Proceedings of

2014 10th IEEE Workshop on Factory Communication Systems

WFCS 2014

May 5-7, 2014

Toulouse, France

Proceedings Editor

Zoubir Mammeri
Paul Sabatier University
Toulouse, France

IEEE Catalog Number: CFP14WFC-ART
ISBN: 978-1-4799-3235-1

IEEE WFCS 2014 Final Program

Monday, May 5, 2014

Full-day program

8:00-	Registration (Registration Desk opens)
8:30-9:00	Opening Session
9:00-10:00	Session 1: Real-Time Scheduling
10:00-10:30	Coffee Break
10:30-12:00	Session 2: Wireless Networks for Automation Systems
12:00-13:30	Lunch Break
13:30-14:30	Keynote 1: Future On-board networks in space systems
14:30-16:00	WIP Session: WIP 1
16:00-16:30	Coffee Break
16:30-18:00	Session 3: Reliable Wireless Communication Networks
19:00-20:30	Welcome Reception

Tuesday, May 6, 2014

Full-day program

8:30-10:00	Session 4: Dependable and Secure Industrial systems
10:00-10:30	Coffee Break
10:30-12:00	Session 5: Real-Ethernet Extensions for Industrial Systems
12:00-13:30	Lunch Break
13:30-14:30	Keynote 2: From Fieldbus to Automation Cloud. Industrial Communication Systems over two decades
14:30-16:00	WIP Session : WIP 2
16:00-16:30	Coffee Break
16:30-18:00	Session 6: Timing Analysis of Real-Time Ethernet
19:00-23:00	Conference dinner

Wednesday, May 7, 2014

Full-day program

8:30-10:00	Session 7: Networked Control Systems
10:00-10:30	Coffee Break
10:30-12:00	Session 8: CAN in Automation systems
12:00-13:30	Lunch Break
13:30-14:30	Session 9: Linux in Networked Industrial Systems
14:30-16:00	Panel: Communication in avionics
16:00-16:15	Closing session
16:15-16:40	Coffee Break

Monday, May 5, 2014

Day program top

Session 1: Real-Time Scheduling

Chair: Gianluca Cena, National Research Council of Italy – (9:00-10:00)

Multi-Variant Time Constrained FlexRay Static Segment Scheduling
Jan Dvorak, Zdenek Hanzalek

A Novel Dynamical Approach to (m,k)-firm Scheduling
Milton Cunguara, Tomas Oliveira E Silva, Paulo Pedreiras

Session 2: Wireless Networks for Automation Systems

Chair: Paulo Pedreiras, University of Aveiro, Portugal – (10:30-12:00)

Carrier-aided Clock Skew Estimation for ToA Ranging with minimal Overhead
Reinhard Exel, Thilo Sauter

Impact of Hard- and Software Timestamping on Clock Synchronization Performance over IEEE 802.11
Anwaq Mahmood, Reinhard Exel, Thilo Sauter

Medium Access Protocol Design for Time-Critical Applications in Wireless Sensor Networks
Tao Zheng, Mikael Gidlund, Johan Akerberg

Keynote 1: Future On-board networks in space systems

Chair: Eduardo Tovar, Polytechnic Institute of Porto, Portugal – (13:30-14:30)

Future On-board networks in space systems
Olivier Notebaert - Airbus Defence and Space, France

WIP Session: WIP 1

Chair: Jean-Luc Scharbarg, IRIT - University of Toulouse, France – (14:30-16:00)

The Generic Device on ARM based hardware - An analysis on cycle time limits
Stefan Matzler, Alexander Dennert, Martin Wollschlaeger

A Proposal for Master Replica Control in the Flexible Time-Triggered Replicated Star for Ethernet
David Gessner, Julian Proenza, Manuel Barranco

A Proposal for Managing the Redundancy Provided by the Flexible Time-Triggered Replicated Star for Ethernet
David Gessner, Julian Proenza, Manuel Barranco

Mixed-Criticality Scheduling of Messages in Time-Triggered Protocols
Zdenek Hanzalek, Tomas Tunys

Session 3: Reliable Wireless Communication Networks

Chair: Roman Obermaisser, University of Siegen, Germany – (16:30-18:00)

Multiple Relaying Protocols for Lifetime Extension in Two-Hop Wireless Networks
Tung-Linh Pham, Dong-Seong Kim

An Enhanced MAC to Increase Reliability in Redundant Wi-Fi Networks
Gianluca Cena, Stefano Scanzio, Adriano Valenzano, Claudio Zunino

Time-Critical MAC Protocol based on IEEE 802.15.4 IR-UWB optimized for Industrial Wireless Sensor Networks
Rafael Reinhold, Lisa Underberg, Ruediger Kays

Tuesday, May 6, 2014

Day program top

Session 4: Dependable and Secure Industrial systems

Chair: Martin Wollschlaeger, GWT-TUD GmbH, Dresden, Germany – (8:30-10:00)

On the security of security extensions for IP-based KNX networks
Aljosha Judmayer, Lukas Krammer, Wolfgang Kastner

On the Description of Access Control Policies in Networked Industrial Systems
Manuel Chemimod, Luca Durante, Lucia Seno, Adriano Valenzano

Design and Analysis of UWB-based Network for Reliable and Timely Communications in Safety-Critical Avionics
Dinh-Khanh Dang, Ahlem Mifdaoui, Thierry Gayraud

Session 5: Real-Ethernet Extensions for Industrial Systems

Chair: Thilo Sauter, Danube University Krems, Austria – (10:30-12:00)

Extending Summation-Frame Communication Systems for High Performance and Complex Automation Applications
David Ganz, Hans Dermot Doran

A Flexible Mechanism for Efficient Transmission of Aperiodic Real-time Messages over EtherCAT networks
Lucia Lo Bello, Gaetano Patti, Giuliana Alderisi, Davide Patti, Orazio Mirabella

Multipath Redundancy for Industrial Networks using IEEE 802.1aq Shortest Path Bridging
Paolo Ferrari, Alessia Flammini, Stefano Rinaldi, Gunnar Prytz, Rahil Hussain

Keynote 2: From Fieldbus to Automation Cloud. Industrial Communication Systems over two decades

Chair: Roman Obermaisser, University of Siegen, Germany – (13:30-14:30)

From Fieldbus to Automation Cloud. Industrial Communication Systems over two decades
Peter Neumann, Rak Magdeburg, Germany

WIP Session: WIP 2

Chair: Marek Miskowicz, AGH University of Science & Technology, Poland – (14:30-16:00)

Integration of Smart Meters into Management Systems in Automation
Andreas Fembach, Wolfgang Kastner

Control-as-a-Service from the Cloud: A Case Study for using Virtualized PLCs
Oeid Givehchi, Jahanzaib Imtiaz, Henning Trsek, Juergen Jasperneite

Concept for a Safety-Controller based on uncertified Hardware
Bernd Thiesmann, Andreas Platschek

An Effective and Easy to Use IoT Architecture
Juan Pimentel

Session 6: Timing Analysis of Real-Time Ethernet

Chair: Ahlem Mifdaoui, University of Toulouse - ISAE, France – (16:30-18:00)

Design of communication systems for networked control system running on PROFINET
Stephan Home, Stefan Palis, Christian Diedrich

Response Time Analysis of Multi-Hop HaRTES Ethernet Switch Networks
Mohammad Ashjaei, Paulo Pedreiras, Moris Behnam, Reinder J. Bril, Luis Almeida, Thomas Nolte

Priority assignment on an avionics switched Ethernet Network (QoS AFDX)
Tasnim Hamza, Jean-Luc Scharbarg, Christian Fraboul

Wednesday, May 7, 2014

Day program top

Session 7: Networked Control Systems

Chair: Zdenek Hanzalek, Czech Technical University in Prague, Czech Republic – (8:30-10:00)

A field level architecture for reconfigurable real-time automation systems
Lars Durkop, Henning Trsek, Jens Otto, Jurgen Jasperneite

Model-based validation of CANopen systems
Alexios Lekidis, Marius Bozga, Saddek Bensalem

Event-Based Communication in Networked Control Systems
Marek Miskowicz

Session 8: CAN in Automation systems

Chair: Marek Miskowicz, AGH University of Science & Technology, Poland – (10:30-12:00)

Effect of Jitter-Reducing Encoders on CAN Error Detection Mechanisms
Gianluca Cena, Ivan Cibrario Bertolotti, Tingting Hu, Adriano Valenzano

Design, Verification, and Performance of a Modbus-CAN Adaptation Layer
Gianluca Cena, Ivan Cibrario Bertolotti, Tingting Hu, Adriano Valenzano

Schedulability Analysis of GMF-Modeled Messages over Controller Area Networks with Mixed-Queues
Meng Liu, Moris Behnam, Thomas Nolte

Session 9: Linux in Networked Industrial Systems

Chair: Eduardo Tovar, Polytechnic Institute of Porto, Portugal – (13:30-14:30)

Parallel Implementation of Real-Time Communication and IP Communication by using Multiple Ring Buffers
Kazuki Ueda, Tatsushi Kikutani, Takahiro Yakoh

Performance evaluation of Linux CAN-related system calls
Michal Sojka, Pavel Pisa, Zdenek Hanzalek

Panel: Communication in avionics

Moderator: Jean-Luc Scharbag – (14:30-16:00)

Concept for a Safety-Controller based on uncertified Hardware

Bernd Thiemann, Andreas Platschek
Vienna University of Technology
Institute of Computer Technology
Vienna, Gußhausstraße 27-29
{thiemann, platschek}@ict.tuwien.ac.at

Abstract

This work suggests new solutions for safe systems in industrial applications. Nowadays automation systems get more complex, so the microcontroller has to be replaced with a more powerful CPU. The preferred solution is to use commercial off-the-shelf (COTS) general purpose CPUs. The hardware has to be analyzed in detail to estimate the behavior in case of a fault. With state-of-the art processors this is not possible anymore. A concept to avoid this is "coded processing" as mentioned in the standard for industrial safety systems IEC 61508 [1]. The goal of this research is to analyze a concept which meets the demands of the IEC 61508 safety integrity level 3 (SIL 3) only based on software techniques to avoid any hardware analysis and dependencies. The evaluation of the concept is done by theoretical analysis based on fault models found in literature. The practical tests are done by a fault injection software which is developed in the course of this research.

1 Introduction

Over the last years industrial safety systems based on microprocessors have widely replaced hard-wired solutions. The advantage of microprocessor based systems is a cost-effective installation and its flexibility during use. Changes in the safety concept can be adopted easily, because there is no need to install new wires. Only a reprogramming of the safety-controller is necessary. The task of a safety-controller is to monitor the industrial process and execute a safe stop to transfer the process into a safe state if dangerous conditions are detected. So it is important for the safety-controller to be safe itself, so it does not cause risks and detects dangerous behavior with a predetermined probability.

To meet the demands of functional safety and error detection the IEC 61508 [1] shows among others design guidelines for safety equipment. Another important part is the definition of the SIL. The four SILs stand for different Safe Failure Fractions (SFF) which the equipment has to meet to comply with the standard. For SIL 3 the probability of failure per hour (PFH) of 10^{-7} [1, Part 2, Table 3] or better has to be fulfilled. In dependency on the hardware fault tolerance (HFT) for SIL 3 a SFF of $> 99\%$ for zero HFT and a SFF of $90\% \leq 99\%$ for single HFT is required.

Common safety-controllers on the market achieve the required SIL 3 due to redundant hardware. A typical safety platform is shown in Fig. 1. Certified components are hatched, not certified components are blank. There are two independent hardware channels, each executing an independent safety application on top of it. The architecture can be 1oo2D (one out-of two including diagnostics) or 2oo2D. The only cross-communication between the channels is for diagnostic purposes. Each channel monitors the other channel. If one channel detects an anomaly in the other one a safe state has to be initiated. The communication with the environment takes place via a network interface card (NIC) and a safe fieldbus. Safety controllers typically do not have any direct interfaces to the industrial process.

The problem of this structure is the certified hardware wherefore a long and costly certification process is required and re-usability is (in contrast to software) limited. The interior of hardware components is typically intellectual property of the manufacturer and not accessible to the developers of safety controllers. Hence it is hard to accomplish guaranteed fault reaction without the knowledge of the internal structures. A solution for hardware independence is to shift the guaranteed fault detection (due to the redundant hardware) entirely into the software. For this case the hardware has to be seen as HFT 0, because with uncertified hardware one can not rely on their fault reaction if any anomaly occurs.

To monitor the hardware and detect nearly all possible hardware faults, coded processing can be used. Coded processing uses an arithmetic code, which fulfills the requirements of Eq. 1. X' and Y' stand for the coded values of X and Y, whereas $\oplus$ is the coded operation for $+$. The coded mathematical operation applied to two coded operands leads to an result which is already encoded.

$$X' \oplus Y' = (X + Y)' \tag{1}$$

One kind of arithmetic codes is the AN-Code. The theoretical construct was introduced by Brown [2] and extended by Forin [3]. The requirement to use this code is

978-1-4799-3235-1/14/$31.00

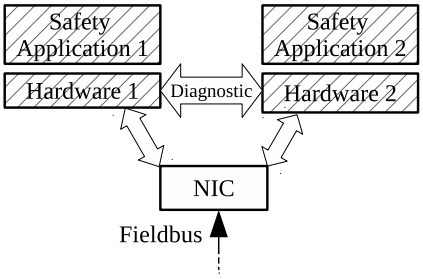

Figure 1. Common safety equipment

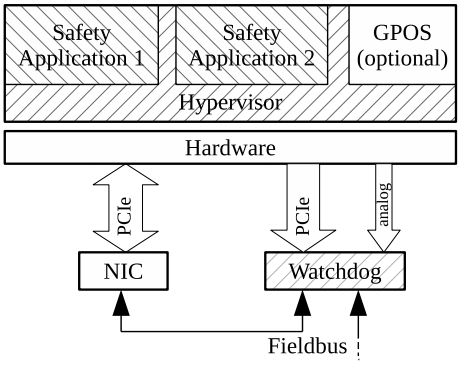

Figure 2. Hardware independent safety approach

that the used processor has to have a wider bitlength than the largest used number in the safety application requires. Nowadays processors typically contain at least an 64 bit arithmetical logical unit and even bigger single instruction multiple data units (up to 256 bit [4]). AN-codes use a constant A to encode a number x_n and generate x_c, as shown in eq. 2.

$$x_c = A \cdot x_n. \tag{2}$$

With this encoding it is possible to detect failed arithmetic operations and some faults in memory. The modulo operation of every variable, regardless if it is in a processor register after a calculation or in the memory, has to be zero to accomplish the encoding. If the modulo test fails a safe state has to be established. The AN-encoding only detects wrong operands but not exchanged operands or wrong operations. For this purpose ANB-coding can be used. B is called the *signature* and is added to every x_c. Thus the result of any arithmetic operation additionally depends on the composition of the signatures. The signature after an addition of two encoded variables $x_{c1} + x_{c2}$ has to be $B_1 + B_2$. This way it is possible to discover wrong or exchanged operands and operators. Forin [3] also introduced an additional *date* D added to the ANB-encoded variable creating the ANBD-encoding to detect a phenomena called "lost update". It happens if a variable is read from the memory, modified and should be stored in the same place in the memory again, but due to an error it is stored somewhere else. Afterwards the previous value is read from the correct position in memory. The dynamic value of D changes every calculation cycle so a lost update is detectable, because the attached date differs from the expected one.

These encodings lead to increased error detecting capabilities but require additional computational performance. Due to the good progress of processor performance and the possibility to use state-of-the art components the disadvantage of the raised computational needs can be compensated. Additionally one gets the opportunity to use uncertified hardware for safety related applications.

2 Concept

Before the concept can be developed some assumptions have been made.

2.1 Assumptions

First there are no special demands on hardware performance. It is assumed that the hardware is fast enough to calculate the results in the predefined period. If this constraint can not be met, it leads to reduced availability, but the safety related behavior must not be effected. For the input and output of data it is mandatory that there are no direct interfaces to the monitored industrial process. Every input and output transaction is sent/received via a safe fieldbus to safe input and output modules. These output modules switch off the power in case of a detected error. The certified software is supposed to be developed according to IEC 61508 part 3. For the uncertified software (GPOS) no safety assumptions can be made. The used hypervisor is a software which can be used to execute different operating systems on the same hardware. It is responsible for the isolation between the guests and a fair sharing of processor time. If an error is detected, the fail-silent state is considered the fail-safe state, e. g. no messages may be sent by the controller. All the safe output modules will switch off to reach an de-energized situation after a certain preconfigured time period without a message from the controller.

2.2 High Level Concept

Based on this assumptions a concept for a safe computer system has been developed. The high level architecture is shown in Fig. 2 and is based on information redundancy and time diversity. The hatched components have to be certified, the blank ones not. On the uncertified hardware a type-1 hypervisor is executed, which isolates the two safety applications against each other. Optionally there can be a general purpose operating system (GPOS) beside the safety applications.

The calculated data is sent via a Peripheral Component Interconnect express (PCIe) to the NIC, which generates the data-packets for the fieldbus. A second unit on the PCIe interface is the watchdog. This item must achieve the high SFF of $> 99\,\%$ requested by the IEC 61508 [1].

The PCIe interface for the watchdog is widespread, so it can be used with a wide range of available industrial PCs on the market. Its tasks are quite simple so it is easy to design and certify. The watchdog is responsible for measuring the analog environmental values like processor supply voltage and temperatures. For this, there are some additional analog inputs which are necessary, because analog data can not be routed via the PCIe interface. The PCIe interface of the watchdog is needed to receive a heartbeat from the safety applications and to check if there is progress in the software control-flow. Additionally the watchdog compares the output results of the two safety applications.

The watchdog can only receive data. If an anomaly is detected no data is generated or sent. The only possibility to interrupt calculations and stop the industrial process is to switch off the network connection. In this way the safe output module does not receive data any more and after a deadline, defined by the safe fieldbus, the output module will switch off.

To achieve verifiable information diversity between the two safety applications coded processing is used. One of the channels calculates with encoded and the other one with uncoded data. To meet the requirements of an $SFF > 99\,\%$ according to the safety standard an encoding called *ANB-encoding* is used. While researchers claim a fault detection a high as $99,56\,\%$ [6, p. 160], at this stage of our work we are not able to confirm or disprove these numbers.

The second channel with the uncoded data additionally calculates the signatures B of the encoded application. In this approach a sightly different encoding is used than postulated by Forin [3]. He designed a code in which the signature B is static and for the dynamic part date D was developed. An easier and faster solution is to link the signature B to the value and not to the variable. In this way the signature B changes after every calculation, leading to a time dependent $B(t)$. The computation of the dynamic $B(t)$ is independent from the encoded or uncoded data and is also executed in the uncoded channel. Due to the two-channel signature calculation even dynamic, not predetermined control-flow can be observed.

For the highest error detection, the calculation results and the signatures of both channels are sent to the watchdog. The comparison outside the main processor increases the error detection probability. With the help of the signatures the watchdog can discover if the two calculation channels used different program sequences to generate the data and thus check the control flow. This constant input of signatures is used as heartbeat to observe the vital function of the safety applications.

3 Analysis

The analysis of the concept is important to accomplish verifiable safety. Due to the hardware independence an accurate and still hardware independent fault model has to be found.

Two fault models are used to analyze the concept in Fig. 2. The model of Goloubeva [5, p. 13] for the analysis on system level and the model given in IEC 61508 [1] for possible physical effects.

3.1 Analysis on System Level

The fault model on system level presented in [5] consists of two error-types including two subtypes. The two error-types are single data error and single code error. A single data error is a logical error in a memory cell. It is not important where exactly the error occurs. A single code error effects an instruction of the program code. As previously, it is not important where exactly the error occurs. The single code error can effect the source-code in two ways. A type-1 fault modifies the operation but does not influence the program flow. For example an addition is exchanged by a multiplication. The mathematics result is erroneous but the control flow is not harmed. Also the operands of the machine instruction can be altered, for example if one operand is exchanged by another one. If an error of this kind modifies the control-flow, it would be a type-2 error. Examples are exchanged jump-instructions or wrong destination addresses.

Coded processing addresses the single data error. If a bit-flip happens anywhere in the data the encoding can detect it with a high probability. For these faults an AN-encoding would be enough, but for the other faults the signature B is needed. If a fault alters the instruction of an arithmetic operation and changes the operand with ANB-encoding using two software channels it is detectable, because the signature in the result is not equal to the one calculated in the second channel, unless the fault alters both calculations the same way. A type-2 error that influences the control flow can also be detected by ANB-encoding. If the control-flow is altered it is also detectable by the signature B. The safe applications in both channels have to execute the same control flow, which can be retraced by the signatures. Due to this it is possible to check if the calculations in both channels executed the same path in the control-flow graph. It would be undetectable if both channels generate identical and wrong results. To generate this kind of dangerous state both software applications have to be altered in very special, different ways.

The fault-model in [5, p. 13] is good for an analysis on system level, but neglects all kinds of analog anomalies. Fault-models on system level can not describe the system behavior for not logical faults, e.g. if the power supply leaves the permitted voltage range. For these faults model in IEC 61508 part 2 Annex A [1] will be used. It consists of many possible anomalies and is beyond the scope of this paper, but the most likely faults will be discussed.

3.2 Physical Analysis

Very often computers struggle with voltage changes, especially in the rough industrial environment electromagnetic disturbances can alter the supply voltage for a short

time. To protect the hardware against dangerous failures the watchdog monitors the voltage and switches the fieldbus off if any limit is exceeded. One extrema of illegal voltage changes is a sudden blackout. In this case both software channels and the watchdog can not work any more, because typical industrial PCs only contain one power supply unit and COTS processors are not designed to be powered by different sources. In this case no data packets can be generated and the output module will switch to a safe state after the deadline of the fieldbus is missed. Another common cause failure for both software channels is the oscillator. In the case where the frequency of the quartz changes slowly, the effect can be detected in two ways. First, the safe output module will recognize the missing data-packets and switch off. Secondly the watchdog, which contains its own timebase and quartz, will uncover the technical issues through unexpected delays of the signatures from the software channels.

4 Proof of Concept

The concept of Section 2 has to be proven. This is done by an exemplary implementation of a mathematical library and a fault injection suite.

4.1 Mathematical Library

A mathematical library has been implemented, which provides ANB-encoded operations for addition, subtraction, multiplication and AN-encoding for the division. Furthermore it handles the dynamic signatures B of the ANB-encoding. The goal is to generate an environment where the programmer of the safety application should not pay attention to the encoding. The encoding, decoding and coded calculations should be done in background so no special knowledge is needed to use the mathematical library and thus coded processing.

A second, simpler mathematical library is needed for the uncoded channel because this library has to calculate the signatures B additionally to allow the comparison with the encoded program.

4.2 Fault Injection Suite

The practical evaluation of the concept in Fig. 2 and the included mathematical library is done by a fault injection software which operates on assembler-code level. With this tool faults on assembler level can be injected during execution by replacing operations with another ones. Operands can be exchanged or modified by a selectable number of bit-flips. A transient or permanent endurance of the fault appearance can be chosen.

First tests of the ANB-library have shown that the performance drawback rate due to the encoding achieves a low double-digit level, compared with the native, uncoded calculation. The second channel calculating with uncoded data and additionally generating the signatures is approximately twice as fast as the encoded channel.

According to the first fault injection tests and the analysis based on Goloubeva's error model [5], the encoded application has a high diagnostic coverage for faults effecting the arithmetic operations, but influences on the control flow can only be detected by comparison with the second, uncoded channel and its additional signature computation.

5 Conclusion and Further Work

The next steps are to test the fault reaction of the encoded safe software application and then analyze the weak points. Depending on the results, the second safe applications will be started in parallel to check if the combination of two channels can avoid previous discovered drawbacks.

In this paper a concept for a hardware independent safety controller has been shown. It is based on information redundancy and time diversity and has to comply with the requirements of the IEC 61508 standard. The presented concept has the advantage of dynamic control-flow. This is a contrast to Forin's [3] design, which only allows static control-flows, because in his concept the signatures are predetermined and stored in an additional read only memory. Another development is the combination of the dynamic date D into the former static signature B. This allows simplification in the mathematical library and therefore improved computational performance.
The advantage of the concept presented here is that through coded and redundant calculations and the additional PCIe watchdog card a computer system gets the ability to perform safety-related applications.

References

[1] IEC, *IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, 2011.*

[2] David T. Brown, *Error Detecting and Correcting Binary Codes for Arithmetic Operations, Electronic Computers, IRE Transactions on , vol.EC-9, no.3, pp.333,337, 1960.*

[3] P. Forin, *Vital Coded Microprocessor: Principles and Application for various Transit Systems, 1989.*

[4] Intel, *Intel 64 and IA-32 Architectures Software Developers Manual Volume 1: Basic Architecture, 2013.*

[5] Goloubeva Olga, Maurizio Rebaudengo, Matteo Sonza Reorda, Massimo Violante, *Software Implemented Hardware Fault Tolerance. 1. Springer, 2006.*

[6] Ute Schiffel, *Hardware Error Detection Using AN-Codes. Technische Universität Dresden, 2011*