design Test Symposium (EWDTS)*, Sept 2017, pp. 1–6.

[3] J. Lojda, J. Podivinsky, O. Cekan, R. Panek, M. Krcma, and Z. Kotasek, "Automatic Design of Reliable Systems Based on the Multiple-choice Knapsack Problem," in *2020 23rd International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2020, pp. 1–4.

[4] J. Lojda, J. Podivinsky, O. Cekan, R. Panek, and Z. Kotasek, "FT-EST Framework: Reliability Estimation for the Purposes of Fault-Tolerant System Design Automation," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, Aug 2018, pp. 244–251.

[5] M. Fingeroff, *High-level synthesis blue book*. Xlibris Corporation, 2010.

[6] Xilinx Inc., "TMRTool: The Industry's First Development Tool to Automatically Generate Triple Module Redundancy (TMR) for Space-grade Re-programmable FPGAs," https://www.xilinx.com/products/design-tools/tmrtool.html, accessed: 2021-04-13.

[7] "Byu edif tools homepage," http://reliability.ee.byu.edu/edif/, accessed: 2021-04-13.

[8] S. Kulis, "Single Event Effects Mitigation with TMRG Tool," *Journal of Instrumentation*, vol. 12, no. 01, p. C01082, 2017. [Online]. Available: http://stacks.iop.org/1748-0221/12/i=01/a=C01082

[9] G. Kanagaraj, S. Ponnambalam, and N. Jawahar, "A Hybrid Cuckoo Search and Genetic Algorithm for Reliability–Redundancy Allocation Problems," *Computers & Industrial Engineering*, vol. 66, no. 4, pp. 1115–1124, 2013.

[10] Z. Wang, T. Chen, K. Tang, and X. Yao, "A Multi-objective Approach to Redundancy Allocation Problem in Parallel-series Systems," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 582–589.

[11] J. Anwer, M. Platzner, and S. Meisner, "FPGA Redundancy Configurations: An Automated Design Space Exploration," in *2014 IEEE International Parallel Distributed Processing Symposium Workshops*, 2014, pp. 275–280.

[12] M. Alderighi, S. D'Angelo, M. Mancini, and G. R. Sechi, "A Fault Injection Tool for SRAM-based FPGAs," in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE.* IEEE, 2003, pp. 129–133.

[13] M. Alderighi, F. Casini, S. d'Angelo, M. Mancini, S. Pastore, and G. R. Sechi, "Evaluation of Single Event Upset Mitigation Schemes for SRAM-based FPGAs Using the FLIPPER Fault Injection Platform," in *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on*. IEEE, 2007, pp. 105–113.

[14] J. Podivinsky, J. Lojda, O. Cekan, and Z. Kotasek, "Evaluation Platform for Testing Fault Tolerance Properties: Soft-core Processor-based Experimental Robot Controller," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 229–236.

[15] H. Kellerer, U. Pferschy, and D. Pisinger, "The Multiple-choice Knapsack Problem," in *Knapsack Problems*. Springer, 2004, pp. 317–347.

[16] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, ser. Algorithms and Combinatorics. Springer Berlin Heidelberg, 2007.

[17] M. Straka, J. Kastil, and Z. Kotasek, "SEU Simulation Framework for Xilinx FPGA: First Step Towards Testing Fault Tolerant Systems," in *14th EUROMICRO Conference on Digital System Design*. IEEE Computer Society, 2011, pp. 223–230.

[18] J.-C. Geffroy and G. Motet, *Design of dependable computing systems*. Springer Science & Business Media, 2013.

[19] Xilinx Inc., "Ml506 Evaluation Platform User Guide," *UG347 (v3. 1.2)*, 2011.