In-circuit Error Detection with Software-based Error Correction - An Alternative to TMR

Gökçe Aydos[1,2] and Görschwin Fey[1,2]

[1] DLR, Institute of Space Systems, Robert-Hooke-Str. 7, 28359 Bremen Germany
[2] University of Bremen, Institute of Computer Science,
Bibliothekstr. 1, 28359 Bremen Germany
goekce@cs.uni-bremen.de

Abstract. FPGAs are often utilized in space avionics. To protect the FPGA application data against radiation effects in space, data redundancy can be used. A well-known method is to triplicate the circuit and eliminate the erroneous circuit output with a local voter (TMR). Alternatively, in-circuit error detection with software-based error correction can be used, if the FPGA works as a co-module next to a processor running the mission software. In this work, we present an implementation of this method on a commonly used spacecraft data handling architecture.

Keywords: fault tolerance, FPGA

1 Introduction

Field-programmable gate arrays (FPGAs) are often utilized in space avionics. FPGAs involved in mission critical applications implement fault tolerance mechanisms. One of the reasons is the ionizing radiation in space, which can flip stored bits in *flip-flops* (FFs). In the best case, the flipped bits are masked and overwritten in next cycles, having no effect on the system. In the worst case, this leads to catastrophic failures [4].

Tolerance against bitflips can be implemented using redundancy in space or time, i.e., by instantiating multiple entities of one circuit (space), or repeating the same operation for multiple clock cycles on one circuit (time), or also by combining the both (spacetime) [3]. Mission critical FPGA applications often use space redundancy, mostly in form of *triple modular redundancy* (TMR).

In TMR, a module, e.g. a FF or a whole circuit, is triplicated and the outputs are connected to a voter, which drives the correct output value in case of a failure in a single module. This facilitates correction of an error in the same clock cycle, i.e., by only using space redundancy. In presence of tight space constraints, this overhead can turn into a hurdle for fulfilling the design timing closure and area requirements. Alternatively, if the overhead of time redundancy is feasible for an application, the redundancy in space can be reduced and, in return, the redundancy in time dimension can be increased. We propose an instantiation of this approach where error detection is done in space, and error correction in time, only in case of an error. In a system constellation with a processor and an FPGA,

the on-demand time redundancy can be easily implemented in software. We call this method *in-circuit error detection with software-based error correction*. In the following, a common spacecraft *on-board data handling* (OBDH) subsystem architecture and an implementation of this method on this architecture will be shown.

2 Application to a Typical OBDH Architecture

The OBDH subsystem of a spacecraft typically handles the communication from the ground station to other subsystems. A simplified model of a typical OBDH architecture is shown on Fig. 1.

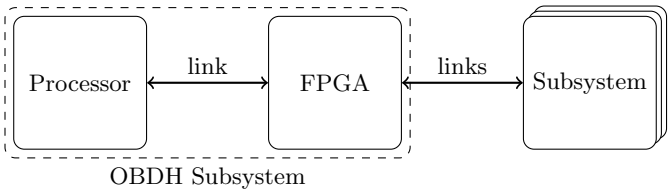

Fig. 1. Simplified model of a typical OBDH architecture.

The processor runs the mission software and the FPGA implements interface protocol circuits required by various subsystems on board of a satellite. The processor uses the FPGA for communicating with the subsystems.

The FPGA implements the memory-mapped interfaces for the subsystems. Consequently, from the software point of view, the FPGA is a remote memory. To access the remote memory, the software sends memory requests to the FPGA and the FPGA replies every request with a response. If a particular request is not responded (within a timeout), then the software can retry the last request. There are two kinds of memory requests, write and read. Every request can contain memory accesses to particular memory addresses.

In our model, the FPGA design to which the fault tolerance method is not applied, consists of three circuits (A), (B) and (C), shown as white boxes in Fig. 2. (A) stores the requests from the software and responses sent from (B). (B) transforms the requests from the software to actual memory signals for (C).

We assume that the processor, the subsystems, and the circuits (A) and (C) are reliable, i.e., immune against bitflips. So, only (B) has to be hardened.

The gray boxes in Fig. 2 show the circuits for error detection and handling after hardening the design. The error detection circuit checks for data integrity in (B) to detect bitflips, e.g., by using *concurrent error detection* (CED) [2]. An example for CED is to generate parity for every register in (B) and check for data integrity in the next cycle. In case of detected bitflips, the error handling module engages the countermeasures. In this implementation, the error handling

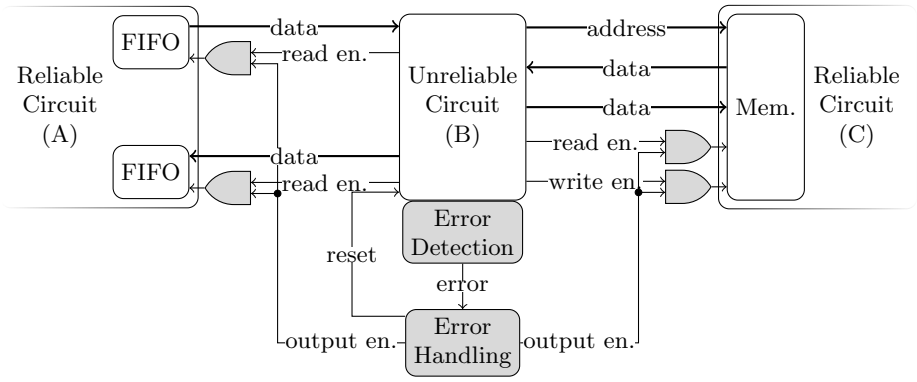

Fig. 2. In-circuit error detection applied on the unreliable circuit (B).

module masks the output signals of (B) and resets it. The masking occurs in one clock cycle, so a fault in (B) does not propagate to the neighboring circuits, i.e., the circuit is isolated.

Through the specified timeout, the software is always aware of a failure in (B). Upon failure of (B), a request is repeated and no request gets lost.

The shown hardening method is useful on circuits like protocol converters, where the circuit acts as an intermediate module. If the circuit interface includes control signals (e.g., write/read enable in Fig 2) which are maskable, then the error can be masked using a relatively short path compared to correcting the error by resetting the circuit.

Like TMR, the method can be applied after the behavioral synthesis on the register transfer level, therefore it is transparent to the application. If the communication protocol between the software and the circuit permits a timeout, then the method is also transparent to the software.

This method concentrates on the circuit bits of an FPGA application and not on the configuration bits, where the FPGA application itself is stored. It is crucial to protect also the configuration bits from the ionizing radiation, if an SRAM-based FPGA is used. If a flash-based FPGA is used, the bitflips on the configuration are negligible [1].

References

1. Battezzati, N., Sterpone, L., Violante, M.: Reconfigurable Field Programmable Gate Arrays for Mission-Critical Applications, chap. 7. Springer (2011)
2. Mitra, S., McCluskey, E.J.: Which concurrent error detection scheme to choose? In: International Test Conference Proceedings. pp. 985–994. IEEE (2000)
3. Nicolaidis, M.: Time redundancy based soft-error tolerance to rescue nanometer technologies. In: 17th IEEE VLSI Test Symposium. pp. 86–94 (1999)
4. Petersen, E.: Single Event Effects in Aerospace, chap. 1. John Wiley & Sons (2011)