Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-6 (2018) 348–353

Implementation and Analysis of Methods for Error Detection and Correction on FPGA

M. Dug*, M. Krstic**
D. Jokic***

**Department of Electrical and Electronic Engineering, International Burch University, Sarajevo, Bosnia and Herzegovina, (email: mehmed.djug@outlook.com)*
*** IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany (e-mail: krstic@ihp-microelectronics.com)*
**** Department of Electrical and Electronic Engineering, International Burch University, Sarajevo, Bosnia and Herzegovina (e-mail: dejan.jokic@ibu.edu.ba)*

Abstract: In this paper, two methods are implemented and analyzed on a Field Programable Gate Array (FPGA) board for the design of fault-tolerant pipelined sequential and combinational circuits. Evaluated methods are named Error Detection and Partial Error Correction (EDPEC) and Full Error Detection and Correction (FEDC). The mentioned methods are based on Error Detection Logic (EDC) in the combinational circuit part combined with fault tolerant master-slave flip-flops with fault tolerant memory elements. Additional to the analysis and implementation of the methods, the enhancement to a method is proposed.

Keywords**:** Soft errors, combinational logic, sequential logic, TMR, FPGA, EDPEC, FEDC.

1. INTRODUCTION

Today as the technology and science are developing fast, the reliability of the electronic components is in focus of the industry and researchers. The fault tolerance requirements have been important in the fields of space industry, nuclear and avionics applications. In the mentioned applications, the effects of the ionising radiation, have an impact on the circuits. The soft transient errors in combinational and sequential circuits could be generated and may lead to temporary or even permanent malfunction. Methods introduced in the (Krstic et al., 2016) give promising results and they are the baseline of the evaluation presented in this paper. There are solutions for such problems, that have been used in praxis, based on the N-modular redundancy (NMR) (Koren and Krishna, 2013). In such architectures multiple hardware replications are used for detection and correction of transient soft errors and permanent faults in the registers and logic. This solution is usually effective for increasing reliability of the system, but the efficiency of the system is not so good since NMR uses enormous additional area, power and time. As it was mentioned before, the NMR is still in use, because it satisfies the requirements of the respective low-volume markets. As the technology develops, the size of transistors is getting smaller, and today it reaches nanometric dimensions. For such small transistor sizes, the upsets in circuits are possible even in terrestrial applications. The CMOS logic operates under probabilistic behaviour where the transistor parameters have large variations due to the nanometric size of transistors. The technology is more sensitive to radiation due to reduced node capacitance consequently leads to reduced critical change. Attenuation of the transient effects is disabled through the utilization of the super-pipelining where the number of gates between the sequential stages is reduced (Dhillon, Diril and Chatterjee, 2005). Additionally, novel methods for low-power, for example adaptive voltage and frequency scaling (AVFS) introduced the requirement for "better then worst case" scenario of the digital logic operation, and timing soft errors are in this constellation expected as a part of regular operation. The errors can happen due to process of ageing, temperature, clock frequency and variation of voltage. As result, the question of reliability and fault tolerance is these days an important problem for the variety of applications.

The fundamental concept of fault tolerance methodology is introduced in (Sogomonyan, Weidling and Goessel, 2013; Weidling, Sogomonyan and Goessel, 2013; Krstic et al., 2016), where the fault tolerant master slave flip-flops protect the sequential logic, while the error detection circuitry is added to the combinational logic. If transient error occurs, the error detection signal can stop the propagation of the signal from the slave-latches in the flip-flops. During the presence of the errors the previous correct state is kept and the system continues to work normally until the error disappears.

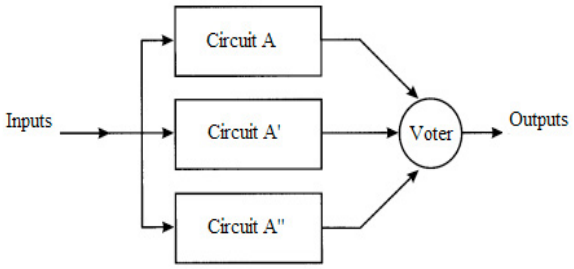

Figure 1. Example of Triple Modular Redundancy

10.1016/j.ifacol.2018.07.178

2. STATE OF ART

In applications that work under extreme environmental conditions, for example space applications, N-Modular Redundancy with majority voting is used for protection of combinational and sequential logic. It works on a principle of replicating the memory elements and the logic (Koren and Krishna, 2013). Triple Modular Redundancy is most used form of the NMR. TMR works on a principle, as we mentioned before, where systems are replicated as it can be seen in Figure 1. The systems have same functionality, in the end they are connected to a voter that gives correct signal by comparing the three inputs. The use of TMR is effective for correction of single permanent faults or single soft errors. The disadvantage of the TMR is that it uses more space and it needs more resources, as it requires triplication of the system.

Looking for more efficient solution, a lot of alternatives were analysed and proposed. Reducing the redundancy from TMR to DMR (Dual Modular Redundancy) the fault tolerance for sequential logic can be satisfied. In the DMR the voter is replaced with the C-element that has the ability of saving the value in the case where the input values are different. Therefore, saved value can be used as the last known correct value.

According to the evaluation provided in (Mitra et al., 2005; Nicolaidis, 2011), the sequential transient errors that occur in registers can be detected and corrected, the problems are however the transient errors that occur at the same time with the active clock edge.

The Razor I flip-flop was developed through the need of power efficient operation of the sequential logic and consequent need for the tolerance to timing errors in combinational logic (Ernst et al., 2004). Mentioned architecture is very useful in scaled technologies where the process variability is disabling effective use of the worst-case margins. Razor I flip-flop has master and shadow latch which use different time events for the activation (rising and falling edge of clock). This approach uses phased clocking for detection of error in main flip-flip latching where the system relies on the time margin between main flip-flop and shadow latch. Razor II flip-flop (Das et al., 2009) has simplified sequential cell and focuses more on the error detection. On the system level, error detection is implemented while the correction uses the principles of architectural replay. Based on RAZOR concept, the Bubble RAZOR (Fojtik et al., 2012) was introduced. Bubble RAZOR uses clock gating to disable error propagation in processor-like structures and in this method bubbles are introduced. Bubbles are used to provide additional cycle in the neighbouring pipeline stages for the architectural replay. Problem with Razor based solution is that they are not effective for single event upsets in flip flops effected by the radiation. Solutions like HiPer (Omana, Rossi and Metra, 2007, 2010) and the DICE latch (Calin, Nicolaidis and Velazco, 1996) are used for such applications, as they prevent single event transients in sequential logic from effecting the state of the memory elements.

There are very few solutions which can in parallel address the single events in flip-flops and logic as well as timing faults. An interesting scheme introduced in (Nicolaidis, 2007) named GRAAL, as a solution applies the polyphase clocking on the latch based designs. The concept uses mutually exclusive latch-clocks that generate safety margin for error detection and correction. Error detection is derived from the data comparison before and after the latching stage while error correction uses shadow flip-flop, which is used for architectural replay. The advantage of this approach is the use of simple circuitry for error detection and correction. On the other hand, the introduction of time constrains in the design requires careful pipeline optimization, balancing and ability to manage only errors which are within some timing boundary.

Recently, solution (Krstic et al., 2016) has been proposed which it addresses soft and timing errors and in some aspects outperforms the reference solutions. This solution is for applications that implement dataflow processing since they cannot rely on software based architectural replay.

3. IMPLEMENTATION AND ANALYSIS

Main purpose of this paper is implementation and evaluation of methods introduced in (Krstic et al., 2016). Using the Altera FPGA and Quartus Prime software Error Detection and Partial Error Correction method has been designed and tested. The RTL simulation waveform from Quartus can be seen in Figure 2. Components that have been used in circuit design are:

Parity Check Block – used for generating parity bit of input signal, one way of checking the system. Two parity checkers

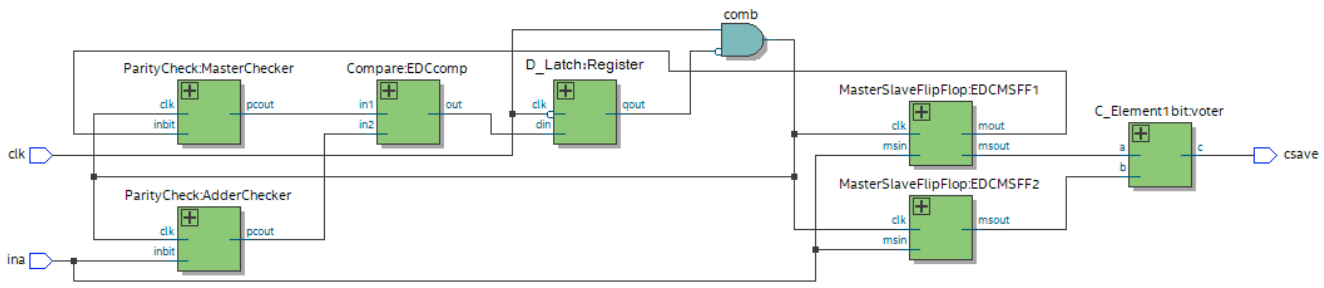

Figure 2. RTL view of EDPEC in Quartus software

are used in this scheme for checking the input signal, in our case for testing the system, and output signal of master latch.

Compare Block – used for checking the difference of input. Outputs of parity checkers are brought to compare block for generation of error, if they are different, the block will signal an error.

D Latch – used for saving a value one clock long. This latch is used for holding the error indicator signal.

DMR Master Slave latch connected to a voter – sequential and combinational logic provide information through few checkers. Voter is the decision maker in the circuit. Master slave latch are gated by the clock for providing state information if there is no error.

Quartus Prime gives opportunity to test the circuit so the user enters different values for input variables. The schematic form in Figure 2 has been analysed using the Quartus software. Signal ina holds the information that needs to be checked. Assuming that logic stage and predictor block give same result, the information is again checked twice, one way of checking the information is the DMR. The information enters and leaves master latch according to clock signal, leaving the master latch signal is passed to parity bit generator and slave latch. Information or data leaving the slave latch it enters the C element for voting. Second way of checking the errors is the parity bit generator in this paper named as the Parity Check block. As it was mentioned before the master latch leaves the information to one parity block where parity bit is generated. The generated bit will be compared with the parity bit generated by the predictor block. It was assumed that logic stage and predictor block give same result, so the ina signal is going to be compared with the signal from master latch. The signal is brought to the parity block where parity bit is generated and passed to the compare block. The compare block compares generated parity bits of ina signal and signal from master latch if they are different error is generated. Error is used for controlling the slave latch. Signal diagram is shown in Figure 3, (a) when there is no error, (b) when error occurs.

Second method introduced in (Krstic et al., 2016), is Full Error Detection and Correction (FEDC). Using same software, Quartus Prime, the method was designed and tested. The RTL view of the FEDC can be seen in Figure 4. In the evaluation, it was assumed that logic stage and predictor block give same result and they were changed by an Adder block. The solution was analysed through inputs ina and inb, which are holding the input information. Inputs are calculated by the Adder block, after the calculation the information is checked. The output of Adder block is proceeded to DMR and parity bit generator, Parity Checker Adder Checker in the Figure 4. In the DMR the information leaving the master latch is send to second parity bit generator, Parity Checker Master Checker in the Figure 4, and proceed to the slave latch. Leaving the slave latch the information is voted by the C-element. The voted information is send

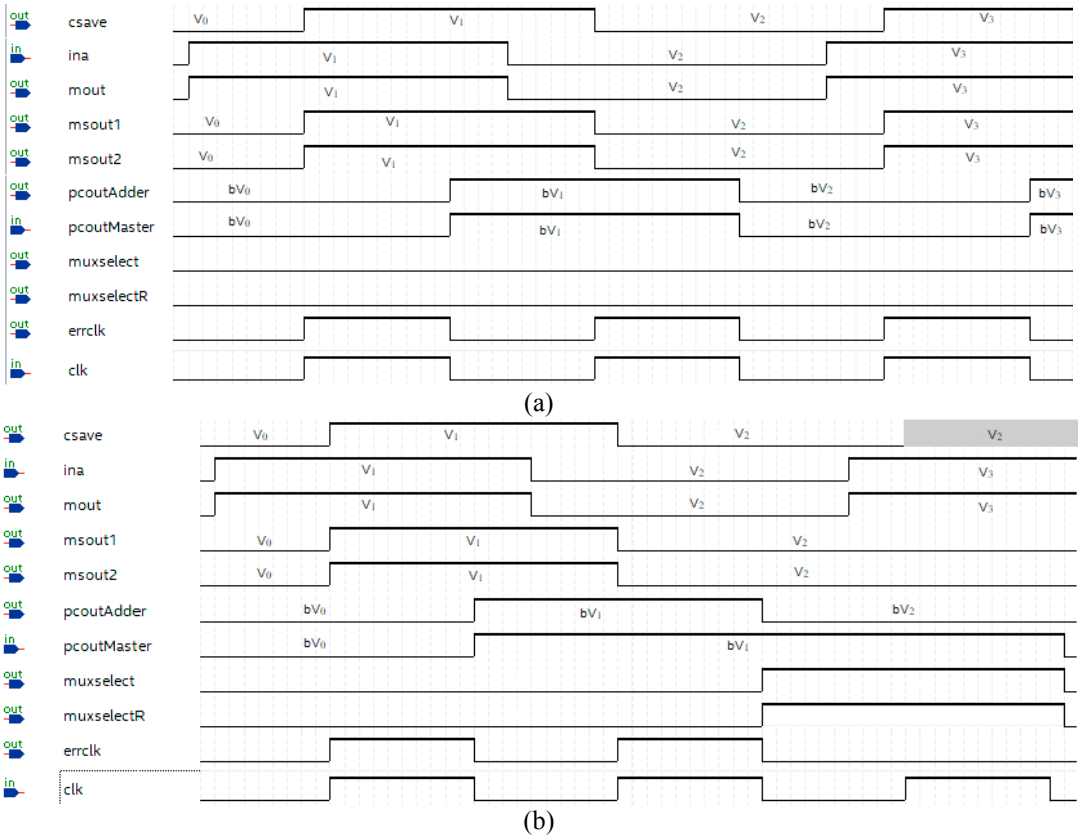

Figure 3. Signal diagram of EDPEC when there is no error (a) and when error occurs (b). Signal description is shown in Table 1.

directly to the mux and to SH latch, where it is hold for one clock cycle. SH latch is used for saving the information for one clock cycle, in the case of error detection the information from SH latch is selected as output. In the scheme two parity blocks are implemented, one is used for parity check of Adder output while the other is used for parity check of master latch. Output of parity blocks is compared in the compare block, where the error signal is generated in the case of different inputs. Error signal is used for few purposes, one use of error signal is the control of mux where error signal selects output signal from SH latch or C-element. Another use of error signal is the control of SH latch. Signal diagram is shown in Figure 5.

3.1 Improvement of FEDC

In this paper an improved FEDC is introduced. The improvement is focused on the input variables, in the case of error detection, the improved FEDC holds the input variable and uses the data for recalculation in the system. Adding a block that consists of D latch and a mux, clocked by an error signal, allows the holding of the input value while the error is present. Signal diagram is shown in Figure 6. It can be seen how the V_2 is held and recalculated because of the error detection. Variable named as crout, in Figure 6., is the output of the introduced block for the improved FEDC. It can be seen how the detected error prolongs the holding of input value which gives the possibility to recalculate the value in the system.

Table 1. Signal description of EDPEC and FEDC

clk	Clock signal
csave	Signal from C element
errclk	Error clock for slave latch
ina	Input information
mout	Master latch output
msout1	Master slave latch output one
msout2	Master slave latch output two
muxselect	Error signal
muxselectR	Latch for holding error signal
pcoutMaster	Parity checker output (master latch)
pcoutAdder	Parity checker output (*ina* signal)
EDCout	Output from the mux (FEDC)
shout	Output signal from SH Latch (FEDC)
crout1	Output signal from CheckReg1 (improved FEDC)
crout2	Output signal from CheckReg2 (improved FEDC)
Vn	Value that propagates through stages of system
bVn	Bit value of parity bit generator

4. RESULTS AND CONCLUSION

This paper provides evaluation and implementation of EDPEC and FEDC methods on Altera FPGA. Beside experimenting with methods, an improvement of FEDC is introduced. Some screenshots of the result evaluation are presented in Figures 3, 5 and 6. In Figure 3, the EDPEC was tested through setting different values to pcoutMaster and pcoutAdder, it can be seen how the value of muxselect and muxselectR changes, they represent the error and they are

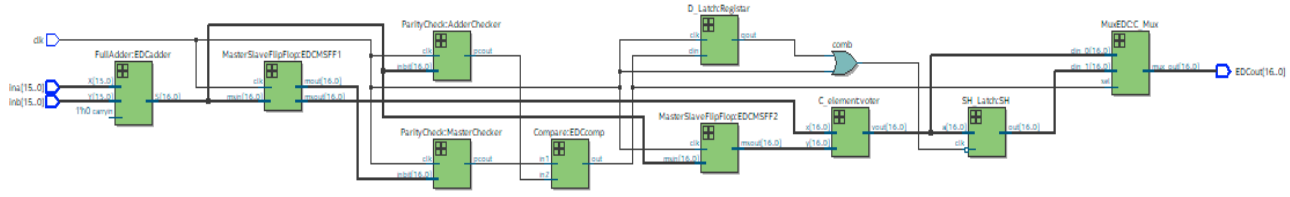

Figure 4. RTL view of FEDC

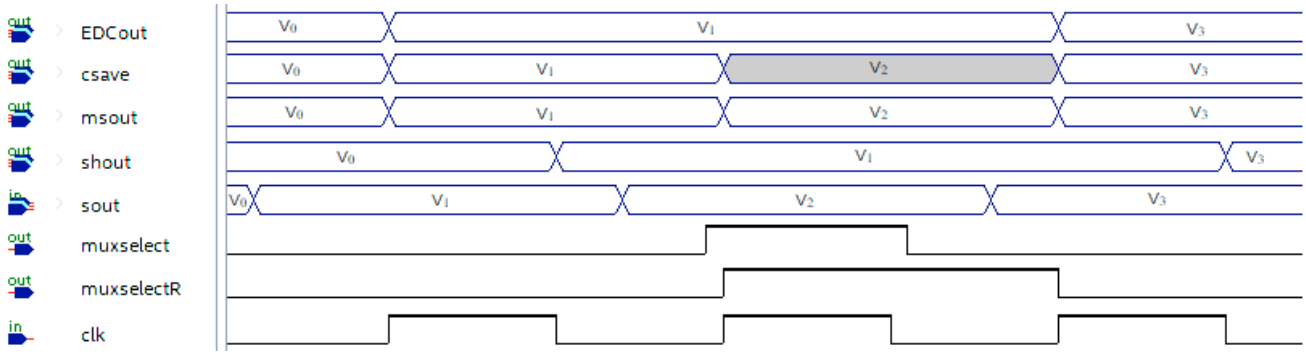

Figure 5. Signal diagram of FEDC, signal description is shown in Table 1.

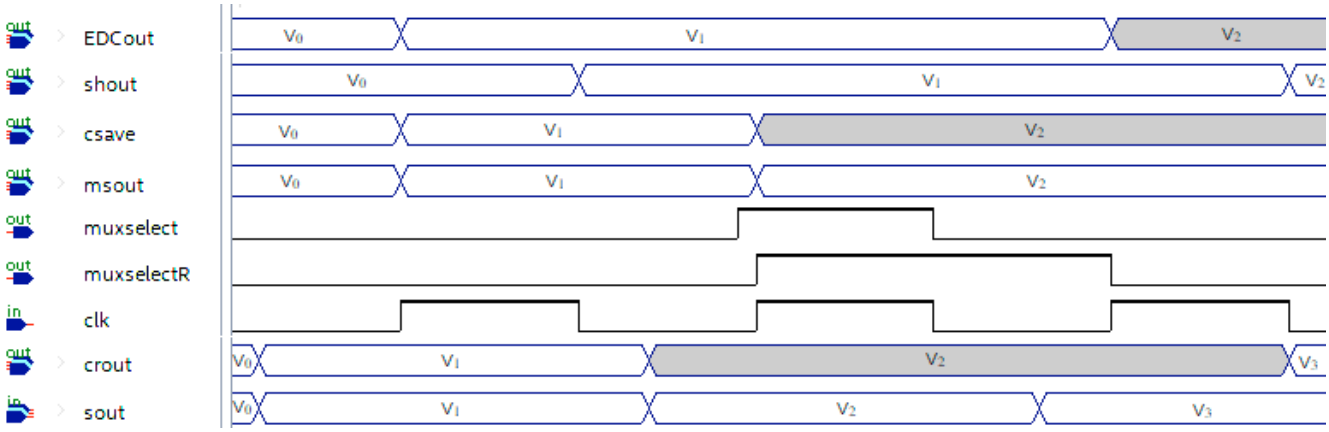

Figure 6. Results of improved FEDC, signal description is shown in Table 1.

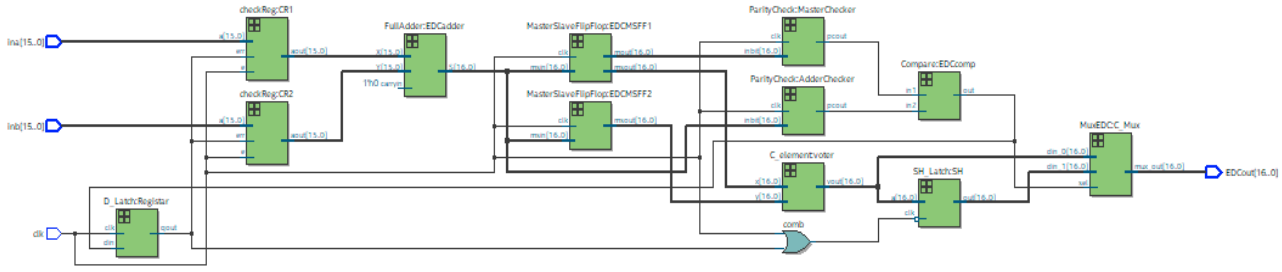

Figure 7. RTL view of improved FEDC

used for controlling the errclk variable that clocks the slave latch. Csave is the output of EDPEC and it can be seen in Figure 3 how it behaves when error is detected, if error exists the output of the system is the last correct state.

Introducing the improvement of FEDC will give the opportunity to make the calculation of inputs that are affected by appearance of error. The values will be held by the D latch inside the new introduced block named as Check Reg with crout output variable. If Figure 5. and Figure 6. are compared, it can be seen how the value V_2 is lost if FEDC is used, that was inspiration to introduce the improvement where the value will be recalculated. The introduced block solved the problem as it can be seen in Figure 6. Both solutions work on similar principle, if error occurs slave latches and SH latch are closed. MuxEDC, block in this paper, selects output from SH latch. The output is last correct state saved in the SH latch. Additional to the FEDC, Check Reg block is added before the Full Adder, in this paper while it will be placed before the logic stage and predictor block in (Krstic et al., 2016). Check Reg is used to resend information in case of the error detection.

Future work as said in the (Krstic et al., 2016) can be testing more complex systems using these methods.

REFERENCES

Calin, T., M. Nicolaidis and R. Velazco (1996). Upset hardened memory design for submicron cmos technology. In: *IEEE Trans. Nucl. Sci.* 43 (6) 2874–2878.

Das, S. et al. (2009). Razorii: In situ error detection and correction for pvt and ser tolerance. In: *IEEE J. Solid State Circuits* 44 (1) (2009) 32–48.

Dhillon, Y., A. Diril and A. Chatterjee (2005). Soft-error tolerance analysis and optimization of nanometer circuits, Design, Automation and Test in Europe. In: *Proceedings, vol. 1*. pp. 288–293.

Ernst, D. et al. (2004). Razor: circuit-level correction of timing errors for low-power operation. In*: IEEE Micro* 24 (6) 10–20.

Fojtik, M. et al. (2012) Bubble razor: an architecture-independent approach to timing-error detection and correction. In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), IEEE International 2012.* pp. 488–490.

Koren, I. and C.M. Krishna (2007). Fault-Tolerant Systems. In: *Morgan Kaufmann Publishers Inc*. San Francisco, CA, USA.

Krstic, M. et al. (2016). Enhanced architectures for soft error detection and correction in combinational and sequential circuits. In: *Microelectronics Reliability.* 56, pp. 212-220.

Mitra, S. et al. (2005). Logic soft errors: a major barrier to robust platform design. In: *Test Conference, Proceedings. ITC 2005. IEEE International 2005*

Nicolaidis, M. (2007). Graal: a new fault tolerant design paradigm formitigating the flaws of deep nanometric technologies. In: *Test Conference, ITC 2007. IEEE International 2007*. pp. 1–10.

Nicolaidis, M. (2011). Soft errors in modern electronic systems. In: *Vol. 41 of Frontiers in Electronic Testing*. Springer, pp. 203–252.

Omana, M., D. Rossi and C. Metra (2007). Latch susceptibility to transient faults and new hardening approach. In: *IEEE Trans. Comput* 56 (9) 1255–1268.

Omana, M., D. Rossi and C. Metra (2010). High-performance robust latches. In: *IEEE Trans. Comput.* 59 (11) 1455–1465.

Sogomonyan, E., S. Weidling and M. Goessel (2013). A new method for correcting time and soft errors in combinational circuits, Design and Diagnostics of Electronic Circuits Systems (DDECS), In: *IEEE 16th International Symposium*. pp. 283–286.

Weidling, S., E. Sogomonyan and M. Goessel (2013) Error correction of transient errors in a sum-bit duplicated adder by error detection. In: *Digital System Design (DSD), 2013 Euromicro Conference.* pp. 855–862.