Design of FPGA-Based Dependable Systems

Martin Straka and Zdenek Kotasek

Brno University of Technology
Faculty of Information Technology
Bozetechova 2, 612 66 Brno, Czech Republic
{strakam,kotasek}@fit.vutbr.cz

Abstract. In this paper, the new methodology from areas fault tolerant systems based on automated generation of checkers in FPGA is presented. Dependability models of architectures based on the use of on-line checkers are described in the paper as well. First, the results of our research in the area of on-line checkers design are described. It is shown how the architectures with on-line checkers can be used in implementing dependable systems into FPGA and required dependability parameters. It is shown how the dependability parameters are derived from the architecture of the system and used for the design of dependable systems into XILINX FPGA.

1 Introduction

With the lower hardware reliability possibly appearing in future technologies, concurrent online testing becomes a strong feature in the design of Fault-Tolerant Systems (FTS). Fault-tolerance (FT) is an important system metric for many operating environments. Different FT architectures are known to improve reliability in real-time systems, Triple Modular Redundancy (TMR) and duplex systems can serve as examples [1]. A possible technique for improving system reliability is through component replication, which usually comes at significant cost: increased design time, testing, power consumption, number of requisite resources in FPGA. As mentioned above, fault-tolerant systems design is one of the areas where on-line testing techniques can be possibly used. One of them, on-line checkers in digital system design can be used for this purposes [2]. Reconfigurable systems implemented using user-programmable logic elements such as FPGA are well suited for applications where high dependability is required [3] [4]. The features of fault-tolerance can be implemented in different levels and applications. In [5], a fault-tolerant approach to reliable microprocessor design is proposed. In [6] the method of highly reliable digital circuit design method based on totally self checking blocks implemented in FPGAs is described. The bases of the self checking blocks are parity predictors. The dependability model and dependability calculations for this method are presented too. The authors in [7] propose a hardware scheme to allow the diagnosis of transient and permanent faults affecting a TMR system implemented by means of FPGA. The scheme allows to easily identify whether a fault affects one of the replicated modules, the voter, or the scheme itself, and whether such a fault is permanent or transient.

2 Motivation for the Research

Hardware units can be implemented on various platforms. From among those which are widely used in many applications, Xilinx FPGA can be mentioned. In our research we tried to evaluate the possibilities of constructing on-line checkers of different functions which can possibly occur in a digital system. The problem we are solving here is how to design FT systems based on FPGA with required dependability parameters.

The problem we try to solve can be also seen as a problem of designing FT system with limited resources available while the required dependability must be guaranteed. We propose the following scheme. Let for the function supposed to be implemented into FPGA various levels of error detection schemes be available, each requiring different amount of resources in FPGA. The function can thus be implemented with various levels of diagnostic security which results in several implementations of the function. Each implementation covers the function completely, the number of errors detected by error detection scheme in each implementation is lower compared with the previous one. Thus, each level requires less resources for the implementation into FPGA than the previous scheme. It can be stated that each level of function implementation is "equipped" with certain level of error detection capabilities which means that different dependability parameters are assigned to each level. The principles of calculating dependability parameters used in this methodology are described later in this paper.

In the paper, the following research activities are described: Our experiments with PSL [8] and FoCs tool, development of ours formal tools allowing to describe properties to be checked by checker and their use in the design of on-line checkers for RTL digital components of various complexity are described in section 3. The principles of digital systems design based on on-line checkers, examples of possible FT architecture together with the evaluation of dependability parameters for them are shown in section 4. Experimental results gained during these activities are presented in section 5. Conclusions and plans for future research are described in section 6.

3 The Use of Checkers In the Design of Fault Tolerant Systems

In the following text, our results gained in the area of design of checkers are briefly described.

3.1 The use of PSL for constructing on-line checkers

In our experiments with PSL it was proven that the use of PSL and FoCs tool for the design of on-line checkers for diagnostic purposes is not an acceptable solution due to the fact that both of these tools are supposed to be used primarily for design verification where the hardware must cover many other additional

functions not needed in our implementations. It was an expected result, the reason why we did this step was to verify this fact and gain some real results based on the implementation into FPGA. It can be summarized that PSL and FoCs tools allow to design checkers for design verification purposes [9] while the design of checkers for diagnostic purposes requires completely different approaches to be used. The results of our research are presented in section 5 - Experimental Results.

3.2 Our approach to the design of on-line checkers for diagnostic purposes

Our methodology is based on constructing checkers for basic digital circuits and their combinations. A methodology allowing to design communication protocol checkers was also developed during our research. For this purpose a specialized language was created which allows to describe properties to be checked, different levels of properties can be described, then it is translated into VHDL checker description created by our software tool. The circuit and its checker are then synthesized into FPGA by XILINX ISE. The methodology is demonstrated in Figure 1. It was described in detail in [10].

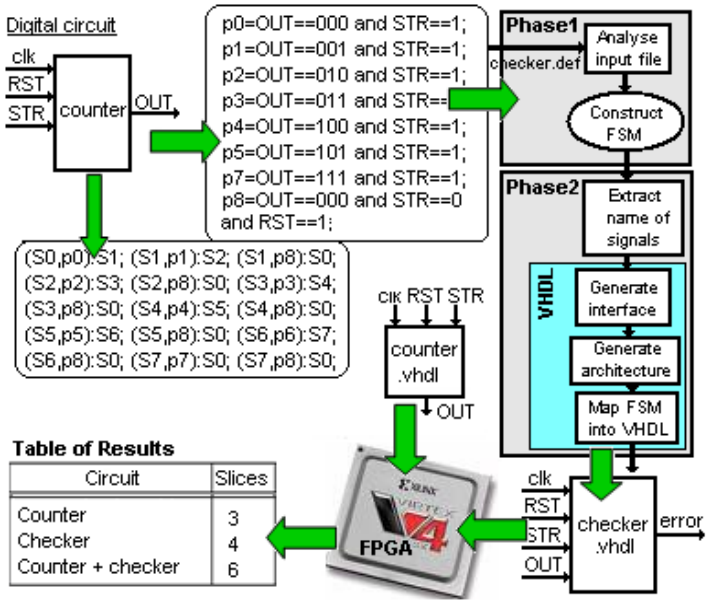

Fig. 1. Demonstration of methodology principles for simple circuit

The size of the checker is the basic criterion of our approach - the checker implementation should not be bigger than the function supposed to be checked. The results of our research are presented in [11].

4 Dependability Models for FT Architectures

To construct FT systems, TMR or duplex architectures are typically used [12]. If implemented into FPGA, then permanent faults which appear in the design can be removed through reconfiguration, the damaged configurable logic is not used by the design. No problem exists if there is enough space available in FPGA - bigger than the design being reconfigured - so that the design can be reconfigured into free space. However, in designs which require more FPGA resources, fault-free elements can be exhausted. This problem can be even stronger for mission oriented applications where fault-free elements can be exhausted due to multiple reconfigurations.

Our methodology is based on the existence of pre-designed configurations, each configuration covers the required function completely but it has different ability to detect faults. The aim to increase availability is an important goal of the methodology. It means that we try to reduce the need for reconfiguration. The problem of availability can be seen on the example of TMR and duplex architectures. When an error appears in each system, the TMR can still function, while the duplex system is not able to operate and must be recovered. This problem can be removed by using TMR and duplex FT architectures based on on-line checkers. Example of configuration of FT architectures with different level of dependability is shown in Figure 2. In our research, we do not solve the problem of faulty voters and checkers. They must be designed as Totally Self-Checking components, they must be able to detect faults in their internal structures.

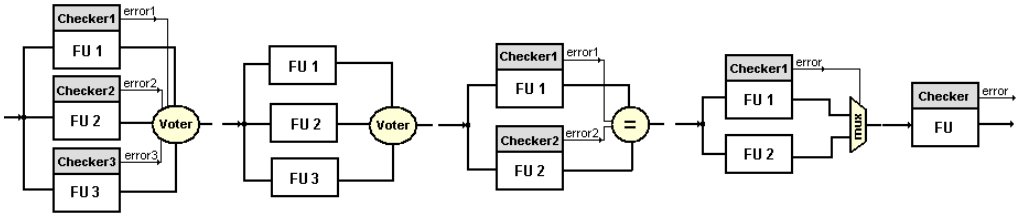

Fig. 2. Example of configuration of FT architectures

The architectures discussed above should be seen as an example of architectures with different level of dependability, each implemented as FU(s) with checkers. It is important to say that the sequence of architectures can possibly consist of architectures different from those demonstrated here. What we show here should be seen as an example.

Methodology of generating configuration of FT architectures is shown in Figure 3. The methodology which has as its input FPGA area and dependability parameters of the system implementation to be achieved (they are defined by the user). It consists of two phases:

PHASE 1 - the development of the architecture with required dependability parameters and availability being one of the goals, and PHASE 2 - the sequence

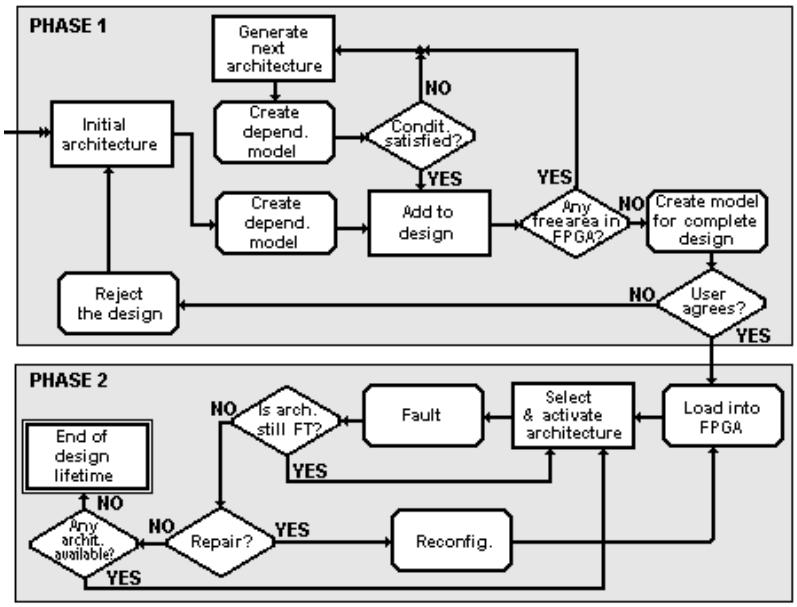

Fig. 3. Demonstration of methodology principles

of actions in FPGA during the lifetime of the architecture including the repairs needed to be done to recover from failures. The initial architecture of the function is chosen (in terms of the redundancy and number of checkers) by the user and dependability parameters calculated. For the first implementation, the size of FPGA resources needed to implement the architecture is evaluated. Since now, this architecture is seen as the initial architecture, the following architecture will reflect this fact - its size of resources and dependability parameters will be lower due to lower ability of checkers to detect failures of FUs.

From the dependability parameters of the whole system, dependability parameters of partial configurations are derived first. They are then used for the decisions whether the partial configuration can be used as a possible solution. As soon as all partial configurations are developed, they are accepted as the proper solution if they satisfy user requirements. If these requirements are not satisfied, the solution is rejected and the process must be repeated. If the solution is acceptable, it can be loaded into FPGA. The dependability models for several examples are described now.

In our methodology, Markov dependability model was used. In the model, two states are distinguished: ready-to-operate (circle) and disabled (square). The states are connected by edges, each marked with a symbol reflecting the intensity of failures (λ). From among other parameters, the following ones are important for the methodology: R(t) - the probability of fault-free state of the system, Q(t) - failure probability, Ts - mean time to failure. In our methodology, dependability parameters of the whole system must be derived from dependability parameters

of single architectures. The system, where failing architecture is replaced by non-failing architecture can be seen as parallel system, thus the following formula can be used:

$$R(t) = 1 - \prod_{i=1}^{n}(1 - Ri(t)),\ where\ i\ denotes\ the\ number\ of\ architectures,$$

$$Ts = 1/\lambda \sum_{k=1}^{n}(1/k),\ where\ k\ is\ the\ number\ of\ architectures.$$

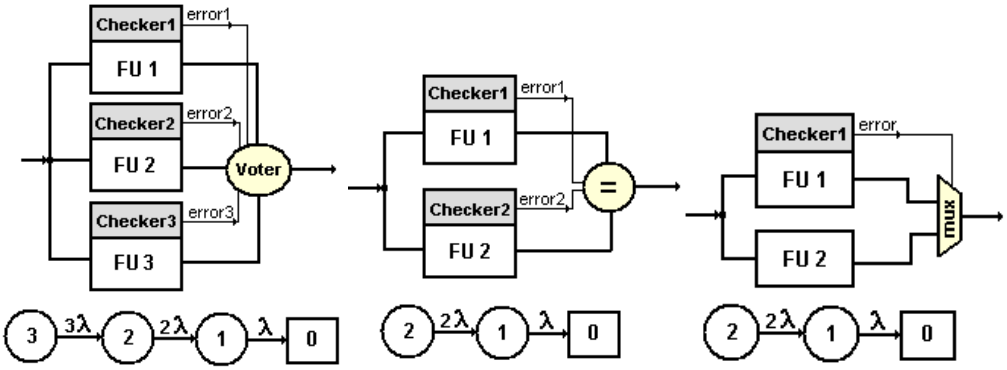

Fig. 4. TMR and Duplex architectures based on checkers with dependability models

For initial architecture, TMR system with additional checkers was chosen (Figure 4). This architecture is fault tolerant even when two modules fail which does not hold for classical TMR architecture (the availability of this architecture is higher). The following architecture is based on duplex system with checkers (Figure 4). Both architectures differ in the size of FPGA resources needed to implement the architecture. From the dependability model the following formulas were gained for both architectures:

TMR+3CH	**Duplex+2CH**
$p3'(t) = -3\lambda p3(t)$	$p2'(t) = -2\lambda p2(t)$
$p2'(t) = 3\lambda p3(t) - 2\lambda p2(t)$	$p1'(t) = 2\lambda p2(t) - \lambda p1(t)$
$p1'(t) = 2\lambda p2(t) - \lambda p1(t)$	$p0'(t) = -\lambda p0(t)$
$p0'(t) = -\lambda p0(t)$	
$p3(t) = e^{-3\lambda(t)}$	$p2(t) = e^{-2\lambda(t)}$
$p2(t) = -3e^{-3\lambda(t)} + e^{-2\lambda(t)}$	$p1(t) = -2e^{-2\lambda(t)} + e^{-\lambda(t)}$
$p1(t) = 3e^{-3\lambda(t)} - 2e^{-2\lambda(t)} + e^{-\lambda(t)}$	
$R(t) = p3(t) + p2(t) + p1(t)$	$R(t) = p2(t) + p1(t)$
$Ts = (1/3 + 1/2 + 1) * 1/\lambda$	$Ts = (1/2 + 1) * 1/\lambda$

where $pi(t)$ is a probability that the system is in p status in time t and λ is intensity of failures based on technology.

5 Experimental Results

Experiments with the implementation of above described architectures were done. The experiments were performed on XILINX FPGA platform. The components were synthesized into Virtex5. We compared the number of slices needed to cover the function and for checker implementation for simple digital components like counters, decoders, serialiser and combinations of them. We performed additional experiments to compare FT techniques, namely TMR with duplex techniques based on the use of checkers. The following structures were implemented: TMR system with checkers (TMR+3CH), TMR system, duplex system with one checkers (Duplex+1CH), duplex system with two checkers (Duplex+2CH) and simple circuit with checker. The number of resources for several combinations of components is summarized Table 1. For some components the sources needed for checker implementation is not significantly higher than the sources needed for the functional unit. Comparison of results for checker created by FoCs tool and our methodology are summarized in Table 2.

Virtex5 - XCV50E circuit	**TMR+3CH** [slices]	**TMR** [slices]	**Duplex+2CH** [slices]	**Duplex+1CH** [slices]	**Simple+CH** [slices]
Counter	21	9	15	11	6
Decoder	27	20	20	18	8
Counter+decoder	30	20	26	22	11
Serialiser	19	12	14	13	6
Shift register	24	13	18	15	7

Table 1. Resources for TMR and duplex architectures in Virtex5

Virtex5 - XCV50E circuit	**Checker developed by FoCs** [slices]	**Checker developed by our tool** [slices]
Counter - full checking	92	7
Counter - only states	48	4
Decoder - full checking	66	5
Shift - only states	52	4

Table 2. Resources needed for checker in Virtex5

6 Conclusions

In this paper, a technique for automated design of FT architectures based on checkers for different types of digital components and its combinations is presented. Tool needed for the research which aims at developing methodology allowing to design FT systems into FPGA were developed. Experiments with our methodology and PSL were performed and experience gained. Formal tool for the description of conditions to be checked in communication protocols and in the design of digital components was developed and implemented. Experiments with the FT architectures and with the implementation of checkers into FPGA were done, the results support the direction of our research.

Acknowledgements

This work was supported by the Research Project No. MSM 0021630528 - Security-Oriented Research in Information Technology and by GACR project No. 102/05/H050 - Integrated Approach to Education of PhD Students in the Area of Parallel and Distributed Systems (Grant Agency of the Czech Republic).

References

1. Shu-Yi Yu and Edward J. McCluskey: On-line Testing and Recovery in TMR Systems for Real-Time Applications. In: ITC '01: Proceedings of the 2001 IEEE International Test Conference. (2001). p.204-249.
2. Fernanda Gusmao de Lima Kastensmidt and Gustavo Neuberger and Renato Fernandes Hentschke and Luigi Carro and Ricardo Reis: Designing Fault-Tolerant Techniques for SRAM-Based FPGAs. Journal: IEEE Des. Test. (2006) p.552–562.
3. Pontarelli, S., Cardarilli, G.C., Malvoni, A., Ottavi, M., Re, M., Salsano, A.: Reconfigurable Architecture for Autonomous Self-Repair. Journal: IEEE Des. Test. (2004). p.185–189.
4. Cristiana Bolchini and Antonio Miele and Marco D. Santambrogio: TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs. In: DFT '07: Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems. (2007). p.87–95.
5. Chris Weaver and Todd M. Austin: A Fault Tolerant Approach to Microprocessor Design. In: DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS). (2001). p.411–420.
6. Pavel Kubalik and Radek Dobias and Hana Kubatova: Dependable Design for FPGA Based on Duplex System and Reconfiguration. In: DSD '06: Proceedings of the 9th EUROMICRO Conference on Digital System Design. (2006) p.139–145.
7. Sergio D'Angelo and Giacomo R. Sechi and Cecilia Metra: Transient and Permanent Fault Diagnosis for FPGA-Based TMR Systems. In: DFT '99: Proceedings of the 14th International Symposium on Defect and Fault-Tolerance in VLSI Systems. (1999). p.330–338.
8. IBM: PSL/Sugar-based Verification Tools. In: http://www.haifa.ibm.com/projects/verification/sugar/index.html. (2008).
9. Marc Boule and Zeljko Zilic: Automata-based assertion-checker synthesis of PSL properties. In: ACM Trans. Des. Autom. Electron. Syst. (2008). p.1–21.
10. Martin Straka and Jiri Tobola and Zdenek Kotasek: Checker Design for On-line Testing of Xilinx FPGA Communication Protocols. In: DFT '07: Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems. (2007). p.152–160.
11. Martin Straka and Zdenek Kotasek and Jan Winter: Digital Systems Architectures Based on On-line Checkers. In: DSD '08: Proceedings of the 11th EUROMICRO Conference on Digital System Design. (2008). p.81–97.
12. Roystein Oliveira and Aditya Jagirdar and Tapan J. Chakraborty: A TMR Scheme for SEU Mitigation in Scan Flip-Flops. In: ISQED '07: Proceedings of the 8th International Symposium on Quality Electronic Design. (2007). p.905–910.