HLS-based Fault Tolerance Approach for SRAM-based FPGAs

Jakub Lojda, Jakub Podivinsky, Martin Krcma, Zdenek Kotasek
Faculty of Information Technology, Brno University of Technology
Bozetechnova 2, 612 66 Brno, Czech Republic
Email: {ilojda, ipodivinsky, ikrcma, kotasek}@fit.vutbr.cz

***Abstract*—This paper presents an approach to fault-tolerant systems design and synthesis based on High-level Synthesis (HLS). A description and evaluation of the impacts of HLS optimization methods are shown as well. The higher reliability is achieved through modification of input description in the C++ programming language on which the HLS synthesis tools are based on. Our work targets SRAM-based FPGAs, which are prone to Single Event Upsets (SEUs). For the evaluation of impacts of HLS optimization methods we use our evaluation platform, which allows us to test fault tolerance properties of the Design Under Test (DUT). The evaluation platform is based on functional verification combined with fault injection.**

Keywords*—*High Level Synthesis, CatapultC, Fault Tolerance, Robot Controller.

I. Introduction

The increase of chip-level integration results in a higher risk of failure. The number of digital systems is penetrating into areas with high demand on reliability, such as medicine, space, industry, is growing as well. Especially in such cases, the reliability is very important, because the consequences of failure can result in injury or heavy financial losses. One of the main approaches of reliability increase is so called fault avoidance. As the name indicates, the main goal is to completely avoid failures in the system using the means of more reliable parts, manufacturing processes etc., which is very challenging and expensive. Another main approach is to use so called fault tolerance [1]. Fault tolerance accepts the fact a fault can appear, but the goal of this approach is to keep the system functional even in the presence of faults, techniques based on redundancy are used for this purpose.

Our research focuses on SRAM-Based *Field Programmable Gate Arrays* (FPGAs). FPGAs are composed of reconfigurable blocks and interconnection network connecting them together. The configuration is saved in the SRAM memory as a bitstream. This results in higher sensitivity of these types of FPGAs to *Single Event Upsets* (SEUs), which are caused by charged particles [2] and can be repaired using *Partial Dynamical Reconfiguration* (PDR) [3]. Many methods to eliminate the impact of SEUs on SRAM-Based FPGAs exist, this paper presents a new one.

The presented method is based on the *High Level Synthesis* (HLS) which is a set of methods transforming a digital circuit description into its RTL representation. At the beginning a designer of the HLS approach prepares a description of the digital circuit. As the input description does not usually contain any information about the timing restrictions, it is up to the synthesis process to infer all the timing information necessary and up to the designer to specify additional constraints. The main decisions such as setting a level of parallel computation of a programming loop (full or partial unrolling) or pipelining a programming loop with an *Initiation Interval* (II) specified are still up to the designer. The II expresses a time between repeated starts of a loop. The *top* of the design is considered an infinite loop. When creating the specification that will be used with HLS, the designer has to keep in mind this fact, e.g. when using C++ as an input description, to use special template data types from supplied library, which allows bit width to be set.

The main idea of HLS is to automate the process of a digital circuit description transformation to the RTL description. As today projects start with some sort of specification and usually an executable model is created, HLS offers an error free way to transform the specification into the RTL description with speeding up the process of design and verification, because the input description can be for example written in an ANSI C or C++ [4].

There are some methods using HLS to improve the level of fault tolerance, for example authors of [5] present an approach to error detection of arithmetic oriented data paths, which operates at the scheduled *Control/Data Flow Graph* (CDFG). As another example, the authors of [6] use HLS to synthesize data paths with concurrent error detection ability. Both of these methods operate on a different level, the method proposed by us operates on the level of the description input source code.

II. HLS-based Fault Tolerance

Our approach is to apply modifications to the specification as the input of HLS. The modifications should produce the resulting RTL description fault-tolerant with resources consumed taken into account. As the modifications are done on the specification level, the result benefits from the advantages of HLS and the set of modifications itself is easier to maintain as well. Our method is based on the modification of the C++ language. There are three types of places the modification can be done at the level of C++ language: data types, arithmetic and logic operations and flow control statements. Our research focuses on finding a method to apply the proper type of modification to the proper instance of the corresponding type while keeping the resources consumed below a desired maximum. As the input description is in the form of an executable code, a profiler tool can be used to determine the frequency of function calls, which could be a good guide to find out the candidates to apply fault tolerance to. The Figure 1 shows the proposed flow.

978-1-5090-5602-6/16/$31.00 ©

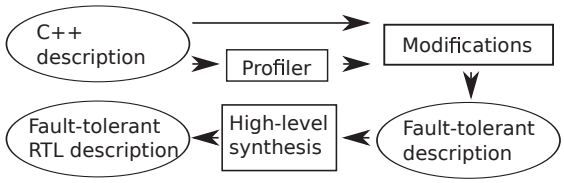

Fig. 1. The new FT approach.

III. Experiments and Results

As our research is in preliminary stage, we focused on experiments with unmodified specification of the robot controller. In our actual experiments, we changed the optimization parameters of the HLS methods and monitored changes in the parameters of the resulting RTL designs. The parameters monitored were the number of slices, slice registers and slice *Lookup tables* (LUTs) occupied.

The next step of our experiment will include the monitoring of the impact of optimization parameters on the susceptibility to failures. The susceptibility to failures will be tested using our Evaluation Platform for Testing Fault Tolerance Methodologies [7], which is based on functional verification and fault injection [8]. The Evaluation Platform uses robot and its robot controller as an experimental system. The goal of the mission of the robot is to find a way through a maze. In these experiments, we swapped the robot control unit, which was originally implemented in VHDL, for a new one, which is synthesized using HLS methods.

The transformation from the description to the RTL level will be made with different optimization preferences set. In this way, we get four different robot controllers, which we are going to evaluate on the Evaluation Platform as a part of our future research. Hundreds of verification runs will be done for each robot controller. Each verification run comprises one passage of the robot through one maze, which is chosen at the beginning of the experiments and remains the same for all verification runs. During verification run a random fault will be injected and the ability of the robot controller to find the right path in the maze will be explored. As a result we will get a number of successful goal achievements for each of the robot controllers. This way the susceptibility to failures will be examined.

The Figure 2 shows the resource requirements for each of the four robot controller units synthezised with different parameters set. The first and the second set of parameters, denoted as *noopt-area* and *noopt-latency*, include area and latency optimizations with no additional requirements added. As can be seen, the results are almost equal, which may be caused by relatively small design size. The third one, *pipeline1-area*, includes the main loop pipelined with II set to 1 and the overall goal set to area. The fourth one, *unroll2-area*, contains the main loop partially unrolled with the level of parallel computation set to 2. As can be seen, the unrolled loop requires more resources as the pipelined one, but it is slightly faster.

The future experiments will be targeted to applying presented modifications which will lead to increased reliability of the robot controller. We will evaluate a susceptibility of modified robot controller to the faults and compare it with the versions without fault tolerance modifications. We will also

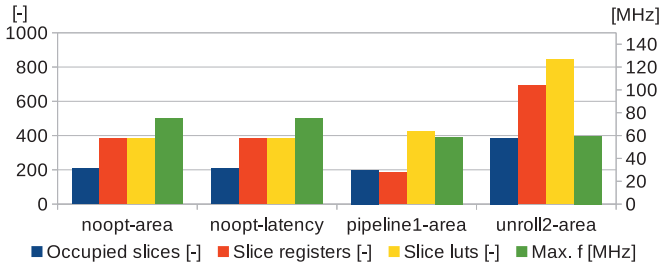

Fig. 2. Comparison of resources consumed for each variant of the HLS synthesized robot controllers.

try to evaluate different modifications of robot controller on achieved level of fault tolerance.

IV. Conclusions and Future Research

In this paper we introduced a newly emerging approach to achieve a certain level of fault tolerance with the usage of HLS. A brief outline to this approach was presented. The experiments described are still in work-in-progress state, as a part of our preliminary work results describing resources consumed to implement various versions of our robot controller unit were shown.

The goal of our future work is to apply various fault tolerance mechanisms to the input description and evaluate the level of fault tolerance achieved in comparison with the additional resources consumed. The objective of our research is to improve this principle to make it generally usable and show its usability on another applications or benchmarks.

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, ARTEMIS JU under grant agreement no 621439 (ALMARVI) and BUT project FIT-S-14-2297.

References

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

[2] D. White, "Considerations surrounding single event effects in fpgas, asics, and processors," http://www.xilinx.com/support/documentation/white_papers/wp402_SEE_Considerations.pdf, Mar. 2012, accessed: 2016-09-15.

[3] XILINX, "Partial Reconfiguration User Guide," http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug702.pdf, Apr. 2012, accessed: 2016-09-15.

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

[5] K. A. Campbell, P. Vissa, D. Z. Pan, and D. Chen, "High-level synthesis of error detecting cores through low-cost modulo-3 shadow datapaths," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 161:1–161:6. [Online]. Available: http://doi.acm.org/10.1145/2744769.2744851

[6] A. Antola, V. Piuri, and M. Sami, "High-level synthesis of data paths with concurrent error detection," in *Defect and Fault Tolerance in VLSI Systems, 1998. Proceedings., 1998 IEEE International Symposium on*, Nov 1998, pp. 292–300.

[7] J. Podivinsky, O. Cekan, J. Lojda, and Z. Kotasek, "Verification of Robot Controller for Evaluating Impacts of Faults in Electro-mechanical Systems," in *Digital System Design (DSD), 2016 19th Euromicro Conference on*. IEEE, 2016, pp. 487–494.

[8] 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.