Reliability Indicators for Automatic Design and Analysis of Fault-Tolerant FPGA Systems

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

***Abstract*—As electronic systems penetrate into areas in which reliable computing is required, new methods incorporating reliability into these systems arise. It is important to properly test and evaluate parameters of such methods before the actual implementation and the practical usage in an application. Generally, in our research, we are focusing on the acceleration of reliable design through creation of automation methods. However, for this purpose, it is important to develop tools to automatically analyze reliability properties of the system after the method is applied. In our previous work, we developed the *Fault Tolerance ESTimation* (FT-EST) framework, which specializes on minimizing the requirement for user intervention. In this paper, we are using the framework to collect the data, however, the research presented in this paper primarily focuses on the possibility to automatically analyze such data. Our previous papers were focused on particular methods of the automatic reliability insertion and evaluation while this paper introduces new reliability indicators based on low-level properties of FPGA configuration bitstreams. Currently, we are limiting our research to SRAM-based FPGA systems and focus on the VHDL and C++ (in the combination with High-level Synthesis) languages.**

***Keywords*—Fault Tolerance Evaluation, Fault-tolerant Design Automation, Analysis, FPGA, Fault Tolerance Estimation Tool, High-level Synthesis, Catapult C, Redundant Data Type**

I. Introduction

To improve electronic circuit reliability, two general approaches exist. The first is called *Fault Avoidance* (FA) [1] and its main objective is to precisely select such electronic components that are able to withstand the conditions of the target environment during their operation. The second approach is called *Fault Tolerance* (FT) [2]. FT, in contrast to FA, accepts the components of the system might occasionally fail, however, the architecture of the system counts with this fact and eventually *masks* the fault. This means that the failure does not propagate and does not influence the controlled task, although, this statement is valid only until all the redundant components are exhausted. In our research, we focus on the FT approach and so does this paper.

The rising complexity of today's systems results in usage of advanced design methodologies, such as the *High-Level Synthesis* (HLS) [3]. HLS-generated systems consist of the so-called *data-path*, which processes the data, and the so-called *control-path*, which is a *Finite State Machine* (FSM) controlling the data-flow through the *data-path*. Generally two approaches to generate FT systems using HLS can be distinguished throughout the literature: 1) modification of the HLS methodology, and, 2) modification of the source code before its processing through HLS.

So far, in our research, we have been considering the reliability of systems implemented in *Field Programmable Gate Arrays* (FPGAs), as they are prone to the so-called *Single Event Upsets* (SEUs) causing a change of bits in the SRAM configuration memory. The bits of the bitstream that propagate a fault to the circuit outputs are generally called *critical* or *sensitive* bits. So far, we have been evaluating the reliability of circuits just and only through the *critical* bits percentage. Our previous research papers were focused towards particular methods of the automatic reliability insertion and reliability evaluation. The main contribution of this paper is the presentation of other novel metrics (i.e. indicators, such as the size of the largest continuous block of sensitive bits or the number of bits leading to a constant failure response) that could be, possibly automatically, evaluated and could improve the expression of the circuit behavior.

The problematic of design analysis is discussed in the literature. The authors of [4] present an approach to mixed-level dependability analysis. Their research evaluates the improvement of the analysis achievable by considering the accurate modeling of the environment in which the circuit performs its application. In the paper [5], the analysis of FPGA platforms made by Digilent Nexys, that are utilizing Xilinx technology, is shown. The author utilizes a commercially available tool, the *Computer Aided Reliability Engineering* (CARE) [6] software, to model and analyze the platforms parameters. The authors of [7] present the formal analysis of a *Finite Impulse Response* (FIR) filter design that utilizes arithmetic residue codes. The paper focuses on the analysis of the so-called missing fault rate, meaning a fault for which the FIR *branch* cannot be identified, mainly. The paper verifies the obtained results using a simulation. The authors of [8] present the reliability analysis of HLS-generated circuits with and without pipelining. The authors also consider the influences on using DSPs in their design. The same authors analyzed and compared the influences on FT properties between a soft-core processor-based implementation and HLS circuit implementing the equivalent algorithm, in their research paper [9].

This paper is organized as follows. An overview of our FT design automation platform, alongside with its components used for the experimental evaluation, is shown in Section II. The analysis process, which is the main contribution of this research paper, is described in Section III. The results are summarized on a case study in Section IV. Section V concludes the paper and suggests our plans for future research.

II. Fault Tolerant Design Automation Platform

Our research is originally based on our intention to develop a platform that would be able to automatically modify system description in such way, that the system, after its implementation, becomes more reliable. Our approach is based on the classical iterative designer's approach, however, we adjusted this approach to prepare it for automatic operation and accelerated execution.

978-1-7281-1756-0/19/$31.00 ©2019 IEEE

At the beginning of the design process, an *unhardened* system in the form of a description code is provided. Desired parameters are defined before the process enters its iterative loop. Each loop starts with the modification of the description code. For this purpose, we developed a concept of *helpers*, which allow to insert particular FT architecture to a given place in the circuit description. As the second part of each iteration, the obtained circuit is synthesized within our *Fault Tolerance ESTimation* (FT-EST) [10] framework, which is strongly focused on the acceleration of the evaluation process. After the evaluation, the decision is made, whether the resulting circuit parameters are satisfactory or whether the iteration loop enters another iteration. The process flow can be seen on Figure 1.

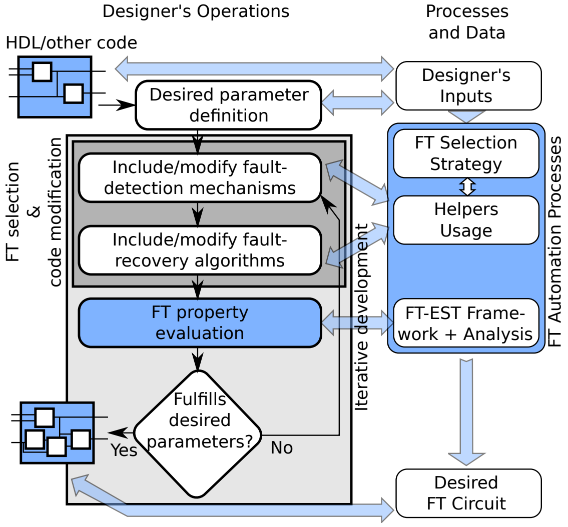
Designer's Operations
Processes and Data
HDL/other code
Desired parameter definition
Designer's Inputs
FT selection & code modification
Include/modify fault-detection mechanisms
Include/modify fault-recovery algorithms
FT property evaluation
FT Selection Strategy
Helpers Usage
FT-EST Frame-work + Analysis
FT Automation Processes
Iterative development
Fulfills desired parameters?
Yes
No
Desired FT Circuit

Figure 1. Design of FT system from an *unhardened* system with the designer's operations mapped to the processes of our automation platform.

A. Evaluation Framework

As a part of our previous research, we developed a *Fault Tolerance ESTimation* (FT-EST) framework, which is able to benchmark FT properties of VHDL-written components and is intended to be a part of the FT design automation platform. The framework is based on the *functional verification* approach and is designed to require as few human interactions as possible. It utilizes some acceleration techniques such as stimuli generation and outputs comparison performed on the FPGA and is prepared for parallel testing of multiple component instances.

The FT-EST framework can be divided into HW and SW part. The HW part is currently written in the VHDL and is synthesized together with the tested component unit, which we call the *Unit Under Test* (UUT). The SW part runs on the PC and communicates with the HW part using the ChipScope Pro *Virtual Input/Output* (VIO) cores [11]. For the SEU simulation, we use the Fault Injector [12] that was previously developed in our research group. The injector has a possibility to filter only utilized bits of the UUT that are used as contents of *Look-Up Tables* (LUTs). The testing of the component can be divided into two types of cycles: 1) the *test cycle* replays one transaction in the UUT's input pins and compares the outputs with the *golden* unit. The transactions are generated using the *Input Generation Unit* (IGU) and outputs are compared in the *Output Compare Unit* (OCU). 2) The *SEU cycle* passes through all the selected bits of the bitstream individually and in each of its iterations, it performs one *test cycle*. The framework architecture can be seen in Figure 2.

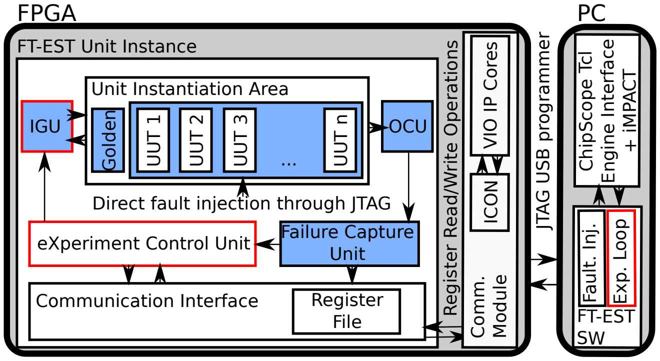

Figure 2. The simplified architecture of the FT-EST system; the parts highlighted in blue are dynamically and fully automatically generated, while the parts highlighted in red are to be provided by the designer to specify the experiment setup.

B. Redundant Data Types

In our approach, we targeted to modify the C++ source description code before its processing by HLS. *Redundant Data Types* (RDTs) [13], [14], [15], as we call them, play the role of *helpers* in our FT design automation platform. RDT is a newly created *Data Type* (DT) that represents a one particular FT architecture (e.g. the *duplex* or the *Triple Modular Redundancy* (TMR). The RDT acts as a *decorator* to the DTs previously used in the algorithm. A set of interconnected operations of equivalent RDT category forms a *subsystem* of the given FT architecture. For example, for the TMR approach, which is implemented using the `triple` RDT, storage elements are triplicated through the creation of multiple instances of the *original* DT and multiple equivalent operations are then executed, one per each instance of the storage element. Finally, a voting component utilizing *bit-* or *word-based* selection is incorporated.

III. The Analysis Procedure

In a classical design approach, a designer decides whether the parameters of a particular system are satisfactory. However, in the automatic FT design, human interactions are very limited. It is possible to use the number of *critical* bits as a reliability indicator, however, in a real application, we usually want to take into account as much information as possible. This motivated us to focus on a way to further statistically analyze the information obtained through the fault injection.

Generally, for the number of erroneous output transactions, three other dimensions (i.e. domains) can be analyzed: 1) the stimuli transactions; 2) the bits of the bitstream; and 3) the type of output errors (e.g. the number of mismatching bits, etc.). The data can be studied after one of these dimensions is collapsed. For this research, we have chosen to study the first two domains (i.e. the stimuli transactions with the bits of the bitstream), thus collapsing the domain of output error type. The data we obtain from the FT-EST framework can be: 1) *2-dimensional* (2D), which means that we obtain the number of erroneous output vectors for each bit of the bitstream; 2) *3-dimensional* (3D), which is similar to (1) except that the input transactions are divided to intervals, resulting in a grid with the input stimuli transactions intervals on x axis and bits of the bitstream on y axis. The z axis then shows the number of erroneous output vectors. 3) 2D with stochastic approximation, which is similar to (1) except that the bits are selected *uniformly at random* and the coverage is not 100 %. As we found out in [10], 30 % SEU coverage gave us results varying by only 3.38 % points at maximum. 4) 3D with stochastic approximation, similar to (3). As the raw data obtained from the FT-EST framework is usually very voluminous, a graphical representation is more illustrative. Examples of the graphical representation of the 2D and 3D formats are shown in Figure 3.

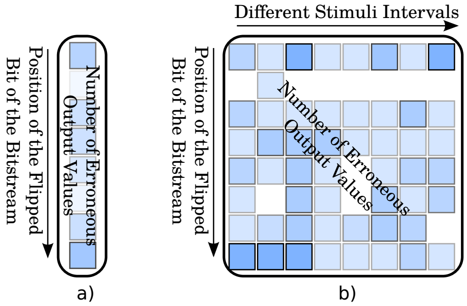
Different Stimuli Intervals
Position of the Flipped Bit of the Bitstream
Number of Erroneous Output Values
Position of the Flipped Bit of the Bitstream
Number of Erroneous Output Values
a)
b)

Figure 3. Two examples of the graphical representation with the description of axis; (a) 2D type with the number of error vectors per bit only; (b) 3D type with the stimuli input transactions divided into intervals, on which the number of error vectors per bit and transaction interval is examined.

To be able to decide whether a given circuit satisfies the requested parameters, we decided to design these indicators, which statistically describe the data obtained in the test phase: 1) the representation of the area without errors (i.e. similar to the critical bit representation); 2) the representation of the largest continuous block of area without errors; 3) the representation of the largest continuous block of erroneous results; 4) the representation of bits leading to constant failure response between intervals; 5) the average representation of erroneous outputs per one bit. The indicators and their usability with the presented data formats are shown in Table I.

TABLE I. THE OVERVIEW OF THE UNITS AND THEIR USABILITY WITH PRESENTED MULTIDIMENSIONAL DATA FORMATS.

		Data Format			
		2D	2D w. Approx.	3D	3D w. Approx.
Unit	Area w/o Errors	✓	✓	✓	✓
	Largest Block w/o Errors	✓	✗	✓	✗
	Largest Block w. Errors	✓	✗	✓	✗
	Constant Response	✗	✗	✓	✓
	Average Errors	✓	✓	✓	✓

IV. CASE STUDY AND EXPERIMENTAL RESULTS

For our case study, we decided to choose two digital circuits. One simpler, which mainly depends on the internal implementation in the HLS tool, and, also, a one, that is composed of more operations. We have chosen a simple *addition* of two 16 bit unsigned integers, which means the input vectors together are 32 bit wide. The *addition* produces one 16-bit unsigned integer on its output. The second circuit implements the *Cyclic Redundancy Check* (CRC) with 8-bit output. The bit width of the input vector of the CRC-8 is also 32 bits. These two circuits were written in the C++ language. In this research, we decided to select the *triple_bit* RDT, which implements the TMR with a bit-based voter component, as the bit-based voter component indicated to be more suitable for this purpose in our previous research paper [16]. The source codes were then modified in such way, that each DT definition, including the input and output interface, was sanitized using the *triple_bit* RDT. We ended up with four C++ implementations, *addition_simple* without any RDT applied (i.e. *simplex*) and *addition_triple_bit* with *data-path* triplicated using RDT approach. Similarly for the *crc8_simple* and *crc8_triple_bit* circuits. We synthesized these C++ source codes using the traditional HLS flow utilizing the Mentor Graphics Catapult C HLS tool [17] to obtain their VHDL RTL implementations. Each version of the circuit was then instantiated inside the FT-EST framework and synthesized using the Xilinx *Integrated Synthesis Environment* (ISE) 14.7 [18]. This way, we obtained a total of four test-beds for our tests. The LUT content sizes of the UUTs are shown in Table II. The tests were held on the ML506 board [19] utilizing Xilinx Virtex 5 FPGA technology.

With the stimuli for the circuits, we tried to explore the whole space of input values, which, in our case, is 2^{32}, as both the circuits have 32 bit wide inputs. However, to minimize the time needed to test all these combinations, we evaluated them from a starting point of 0 to $2^{32} - 1$ with the step of 43, resulting in approximately 100 millions combinations. The length of the step was selected with the binary divisibility in mind, as some steps might result in a lower coverage of the input bits (e.g. the step of 2 would result in an impossibility to set, and thus test the first bit in the input vector). In this research, we focused on the novel approach of 3D format of data, and thus we configured the IGU inside of the FT-EST framework for an ability to select the starting and ending points of the stimuli dynamically from the SW. This allowed us to obtain results individually for each interval. The interval of stimuli was divided into 86 sub-intervals based on the trade-off between the resolution and the time required.

The voluminous data we obtained through the experimentation are demonstrated graphically in Figure 4. As can be seen, for the *addition_simple*, the errors are spread all over the grid, indicating a lower resistance against faults. As the circuit without RDT is fairly simple, a failure inside the *control-path* does not cause a complete failure, in other words, there is a large number of less numerous discrepancies. For the *addition_triple_bit*, the discrepancies are more numerous, however, there is a smaller number of them. A large continuous block of *safe* bits, covering 25% of utilized LUT area, can be recognized here. We believe this large block belongs to bits creating the *data-path* of the circuit, however, as the RDT version is slightly more complex, the *control-path* is more extensive, creating weak points in the circuit. The fact, there is a smaller number of bits creating more numerous errors would support the assumption, that these *sensitive* bits belong to the *control-path*. We are convinced, the bits that create an *alternation* in the error rate are part of the very close end of the *data-path*, causing a *stuck-at* faults in the circuit outputs.

For the CRC-8 circuits, the situation is similar with the bits, we believe belong to the *control-path*, forming a darker places on the screen. However, as can be seen, the biggest difference to the previous circuit is that for the CRC-8 circuits there is no large continuous block of *safe* bits as in the case of the *addition*. This phenomenon can be explained with the fact the CRC-8 circuit is composed of multiple operations, which is blurring the possibility to view the internal context. The important part of the analysis is also a numerical expression of the indicators we proposed in this text. The exact numeric values for the indicators can be seen in Table II.

The main benefit of this evaluation approach is, that, though, it is oriented on the lower-level data format, the circuit parameters can be described by statistical methods, and thus returning the development back to a higher-level. The presented indicators will certainly serve as a starting point for our automation platform. Although meant for a machine processing, if the results are visualized using the representation

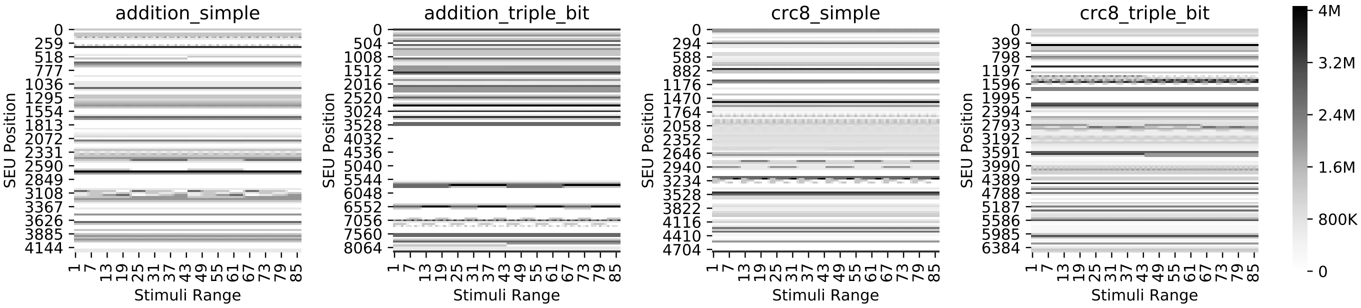

Figure 4. The graphical representation of data obtained through the experimentation; the stimuli intervals are displayed on the *x* axis; bits of the bitstream are displayed on the *y* axis; the darker the point in the grid is, the larger the number of erroneous outputs was observed.

TABLE II. THE EXACT ENUMERATION OF THE PROPOSED CIRCUIT RELIABILITY INDICATORS WITH THEIR PERCENTAGE REPRESENTATIONS.

Indicator	Circuit Implementation							
	addition_simple		*addition_triple_bit*		*crc8_simple*		*crc8_triple_bit*	
	Abs. #	Repr. [%]	Abs. #	Repr. [%]	Abs. #	Repr. [%]	Abs. #	Repr. [%]
Tested LUT bits [b] | [%]	4288 b	100.00 %	8320 b	100.00 %	4800 b	100.00 %	6592 b	100.00 %
Area w/o Errors [b] | [%]	4126 b	96.22 %	8157 b	98.04 %	3823 b	79.65 %	5773 b	87.58 %
Larg. Bl. w/o Err. [b] | [%]	299 b	6.97 %	2102 b	25.26 %	190 b	3.96 %	385 b	5.84 %
Larg. Bl. w. Err. [b] | [%]	2 b	0.05 %	1 b	0.01 %	225 b	4.69 %	34 b	0.52 %
Constant Response [b | [%]	4198 b	97.90 %	8290 b	99.64 %	4069 b	84.77 %	6077 b	92.19 %
Avg. Err. per bit [-] | [%]	1583965 errors	0.04 % out of 2^{32}	996429 errors	0.02 % out of 2^{32}	1612452 errors	0.04 % out of 2^{32}	1568190 errors	0.04 % out of 2^{32}

described in this paper, the method can also be used for designers in the FT system development process.

V. CONCLUSION AND FUTURE RESEARCH

The paper described the approach of accelerated fault tolerance estimation and the concept of *helpers* which for the C++ language are called RDTs. The overview of the automation platform utilizing these techniques was also presented. The main contribution of this paper consists of the proposal of new reliability indicators that are tightly connected to the low-level *functional verification* and *fault injection* approaches, while trying to move them to a higher abstraction level with the use of statistical methods. The indicators were presented on a case study utilizing our previously mentioned RDT FT approach and other components of our FT automation platform.

As a part of our future research, we would like to develop a better strategy to the FT architecture selection than the presented arrangement which was utilizing equivalent FT architecture for each operation.

ACKNOWLEDGEMENTS

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science – LQ1602, the BUT project FIT-S-17-3994 and the JU ECSEL Project SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems), Grant agreement No. 783119.

REFERENCES

[1] J.-C. Geffroy and G. Motet, *Design of Dependable Computing Systems*. Kluwer Academic Publishers, 2002.

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

[3] M. Fingeroff, *High-level Synthesis Blue Book*. Xlibris Corporation, 2010.

[4] R. Leveugle, D. Cimonnet, and A. Ammari, "System-level Dependability Analysis with RT-level Fault Injection Accuracy," in *19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings.*, Oct 2004, pp. 451–458.

[5] M. Radu, "Reliability and Fault Tolerance Analysis of FPGA Platforms," in *IEEE Long Island Systems, Applications and Technology (LISAT) Conference 2014*, May 2014, pp. 1–4.

[6] B. R. Engineering, "Computer Aided Reliability Engineering Software (CARE)," https://www.bqr.com/care/, 2018, accessed: 2018-11-07.

[7] Z. Gao, W. Yang, X. Chen, M. Zhao, and J. Wang, "Fault Missing Rate Analysis of the Arithmetic Residue Codes based Fault-tolerant FIR Filter Design," in *2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*, June 2012, pp. 130–133.

[8] J. Tonfat, L. Tambara, A. Santos, and F. Kastensmidt, "Method to Analyze the Susceptibility of HLS Designs in SRAM-Based FPGAs Under Soft Errors," in *Applied Reconfigurable Computing*, V. Bonato, C. Bouganis, and M. Gorgon, Eds. Cham: Springer International Publishing, 2016, pp. 132–143.

[9] L. A. Tambara, J. Tonfat, A. Santos, F. L. Kastensmidt, N. H. Medina, N. Added, V. A. P. Aguiar, F. Aguirre, and M. A. G. Silveira, "Analyzing Reliability and Performance Trade-Offs of HLS-Based Designs in SRAM-Based FPGAs Under Soft Errors," *IEEE Transactions on Nuclear Science*, vol. 64, no. 2, pp. 874–881, Feb 2017.

[10] 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.

[11] Xilinx Inc., "ChipScope Pro VIO Documentation," https://www.xilinx.com/support/documentation/ip_documentation/chipscope_vio.pdf, Sep. 2009, accessed: 2018-02-15.

[12] 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.

[13] J. Lojda, J. Podivínský, M. Krčma, and Z. Kotásek, "HLS-based Fault Tolerance Approach for SRAM-based FPGAs," in *Proceedings of the 2016 International Conference on Field Programmable Technology*. IEEE Computer Society, 2016, pp. 297–298.

[14] 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. 273–278.

[15] J. Lojda, J. Podivinsky, and Z. Kotasek, "Redundant Data Types and Operations in HLS and Their Use for a Robot Controller Unit Fault Tolerance Evaluation," in *2017 IEEE East-West Design Test Symposium (EWDTS)*, Sept 2017, pp. 359–364.

[16] J. Lojda, J. Podivinsky, Z. Kotasek, and M. Krcma, "Majority Type and Redundancy Level Influences on Redundant Data Types Approach for HLS," in *16th Biennial Baltic Electronics Conference (BEC2018)*, Oct 2018.

[17] M. Graphics, "Catapult HLS," https://www.mentor.com/hls-lp/catapult-high-level-synthesis/, 2017, accessed: 2017-07-07.

[18] Xilinx, "ISE Design Suite," https://www.xilinx.com/products/design-tools/ise-design-suite.html, 2017, accessed: 2017-07-07.

[19] Xilinx Inc., "ML506 Evaluation Platform User Guide," *UG347 (v3. 1.2)*, 2011.