State Synchronization of Faulty Soft Core Processors in Reconfigurable TMR Architecture

Karel Szurman
5th year, part-time study
Supervisor: Zdeněk Kotásek

Faculty of Information Technology, Brno University of Technology
Božetěchova 1/2, 612 66 Brno
iszurman@fit.vutbr.cz

***Abstract*—Fault-tolerant systems implemented into SRAM-based FPGA are frequently protected by combination of triple modular redundancy and partial dynamic reconfiguration. When a part of the SRAM configuration memory with the copy of the protected circuit is reconfigured on the run, the system restart is the easiest way how to bring all three copies of the circuit back to fully synchronous and operating state. Soft core processors are complex systems which require more precise technique for synchronization of the system state space and data gained from previous calculations without disruption of processors functionality and executed program. This paper presents current state of our research focused on the state synchronization methodology for soft core processors.**

Keywords—Fault tolerant system, FPGA, state synchronization, partial dynamic reconfiguration, recovery, soft core processor.

I. INTRODUCTION

Emerging technologies used in avionics and space systems have growing demands on computing frequency and data throughput. Examples of such systems can be LiDAR (Light Detection and Ranging) system for 3D sensing of the Earth surface in real time, software defined radio system for space telecommunication or research satellite using a number of highly accurate sensors for data acquisition during its exploration mission. These digital systems are usually based on combination of Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuits (ASICs) and custom-made electronic hardware. Avionics and space systems are safety-critical systems the failure of which can lead to catastrophic consequences. These systems are exposed to various failure conditions during their lifetime. Radiation effects, caused by energetic particles in the space radiation environment, are one of the most serious. Nowadays, SRAM-based FPGAs become broadly used inside these systems for their low price, high performance, ability for reprogramming and flexibility even when they are very sensitive to radiation effects and mainly to Single Event Upset (SEU) effect. SEUs can cause changes in a state of bistable element and affect configuration memory and user logic. Usage of SRAM-based FPGA requires implementation of SEU mitigation technique and employing of fault tolerance strategy to operate correctly even in the presence of failure.

Two main SEU mitigation strategies for Fault Tolerant Systems (FTSs) exist [1]. Both approaches employ hardware redundancy. The most used form is Triple Modular Redundancy (TMR) due to its fault-masking ability, possibility to scale the TMR protection by changing its granularity, tolerable overhead and the availability of tools for automated TMR generation [2]. The first technique, referred as scrubbing, is based on periodic writing a known copy of the original bitstream to configuration memory in order to correct corrupted bits. The copy is stored usually in external radiation-hardened memory. The scrubbing has various implementations and it is also often combined with hardware redundancy applied automatically on netlist of the circuit. The second technique is based on the usage of hardware modular redundancy together with Partial Dynamic Reconfiguration (PDR). In our terminology, TMR consists of replicated circuits, the circuits contain components. Fault detection in a TMR is operating by means of majority voting from copies of protected circuit. When a failure caused by SEU in one of the circuit copies is detected then corresponding TMR module located in FPGA configuration memory is reconfigured through PDR process. In our research, the PDR is controlled by Generic Partial Dynamic Reconfiguration Controller (GPDRC) [5]. After the faulty circuit reconfiguration is finished, its operational state is not up-to-date and needs to be synchronized with correctly operating circuit copies in TMR architecture before it is incorporated back into the system.

The aim of our research is to propose a new methodology for a state synchronization of faulty soft core processors in reconfigurable TMR architecture implemented into SRAM FPGA. We targeted soft core processors as a computation platform alternative to microprocessor, DSPs or general-purpose processor, which can be synthetized for any FPGA design and easily modified and customized by implementation of additional features or application of a specific optimization. The new methodology should complete existing FT methodologies which address fault masking, fault detection and fault recovery based on hardware redundancy and PDR.

II. State Synchronization Methodology

At the beginning of the research, we focused on methods for state synchronization for digital system implemented by random logic and state machines, synchronization strategies and parameters which could be evaluated. Then, we started to deal with the state synchronization for soft-core processors as the platform with complex functionality and structure where the state synchronization is more demanding.

A. Initial development phase

Results gained during the initial development phase of our state synchronization methodology are as follows [3] [4]:

- We defined a set of parameters (criteria) for evaluation and optimization of various state synchronization techniques.
- We developed fundamental state synchronization methodology and described the basic principles of state synchronization.
- We implemented reconfigurable fault tolerant CAN bus control system together with implementation of specific synchronization strategy based on our methodology.

We realized that the principles of state synchronization and its implementation have a strong impact on the FTS and its parameters. Therefore, all aspects including requirements on its real time behavior, principles of performing its function, the type and volume of the synchronized context must be taken into account when the method of a state synchronization after fault occurrence is developed.

B. State synchronization parameters

As the first, we identified set of dynamic and static parameters. The dynamic parameters reflect the impact of the synchronization on operation and function of the system and overall timing. According to the synchronization impact, the methods can be divided into function blocking and function non-blocking methods. Another dynamic parameter is the time needed to perform the synchronization which is closely associated with other parameters, requirements on the synchronization implementation and the volume of data which needs to be synchronized. On the other hand, the static parameters have an indirect impact on system features and have a close relation to an algorithm used to implement the synchronization procedure. This set of parameters include area and FPGA resource overhead, the power demands and reliability of implemented synchronization.

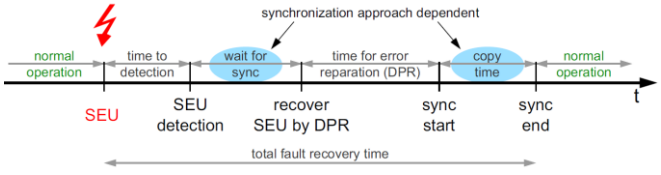

Figure 1: SEU recovery process [2]

The time needed for system resynchronization is the one of the most important criteria since the goal is to maximize Mean Time Between Failures (MTBF) metric. The overall timing for SEU recovery process in shown in Figure 1. The total fault recovery time is given by the sum of time durations of all following recovery phases:

- the time needed for error detection,
- the wait for synchronization time,
- the reconfiguration time for repairing the SEU,
- the synchronization time.

The time needed for error detection is the time between SEU occurrence and the moment when an error caused by this fault is detected in majority voter. The reconfiguration time is proportional to the size of reconfigured partition and speed. The wait for synchronization time is the time needed for finishing ongoing calculations. The synchronization time is the time required to copy the correct values from reference circuit instance to a recently reconfigured circuit.

C. Essential design considerations

We determined the set of essential design considerations which must be satisfied by the implemented synchronization method. The design considerations are following:

1) The selection of the state in which the synchronization of a reconfigured circuit copy will be performed – the proper state must be considered with respect to the synchronization method feasibility, real-time requirements, the system architecture and data consistency.
2) The definition of the system context which will be used for the synchronization – the context is defined by data type and its volume, the two types of data can be distinguished: the data which are processed and reproduced (i.e. application-related data) and the data which are important for system function and operation (i.e. system related data).
3) The design of the interconnection between redundant components and the control mechanism which will be needed for performing the synchronization process.

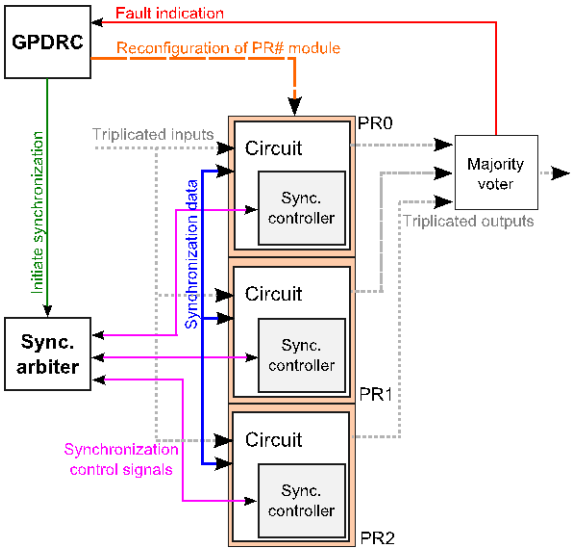

Figure 2: Generic architecture for state synchronization

D. *State synchronization implementation*

We designed generic architecture for synchronization with arbiter and controllers [3]. This architecture is shown in Figure 2. It was designed on the basic of consideration that the FTS architecture can consists of various components with complex hierarchy and in each of components, registers with data of the system state can be stored. For this reason, the synchronization process of a reconfigured circuit copy in the TMR architecture has to be controlled on two levels. From the outside, on the level of individual circuits, and inside on the level of circuits for synchronization of their internal components.

The basic principle of the fault recovery strategy is as follows. A protected circuit is implemented within coarse grained TMR architecture. Majority voter performs voting from input signals and indicates any mismatch to GPDRC and initiate reconfiguration of faulty circuit. After the process of reconfiguration is finished, GPDRC indicates the index of reconfigured unit to the synchronization arbiter which needs to be synchronized. The arbiter determines the roles of replicated units in TMR architecture (it specifies which of them is synchronized, referenced or paused during the synchronization process). The arbiter controls all state synchronization phases for components and data objects inside the synchronized circuit. In the end, the arbiter exits the synchronization process and switches the circuit copies back into its operational mode.

The crucial task for state synchronization strategy is the implementation of the interconnection between redundant modules which must be designed with compromise between FPGA resource utilization overhead, implementation complexity and the goal to complete the synchronization in the shortest possible time.

III. RECONFIGURABLE FT CAN BUS CONTROL SYSTEM

The proposed state synchronization methodology for digital system implemented on random logic and state machines was evaluated by experiments with reconfigurable FT CAN bus control system, published in [3] and [4].

IV. FAUL TOLERANCE AND SYNCHRONIZATION OF SOFT CORE PROCESSORS

Our current research is focused on soft core processors, aspects of their fault recovery with the usage of PDR process and state synchronization. Till now, we were considering only coarse grained TMR architecture for protection of target system in our experiments. The scheme of processors protected by coarse grained TMR and the illustration of fault recovery strategy is shown in Figure 3.

Moreover, our goal is also to explore possibilities of fine grained TMR architecture used for protecting internal components within soft core processor.

A. *The state of the art methods*

The paper [2] describes methods for synchronization of faulty processors in coarse grained TMR architecture. The aim of authors was to research essential steps for synchronization in more details since this topic is insufficinelty addressed by researchers studying various FT methodologies and PDR. They proposed and evaluated four different methods for Xilinx PicoBlaze soft core processor which span from an implementation with minimal hardware overhead to completely hardware-based synchronization technique.

- Synchronization by reset – processors are brought into synchronized and known state of program execution by the system reset.
- Synchronization with shared memory – processors are synchronized by concurrent writing into TMR protected memory by all processors, followed by a concurrent reading of the data. Synchronization has to be triggered externally by program at the moment when no interrupt is executed.
- Synchronization with shared memory driven by interrupt – processors are synchronized by interrupt and it can be performed almost immediately after reconfiguration is finished. However, it requires hardware synchronization of the stack context.
- Complete hardware-based synchronization – processors are synchronized through hardware synchronization interface which allows copying of processor core registers with flags, stack pointe, the stack data and the scratchpad memory stored in block RAM (BRAM). Multiplexers and synchronization counters are implemented for each memory element enabling synchronization of one element in one cycle.

The evaluation of synchronization methods and experiments demonstrate significant increase in utilization FPGA resources and impact on the system frequency for synchronization methods exploiting hardware implementation and modifications inside processors [2].

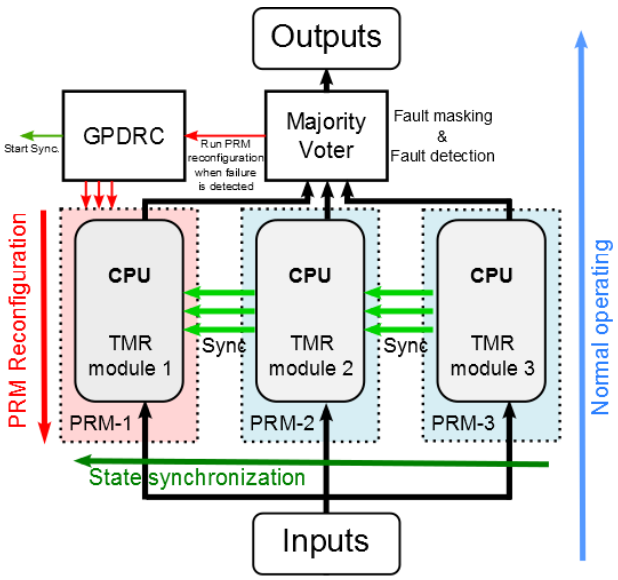

Figure 3: Recovery process for soft core processors protected by TMR architecture

B. *Design of soft core processor synchronization*

We evaluated several open-source soft core processors available for our research including LEON3, Plasma, ZPU and neo430. The neo430 was selected as the main platform for our experiments since it is a small, powerful and customizable 16-bit soft-core microcontroller, compatible to TI's MSP430 [6]. This microcontroller uses separated instruction and data memory and integrates high precision timer, watchdog timer, serial interface UART, GPIOs, Wishbone bus interface and internal bootloader.

Synchronization procedure for a soft core processor must synchronize all processor registers, program stack pointer and stack data. Moreover, instruction and data caches and program memory can be synchronized.

The scope of synchronization strategy depends on soft core processor complexity and application requirements for the synchronization itself and the recovery process.

V. PH.D. THESIS GOALS

The aim of our research is to propose new methodology for design and implementation of state synchronization procedure for reconfigurable FTS based on soft core processors protected by coarse grained or fine grained TMR architecture with respect to defined criteria for state synchronization procedure.

Recent results and goals satisfied during previous research were summarized in section II.A. The future goals of the research and Ph.D. thesis are as follows:

1. Implementation of CAN bus control system in neo430 soft core processor protected by coarse grained reconfigurable TMR architecture. Design state synchronization method, perform experiments and compare results with previous hardware implementation of the FT CAN bus control system.
2. Implementation of the robot controller algorithm [7] in neo430 soft core processor protected by fine grained reconfigurable TMR architecture. The design of state synchronization method and execution of the experiments. The motivation is also in further participation on collective research activities within Fault Tolerant Systems Design, Diagnostics and Testing group at Brno FIT. The robot controller is an integral part of the verification environment for evaluating impacts of faults in electro-mechanical systems.
3. Comparison and generalization of advantages and disadvantages of various state synchronization methods for soft core processor with respect to selected granularity of the TMR architecture based on performed experiments.
4. Evaluation of various synchronization methods for soft core processors against defined design criteria for synchronization method and execution (impact on the system function, speed, overhead).
5. Evaluation of reliability of unprotected soft core processor and soft core processor protected by coarse grained and fine grained TMR architecture, the system with and without reconfiguration and implementation of a state synchronization strategy.
6. Comparison of different ways for construction of synchronization circuit for digital system based on random logic and soft core processors.

VI. CONCLUSION AND FUTURE RESEARCH

In this paper, the present research related to development of new state synchronization methodology for FTS based on soft core processors was described.

The future work will be mainly focused on the implementation of various synchronization methods for soft core processors with respect to selected TMR architecture granularity, defined synchronization criteria and the evaluation of experiments and results with the aim to generalize gained knowledge for any soft core processor platform.

VII. ACKNOWLEDGEMENTS

This work was supported by The Ministry of Education, Youth and Sports from the National Program of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602. This work was also supported by Brno University of Technology under number FIT-S-14-2297 and by ARTEMIS JU under grant agreement no 641439 (ALMARVI).

REFERENCES

[1] Siegle, F.; Vladimirova, T.; Ilstad, J.; Emam, O.: Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications. ACM Comput. Surv. 47, 2, Article 37, pp 1-34, January 2015. ISSN 0360-0300.

[2] Kretzschmar, U.; Gomez-Cornejo, J.; Astarloa, A.; Bidarte, U.; Del Ser, J.:Synchronization of faulty processors in coarse-grained TMR protected partially reconfigurable FPGA designs. Reliability Engineering & System Safety, Volume 151, 2016, pp.1-9. ISSN 0951-8320.

[3] Szurman, K.; Mičulka, L.; Kotásek, Z.: State Synchronization after Partial Reconfiguration of Fault Tolerant CAN Bus Control System. 17th Euromicro Conference on Digital Systems Design. Verona: IEEE Computer Society, 2014, pp. 704-707. ISBN 978-0-7695-5074-9.

[4] Szurman, K.; Mičulka, L.; Kotásek, Z.: Towards a State Synchronization Methodology for Recovery Process after Partial Reconfiguration of Fault Tolerant Systems. 9th IEEE International Conference on Computer Engineering and Systems. Cairo: IEEE Computer Society, 2014, pp. 231-236. ISBN 978-1-4799-6593-9.

[5] Mičulka, L.; Kotásek, Z.: Generic Partial Dynamic Reconfiguration Controller for Transient and Permanent Fault Mitigation in Fault Tolerant Systems Implemented Into FPGA. 17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, Warszawa, PL, 2014, pp. 171–174, ISBN 978-0-7695-5074-9.

[6] Nolting, S.: The NEO430 Processor - A small, powerful and customizable open-source 16-bit soft-core microcontroller, compatible to TI's MSP430 ISA. https://opencores.org/project,neo430, available online, april 2017.

[7] Podivínský, J.; Čekan, O.; Lojda, J.; Kotásek, Z.: Verification of Robot Controller for Evaluating Impacts of Faults in Electro-mechanical Systems. Proceedings of the 19th Euromicro Conference on Digital Systems Design. Limassol: IEEE Computer Society, 2016, pp. 487-494. ISBN 978-1-5090-2817-7.