

# Radiation Mitigation for SRAM-Based FPGAs in the CBM Experiment

A dissertation  
in fulfillment of the requirements for the academic degree of  
*doctor rerum naturalium* (Dr. rer. nat.)

submitted to the Faculty of Computer Science and Mathematics  
of the Johann Wolfgang Goethe University  
in Frankfurt am Main, Germany

by  
Sebastian Andreas Manz  
born in Friedrichshafen, Germany

Frankfurt 2015

(D 30)

accepted by the Faculty of Computer Science and Mathematics of the  
Johann Wolfgang Goethe University as a dissertation.

Dean: Prof. Dr. Uwe Brinkschulte

Expert assessor: Prof. Dr. Udo Kebschull  
Prof. Dr. Lars Hedrich

Date of disputation: September 24, 2015







## Radiation Mitigation for SRAM-Based FPGAs in the CBM Experiment

Detectors of modern high-energy physics experiments generate huge data rates during operation. The efficient read-out of this data from the front-end electronics is a sophisticated task, the main challenges, however, may vary from experiment to experiment. The *Compressed Baryonic Matter* (CBM) experiment that is currently under construction at the *Facility for Antiproton and Ion Research* (FAIR) in Darmstadt/Germany foresees a novel approach for data acquisition. Unlike previous comparable experiments that organize data read-out based on global, hierarchical trigger decisions, CBM is based on free-running and self-triggered front-end electronics. Data is pushed to the next stage of the read-out chain rather than pulled from the buffers of the previous stage. This new paradigm requires a completely new development of read-out electronics.

As one part of this thesis, a firmware for a read-out controller to interface such a free-running and self-triggered front-end ASIC, the GET4 chip, was implemented. The firmware in question was developed to run on a *Field Programmable Gate Array* (FPGA). An FPGA is an integrated circuit whose behavior can be reconfigured “in the field” which offers a lot of flexibility, bugs can be fixed and also completely new features can be added, even after the hardware has already been installed. Due to these general advantages, the usage of FPGAs is desired for the final experiment. However, there is also a drawback to the usage of FPGAs. The only affordable FPGAs today are based on either SRAM or Flash technology and both cannot easily be operated in a radiation environment. SRAM-based devices suffer severely from Single Event Upsets (SEUs) and Flash-based FPGAs deteriorate too fast from Total Ionizing Dose (TID) effects.

Several radiation mitigation techniques exist for SRAM-based FPGAs, but careful evaluation for each use case is required. For CBM it is not clear if the higher resource consumption of added redundancy, that more or less directly translates in to additional cost, outweighs the advantaged of using FPGAs. In addition, it is even not clear if radiation mitigation techniques (e.g. *scrubbing*) that were already successfully put into operation in space applications also work as efficiently at the much higher particle rates expected at CBM.

In this thesis, existing radiation mitigation techniques have been analyzed and eligible techniques have been implemented for the above-mentioned read-out controller. To minimize additional costs, redundancy was only implemented for selected parts of the design.

Finally, the radiation mitigated read-out controller was tested by mounting the device directly into a particle beam at Forschungszentrum Jülich. The tests show that the radiation mitigation effect of the implemented techniques remains sound, even at a very high particle flux and with only part of the design protected by costly redundancy.

The promising results of the in-beam tests suggest to use FPGAs in the read-out chain of the CBM-ToF detector.



# Contents

<b>1. Introduction</b>	<b>13</b>
1.1. Objectives and Contributions . . . . .	14
1.2. Thesis Outline . . . . .	15
<b>2. Motivation</b>	<b>17</b>
2.1. The CBM Experiment at FAIR . . . . .	17
2.1.1. Scientific Motivation . . . . .	17
2.1.2. The CBM Detector Setup . . . . .	18
2.1.3. Self-Triggered and Time Stamped Paradigm . . . . .	19
2.1.4. CBM-ToF Read-Out Chain . . . . .	21
2.2. Electronic Devices in Radiation Environments . . . . .	24
2.2.1. Cumulative effects . . . . .	25
2.2.2. Single Event Effects . . . . .	26
2.3. FPGA Architecture . . . . .	29
2.4. Expected Impact of Radiation on CBM DAQ . . . . .	31
2.4.1. Placement of SRAM-based FPGAs . . . . .	31
2.4.2. Detector Dead Time . . . . .	33
2.4.3. Situation for CBM-ToF . . . . .	34
<b>3. State of the Art</b>	<b>37</b>
3.1. CBM-ToF Read-Out Controller . . . . .	37
3.1.1. Read-Out at Other HEP Experiments . . . . .	37
3.1.2. CBM Constraints . . . . .	38
3.1.3. Existing Implementations . . . . .	41
3.1.4. The Read-Out Controller Hardware Platform . . . . .	43
3.2. Radiation Tolerance for FPGAs . . . . .	43
3.2.1. Hardware Approach . . . . .	43
3.2.2. Redundancy . . . . .	45
3.2.3. Configuration Scrubbing . . . . .	52
3.2.4. Fault Injection Tests . . . . .	54
3.2.5. Fault Tolerance in High Energy Physics . . . . .	54
3.3. Verification Through In-Beam Tests . . . . .	55
3.3.1. SEU Counting . . . . .	56

<b>4. Approach</b>	<b>59</b>
4.1. Modular Firmware Concept for the CBM Read-Out Controller . . . . .	59
4.1.1. Motivation for Modularization . . . . .	59
4.1.2. Interface Requirements . . . . .	62
4.1.3. Towards Radiation Tolerance . . . . .	63
4.2. Choice of Radiation Mitigation Techniques . . . . .	64
4.2.1. Blind Scrubbing . . . . .	64
4.2.2. Selective TMR . . . . .	64
4.2.3. Fault Tolerant Protocol . . . . .	66
4.2.4. Fault Injection Tests . . . . .	67
4.3. In-Beam Tests . . . . .	67
4.3.1. Test Setups . . . . .	68
4.3.2. Beam Diagnostics . . . . .	69
<b>5. Implementation</b>	<b>73</b>
5.1. Main FPGA Design . . . . .	73
5.1.1. The Modules . . . . .	73
5.1.2. Handling the Multiplicity of Firmwares . . . . .	78
5.2. Radiation Mitigation for the FPGA . . . . .	79
5.2.1. Scrubbing . . . . .	79
5.2.2. Redundancy . . . . .	81
5.2.3. Fault Tolerance in Higher Design Levels . . . . .	84
5.2.4. Identification of Critical Components . . . . .	85
5.3. In-Beam Tests . . . . .	87
5.3.1. The Experiment Setups . . . . .	87
5.3.2. Preparation for In-Beam Tests . . . . .	90
5.3.3. Beam Diagnostics . . . . .	92
<b>6. Results</b>	<b>99</b>
6.1. Modular ROC Usage . . . . .	99
6.1.1. Operative Firmware . . . . .	100
6.1.2. Software Integration . . . . .	101
6.2. Radiation Mitigation Techniques . . . . .	101
6.2.1. Blind Scrubbing . . . . .	101
6.2.2. Fabric Resource Consumption . . . . .	102
6.3. In-Beam Tests . . . . .	103
6.3.1. Test Setup 2012 . . . . .	104
6.3.2. Test Setup 2013 . . . . .	110
6.3.3. Beam Diagnostics . . . . .	112
<b>7. Discussion</b>	<b>115</b>
7.1. Applying Results on the CBM Use Case . . . . .	115

7.2. Detector Dead Time Estimations Based on Parameters Measured at In-Beam Tests . . . . .	116
<b>8. Conclusion</b>	<b>121</b>
8.1. Summary . . . . .	121
8.1.1. Implementation of the CBM-ToF Read-Out Controller Firmware . . . . .	121
8.1.2. Evaluation and Implementation of Radiation Mitigation Techniques . . . . .	121
8.1.3. In-Beam Verification of the Implemented Radiation Mitigation Techniques . . . . .	122
8.2. Outlook . . . . .	122
8.2.1. Fault Tolerant Communication Module . . . . .	122
8.2.2. SEU Mitigation in Xilinx Series 7 FPGAs . . . . .	123
8.2.3. Resilience . . . . .	123
<b>9. Acknowledgments</b>	<b>125</b>
<b>A. List of Publications</b>	<b>127</b>
A.1. As Main Author . . . . .	127
A.1.1. Peer-Reviewed . . . . .	127
A.1.2. At Conferences . . . . .	127
A.1.3. Others . . . . .	128
A.2. As Coauthor . . . . .	129
A.2.1. Notable . . . . .	129
A.2.2. Others . . . . .	129
<b>B. Documentation for the GET4 Read-Out Controller</b>	<b>133</b>
B.1. Introduction . . . . .	133
B.2. Basic Functionality . . . . .	134
B.2.1. Slow Control . . . . .	134
B.2.2. Data Readout . . . . .	139
B.2.3. Deterministic Latency Messages (DLMs) . . . . .	141
B.3. The Modules . . . . .	142
B.3.1. Frontend Readout: GET4 . . . . .	142
B.3.2. Transport: Optics . . . . .	144
<b>C. Registers in the GET4 Read-Out Controller</b>	<b>149</b>
C.1. Addresses in the <i>Common</i> Address Space (0x 00 00 00 - 0x 0F FF FC) . . . . .	149
C.2. Addresses in the Module: Optics (0x 20 00 00 - 0x 2F FF FC) . . . . .	152
C.3. Addresses in the Module: GET4 (0x 50 00 00 - 0x 5F FF FC) . . . . .	153
<b>D. The Read-Out Controller Hardware Platform</b>	<b>157</b>
D.1. SysCore Architecture . . . . .	157

*Contents*

---

D.2. SysCore Boards . . . . .	159
D.2.1. SysCore Board Version 1 . . . . .	159
D.2.2. SysCore Boards Version 2.X . . . . .	160
D.2.3. SysCore Board Version 3 . . . . .	161
D.3. Final CBM-ToF ROC Board . . . . .	162
<b>Bibliography</b>	<b>165</b>
<b>E. German Summary</b>	<b>175</b>

# List of Figures

2.1. Overview of the planned research project FAIR. . . . .	18
2.2. Schematic illustration of the QCD phase diagram. . . . .	19
2.3. The two configurations of the CBM experiment. . . . .	20
2.4. Stages of a classic high energy physics detector read-out chain. . . . .	21
2.5. Example of a radiation caused electronics failure. . . . .	24
2.6. Illustration of the effect of ionizing radiation in semiconductor material. . . . .	27
2.7. Weibull curve. . . . .	28
2.8. Representation of combinational logic in look-up tables. . . . .	29
2.9. Very simplified illustration of the basic components of an FPGA. . . . .	30
3.1. Time representation with Epoch markers. . . . .	40
3.2. Illustration of the implementation of TMR. . . . .	47
3.3. Illustration of the vulnerability of TMR'ed logic and non-redundant logic. . . . .	49
3.4. Plot comparing (idealized) TMR'ed logic with non-redundant logic. . . . .	51
3.5. Comparison of traditional approach and SEU Counter approach to estimate failure rate of CBM detector. . . . .	56
4.1. The detector systems of the CBM experiment. . . . .	60
4.2. Basic illustration of the CBM read-out controller firmware design based on two modules. . . . .	63
4.3. Analysis of the GET4 read-out design. . . . .	65
4.4. Illustration of the resource consumption for the GET4 read-out firmware. . . . .	66
4.5. Illustration of the two setups for the in-beam tests in 2012 and 2013 respectively. . . . .	69
5.1. The modular design of the read-out controller firmware. . . . .	74
5.2. A schematic overview of the front-end module for the GET4 read-out. . . . .	77
5.3. A schematic overview of the configuration controller. . . . .	80
5.4. Comparing the SEU susceptibility of Hamming Coded FSMs, TMR'ed FSMs, and FSMs without redundancy. . . . .	82
5.5. VHDL implementation of a TMR'ed flip-flop . . . . .	83
5.6. VHDL implementation of CRC calculation. . . . .	84
5.7. VHDL code snippet showing how to implement FSMs with recovery state functionality. . . . .	85
5.8. Evaluation of three designs with the SEU injection method. . . . .	86
5.9. Illustration of the test procedure performed during the in-beam tests. . . . .	88

5.10. Oscilloscope measuring the response of an in-beam scintillator at COSY. . . . .	93
5.11. Usage of self-developing dosimetry film during 2012 beam test. . . . .	94
5.12. Information of beam characteristics provided by COSY on their homepage. . . . .	95
5.13. Two board setup used for the in-beam tests of section 6.3. . . . .	97
6.1. The modular design of the read-out controller firmware. . . . .	99
6.2. The graphical user interface of the <i>roclib</i> software. . . . .	102
6.3. Comparison of fabric resource consumption for the GET4 read-out firm- ware used for the 2012 in-beam test. . . . .	103
6.4. Illustration of the design behavior during the in-beam test 2012. . . . .	105
6.5. Analysis of 2012 in-beam tests results. . . . .	107
6.6. Analysis of the data recorded during the 2012 in-beam tests. . . . .	108
6.7. Analysis of TMR efficiency. . . . .	109
6.8. Illustration of the design behavior during the in-beam test 2013. . . . .	110
6.9. Analysis of 2013 in-beam tests results. . . . .	111
6.10. Analysis of the data recorded during the 2013 in-beam tests. . . . .	112
B.1. Schematic overview of the modular approach for the ROC design. . . . .	134
B.2. Bit organization of the GET/PUT commands . . . . .	135
B.3. CMD-Lists memory mapping. . . . .	137
B.4. Data path massages . . . . .	140
B.5. A schematic overview of the front-end module for the GET4 read-out. . . . .	142
B.6. A <i>CBMNet</i> data path packet. . . . .	146
B.7. The protocol on the <i>CBMNet</i> control path. . . . .	146
B.8. Data arrangement for DLMs. . . . .	147
D.1. Illustration of the <i>SysCore Architecture</i> . . . . .	157
D.2. Overview of the <i>SysCore Board Version 3</i> configuration system. . . . .	158
D.3. The <i>SysCore Board Version 1</i> . . . . .	159
D.4. The <i>SysCore Board Version 2</i> . . . . .	160
D.5. The <i>SysCore Board Version 3</i> . . . . .	161
E.1. Ressourcenverbrauch der GET4 Auslesefirmware. . . . .	178
E.2. Das Verhalten der Auslesefirmware während des Strahltests 2012. . . . .	179



# 1. Introduction

When Galileo Galilei constructed his experiments to examine the laws of objects in free fall he used a very simple setup of bodies of various materials rolling down ramps. Over the years, experiments that were constructed to push the boundaries of science and technology have become more and more complex. Today, one person alone can hardly design and construct an experiment that would be able to gain new knowledge about the laws of nature. Most of today's state-of-the-art experiments of this kind are designed and constructed within a collaboration of thousands of scientists and non-scientists.

This thesis is carried out in the context of such an experiment, the *Compressed Baryonic Matter* (CBM) experiment. CBM is a high energy physics experiment that is - at time of writing - under construction as part of the new particle accelerator facility *Facility for Antiproton and Ion Research* (FAIR) in Darmstadt/Germany. The goal of CBM is to create extremely dense matter and to analyze its characteristics. Therefore, heavy ion particles are accelerated to a very high momentum and are then aimed at target material. The impact of the heavy ions with the atomic nuclei of the target matter results in the creation of many different particles that can be traced and characterized in several types of particle detectors. The information from the detectors allows to draw conclusions concerning the laws of physics of the very dense matter that existed during the collision of the heavy ion and the atomic nucleus of the target material. Especially challenging is the vast amount of data that is produced by the detectors. The detector data cannot be stored completely but needs to be analyzed and reduced on-the-fly. Online analysis and preprocessing of the data is required already in early steps of the detector read-out chain.

The particular topic that is addressed in this thesis deals with the operation of SRAM-based *Field Programmable Gate Arrays* (FPGAs) close to the detector, an environment with a significant level of ionizing radiation. SRAM-based FPGAs are very flexible devices because they can be reprogrammed in the field, this means that their behavior can be enhanced even after they have been installed. This makes them an ideal candidate to execute such online data processing algorithms. The downside is that they are susceptible to the ionizing radiation in the detector cave. Radiation can disturb the operation of SRAM-based FPGAs. The usage of SRAM-based FPGAs in an environment such as the CBM detector cave is only possible with appropriate radiation mitigation techniques.

The alternative would be to use specially designed microchips, known as *Application Specific Integrated Circuits* (ASICs), for the read-out functionality. ASICs are much less susceptible to ionizing radiation than SRAM-based FPGAs because their logic is hard-wired for a specific task. However, ASICs require more development effort, as a whole chip needs to be designed and produced, solely to interface the front-end electronics. In

addition, ASICs are not very flexible, their functionality cannot be altered after the chip has been build.

For some areas of the CBM cave the radiation level is so high, that the operation of FPGAs is definitely not an option and the usage of ASICs is mandatory. Nevertheless, there are some other areas with (comparably) modest radiation levels where SRAM-based FPGAs might be the better option to use. The CBM-ToF detector electronic is to be placed in an environment where the expected radiation level is relatively low compared to environments of other CBM detectors. Nevertheless, the expected radiation level is still considered to be tough for operation of FPGAs. If the operation of an SRAM-based FPGA is not feasible in case of CBM-ToF front-end electronics, it is also not feasible for the other CBM detectors.

### 1.1. Objectives and Contributions

The aim of this thesis is to evaluate the feasibility of using commercial off-the-shelf hardware, in particular SRAM-based FPGAs, on or close to the CBM-ToF detector in a harsh radiation environment. This would allow a flexible and cost efficient design of the CBM-ToF read-out chain.

The focus of this thesis is not on the development of new radiation mitigation techniques for SRAM-based FPGAs, but on the evaluation and implementation of established techniques for the special use case at CBM. Most of the existing techniques were originally developed for space and military applications where the radiation level is much smaller than it is expected for CBM. On the other hand, the demand for reliability is much softer in the CBM use case. Since not all radiation mitigation techniques work equally well in all situations, detailed evaluation of established techniques is necessary.

State of the art radiation mitigation techniques were evaluated and selected techniques were applied to a complex detector read-out firmware. In some cases, state-of-the-art techniques had to be modified to suit the CBM use case. Also some extra considerations for higher abstraction levels of the system design had to be taken into account.

Finally, the efficiency of the applied radiation mitigation techniques was measured in two in-beam experiments.

The main contributions of this thesis are listed below.

- Implementation of a CBM-ToF Read-Out Controller Firmware.
  - The starting point for the work on radiation mitigation is a complex firmware to read-out data from the CBM-ToF front-end electronics. The firmware reads multiple front-end channels and multiplexes them to deliver the data to the CBM data transport network on a single optical link. Important information, such as channel numbers and extended time stamps, is added and the data is arranged according to a defined data format.
  - The firmware was not only used for this thesis, but also by the CBM-ToF group to read-out detector prototypes in the laboratory and during in-beam tests.

Maintenance (bug fixing) and improvements (implementation of additional features) of the firmware design is also provided in the scope of this thesis.

- Evaluation and implementation of radiation mitigation techniques
  - Two different techniques to implement redundancy for finite state machines, TMR'ed FSMs and Hamming-coded FSMs, were analyzed using a technique called fault injection. TMR'ed FSMs were chosen over Hamming-coded FSMs.
  - *Selected Module Redundancy* was introduced as a new approach to attenuate the TMR-overhead of fabric resource consumption to an acceptable level.
  - An existing implementation of *Scrubbing* was reactivated in order to evaluate its efficiency when applied on a complex detector read-out firmware that implements the aforementioned *Selected Module Redundancy* instead of the cost and resource intensive “full TMR”.
  - Higher system levels (e.g. communication protocols) are designed to automatically recover to an operational state after periods of erroneous system behavior.
  - Fault injection tests were used to identify and clean up issues in existing protocol implementations.
  - The selected radiation mitigation techniques have been implemented for the CBM-ToF read-out firmware.
- Verification of the implemented radiation mitigation techniques in two in-beam tests at the *Forschungszentrum Jülich/Germany*.
  - Design of an experimental setup to measure the efficiency of radiation mitigation techniques
  - Simplification of beam diagnostics for blind scrubbing setups by counting SEUs directly in parallel in a second device (SEU Counter Approach)
- Estimation of the impact of radiation effects on CBM read-out electronics
  - The detector dead time due to radiation-caused electronics failure was estimated based on the results of the in-beam tests.

## 1.2. Thesis Outline

First, in chapter 2, the reasoning that motivates this work is presented. A very brief overview of the Compressed Baryonic Matter experiment is given and the problems with electronics that are operated in radiation environments are explained.

The subsequent chapters 3, 4, 5, and 6 will then each be organized in three sections, addressing the following three, not really independent, yet distinguishable tasks respectively.

- implementation and maintenance of the CBM-ToF read-out controller firmware for interfacing the GET4 ASIC
- evaluation and implementation of radiation mitigation techniques, adapted for the use case of the CBM-ToF read-out controller
- in-beam verification of the radiation mitigated design that was implemented

The first bullet in the list involves a lot of engineering work that, although it is a part of the work for this thesis, shall not be emphasized too much. For that reason, many details of the actual functionality of the GET4 read-out controller are not presented in the main body of the thesis, instead they can be found in appendix B.

Chapter 3 summarizes previous work, known techniques, and existing implementations. The basic ideas behind the present work are then described in chapter 4 and implementation details are given in chapter 5. Results are presented in chapter 6 and discussed in chapter 7.

Finally, a summary of the thesis and an outlook to possible future tasks is given in chapter 8.

## 2. Motivation

This chapter provides the reader with information that is required to understand the motivation for this work.

First, section 2.1 gives some background concerning the CBM experiment, including the scientific motivation and the basic detector setup. Later on, the focus shifts towards the read-out chain for the CBM *Time of Flight* (ToF) system as this is the most relevant application for this thesis.

Then, section 2.2 gives a brief overview of radiation effects in electronic devices. It includes a basic introduction to the architecture of FPGAs.

The chapter concludes with section 2.4 where the specific problems that arise in the special use case of CBM, when operating FPGAs in the radiation environment close to the detector, are explained.

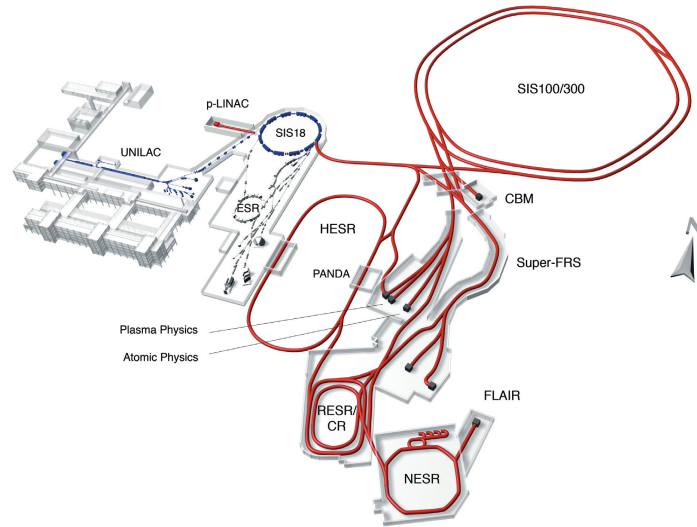
### 2.1. The CBM Experiment at FAIR

The *Compressed Baryonic Matter* (CBM) experiment is a high-energy physics experiment that aims at the investigation of nuclear matter at very high baryon densities but still modest temperatures. The experiment is an international collaboration of currently 57 institutes from 12 countries [FS13]. It is planned as part of the *Facility for Antiproton and Ion Research* (FAIR) that is currently under construction in Darmstadt/Germany as an extension to the already existing GSI facility. Figure 2.1 shows an overview of the planned facility.

Detailed information about the CBM experiment and the underlying physical phenomena can be found in the [FHK<sup>+</sup>11].

#### 2.1.1. Scientific Motivation

The general goal of high energy physics experiments is to gain a better understanding of the properties of matter under extreme conditions. Extreme conditions can for example be very high temperatures explored at the *Relativistic Heavy Ion Collider* (RHIC) and the *Large Hadron Collider* (LHC). These facilities can reach very high temperatures due to their very high collision energies. The CBM experiment is a fixed target experiment that cannot reach the energies of today's large collider experiments. Instead the focus of CBM is the exploration of the QCD (quantum chromo dynamics) phase diagram at very high baryon densities which is complementary to the investigations performed at RHIC and LHC.



**Figure 2.1.:** Overview of the planned research project FAIR. The existing GSI facility is shown in blue and the planned extension for FAIR is shown in red. The CBM experiment will be located close to the two large accelerator rings. The illustration is taken from [Aug06a].

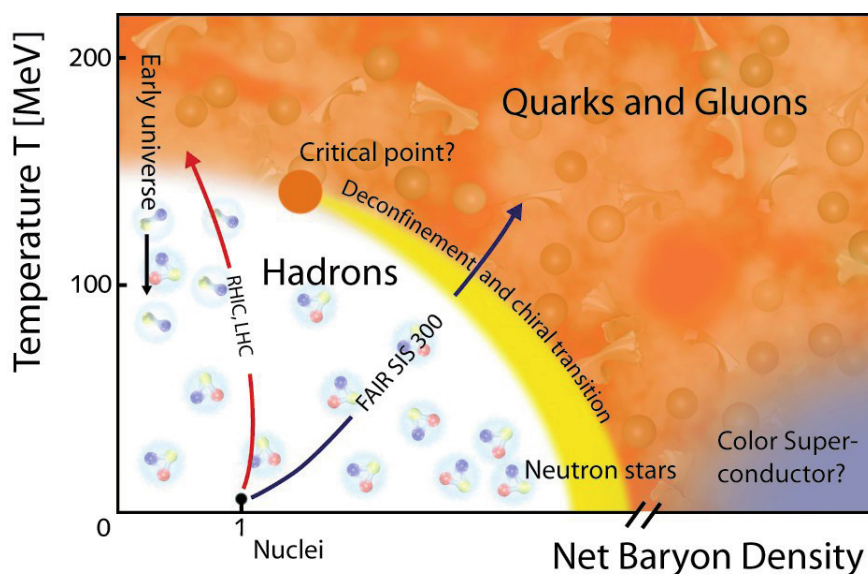
The term “phase diagram” is already known from basic chemistry when referring to the different states of matter (e.g. solid, fluid, or gaseous) depending on external conditions, usually pressure and temperature. However, under more extreme external conditions, further states of matter can be observed. Figure 2.2 shows QCD phase diagram with the hadronic phase at lower temperatures and densities while at higher temperatures, higher densities or both, the state known as *quark-gluon plasma* phase can be found.

The possible, but not yet experimentally observed, first-order phase transition between the hadronic phase and quark-gluon plasma is of special interest. CBM aims at exploring a region of the QCD phase diagram where such a phase transition can be expected.

### 2.1.2. The CBM Detector Setup

CBM is designed as a fixed target experiment to study the interaction of heavy ions that are collided at high energies. Therefore, high-energy heavy ions are aimed at a stationary target consisting of heavy ions as well. Different types of particles with various different properties are created as a byproduct of such a collision. When the types of these particles and their properties are known, it is possible to draw conclusions about the laws of nature governing the subatomic world. The purpose of the CBM experiment is to measure the type and properties of the particles created in such heavy ion collision events.

The full CBM experiment consists of several detectors that can be arranged in two different configurations, one with electron detectors (figure 2.3(a)) and one with muon detectors (figure 2.3(b)). Most relevant for this thesis is the *Time of Flight* (ToF) detector



**Figure 2.2.:** Schematic illustration of the QCD phase diagram. The hadronic state can be found in the region of lower temperatures and densities while at higher temperatures, densities, or both the state known as quark-gluon plasma is expected. Of special interest for CBM is the transition between these two states of matter. The illustration is taken from [Aug06a].

that is present in both layouts.

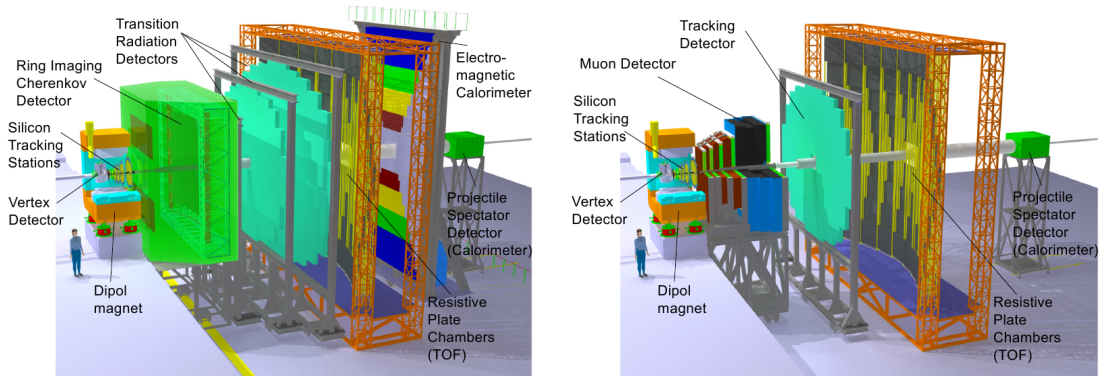
The main task of the CBM-ToF detector is to very accurately measure the arrival time of charged particles. CBM-ToF in conjunction with CBM-STs (*Silicon Tracking System*), allows to identify charged hadrons<sup>1</sup> i.e. tell if it was a proton, pion, kaon, etc..

The CBM-ToF system will be placed at a distance of about 10 m from the target and stretches out over an area of approximately  $12 \times 9 \text{ m}^2$ . The conceptual design of the CBM-ToF detector is presented in [DHA<sup>+</sup>14] and also in [TOF].

### 2.1.3. Self-Triggered and Time Stamped Paradigm

Since interesting events are also very rare events, CBM requires a very high event rate. CBM data acquisition is designed for event rates of up to 10 MHz which corresponds to a data rate of approximately 500 gigabyte per second (assuming a data volume of 50 kB per event), in some scenarios even a data rate of one terabyte per second is assumed [Aug06b, page 18]. In any case, this data rate is way too high to be completely stored in a realistic scenario. A realistic number would be in the order of one gigabyte per second. In consequence, CBM requires an online data event selection mechanism that rejects uninteresting background events to reduce the data rate by a factor of 500 or more.

<sup>1</sup>Hadrons are particles that consist of quarks, e.g. protons, neutrons, pions, but not electrons or muons.



(a) The CBM experimental facility with the electron detectors (RICH and TRD). (b) The CBM experimental facility with the muon detection system.

**Figure 2.3.:** The two configurations of the CBM experiment. Depending on the goal of the measurement either the electron detectors (*Ring Imaging Cherenkov Detector* (RICH) and *Transition Radiation Detector* (TRD) or the muon detector are included in the setup. The images are taken from the CBM-ToF technical design report [TOF].

Since the event topologies and resulting signal signatures of interesting events are very complex, no hierarchical trigger mechanism in the early hardware stages of the read-out chain is planned. Instead, all front-end electronics generate data autonomously and tag it with a time-stamp. The data is then pushed from the front-end electronics to a high-performance computer farm where the online event selection then happens exclusively in software [TOF, page 8]. This concept is different from the implementations in existing high-energy physics experiments.

The advantage of such a concept is that very complex event selection algorithms that cannot realistically be implemented as hardware triggers become possible with software. In addition, the software algorithms, compared to hardware-based trigger algorithms, are relatively easy to adapt to new criteria later. A further advantage compared to a triggered system is that there is no detector dead-time due to buffer read-out. The dead-time is reduced to the double-hit capability of the detector and front-end electronics.

On the other hand, new challenges emerge which are the high data volume that has to be pushed from the front-end to the computing farm and the global time distribution and time synchronization. Without global trigger, the event selector needs to be able to correlate all data from all parts of the detector by their time-stamp. A common clock for all front-end electronics of the whole detector is required and the global distribution of a common clock is not an easy task. In addition, the time in all front-end electronics needs to be synchronized to a common value.

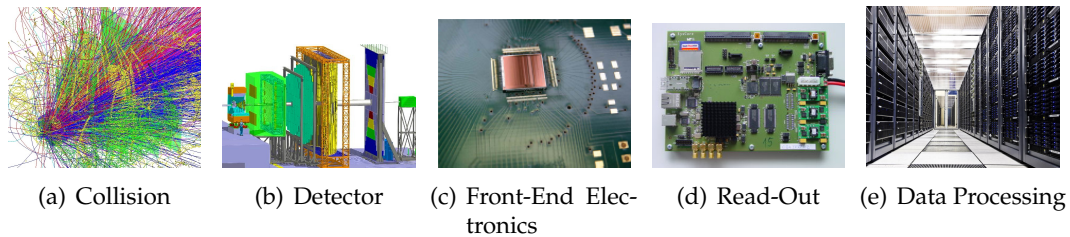
Naturally, one has to deal with many possible pitfalls when implementing such a conceptually new read-out chain for the first time. A first fully free-running and time-stamped detector read-out chain was set up by Pierre-Alain Loizeau during his doctoral thesis [Loi14]. One part that he used for this cutting-edge read-out chain prototype was



the read-out firmware that is described here in sections 4.1 and 5.1 as well as in appendix B.

#### 2.1.4. CBM-ToF Read-Out Chain

A classic detector read-out chain consists of several steps, as illustrated in figure 2.4. After the collision (figure 2.4(a)), a multitude of particles is generated that then pass through the active areas of the various detectors (figure 2.4(b)). The different detectors exploit different physical effects to gain the required information to characterize the particles. In most cases an electric pulse is created and fed to front-end electronics for further processing (figure 2.4(c)). There the analog pulse is amplified and the shape of the pulse is optimized before it is finally digitized. The digital information is then read out from the front-end electronics (figure 2.4(d)), converted to a global data format, and then transported to the global data processing stage. First pre-processing of the data can already happen at this read-out stage. The major part of data processing is then executed on a large computer cluster where the information from all detectors converges (figure 2.4(e)).



**Figure 2.4.:** Stages of a classic high energy physics detector read-out chain. The work of this thesis is centered around the read-out stage. Pictures: a) and b) from [FHK<sup>+</sup>11], c) from [Sch07], d) from GSI<sup>2</sup>, and e) from CERN<sup>3</sup>.

In a triggered system, the front-end electronics and the read-out stage need to implement the referring trigger algorithms, however, CBM follows a trigger-less approach (see section 2.1.3).

The CBM-ToF detector mostly follows this classic approach, the according stages are explained in the following.

**Multi-Gap Resistive Plate Chambers** The main purpose of the Time-of-Flight wall is the identification of charged hadrons. A main challenge is the coverage of the wide range of particle rates. The simulated particle flux reaches some  $10^4 \text{ s}^{-1} \text{ cm}^{-2}$  at the center region and drops almost exponentially with larger distance from the center. The particle rate at the outer boundary of the wall is three orders of magnitude less than at center region. A

<sup>2</sup><https://cbm-wiki.gsi.de/foswiki/bin/view/Public/PublicNxyter>

<sup>3</sup><http://cds.cern.ch/record/1103476>

further key requirement for CBM-ToF is the very fine time resolution of better than  $80\text{ ps}$ , including all contributions from start time, electronics, clock, cables, etc..

The CBM-ToF detector will be implemented with Multi-Gap Resistive Plate Chambers (MRPCs) as they can satisfy the requirements in terms of time resolution and rate capability. The huge particle flux gradient is addressed by using three different modules with different size and also different material depending on the expected particle flux.

A particle traversing the MRPC volume generates an avalanche of charged particles, that can be detected as a small current peak in the read-out electrode.

A detailed explanation of the MRPCs to be used for CBM can be found in [Dep13], [DHA<sup>+</sup>14], and [TOF].

**PADI / GET4 / CLOSY** The small signal of the read-out electrode is fed to an analog preamplifier and discriminator circuit (PADI) [CHH<sup>+</sup>13] that enhances the signal before it is digitized in the *GSI Eventdriven TDC with 4 channels* (GET4) ASIC [DF09]. About 25 000 GET4 ASICs will be assembled in the CBM-ToF detector. The GET4 ASIC performs a time-to-digital conversion (TDC) on four independent input channels by detecting rising and falling edges of the input signal and then creating a hit message. The GET4 can measure very precise hit time information of better than  $15\text{ ps}$  [Har13], the double hit resolution is better than  $5\text{ ns}$ . The digitized hit data is pushed to the next read-out stage via a serial protocol over an LVDS link.

A very precise and low-jitter clock is mandatory for the GET4 to be able to achieve the required precision. The chip is specified for a clock frequency of  $156.25\text{ MHz}$ , which is  $5/8$  of  $250\text{ MHz}$ . The appropriate low-jitter clock for the GET4 is provided by the CLOSY clock distribution system, that also generates a sync signal for synchronization. More details about the CLOSY system can be found in [Koc09].

In the CBM-ToF Technical Design Report [TOF], an alternative to the GET4 solution is considered as well. This alternative solution is based on an FPGA-TDC implemented on the TRB3 board [NAMH<sup>+</sup>13, UBKT12] and is currently planned as backup solution in case the GET4 approach fails for unforeseen reasons.

**ROC** The Read-Out Controller (ROC) is a data aggregator, an early data processor, and provides the controls interface to the front-end electronics. The currently planned system assumes data aggregation from 80 GET4 ASICs.

Since the work of this thesis is centered around this stage of the read-out chain, more detailed information about the functionality of the ROC is given later. The full documentation for the GET4-ROC can be found in appendix B and the underlying design approach is described in sections 4.1 and 5.1. The main topic of this thesis, however, is not the functionality of the GET4-ROC but the radiation mitigation techniques required for its implementation on an SRAM-based FPGA platform which are presented in sections 4.2 and 5.2.

**DPB** The Data Processing Board (DPB) is a second, hardware based data combining and data processing step foreseen in the CBM-DAQ. Sometimes the term *Data Combiner Board* (DCB) is used as well, mostly when the board does not perform any mentionable data processing but only acts as a channel combiner. The DPB will be operated in a non-radiation environment and will be available in most of the subdetector read-out chains. More complex data processing can be implemented here, for example the feature extraction for the SPADIC ASIC in read-out chain of the *Transition Radiation Detector* (TRD) (cf. [Garss]).

In case of CBM-ToF, the planned DPB implementation is more a “combiner” than a “processing” board, however, first evaluations for data processing in this stage have been made as well [XHD<sup>+</sup>13].

The DPB also packs the data in *micro slice containers*, the global data format required by the *First Level Event Selector* (FLES, see later).

**ABB / FLIB** The last hardware-based stage in the planned read-out chain is an FPGA card with PCIexpress capability. It will be installed in the entry nodes of the computing cluster where the software-based event selection is to be executed. The PCIexpress card receives the data from the DPB layer via optical connection.

The prototype that was available from the beginning of the work on this thesis is the *Active Buffer Board* (ABB) [GKW<sup>+</sup>09]. The ABB hardware is a commercial FPGA development board that could be purchased from Avnet, listed as *AES-XLX-V5LXT-PCIE110-G*. The firmware that received the data from its two optical connections, buffers it in on-board SRAM, and delivers it to the computing node via PCIexpress was developed and maintained by Wenxue Gao for his PhD thesis [Gao12].

The ABB also requires integration in the host operating system. The according Linux kernel driver is written by Guillermo Marcus, as part of his PhD thesis [Mar11].

For all the work of this thesis, the ABB was used. At time of writing, however, the PCIexpress board for receiving data at CBM is already the successor model to the ABB, the *FLES Interface Board* (FLIB). The FLIB is again a commercially available hardware, a combination of an FPGA development board from High Tech Global, listed as *HTG-K7-PCIE-325-2*, and an add-on board from Faster Technology, listed as *FM-S14/FM-S18/FM-S28*. The FLIB firmware and Kernel driver is developed and maintained by Dirk Hutter for his PhD thesis [Hutss].

For the real CBM experiment another reimplementation is foreseen with details not yet decided.

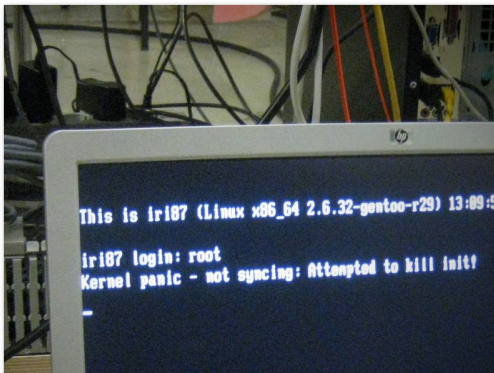
**Computing Node / FLES** At CBM, the first stage where data from all subdetectors is available is the *First Level Event Selector* (FLES). This high-performance cluster executes the software for online event reconstruction on the incoming streams data. The reconstructed events are then analyzed (also online) and only interesting events are selected for storage.

In the field of high performance computing, technical progress develops very fast, it cannot be foreseen today what technology will be available at the time the CBM experiment starts. To be able to select the best technology available at the time when the experiment starts, hardware and architecture of the FLES is not yet fully defined, only that it will be a state-of-the-art high performance cluster internally connected with a high-speed network.

For the experiments of this thesis (and also in other early prototype experiments), no online physics event selection is necessary yet as only few data streams are generated and hence the data rates are still modest. The incoming data stream can be fully stored, a single computing node running Linux is sufficient.

The work of this thesis was carried out using the early available CBM-DAQ reference software developed mainly by Sergey Linev at GSI and known by the name of *roclib* and *DABC* [AMEKL10b]. This software is compatible with the ABB board. For better compatibility with the future FLES design, a new implementation of the online event selection software is currently under development.

## 2.2. Electronic Devices in Radiation Environments



**Figure 2.5.:** Example of a radiation caused electronics failure: a computer crash during an in-beam test at COSY, Forschungszentrum Jülich.

Electronic devices that function flawlessly under normal conditions can be severely disturbed by radiation and they might fail in a radiation environment. Figure 2.5 shows such a failure, a PC that crashed because it was exposed to ionizing radiation. Radiation can affect the operation of electronic devices in different ways. In case of the experiment PC, the operation was disturbed only temporally and after a reboot the PC was fully functional again. However, some effects affect the physical structure of electronic devices which might even result in permanent device damage.

When particles traverse through matter they can interact with it and thereby deposit energy in the material. The amount of energy transferred into the silicon of an electronic device is a key parameter for the characterization of many radiation effects. The underlying effects can for example be ionization effects, bremsstrahlung, coulomb scattering, or nuclear effects. Important for radiation damage in electronic devices are ionization and nuclear effects.

Nuclear effects occur if a nucleus in the material is hit by the radiation particle. The scattering of the radiation particle and the nucleus can lead to displacement, decay, or excitation of the nucleus. Nuclear effects are not directly ionizing, but as a result particles

can be created in the material and they can be ionizing.

Ionization effects are caused by charged particles that interact with the electrons of the target material. The electrons are removed from their atoms, thereby electron hole pairs are created. If an electric field is present, electrons and holes are separated, otherwise they will recombine. The energy loss per distance of hadrons (protons, pions, ions) due to ionization effects is given by the Bethe formula, as shown in equation 2.1. For electrons the situation is slightly different. Due to their much smaller mass they also experience energy loss from *bremsstrahlung*, see [Leo94, chapter 2.4] for details.

The Bethe formula is:

$$-\frac{dE}{dx} = \frac{4\pi}{m_e c^2} \cdot \frac{z^2}{\beta^2} \cdot \left(\frac{e^2}{4\pi\epsilon_0}\right)^2 \cdot \frac{Z \cdot \rho \cdot N_A}{A \cdot M_u} \left[ \ln \left( \frac{2m_e c^2 \beta^2}{I(1-\beta^2)} \right) - \beta^2 \right] \quad (2.1)$$

with  $m_e$  the electron mass,  $c$  the speed of light,  $z$  the particle charge,  $\beta = v/c$ ,  $v$  the particle speed,  $e$  the electron charge,  $m_e$  the electron rest mass,  $\epsilon_0$  the vacuum permittivity,  $Z$  the atomic number of the material,  $\rho$  the density of the material,  $N_A$  the Avogadro number,  $A$  the relative atomic mass of the material, and  $M_u$  the Molar mass constant.

The relevant radiation effects on electronic devices can be characterized in the following categories.

- Cumulative Effects (destructive)
  - due to energy deposition, Total Ionizing Dose (TID)
  - due to lattice displacement, Non-Ionizing Energy Loss (NIEL)
- Single Event Effects
  - Destructive Effects or “Hard Errors”: Single Event Burn-Out (SEBO), Single Event Gate Rupture (SEGR), Single Event Latchup (SEL)
  - Non-Destructive Effects or “Soft Errors”: Single Event Transient (SET), Single Event Upset (SEU)

The next subsections will give a very brief description for the different effects. More detailed explanations can be found in corresponding literature, e.g. in [BSV11], [Leo94], or [Bau05].

### 2.2.1. Cumulative effects

Cumulative effects are gradual effects, depending on factors that are integrated over the time the device is exposed to radiation. A sensitive device will then fail after a device specific tolerance limit has been reached. The time of failure can be predicted if the tolerance limits of the device is known.

**Total Ionizing Dose** Ionizing particles, e.g. charged hadrons, electrons, gammas, or neutrons, deposit energy in the material when they pass through it. Gammas and neutrons are not directly ionizing, however, they can still induce ionizing energy depositions. The Total Ionizing Dose (TID) is the total amount of energy that has been deposited in the material by ionizing radiation over time. It is given in the SI-unit Gray (Gy) or in the out-dated unit *rad* which is still widely used ( $1 \text{ Gy} = 100 \text{ rad}$ ).

Electron-hole pairs that are created in silicon dioxide do not quickly recombine and are separated at presence of an electric field, e.g. at a transistor. Electrons are more mobile than holes and can leave the oxide whereas holes can be trapped in defect centers. This process also creates more defects at the interface between silicon and silicon dioxide. Charge and defect buildup are the reasons for device degradation in terms of TID, that can finally lead a broken device.

When a device is removed from the radiation environment, an opposing effect called annealing takes place. Holes can be detrapped at thermal energy, an effect that depends on temperature, type of semiconductor, process technology, etc.. Longer periods without radiation and at higher temperature helps to increase the lifetime of a device (careful, higher temperature during irradiation enhances the damage).

**Displacement Damage** Particles passing through matter, neutral or charged, can displace single atoms from their position due to non-ionizing energy loss (NIEL). Lattice structure and doping of semiconductor's material is disturbed, altering its electrical behavior.

The relevant parameter to characterize the bulk damage is particle fluence. Since it differs amongst different particles and also varies with the particle energy, particle fluence is commonly given normalized to the equivalence of 1 MeV-neutrons,  $n_{eq}/\text{cm}^2$ .

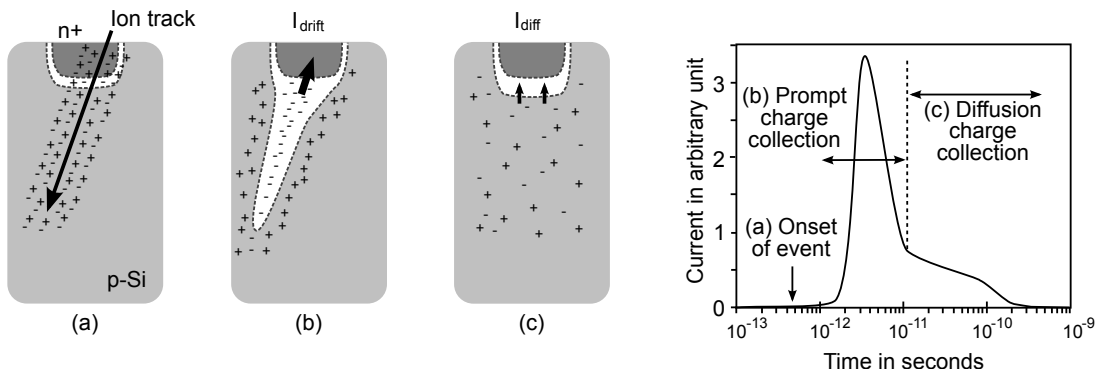
### 2.2.2. Single Event Effects

Unlike cumulative effects, Single Event Effects are spontaneous effects that are caused by a single particle and can happen at every moment. Their occurrence cannot be predicted, only a probability can be given which is usually expressed in terms of cross section.

An ionizing particle traveling through matter creates a track of electron-hole pairs that can cause various single event effects. In presence of an electric field, as in the depletion region of doped semiconductor material, the charges are separated and a current pulse is generated. Figure 2.6 illustrates the process.

Single Event Effects can be categorized as destructive ("hard error") and non destructive ("soft error"). While destructive effects damage mostly power MOSFET devices, non-destructive SEUs cause severe problems especially for SRAM-based devices.

**Hard Errors** Hard errors cause physical damage to the device. They are most often caused by heavy ions which can deposit the required amount of energy in the device.



**Figure 2.6.:** Generation of electron-hole pairs, charge collection and the resulting current pulse. This figure is adopted from [Bau05, Figure 2].

- **Single Event Burn-Out (SEBO)** In power MOSFETs, heavy ions can cause a drain-source voltage that is higher than the breakdown voltage of the element. The resulting current causes high temperatures and may melt the device locally.
- **Single Event Gate Rupture (SEGR)** Heavy ions can also cause an electric field strong enough to destroy the isolator of the gate oxide in a power MOSFET device. This causes a current flow that eventually heats and melts the device locally.
- **Single Event Latchup (SEL)** A short circuit that is caused by improper configuration of a semiconductor element is called latchup. Electrical latchups can be caused by an improper power supply or by transients on input/output lines. An SEL is a latchup induced by an ionizing particle.

Without protection measures, this effect can lead to thermal destruction of the circuit. Semiconductor manufacturer are aware of latchups caused by improper powering and transients on input/output elements and most devices are protected against these effects. Latchups caused by ionizing radiation (SELs) are not a very common scenario and therefore semiconductor manufacturers are less aware of them, but still, modern devices are usually hardly sensitive to SELs. Xilinx even claims their devices to be “immune to destruction by parasitic bipolar structure latchup” [LDF<sup>+</sup>05].

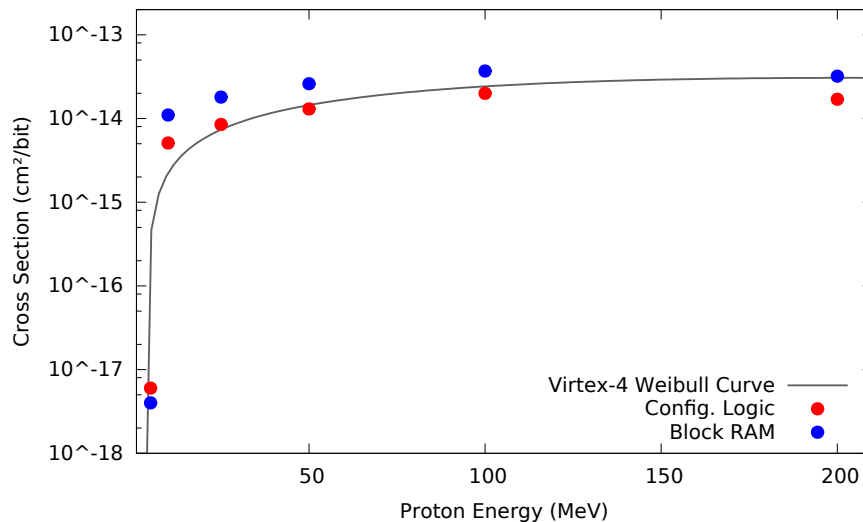
**Soft Errors** Soft errors do not damage the device but they disturb their operation. Single hadrons (protons, pions,...) cannot deposit enough energy to upset modern devices by direct ionization, however, the recoil from an interaction of a single hadron with a nuclei can [Fac99, p. 32]. The relevant parameter to estimate soft error rates is the flux of high-energy hadrons (in  $1/cm^2/s$ ). The following soft errors exist:

- **Single Event Transient (SET)** An ionizing particle can induce sufficient charge on a transistor to change the voltage over or under the threshold limit. This generates an

asynchronous glitch at the output of the transistor path which propagates through the circuit. It is very short lived and readjusts quickly, in the order of picoseconds to nanoseconds [BSV11, p. 46]. This is called a Single Event Transient (SET).

- **Single Event Upset (SEU)** If an SET reaches a latch or a flip-flop, or occurs at a transistor that belongs to a latch or flip-flop, it can be sampled and become a static error. In this case it is referred to as *Single Event Upset* (SEU), or simply *upset*, or *bit-flip*. The term SEU is used in general when an ionizing particle changes the state of a memory cell. It is a non-persistent error that can be remedied by rewriting the correct value to the memory cell. In worst case, a power cycle is required.

In real-life, the components suffering most from SEUs are SRAM cells. SRAM cells are relatively susceptible, widely used, and when they are used, usually a large number of cells is present in the system. SEUs occur when enough energy is generated in the sensitive volume of the SRAM cell. Figure 2.7 shows the dependency of the SEU probability of Virtex-4 devices when they are hit by protons of various kinetic energies. The curve is different for other particles, it depends on how much energy they deposit in the material, a value called “linear energy transfer” (LET). The LET value depends on the kind of particle (proton, electron, heavy ion, ...) and its kinetic energy.

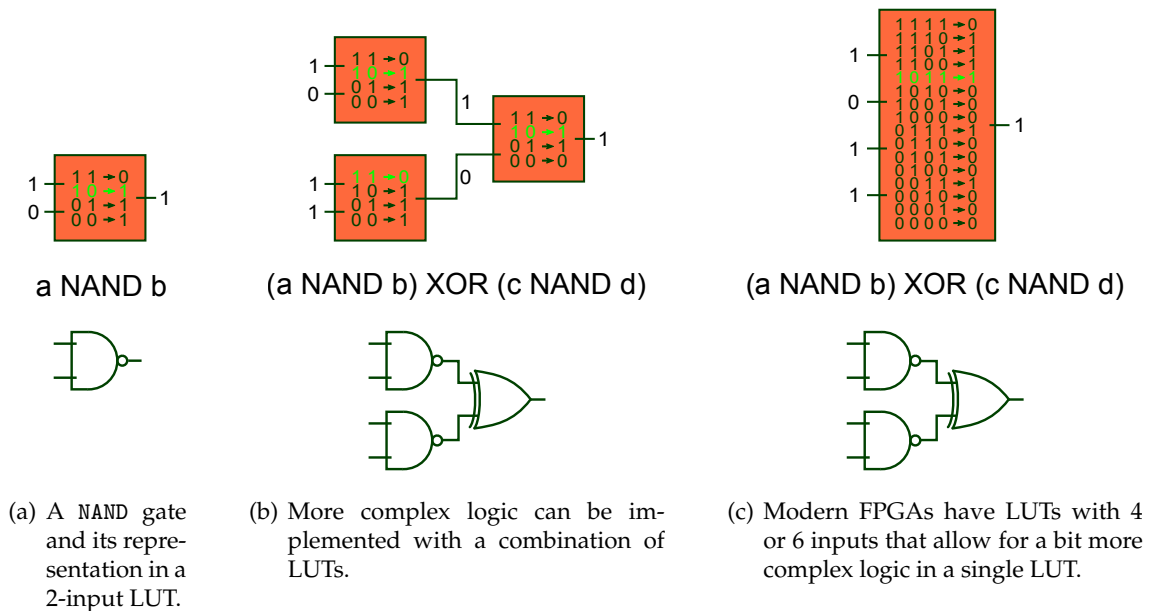


**Figure 2.7.:** The dependency of the SEU cross section for configuration bits of Xilinx Virtex-4 devices from the energy of the protons. The line plot represents what is known as the “Weibull curve”, the most widely used model to fit SEU cross section data (formula and Virtex-4 fit parameters are taken from [EMWG06]). The values plotted in red and blue refer to experimental data (extracted from [GKS<sup>+</sup>06, Figure 6]).



### 2.3. FPGA Architecture

Before discussing the placement of SRAM-based FPGAs, it is important to understand the basic concept of how FPGAs work. This section gives a very brief overview of principles of an FPGA. The emphasis is thereby on SRAM-based structures in the FPGA as they are important for this thesis.



**Figure 2.8.:** Illustration of the representation of combinational logic in look-up tables (LUTs) as it is done in FPGAs.

Any Boolean function can be represented in a truth table and any combinational logic<sup>2</sup> can be implemented by a combination of Boolean functions. The basic idea for modeling combinational logic is to store the result of every possible combination of input values of the underlying Boolean functions in look-up tables. Figure 2.8 illustrates this principle.

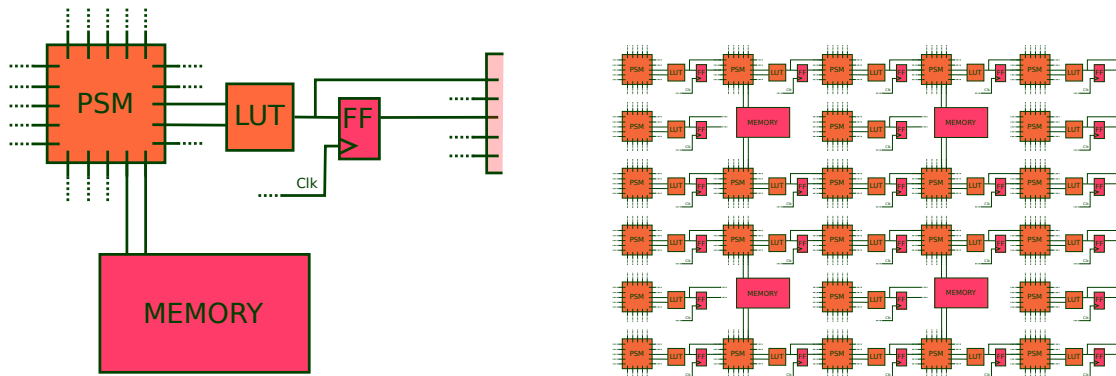
For the realization of sequential logic<sup>3</sup>, like a counter or a shift register, an additional component called flip-flop (FF) is required to store interim values. A flip-flop stores its current input value at time of a rising (or falling) edge of a clock signal and provides this value at its output. Flip-flops allow to store the output of combinational logic for the next clock cycle.

Very complex sequential logic can be implemented by cleverly connecting many look-up tables and flip-flops. Since LUTs and FFs are implemented with SRAM cells, their content can be arbitrarily configured. The interconnection of the components, however,

<sup>2</sup>Combinational logic is logic, where the output vector only depends on the current input vector, but not on the history of the input vector.

<sup>3</sup>Sequential logic is logic where the output not only depends on the current input vector but also on the history of the input vector. It requires the storage of interim values.

also contributes to the full description of the logic. To be able to fully configure any sequential logic, the interconnections between the components need to be configurable as well.



(a) The main building blocks of an FPGA are Programmable Switch Matrices (PSMs), Look-Up Tables (LUTs), and Flip-Flops (FFs), in most cases there are also some on-chip memory blocks available. LUTs are used to reflect the combinational logic of a design, FFs and memory store the current state of the system, and PSMs are used to interconnect everything.

(b) To put in a nutshell, an FPGA is build from a matrix of many of the units that are shown in figure 2.9(a). The logic components are interconnected via the reconfigurable PSMs, they determine the routing. Using several LUTs in an hierarchical connection scheme allows for describing more complex combinational logic (see figure 2.8(b)).

**Figure 2.9.:** Very simplified illustration of the basic components of an FPGA. Not shown are for example IO-buffers and special components like clock managers. The basic components of an SRAM-based FPGA (PSMs, LUTs, FFs, and Memory) are implemented using SRAM cells, and hence they are susceptible to single event upsets (SEUs). Important to remember is, that the values stored in PSMs and LUTs are static (this means they do not change during operation of the chip), while the values stored in FFs and on-chip memory are dynamic (these values may change during operation).

In FPGAs the configuration of the interconnections is done by programmable switch matrices (PSMs), which are usually also SRAM-based. The components are connected to a large net of static wires. The trick is, that two wires that end in the same PSM can be connected or disconnected depending on the configuration of the SRAM cells of the PSM. This allows for very versatile interconnections of the various components of the FPGA. Figure 2.9 shows a very simplified illustration of the basic architecture of an FPGA. An FPGA is basically a matrix of LUTs, FFs, and PSMs, normally equipped with some on-chip memory as well.

A clever usage of LUTs, FFs, and PSMs now allows to define any sequential logic as long as enough fabric resources are available on the FPGA.

With SRAM-based FPGAs, all the reddish units in figure 2.9 are implemented with SRAM cells, and hence, they are susceptible to single event upsets (SEUs). Important to remember is, that the values stored in PSMs and LUTs are static (this means they do not change during operation of the chip), while the values stored in FFs and on-chip memory

are dynamic (these values may change during operation).

### 2.4. Expected Impact of Radiation on CBM DAQ

CBM is aiming at very high interaction rates, which entail a very harsh radiation level in the detector cave. The radiation level, however, varies strongly with the location inside the cave. Therefore, selection and placement of electronic components have to be compatible with the fast hadron flux as well as with the radiation dose they are exposed to during the experiment.

The radiation level at the CBM silicon tracker system (STS) is so high, that even ASICs have to be designed very carefully [Aug06b, chapter 10.1]. As other radiation tolerant solutions did not fit into the limited space available for STS electronics, the *HUB chip* was designed and first prototypes have been produced [Lem12].<sup>4</sup>

The most relaxed radiation requirements for the read-out electronics can be found at the CBM-ToF detector. Also, ToF does not entail too complex data manipulations (in contrast to TRD for example [Gar14]) that further complicate the implementation of radiation mitigation techniques. If usage of SRAM-based electronics is not feasible for ToF, it is not an option for all the other detector systems as well.

At this point it might be noteworthy, that although this thesis focuses on radiation mitigation for SRAM-based FPGAs, there are also other components that can have issues when operated in a radiation environment, for example Flash-based technology, power regulators, or the receiver diodes of optical connections. Such components also require attention, but this is beyond the scope of this thesis.

#### 2.4.1. Placement of SRAM-based FPGAs

The general advantage of FPGAs is their flexibility. FPGAs can be reconfigured which allows to adapt their behavior even after the full system is built. This means, that bugs in the FPGA firmware that are discovered after commissioning of the detector can be fixed and also that additional features can be implemented, allowing to adapt to new use cases.

The downside of FPGAs is that they cannot reach the compactness and the very high clock frequencies of ASICs. In the CBM use case, an additional major problem is radiation. FPGAs which are competitive in terms of size and speed are based on SRAM technology. For that reason, they are susceptible to radiation, mostly to single event effects.

**Single Event Effects** Due to their large capacitive loading of single paths, single event transients (SETs) do not constitute a severe problem for Xilinx FPGAs [LDF<sup>+</sup>05].

---

<sup>4</sup>Since more space for readout electronics became available with the new magnet design end of 2013, a more mature project from CERN, the *GBTx* ASIC, could be adopted for CBM as a drop-in replacement for the *HUB chip* [Mül14]. At time of writing, the *GBTx* ASIC is the prioritized technology choice for STS and also for other detector systems.

However, all SRAM-based FPGAs suffer severely from Single-Event Upsets (SEUs) that change the storage value of an SRAM cell. Since so many parts of the internal structure of an FPGA are implemented with SRAM technology, SEUs are the most critical radiation effect for the operation of FPGAs on or close to the CBM detector.

The “static” SRAM cells of the FPGA (LUTs and PSMs) can be refreshed during runtime without interrupting the operation of the device. This technique, which is known as *scrubbing* is described in section 3.2.3. It is important to understand, that scrubbing is not a universal remedy against SEUs. Scrubbing does not prevent SEUs, it only repairs the induced error. Even with scrubbing enabled, SEUs can affect the static FPGA configuration for a short time until they are repaired. The dynamic FPGA memory (FFs and Memory) are not SEU-mitigated at all by scrubbing.

Additional countermeasures for the protection of “dynamic SRAM cells” and to sustain temporal occurrences of SEUs in “static SRAM cells” are required. This is usually achieved by implementing the design with redundancy (see section 3.2.2). The problem with redundancy is, that it only protects against single bit upsets but not against multi-bit upsets. If SEUs are not repaired, SEUs will accumulate and multi-bit upsets will occur eventually.

So, only the combination of both, “scrubbing” and a redundant system design can provide good protection against SEUs. Unfortunately, there are also limits to this approach. “Scrubbing” does not instantaneously repair an SEU but usually takes some 10 – 100 ms. If the radiation level is high enough, so that multiple SEUs occur already during one scrubbing cycle, the approach fails as well.

For a reasonably complex system, the implementation of full redundancy (including critical components such as clock managers) cannot realistically be implemented anyway and full reliability cannot be guaranteed.

Fortunately, full reliability is not required as the detector cannot provide 100% efficiency anyway, see next section (2.4.2).

The decision how deep into the radiation zone SRAM-based FPGAs should be placed, has to maintain the appropriate balance between reliability requirements and saving cables.

**Cumulative Radiation Effects** Cumulative radiation effects are not a severe problem for SRAM-based FPGAs at CBM. Modern SRAM-based FPGAs can be operated until a total dose of up to 300 *krad* / 3 000 *Gy* is accumulated [DFLH08]. And this does not take into account several device annealing effects that happen at room temperature when the particle beam is switched off. According to FLUKA simulations for CBM-ToF [Sen11, pages 31/32], critical TID values are not reached during lifetime of CBM when operated not too close to the beamline and operating SRAM-based FPGAs that close to the beamline is not possible anyways due to too severe single event upset effects.

Although cumulative effects can be neglected for SRAM-based FPGAs, they cannot for Flash-based FPGAs. At time of writing, more and more Flash-based FPGAs become available that come in sizes and achieve frequencies one might be tempted to consider

as a viable option for read-out controllers. However, TID effects are a much more critical problem for Flash-based devices than for SRAM-based FPGAs and this rules out such Flash-based FPGAs as an option for CBM.

It should be noted, that some solutions for SRAM-based FPGAs systems entail Flash technology as well. For example, the current solution for the radiation mitigation technique “scrubbing” (see section D.1) is based on an on-board Flash memory in conjunction with a small Flash-based FPGA that acts as configuration controller for the main SRAM-based FPGA. Due to the low TID tolerance of Flash technology, this can only be a preliminary solution during research and development. Therefore, for the real experiment a new scrubbing scheme is planned that exploits the internal scrubbing controllers that became available with the Xilinx Series 7 FPGAs (see section 8.2.2).

### 2.4.2. Detector Dead Time

Detectors are built to be sensitive to certain kinds of particles passing through them, this is the purpose of the detector. During operation of a particle detector, it is desired for the detector to be sensitive to the according particles all the time. However, this is not achievable for multiple possible reasons.

After a particle was detected, the detector can lose its sensitivity for a short time, e.g. because the passing particle caused the discharge of an electric potential that then first needs to recover before another particle can be detected. For the short time until the potential is recovered the detector is called to be “blind” or “dead”.

Another reason affects detectors that measure the energy that a passing particle deposits in the detector’s material. If two particles pass in rapid succession, or even in partially overlapping time frames, the resulting signal might not be separable into two events but appear as only one event. This is called a “pile-up”.

The analog read-out electronics can also be a reason for dead time. For example the shape time or the digitizer can contribute to dead time.

Furthermore, in modern high energy physics experiments there are a lot of read-out channels. For cost efficiency the read-out chain cannot be designed to transport and process the worst case data rate for a long time. Even if shorter peaks with increased data rate can be buffered, longer periods of too high data rate completely fill the buffers. And such longer periods might be inevitable in some cases, hence, there is always the chance of a lost data due to full buffers.

Full buffers are a particularly serious problem for self-triggered data acquisition systems. Without a global trigger it can easily happen that all the time part of the channels drop data. A “full picture” of an event is then never recorded. In such a situation of overload, it is desirable to drop data from all channels synchronously and thereby assure that part of the time no data is lost due to full buffers. This requires intelligent throttling mechanism.

Another cause for detector dead time is device failure. For example, if the configuration of an FPGA gets corrupted (e.g. by an SEU) the device might no longer work properly

and data is lost. The further up in the read-out chain hierarchy the failing device is located, the more channels are affected, and hence the more dead time is generated. A crash of the high performance cluster for online data analysis might cause a significant down time of the whole experiment.

The bottom line is, that there is detector dead time anyway, for various reasons. So a small contribution to detector dead time as a result of radiation caused electronics failure can be accepted, as long as it does not significantly increase the overall detector dead time.

### 2.4.3. Situation for CBM-ToF

The numbers of the following section are mostly taken from the ToF technical design report (ToF-TDR) in the version as of October 2014 [TOF].

The CBM-ToF design foresees about 100 000 electronic read-out channels distributed over an active area of about  $120 m^2$ . About 30 000 channels originate from the inner region where the radiation level is significantly higher.

For a system with such a large amount of read-out channels, the price per channel is a critical parameter, it cannot be too high. A channel combiner is required in the early stage of the read-out chain in order to avoid the tremendous cost for cabling. Cost is not the only reason for such a data combiner. Just physically building such a huge amount of copper connections is not a very easy task, and on top of that, it would result in a physical design that is hard to maintain. Furthermore, it would add a significant amount of material to the detector which should be avoided if possible.

The GET4 ASIC already combines four analog input channels on one digital output link, but this is not enough. The current plan is to connect 80 GET4s to one read-out controller board that then combines the data and sends it over an optical link to the next stage in the read-out chain.

The usage of a commercial off-the-shelf SRAM-based FPGA for this read-out controller would bring along the huge benefit of flexibility at relatively low cost. It allows for flexible data processing at a very early stage in the read-out chain.

However, the FPGAs have to operate in an environment with a significant radiation level. Single event upsets (SEUs) created by fast hadrons are expected to be the major obstacle for operation of SRAM-based FPGAs in such an environment. According to FLUKA simulations, the flux of fast Hadrons in the inner region is about  $\sim 10^4 s^{-1}cm^{-2}$ , and in the outer region  $\sim 10^2 s^{-1}cm^{-2}$  [Sen11].

Techniques are available to mitigate SEU effects in SRAM-based FPGAs, however, for reasonably complex systems full reliability cannot be guaranteed. Since those techniques also lead to increased cost, compromises might have to be made to trade off reliability for cost efficiency.

As a requirement for CBM-ToF, the overall detector efficiency should be better than 95%, the contribution of electronic failures to this 5% accepted efficiency loss should not be appreciable. On the other hand, additional detector dead time of a few per mille is

indeed acceptable.

However, the question how high the expected contribution of SEU effects to detector dead time actually remains. Without additional information, this contribution cannot be predicted very accurately. The following points remain vague.

- How many FPGA resources are required for the implementation of a GET4 read-out controller firmware? Especially, when radiation mitigation techniques have to be implemented on top of the basic functionality.
- What is the average number of SEUs that a complex read-out firmware can sustain until a functional failure occurs?
- How good can radiation mitigation techniques work when cost efficiency is also a criterion?
- How long does it take to repair a device after a failure?
- Are there any show-stoppers hidden in the details of implementing such a radiation mitigated system at reasonable cost?

Whether the usage of SRAM-based FPGAs in early stages of the CBM-ToF read-out chain is viable or not cannot be decided without clarifying these points. The work of this thesis was carried out to obtain the missing information. For that, a firmware to read-out the GET4 ASIC was implemented, cost-efficient radiation mitigation techniques were applied to that firmware, and finally the efficiency of the applied techniques was experimentally evaluated.





## 3. State of the Art

This chapter presents the state-of-the-art of science and technology at the time when the work on this thesis began. It is organized in three sections that reflect the three topics already mentioned in section 1.2.

Section 3.1 covers the situation of the CBM-ToF read-out controller. The subject concerning radiation mitigation techniques for SRAM-based FPGAs is then presented in section 3.2. Details are given for *Triple Modular Redundancy* (TMR) and *Scrubbing*, the techniques that have been selected to be used in the present work. Finally, section 3.3 discusses very briefly the common procedure of using in-beam tests for verification and efficiency evaluation of radiation mitigation techniques. This also induces a small subsection about SEU counting by configuration read-back which is relevant later for beam diagnostics.

### 3.1. CBM-ToF Read-Out Controller

This section describes the status of the CBM-ToF read-out controller before the work of this thesis was started. It consists of a comparison to other high-energy physics experiments, an explanation of the specialties of CBM, and also sheds some light on the status of similar implementations that already existed within the CBM community.

#### 3.1.1. Read-Out at Other HEP Experiments

Every high energy physics (HEP) experiment is different. Their data read-out components are highly integrated into the individual system and therefore are not easily comparable amongst each other. A comparison of CBM read-out electronics to the read-out electronics of existing HEP experiments is even more difficult. The reason for that is discussed now.

Present generation of major HEP experiments implement their data acquisition architecture based on central triggers, the data rate is reduced for read-out in multiple steps. The first step of the data acquisition chain has to decide on a very small time scale whether the recorded data shows characteristics of an interesting event or not. This decision is called a trigger. Only if the event is classified as interesting it is transported to the next step in the chain. In the next step the incoming data rate is already reduced since data that does not qualify to a triggered event is discarded. Therefore, in each step more time is available for the trigger decision, and hence a more precise decision can then be made.

The major challenge in implementing components for such triggered data acquisition systems is the requirement for low latency [dC09]. The step needs to come to a decision as fast as possible. For example, the ALICE DAQ (for Run 1, 2009-2012) restricts the maximum latency of the level-0 trigger to 800 ns, the level-1 trigger to 6.5  $\mu$ s, and the level-2 trigger to 87.6  $\mu$ s [Kir07, page 41]. Data has to be kept in the buffers until a trigger decision is reached and no further data can be taken, this can cause detector dead time.

CBM data acquisition comes with the novel approach of a free-streaming, self-triggered architecture. This is a completely new paradigm for the read-out electronics. Latency is no longer a crucial parameter, the main problem for readout is now data throughput and online data reduction. For that reason, a comparison to existing HEP data acquisition systems based on central triggers is not practical.

At time of writing, none of the major HEP experiments already implement self-triggered data acquisition. LHCb<sup>1</sup> and PANDA<sup>2</sup> plan to implement free-streaming read-out architectures as well, LHCb for their next upgrade in 2018 [Ale13], PANDA for their start version [KBD<sup>+</sup>12] that will be put into operation at FAIR, at the same time as CBM.

Today, at the end of 2014, LHCb is evaluating the usage of FPGAs in radiation environment, but did not come to a conclusion so far [FUW<sup>+</sup>14].

PANDA follows a similar approach as CBM, including an FPGA in radiation environment for online data reduction. However, their radiation level is lower ( $\sim 60$  hits/cm<sup>2</sup>s) and currently, they do not foresee any radiation mitigation techniques for their FPGAs [KBD<sup>+</sup>12].

### 3.1.2. CBM Constraints

The CBM data acquisition system differs from those of previous high-energy physics experiments as it is planned as a free-streaming, data-push architecture rather than a system that is based on a hierarchical set of trigger decisions. This induces some constraints on the CBM system architecture.

#### CBMNet

One of the most challenging tasks with the concept of a free-streaming detector read-out is the common representation of time in all detector components. Unlike in triggered systems, there is no global decision to start the read-out of detector data. Every component decides locally when to send out data. To later enable a correlation of data from different components, all data needs to be tagged with an unambiguous time stamp. Since two clocks derived from two different sources never run at the exact same frequency, they drift apart over time. Therefore, either all the clocks that are used in all time critical components are derived from the same clock source, or the clock drift of all clock sources is known very precisely. In laboratory setups with only few components it is feasible to

---

<sup>1</sup>LHCb: Large Hadron Collider beauty experiment

<sup>2</sup>PANDA: Proton Antiproton Annihilations at Darmstadt

record and log the drift of all clocks involved. However, this approach does not scale to a big experiment like CBM with some hundreds or even thousands of independently operating entities. Therefore, distribution of a common clock and the synchronization of the front-end electronics is required.

CBMNet is a connection protocol designed at the *Computer Architecture Group* in Mannheim/Germany which provides the features required by CBM [Lem12]. It can be used to

- distribute the clock,
- synchronize the time with special messages that travel through the network with deterministic timing, so called deterministic latency messages (DLMs),
- transport control messages on a reliable channel,
- and transport data messages on a fast channel.

A project with comparable objectives is the GigaBit Transceiver (GBT) project developed at CERN [MMK07]. However, the present work is based on CBMNet.

**Clock Distribution** Clock distribution is one of the key features of CBMNet. CBMNet is designed to use the recovered clock from one dedicated optical receiver as system clock and also to use this recovered clock as reference clock for other optical connections, The clock can thereby be further distributed to the next components in the read-out chain. However, to be able to do so, the jitter of the recovered clock needs to be below 40 ps RMS [Lem12, page 68]. Since each step that tempers with the clock also adds jitter to the clock signal [Xil08, page 74], clock distribution over several hierarchy levels requires a dedicated jitter cleaning device. This is the reason why such a the jitter cleaner being present on the *SysCore Board Version 3* (see section D.2.3).

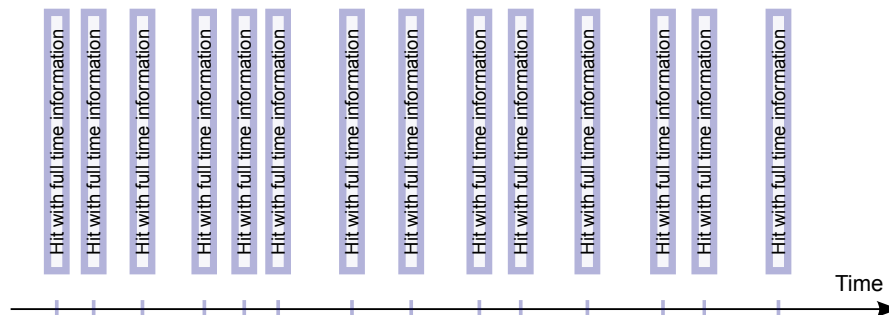
Finally, with such a clock distribution, a clock from a single time master can be distributed down the tree of the read-out chain and all components will be clocked synchronously without clock drift.

**Time Synchronization** Another key feature of CBMNet is the system-wide time synchronization with so called “Deterministic Latency Message” (DLM). DLMs are 4 bit wide messages which are transported through the CBMNet cores and over the optical fiber with a deterministic delay. The delay might be different from connection to connection because of different cable length, but it will not vary for two DLMs over the same connection. Since DLMs work in both directions, the delay can be measured easily via loopback. Once the delays of all connections are known, DLMs can be used to start the time stamp counters in all components coherently. A dedicated DLM for synchronization can be sent periodically with a defined interval to allow for components to check if they are out of sync.

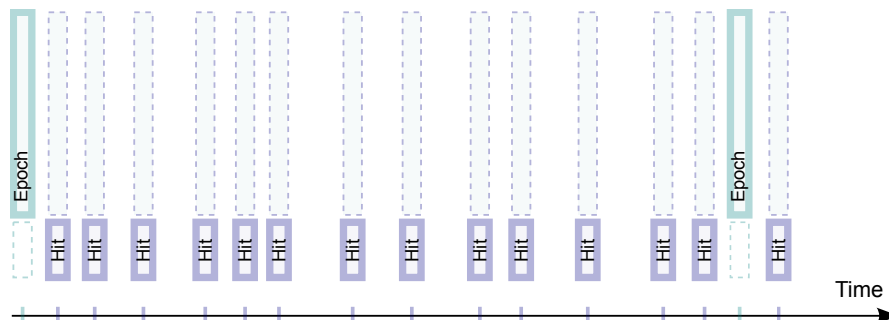
## Time Encoding

The self-triggered and free-streaming approach of the CBM experiment requires a unambiguous time encoding for the time span of a whole run. A run typically lasts from several minutes to hours, or even days. Time precision, on the other hand, needs to be in the order of nanoseconds, for CBM-ToF even in the order of tens of picoseconds. A lot of bits are required to encode time with the required precision while still covering the full run time. For example, already 48 *bits* are required to cover a 3 hours run at a precision of 50 *ps*.

To transport all those bits for every hit in every component would result in an unnecessary high utilization of bandwidth. “Unnecessary”, because the more significant bits of different hits do not change very frequently and are therefore in most cases just a repetition of the information of the previous hit.



(a) Every hit carries the full time stamp.



(b) Upper bits of the time stamp counter are communicated via Epoch markers, hits messages only carry the lower bits.

**Figure 3.1.:** Both diagrams show the same hit sequence. Less bandwidth is required when Epoch markers are used as for each hit message the number of bits can be significantly reduced.

The repetitive sending of the more significant bits is avoided in CBM with the introduction of Epoch markers in the data stream. The bits of the time stamp counter are subdivided into two groups, the “higher bits” vector and the “lower bits” vector. The “higher

bits” only change when the “lower bits” counter overflows. Every time the “higher bits” change, an Epoch marker is inserted in the data stream carrying the updated value of the higher bits. Hit data is only tagged with the lower bits of the time stamp counter. Figure 3.1 illustrates this principle.

However, this concept also entails some challenges. The insertion and handling of Epoch markers has to be implemented very accurately. It would result in a huge time difference for a hit if it is sent out before or after an Epoch marker. In addition, merging of multiple data streams becomes very complicated, especially if the preceding data path contains buffers. In that case, hit data from different data streams is not correlated in time and might belong to different epochs. Furthermore, a lost Epoch marker would corrupt a lot of data.

At the time of writing, the implementations used for the CBM read-out chain are based on Epoch markers. In case of the GET4, a hit message comprises 18 *bit* time information, and an Epoch Message, which is inserted in the data stream every 26.2144  $\mu$ s, comprises 32 *bit*.

### 3.1.3. Existing Implementations

At the beginning of the work for this thesis only one self-triggered and free-streaming front-end ASIC was available in the CBM community: the nXYTER [Sch07]. The nXYTER chip is the first prototype ASIC used for the CBM read-out chain. In contrast to the GET4 chip, for which the read-out controller (ROC) was developed in the scope of this thesis, the nXYTER chip is only intended for research and development and not for the final experiment, thus it does not implement measures countering radiation effects.

To read out data from the nXYTER, a SysCore Version 2 firmware was developed by Norbert Abel [Abe07]. This read-out controller firmware was already based on the free-streaming data acquisition paradigm of CBM. At first, the firmware was designed as one monolithic piece of logic that receives the data from the nXYTER chip and then transports it via Ethernet to a PC.

With the start of this thesis, the nXYTER firmware was redesigned with modularity in mind as described later in section 4.1. With this approach, further nXYTER read-out developments and initial GET4 read-out developments could both profit by sharing logic.

**Ethernet Transport** Prior to modularization of the firmware, Ethernet was used exclusively for data transport to the DAQ PC. The Ethernet logic is implemented using the Xilinx EDK platform and provides basic functionality. The data throughput is in the order of 40 MB/s which is too slow for a bigger setup, but absolutely sufficient for the small setups of the early development phase. Clock distribution and synchronization are not implemented. However, for smaller setups a workaround was implemented (see the next paragraph).

The advantage is that there is no special hardware required aside from the ROC. This was especially important in the early R’n’D phase, where many different setups in many

different laboratories were tested. It was not affordable to provide an expensive PCIe board, which is required for optical read-out, to all of these laboratories.

However, the Ethernet transport logic was developed as a preliminary solution and is missing some important features (high bandwidth, clock distribution and time synchronization) which are required for the final experiment.

**“Poor Man’s Synchronization”** General purpose IO (GPIO) pins can be used as a workaround to synchronize a few ROCs. This has become known in the collaboration as the “poor man’s synchronization”. It was first developed for the nXYTER read-out as a workaround until clock distribution and time synchronization via CBMNet became available. The time stamp of one “master ROC” is periodically communicated via GPIO pins to the other “slave ROCs”. Every time the slave receives such a synchronization message from the master, it inserts its own local time stamp and also the received master time stamp in the data stream. For smaller setups, this allows for reconstruction and correction of the clock drifts in the analysis software on the DAQ-PC.

### Configuration Controller Firmware

For executing the radiation mitigation technique *scrubbing* (see section 3.2.3), a configuration controller is required. The SysCore Boards are equipped with an auxiliary Flash-based FPGA for that purpose (see section D.1). This configuration controller also requires a firmware.

With assistance of the author of this thesis and also with support from Heiko Engel, a reimplementations of the configuration controller firmware was implemented by Andrei Oancea for his diploma thesis [Oan13]. He combined the functionality of two existing firmwares in one clean reimplementations and also provided the integration into the new generation of the CBM read-out chain (see section 5.2.1).

A firmware that implements “blind scrubbing” has been described by Heiko Engel in [Eng09]. A second firmware for the configuration controller exists that implements the power-up configuration of the main FPGA, but not scrubbing. This firmware was written by former student Stefan Müller-Klieser based on previous work of David Rohr. It is supported by the main CBM software “roclib” and therefore used in CBM community.

The scrubbing engine used for this thesis follows the concepts of Heiko Engels implementation.

### Software

The DAQ software used for this thesis was mostly developed by Sergey Linev and Jörn Adamczewski-Musch at GSI. They provided the *Data Acquisition Backbone Core* (DABC) [AEKL08] for data analysis, the *GSI Object Oriented Online Offline system* (Go4) [AMEL11] for online monitoring, and also the software that goes by the name of “roclib” which includes low level communication and drivers for the different data sources.

### 3.1.4. The Read-Out Controller Hardware Platform

The hardware used primarily to carry out this thesis is the *SysCore Board Version 2*, an implementation of the *SysCore Architecture*. Both, the specification of *SysCore Architecture* and its implementations, the *SysCore Boards*, are described in appendix D.

## 3.2. Radiation Tolerance for FPGAs

As the CBM-ToF read-out controller is going to be operated in a radiation environment, mitigation of harmful radiation effects is required. In this section, possible state-of-the-art radiation mitigation techniques for SRAM-based FPGAs are presented and reviewed for application in CBM. The particular techniques that have been implemented are evaluated more precisely. Finally, a comparison to the radiation mitigation approaches of other high energy physics experiments is presented.

### 3.2.1. Hardware Approach

Radiation mitigation already starts with the choice of components and the overall construction of the system. Those aspects are explained in the following.

#### Shielding

The first idea that usually comes to one's mind is to protect the electronics by installing additional material to shield it from the radiation. While this can work for "slow" particles in some situations, there are a couple of reasons why it is not applicable for electronics of a HEP detector:

- Shielding would not only shield electronics but also the detectors located behind the electronics. The results of those detectors would then be falsified.
- As the name "high energy physics" suggests, some radiation particles have high kinetic energy. The amount of material that is required to shield the electronics would be either too expensive or would require too much space. In any case it would add a significant amount of weight to the electronics which then would even require an additional support structure.
- The additional matter also favors the production of secondary particles. The detectors also respond to these secondaries, i.e. additional data is produced and detector dead time increases.
- Backscattered particles (primary and secondary) can also affect the detectors in front of the shield.
- Generally, additional material in the detector should be avoided. Interaction of particles with material pollutes the particle signatures.

Opting for compromise by putting in a too thin layer of shielding material that does not fully absorb the particles might even be counter productive. Slower particles have higher probability to interact with matter, and hence also electronics are more susceptible to decelerated particles (this can be revealed when plotting equation 2.1).

## Flash-Based FPGAs

While most FPGAs are SRAM-based, it is also possible to purchase Flash-based FPGAs. Flash memory has two major advantages over SRAM memory. First, Flash is non-volatile. The memory content is preserved even when power is cut off, while SRAM loses the configuration without powering. Second, Flash-based devices are less susceptible to SEUs than SRAM-based devices.

In the past, Flash-based FPGAs were not available in sizes that are comparable to those of SRAM-based FPGAs. Recently, however, commercial Flash-based FPGA devices became available as consumer products in sizes that can compete with SRAM-based FPGAs, e.g. the *MicroSemi Igloo 2* [Mic14].

However, the major problem with flash-based technology remains, such devices show a rather low threshold in terms of TID (see section 2.2.1 for TID). For example the *Radiation-Tolerant ProASIC3* from Actel only sustains a total dose of 40 krad before it shows degradation effects [Act10, page 12]. Even lower values have been reported. The ALICE experiment foresees (Flash-based) MicroSemi SmartFusion 2 FPGAs for the consolidation upgrade for Run2 [AAB<sup>+</sup>13]. “They found (...) that the FPGA can’t be re-programmed after a dose of 2.5-2.5 krad. This rather low TID threshold was unexpected. That strongly limits the range of places where these devices can be used.”<sup>3</sup>

A second problem occurs when data is not only read from but also written to Flash memory during irradiation. Then the chance for single event gate ruptures (SEGR) exist due to the charge pumps used internally in Flash devices.

Flash-based FPGAs might be a considerable option for detector read-out electronics. However, the opinions concerning the usage of Flash-based FPGAs is differ. There are certainly some advantages (good SEU tolerance, non-volatile) but also some serious arguments against Flash-based FPGAs (low TID, chance for SEGR).

For the final CBM experiment, a Flash-based solution is not foreseen. Nevertheless, as an intermediate solution for research and development activity, the *SysCore Boards* are equipped with an auxiliary Flash-based FPGA (see appendix D). This small Flash-based FPGA acts as the configuration controller for the big SRAM-based FPGA, providing initial power-up configuration and also the scrubbing functionality. Fortunately, with the new Xilinx Series 7 FPGAs, an on-chip scrubbing mechanism is introduced that allows CBM to plan a flash-free scrubbing solution for the future (see section 8.2.2).

---

<sup>3</sup>Reported by Walter F. J. Müller at an internal CBM-DAQ meeting on July 9<sup>th</sup>, 2014.



### **Military Grade FPGAs**

Besides their main line of FPGAs, Xilinx also produces military and space grade FPGAs, which are supposed to be radiation tolerant [Xil10a, Xil14c]. Those chips fall under US export restrictions and cannot be simply ordered in Europe. In addition, those chips are significantly more expensive than the main line of FPGAs. The cost factor alone is reason enough for military and space grade FPGAs not to be considered for CBM.

### **Antifuse Devices**

Antifuse devices come with a similar architecture as FPGAs, however, their static configuration memory is one-time programmable. After their initial programming, the static configuration memory cannot be changed anymore. This almost completely removes the flexibility advantage of an FPGA. Antifuse devices are not considered for this thesis.

### **FRAM Technology**

FRAM is an emerging technology that combines the advantages of Flash-based devices (non-volatile, not very susceptible to SEUs) with SRAM-based devices (fast). More details can be found in [Vog14, Kum12]. The technology looks very promising and might be considered in future projects. However, the technology is very new and at time of writing, no FRAM-based FPGAs are commercially available.

#### **3.2.2. Redundancy**

When an SRAM-based FPGA is operated in a radiation environment, errors in the configuration memory of the FPGA cannot be prevented completely. Even if errors in the configuration memory are repaired, they will be present in the system for a certain time because error correction cannot happen instantaneously. Therefore, the design needs to tolerate those errors for a flawless continuation of operation, at least until the error is repaired. This is usually achieved by adding redundancy to the design.

#### **Temporal Redundancy**

An approach to further mitigate radiation effects in FPGAs is called *Temporal Redundancy*. The idea is to sample the logic's output at different points in time. A short glitch (SET) is then only sampled once, and can be corrected by a majority voter.

Temporal Redundancy does not protect against SEUs but only against SETs. However, SEUs are the main problem in FPGAs. TMR (see below) is easier to implement on an FPGA where clocking resources are very limited and in addition, TMR protects against both, SEUs and SETs. Therefore, TMR is preferred over Temporal Redundancy for FPGA implementations. In fact no complex real life FPGA implementation of Temporal Redundancy is known to the author of this thesis, Temporal Redundancy is not considered for this work either.

## Triple Modular Redundancy

The canonical approach is known as *Triple Modular Redundancy* (TMR), sometimes also called *Triple-Module Redundancy* [Car06]. The basic idea is to triplicate units of logic and vote for majority on the outputs.

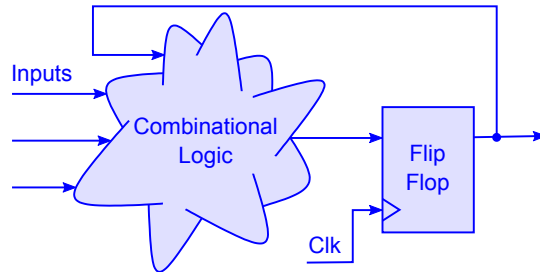
Figure 3.2 illustrates the principle. Even if one of the three parts fails, there is still a majority of two correct outputs against one erroneous output (see figure 3.2(b)).

The remaining problem is that a single SEU in the logic of the voter can still result in erroneous system behavior. This can be overcome by also triplicating the voter logic (see figure 3.2(c)). This leads to three outputs instead of one, the three outputs can be connected to the corresponding inputs of the next TMR'ed logic unit.

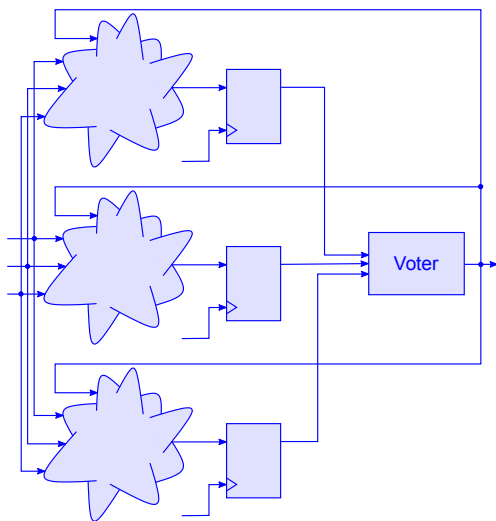
The units of logic on which triplication is applied ranges from a very fine-grain level where every flip-flop is tripled and voted on to a very coarse level where the tripled unit is the whole device. The advantage of a coarse implementation is the comparably small overhead due to the voters. The smaller the tripled logic units are the more voters are required, resulting more overhead. However, a problem with a coarse implementation is the higher probability of two simultaneous errors in the TMR'ed logic. With simultaneous errors in two of the three logic units, the TMR approach fails. A second problem of a coarse approach is that the more complex a unit becomes, the more complicated it is to repair it. Especially if the state of the logic unit does not only depend on its inputs but also on internal states. In some situations, a fine grain implementation might not be possible at all, e.g. when using a third party core that cannot be modified.

**Problems with TMR** TMR is neither perfect, nor does it come for free. Several issues have to be considered.

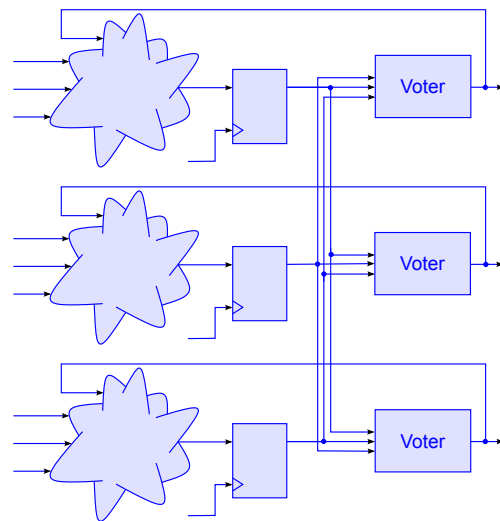
- The most severe drawback is the additional resource consumption. Since the logic is tripled and voters are added, more than three times the resources are required compared to the original design. The actual overhead depends on the implementation, reports from different publications vary from below four to over six [BSV11, page 191], [WRGC03b], [WRGC03a], [MMPW07], [Wir14, page 30].
- An airtight TMR implementation would utilize three times more input and output pins of the device than the original logic. Majority voting in the fabric of the FPGA would induce a single point of failure. Perfect TMR would therefore not only require more resources in terms of routing, look-up table and flip-flops, but also in terms of IO pins. This dramatically reduces the connectivity of the FPGA which is in many situations a critical parameter.
- With more routing and additional logic complexity, the maximum clock frequency of the design decreases [WRGC03a], [Wir14, page 30].
- The approach with tripled voters does not work on clock domain crossings that are common in complex designs (see also [Wir14, page 16]). When crossing a clock



(a) The original logic, consisting of an combinational part and an sequential part (flip-flop). Depicted here is also a feedback circuit, the output of the flip-flop is used as input for the combinational logic.



(b) The logic is triplicated and a voter on the outputs decides for the correct result by selecting the majority of the three inputs. Either all three outputs are the same (normal operation) or, in case of an error in one path, the correct result is still chosen by two against one. The feedback path corrects potentially erroneous values in the next clock cycle.



(c) The problem with an implementation as depicted in figure 3.2(b) is that a single error in the voter can still result in an erroneous behavior of the design. This can be avoided by also triplicating the voters. The design then has three outputs, that are connected to the corresponding input of the subsequent TMR'ed logic unit.

**Figure 3.2.:** Illustration of the implementation of TMR.

domain, it cannot be assured that all three paths are sampled in the same clock cycle. Due to metastability, one path might be sampled one clock cycle too late or too early compared to the other paths. Either a complicated synchronization method is required or the clock domains are crossed using a single path only (the variant depicted in figure 3.2(b)), inducing a single point of failure.

- TMR does not protect against multi-bit upsets (two simultaneous SEUs). In many applications, this is not a problem because the probability for a multi-bit upset is so low that it can be safely neglected. However, in some applications, e.g. when operating in high radiation environments, the probability for multi-bit upset becomes relevant.
- TMR is only effective for a certain run time. After that, the probability to fail is even better without TMR (see subsection *TMR Needs Repair* below).

**Automated TMR Tools** On his way to a working implementation of TMR the developer has to avoid several pitfalls. Not only the implementation in the hardware description language needs to be correct, but also some default optimization routines of the mapping and place-and-route tools need to be manually disabled. This requires a very deep and precise knowledge of the used design tools.

Xilinx offers a tool called *XTMR* that automatically performs some of the required steps towards a TMR'ed implementation [BCT08, page 5].

Using this tool helps in avoiding some pitfalls, however, it creates some additional problems. First of all, it works on netlists and is a brute force approach, triplicating every part of the logic in the netlist. It is therefore difficult to restrict TMR-implementation on selected logic components. The additional resource consumption cannot be easily reduced.

Second, *XTMR* is coupled to a specific version of the vendor tools suite. The developer has to commit himself to stick to this specific version, e.g. Xilinx ISE version 9.2i. An upgrade to a later version is not easily possible.

Third, the tool falls under US export restrictions. Xilinx states in the *XTMR* product brief manual [Xil]: "The Xilinx TMRTool is an ITAR-controlled product and as such certain documents and declarations must be collected from the customer when an order is placed." As a consequence, Xilinx *XTMR* tool cannot be easily used, especially not outside the United States.

Another automated tool called *BYU-LANL TMR Tool* exists [Con09] but is not further considered here.

For the sake of flexibility, a manual approach was chosen for this thesis, see section 4.2.2.

**Partial TMR** Under certain circumstances, the resource consumption can be mitigated by only applying TMR to selected logic components. *Partial TMR* is such an approach [PCG<sup>+</sup>05]. The designer can trade off resource consumption with better reliability. The

*Partial TMR* tool analyzes a specific bitfile for sensitive bits, and only triples the logic that corresponds to these sensitive bits.

This thesis uses a slightly different approach that has been named *Selective TMR*. A program like the *Partial TMR* tool that automatically identifies all sensitive bits in a design can only detect whether an SEU in a certain SRAM cell causes an error or not, but it cannot qualify the impact of the error to the surrounding setup.

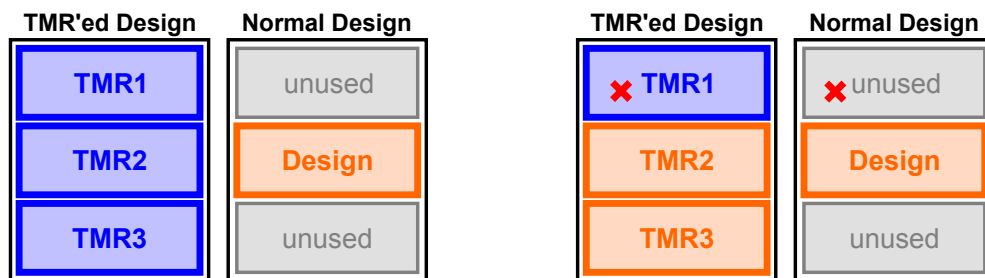
*Selective TMR* is a manual approach that exploits the knowledge of the firmware designer to qualify the severity of the error. It is presented in section 4.2.2.

### TMR Needs Repair

After running for a certain time, TMR without repair mechanisms even worse than implementing no redundancy at all.

Intuitively one expects TMR'ed logic to be more robust against SEUs than logic without redundancy. Indeed, this is true until the first SEU affects the TMR'ed logic.

The TMR'ed logic is bigger than an equivalent logic without redundancy. For the argument we assume an idealized TMR design, only taking into account the triplication of the logic and omitting additional overhead by voters and additional routing. Such an idealized TMR design consumes (only) three times the resources of the non-redundant design, in reality a factor of four to six is more realistic (see "Problems with TMR", page 46).



(a) Without SEUs in the logic only the normal design is vulnerable.

(b) After the first SEU, the vulnerable area of the TMR'ed logic unit is twice the area of the non-redundant logic.

**Figure 3.3.:** Illustration of the vulnerability of a TMR'ed logic unit and a non-redundant logic unit. Vulnerable parts are shown in orange, the parts of the TMR'ed logic unit that can tolerate an SEU are shown in blue. The non-redundant logic unit is always completely vulnerable, but it is also just  $1/3$  of the resources of the TMR'ed logic unit.

In reality, a fully TMR'ed design consists of many TMR'ed logic units. Depicted in figure 3.3 is one such unit. In the following we pick out one logic unit and consider only SEUs affecting this one logic unit.

Only one in three SEUs that affect the TMR'ed logic would also affect the non-redundant logic. If we only consider SEUs that affect the TMR'ed logic, the probabilities to affect the design is the given by:

$$p_{\text{SEU affects design}} = \begin{cases} 1 & \text{TMR'ed design} \\ \frac{1}{3} & \text{non-redundant design} \end{cases} \quad (3.1)$$

In case of the TMR'ed design, however, the first SEU that affects the logic does **not** corrupt operation of the logic. This is the basic idea behind TMR in the first place. On the other hand, in the non-redundant design, it immediately leads to an error. The probability for erroneous behavior after the first SEU is therefore given by:

$$p_{\text{error}}^{(1)} = \begin{cases} 0 & \text{TMR'ed design} \\ \frac{1}{3} & \text{non-redundant design} \end{cases} \quad (3.2)$$

The probability to have a functional design after the first SEU is:

$$p_{\text{functional}}^{(1)} = 1 - p_{\text{error}}^{(1)} = \begin{cases} 1 - 0 = 1 & \text{TMR'ed design} \\ 1 - \frac{1}{3} = \frac{2}{3} & \text{non-redundant design} \end{cases} \quad (3.3)$$

After the first SEU in the TMR'ed logic unit the operation is not disturbed, but one of the three branches is corrupted.<sup>4</sup> While the first SEU does not have an immediate effect on the operation of the TMR'ed logic, it does change the probability for subsequent SEUs to affect the logic. In case the error is not repaired, the next SEU in one of the other two branches of the TMR'ed logic unit leads to an error. Hence, the probability for subsequent SEUs in this logic unit causing an error is  $p_{\text{error}}^{(2)} = 2/3$ .

For the non-redundant logic the situation does not change, a subsequent SEU causes an error with a probability of  $p_{\text{error}}^{(2)} = 1/3$ .

$$p_{\text{error}}^{(2)} = \begin{cases} \frac{2}{3} & \text{TMR'ed design} \\ \frac{1}{3} & \text{non-redundant design} \end{cases} \quad (3.4)$$

The probability to have a functional design after the second SEU is then:

$$\begin{aligned} p_{\text{functional}}^{(2)} &= \left(1 - p_{\text{error}}^{(1)}\right) \cdot \left(1 - p_{\text{error}}^{(2)}\right) \\ &= \begin{cases} (1 - 0) \cdot \left(1 - \frac{2}{3}\right) = \frac{1}{3} & \text{TMR'ed design} \\ \left(1 - \frac{1}{3}\right) \cdot \left(1 - \frac{1}{3}\right) = \frac{4}{9} & \text{non-redundant design} \end{cases} \end{aligned} \quad (3.5)$$

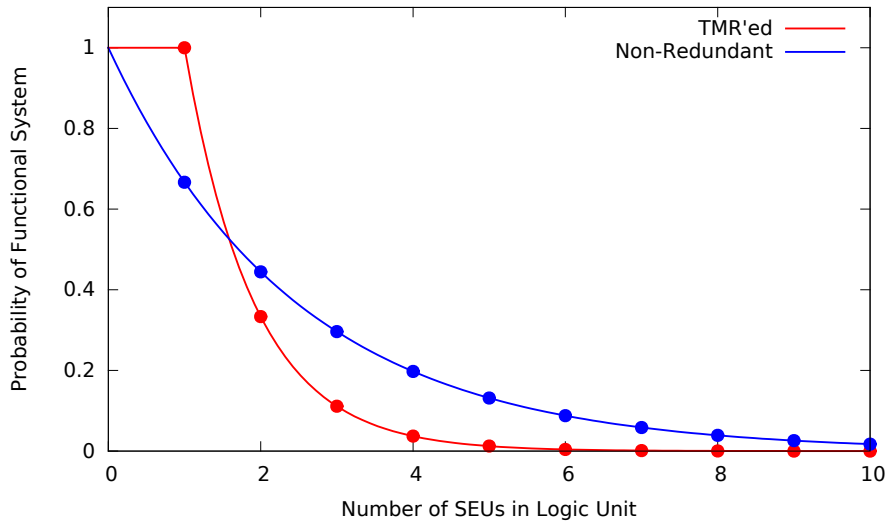
---

<sup>4</sup>Here SEUs in the dynamic configuration memory (flip-flops) are neglected and only SEUs in static configuration memory (look-up tables and routing) are considered. SEUs in static memory are much more likely as there are much more static configuration bits than dynamic configuration bits contributing to the design [Whi14, page 25]. An SEU in the dynamic part will indeed be repaired by TMR.

We see, that already after the second SEU, the probability to be functional is higher for the non-redundant logic unit ( $p = 4/9$ ) than for the TMR'ed logic unit ( $p = 1/3$ ).

The generalized equation for a number of  $n$  SEUs is:

$$p_{functional}^{(n)} = \prod_{i=1}^n (1 - p_{error}^{(i)}) = \begin{cases} (\frac{1}{3})^{n-1} & \text{TMR'ed design} \\ (\frac{2}{3})^n & \text{non-redundant design} \end{cases} \quad (3.6)$$



**Figure 3.4.:** Comparison of (idealized) TMR'ed logic with non-redundant logic. The dots refer to the probability of an error-free operation despite a discrete number of SEUs. A plot of the equation 3.6 with continualized  $n$  connects the dots. The red plot shows a plateau from 0 to 1 where it is protected against errors. Already for the second SEU in the TMR'ed logic, it has a higher probability to fail than the non-redundant logic. If SEUs are **not repaired** and **more than one** critical SEU is expected in one of the tripled logic units during runtime of the system, it would even be better to do nothing than to implement TMR.

A fully TMR'ed design usually consists of many tripled logic units. The number of SEUs that the whole design can sustain until the probability to still have an operative system becomes higher for the design without TMR, the so called "break even point", is higher than two SEUs. The actual value varies with the granularity of the TMR implementation.

However, the underlying principle described in this section is valid for each TMR'ed logic unit independently and therefore also for the whole design. It is important to repair SEUs before the number of SEUs in the design reaches the "break even point", otherwise TMR should be omitted.

The principle was presented before, e.g. in [Wir14].

## FSM Encoding

For the implementation of finite state machines (FSMs), an alternative to TMR can be chosen. The state vector of the FSMs can be Hamming-coded [Ham50]. If the hamming codes are chosen with hamming distance of three, this allows for online detection and correction of corrupted states in case of single bit flips.

In [KZ04] Hamming-coded FSMs have been compared to FSMs implementing TMR. Several sample FSM designs with varying complexity have been analyzed by implementing TMR and Hamming-coding. Hamming-coded FSMs perform slightly better than TMR'ed in terms of resource consumption, but TMR shows better results when it comes to clock frequency penalties. Also the results vary very much depending on the implementation.

Good results concerning SEU mitigation have been achieved with test designs implementing Hamming-coded FSMs [Geb12]. However, other tests that compare Hamming-coded FSMs to TMR'ed FSMs still show better results for TMR [Ber14].

Since the situation is ambiguous, an analysis with respect to SEU tolerance has been conducted that compares Hamming-coded FSMs with FSMs implementing TMR. It is presented in section 5.2.2.

### 3.2.3. Configuration Scrubbing

Since SEUs are not temporary errors, but remain in the configuration memory, they accumulate over time. As described in detail in section 3.2.2, this is a problem especially for redundancy-based mitigation techniques such as TMR. As shown in figure 3.4, TMR works when SEUs do not accumulate. Unfortunately, without countermeasures, they do accumulate over time and TMR will fail eventually.

Therefore, TMR requires a technique to remove SEUs from the configuration memory. This can be achieved by periodically refreshing the FPGA's configuration. A full reconfiguration, however, interrupts the operation of the FPGA until it is reprogrammed. On top of that, all internal states are lost and the device then first needs to be resynchronized into the running system.

Fortunately, Xilinx developed a technique called *Dynamic Partial Reconfiguration* (DPR) which they first described for their Virtex FPGAs in 2000 [Xil00]. DPR allows to reprogram the FPGA configuration memory **without interrupting the operation** of the device. Since the operation of the device continues, the internal states are of course preserved as well. This technique allows to dynamically change the logic in the FPGA during runtime, this enables for example to share fabric resources among tasks that do not run simultaneously. More information about DPR can be found in [Eto07], [Xil13], and in [Abe11].

The same technique can also be used to refresh the configuration memory in case of SEUs. The idea to use DPR for refreshing the configuration memory was first described in [CCS00] and by now it is state-of-the-art technology for SEU mitigation. The documentation for implementing scrubbing for Xilinx Virtex-4 FPGAs, the particular devices



used for this thesis, is described in [CT09]. Scrubbing runs in the background, without interrupting or disturbing the execution of the running firmware.

Several different approaches for the implementation of scrubbing exist:

- One approach for scrubbing is to read-back the configuration memory and check for SEUs. Only if an SEU is detected, the configuration memory is scrubbed. This brings the advantage of providing valuable statistics like the number of SEUs that are corrected. In low radiation environments, where the probability for suffering from multi-bit upsets is relatively low, this is a valid option. This approach has been followed for the ALICE RCU [RAF<sup>+</sup>11a].
- The faster approach is called “blind scrubbing”. The configuration controller continuously refreshes the configuration memory without checking for SEUs. As this is the fastest method available for the Virtex-4 FPGA and a ready-to-use implementation was available in the working group (by Heiko Engel [Eng09]), it was chosen for this thesis.
- With Xilinx Series 5 and 6 FPGAs, Xilinx introduced a feature based on the on-chip calculation of CRC checksums from the configuration memory [Cha10]. It allows for much faster SEU detection than read-back of the full configuration memory. This feature was not available on the hardware used for this thesis.
- Xilinx Series 7 FPGAs are shipped with an integrated scrubbing engine that allows to omit the external on-board configuration controller [Xil14a, chapter 8]. This solves the problem with the low TID of Flash-based devices that have previously been considered as the only viable choice to implement such configuration controllers. Future SEU mitigation strategies for CBM will most likely exploit this feature, see section 8.2.2 for more details.

It should be noted that scrubbing only prevents the accumulation of SEUs in the static part of the FPGA configuration memory, but not the occurrence of SEUs per se. Scrubbing has to write the correct configuration into the chip and this takes some time. An SEU remains in the logic until it is repaired and until then it can disturb the operation of the design. This is especially problematic, when an SEU leads to corruption of the dynamic part of the configuration memory, for example the state vector of a finite state machine (FSM). This can happen directly when a bit is flipped that is part of the dynamic memory or indirectly when an erroneous output of corrupted logic from the static part is latched in a memory cell of the dynamic part. Scrubbing does not refresh memory cells that belong to the dynamic part of the chip’s configuration. This would overwrite any internal states in the logic, sequential logic would therefore not be possible. An additional technique, e.g. TMR, is required to sustain the SEU for this short time period and to mitigate SEU effects in the dynamic part of the FPGA’s memory.

TMR and scrubbing work best together: TMR covers aspects that scrubbing leaves out and TMR is not efficient without scrubbing.

### 3.2.4. Fault Injection Tests

The technique used for scrubbing (see section 3.2.3) can also be used to emulate SEU events in the laboratory [LCF<sup>+</sup>01, AMZE03, Eng09, and many more]. When scrubbing is performed, the configuration memory of the FPGA is refreshed with correct values and thereby potential configuration memory errors, e.g. caused by radiation, are corrected. This procedure also works the other way around. Instead of writing correct values into the configuration memory it is also possible to refresh with a bit pattern where one or several bits are flipped. This results in a chip configuration with one or several flipped bits, a similar situation as expected when radiation is causing bits to flip. Radiation induced SEUs can be emulated and the behavior of the running firmware in such situations can be evaluated without the need to actually expose the device to radiation. However, not all of the possible SEUs can be emulated that way, e.g. SEUs in the dynamic configuration memory (flip-flops) are not considered. Nevertheless, fault injection is a powerful tool to analyze a design and to prepare for real in-beam tests.

Fault injection tests have been used in this thesis to evaluate the efficiency of Hamming-coded finite state machines and triple modular redundancy to (section 5.2.2) and for preparation tests preceding the in-beam tests (section 5.3.2).

### 3.2.5. Fault Tolerance in High Energy Physics

According to sales promotion of FPGA vendors, the main use case for FPGAs in radiation environments are military applications and space flight missions. For military applications, not much information is available. For space missions there are many publications, mostly describing the same techniques, triple modular redundancy in conjunction with scrubbing. However, in space applications, the situation is slightly different, they require very high reliability. A failing device might be mission critical and nobody can press a reset button once the device is in space.

Yet more and more FPGAs are operated in radiation environments of high energy physics (HEP) experiments. At the “Workshop on FPGAs in High Energy Physics” that was held at CERN in March 2014, the conclusion at the end one talk was: “The new hardware and error mitigation techniques allow HEP front-end electronics to follow the general trend of replacing ASICs with FPGAs in areas with moderate radiation levels” [HAB14]. Nowadays, most HEP experiments operate or plan to use FPGAs in radiation environments:

- After using an SRAM-based Xilinx Virtex II device for Run1, ALICE plans to update their Readout Control Units to a Flash-based SmartFusion 2 FPGA. For Run2, they expect  $\sim 45$  SEUs/h. [JA14]
- ATLAS plans to mount SRAM-based Xilinx Kintex-7 devices in an environment where they expect up to 10 SEUs per day. They mitigate by duplicating the devices on the read-out board. [HAB14]

- LHCb plans to operate an SRAM-based Altera FPGA in an environment where they expect per single FPGA on average one error every  $1.5 \cdot 10^6$  seconds [FUWL14], that translates to only 19 SEUs per year.
- CMS plans to replace their existing Antifuse chip with a Flash-based MicroSemi FPGA and use a tool for automatic TMR implementation. [Gra14]

There are also existing solutions that are based on FPGAs. ALICE has seen several SEUs in their read-out electronics and decided to use scrubbing for mitigation [RAF<sup>+</sup>11b], even at a much lower SEU rate (about one SEU every two hours) than what is expected for CBM (one SEU every two seconds, see section 7.2).

There is reason enough to take SEU effects serious. As presented in [BCMS12], the radiation induced failures for the LHC operation in 2011 summarize to about 70 beam dumps, causing a downtime about 400 hours. This of course includes not only failures in FPGAs but also in all other kinds of devices like optical fibers, or voltage regulators.

### 3.3. Verification Through In-Beam Tests

A common procedure to evaluate the susceptibility of electronics with respect to radiation is to operate the electronics in a high-radiation environment and measure radiation effects under controlled conditions [Eng09, Rø09, Mar04, Xil14b, and many more]. Radiation environments are usually produced by either placing the device close to a radioactive source or by mounting it directly in the beam line of an accelerated particle beam.

Not all classes of radiation cause the same effects in electronics. SEU effects, for example, require a certain energy, and thus they are dominated by the level of high energy hadrons<sup>5</sup> (see section 2.2.2). The radiation environment has to be chosen according to the effect that one wants to evaluate. For example: using a beam of accelerated electrons to measure SEU effects would not work, because SEUs are hardly caused by electrons. The in-beam tests carried out for this thesis evaluate SEU effects. The beam particles were protons with a kinetic energy of about  $2 \text{ GeV}$ .

For classification of measured SEU effects one needs to know the number of SEUs that occurred in the device under test. The number of SEUs in a device can be calculated from the number of beam particles and the SEU cross section.

$$\text{no. of SEUs} = \text{no. of particles} \times \text{device cross section} \quad (3.7)$$

$$= \text{no. of particles} \times \underbrace{\text{bit cross section} \times \text{no. of bits in device}} \quad (3.8)$$

Mature techniques exist for measuring beam particle rates, many details can be found in literature, for example in [Kno00]. However, all of these techniques are hard to apply, an expert on-site is required for this task.

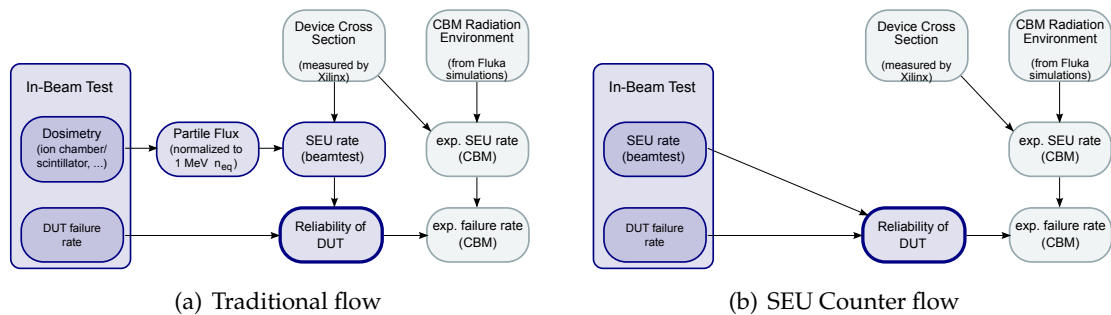
<sup>5</sup>For this reason, FLUKA simulations for CBM give also the flux of high energy hadrons [Sen11]. This allows for an estimation of SEU rates in the electronics.

For the SEU cross section, the situation is different. The SEU cross section depends on the particle type and on the energy of the particle. If the SEU cross section for the specific particle type at the given energy is not known, it has to be measured in a separate test. For such a test, it is very important to have precise knowledge about the internal structure of the device which is not always available. Fortunately, Xilinx publishes device specific bit cross sections for neutrons [Xil14b]. They give their results with the statement: “Neutron cross sections are determined from LANSCE beam testing according to JESD89A/89-3A”. According to JESD89A, LANSCE neutrons are high energy neutrons ( $E > 10 \text{ MeV}$ ) [JED06, pages 33 & 81].

The SEU cross section for protons and neutrons at higher energies do not differ much since only nucleus-nucleus collisions deposit enough energy in the silicon to induce SEUs. Figure 2.7 (page 28) shows that protons at high energies show only small variations in their effective SEU cross section. For that reason the values published by Xilinx can be used directly to estimate the number of SEUs from the beam particle rate.

However, to be able to compare measurements under different conditions (different kind of particles, different spectrum of particle energies), normalization is required. A common method is to normalize the particle rate to  $1 \text{ MeV} n_{eq}$ .  $1 \text{ MeV} n_{eq}$  is the equivalent rate of neutrons with a kinetic energy of  $1 \text{ MeV}$  that would cause the same radiation damage as the actual particle beam that was used for the test.

When testing electronics for radiation effects using a particle beam, it is standard procedure to measure the particle flux and then calculate the SEU rate from the device cross section. Based on the SEU rate, the efficiency of radiation mitigation techniques can then be evaluated (see figure 3.5(a)).



**Figure 3.5.:** Experiment flow to estimate failure rate of CBM detector (traditional and SEU Counter approach). The complicated and error-prone task of particle rate measurement is not needed when SEUs are directly counted.

### 3.3.1. SEU Counting

If only SEU effects are evaluated, the actual particle flux and the SEU cross section might not be required for classification of SEU effects. In many devices, SEUs can directly be counted by readback of the configuration memory. The number of SEUs sustained by

the device until it fails can be directly measured instead of getting it from complicated measurements and/or lengthy calculations. This has been done several times, see for example: [HSPG<sup>+</sup>08], [HDFC<sup>+</sup>11],[RBK<sup>+</sup>12], [Geb12].

Figure 3.5 shows a comparison between a traditional flow and the SEU Counter flow. Instead of first measuring the particle rate and then calculating the SEU rate from that, which is a complicated and error-prone task, the SEU rate can be measured. With direct SEU counting it is still possible to evaluate the efficiency of radiation mitigation techniques.

Nevertheless, estimations based on device cross section and radiation environment are still required for determination of the expected failure rate of the device in the experiment, but this is a different task (see section 7.2).



## 4. Approach

Before going into implementation details in chapter 5, this chapter describes the approach for the implementation of a radiation mitigated CBM-ToF read-out controller and furthermore the basic setup for in-beam testing of the mitigation efficiency is presented. As the previous chapter, this one is also organized in the three sections that represent the same three topics.

First, section 4.1 presents the GET4 read-out controller that follows the modular design approach of the CBM read-out controller family. The general advantages of this modular design approach are discussed. Then, in section 4.2, the choice of radiation mitigation techniques and the necessary adaptations for the techniques to fit for the CBM use case are discussed. Section 4.3 finally describes the basic ideas to verify the efficiency of the implemented techniques at in-beam tests.

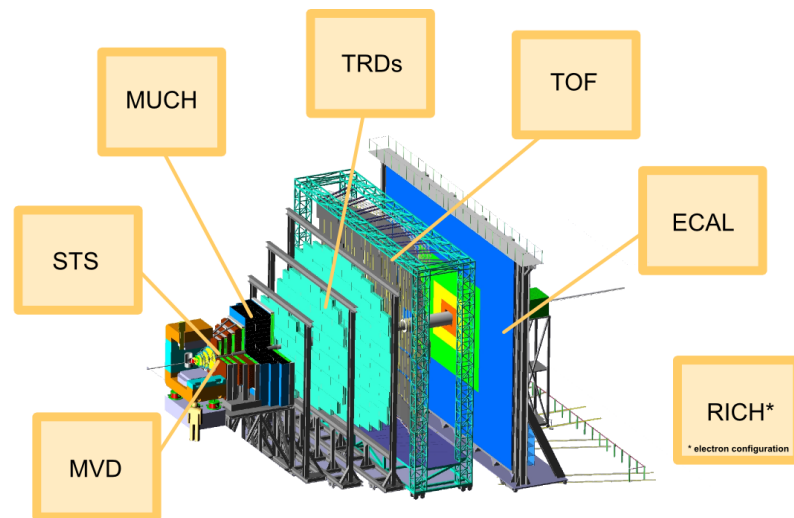
### 4.1. Modular Firmware Concept for the CBM Read-Out Controller

After digitization of the detector signals in the front-end ASICs the data is sent to the CBM read-out controller. The read-out controller receives the data, reformats it to the CBM data format, and then pushes it to the computing node for further processing. A working FPGA firmware for interfacing nXYTER ASICs already existed, the part of the logic that handles the communication towards the computing node could be reused for the GET4 ASIC. The modular approach allows for sharing logic between read-out controller firmwares for different front-end ASICs. This idea is described in more detail in this section, it was also presented at TWEPP conference in 2010 [MAGK10].

#### 4.1.1. Motivation for Modularization

For a project at the scale of the CBM read-out controller firmware it is almost always a good idea to implement the design in clean structure with several encapsulated entities and clear interfaces. Especially if there is more than one designer working on the project. In addition, another two strong motivations to divide the firmware of the read-out controller in different “modules” are described in the following, one regarding the front-end interfacing logic and one regarding the transport logic towards the DAQ.

**Multiple Different Front-End Interfaces** As depicted in figure 4.1, CBM consists of multiple different subdetectors, each with individually designed front-end electronics.



**Figure 4.1.:** The CBM experiment at FAIR/GSI consists of several detector systems as illustrated here in its muon configuration. Picture as published in [MAGK10].

The different front-end chips require different interfacing logic in the read-out controller (ROC). With the *SysCore Boards* being very flexible regarding these kind of interfaces, the goal is, of course, to support as many of the front-end electronics interfaces as possible. The first chip that was read out by a *SysCore Board* was the nXYTER in 2007 [Abe07]. This read-out firmware, developed by Norbert Abel, quickly became the standard solution for nXYTER read-out and is still widely used in CBM community. With the work of this thesis, the read-out of the first prototype of the GET4 followed in 2009. As of 2012, further iterations of the GET4 with slightly different interfaces are supported. Later on, the *SysCore Board Version 3* was used to interface the SPADIC [Gar11] and the STS-XYTER [Sch14].

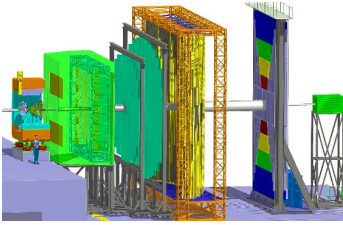
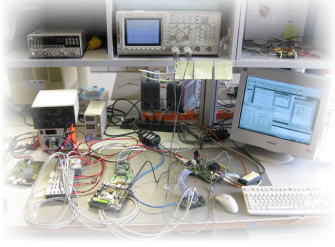
Plans foresee to read out some of the front-end electronics with ASICs instead of FPGA-based ROCs, e.g. in case of the STS. The reason is the extreme radiation level in those regions. Nevertheless, even if the read-out of some of the detector front-ends is planned to be accomplished by an ASIC in the final detector setup, the prototyping of those read-out ASICs is still done on an FPGA.

In the end, several different firmwares for interfacing the different front-end electronics are required. However, the different firmwares have very common requirements regarding the interface towards the DAQ. The transport logic towards the DAQ can be the same, and with a modular design and a clear interface, the same logic can easily be reused.

**Different Requirements for Data Transport** The *SysCore Boards* are designed to support the research and development of the CBM read-out chain and also to providing a working data read-out solution for detector development. The challenge here is to develop



towards an implementation that meets the full list of future requirements, while at the same time permanently providing a working read-out solution for the smaller laboratory setups.

Final experiment	Laboratory
	
many ROCs involved	↔ few ROCs involved
synchronization of all the ROCs is very complicated	↔ rather simple synchronization method that does not scale to more than a few ROCs is also acceptable
huge data bandwidth required	↔ significantly lower data bandwidth is sufficient
data is processed on a high performance cluster	↔ data is processed on a commodity PC
significant amount of radiation is affecting the electronics	↔ no radiation issue
different detector ground levels require electrically isolated readout	↔ only one ground level
needs to be available at the end, when the experiment is constructed	↔ needs to be available as early as possible, during R'n'D

**Table 4.1.:** The different requirements on the transport logic during the research and development phase and for the final detector setup. Detector picture from [FHK<sup>+</sup>11]

The final experiment setup and the smaller laboratory setups are two different scenarios and both have different requirements. The requirements for the final experiment setup are tighter than the requirements for laboratory setups. In the final experiment many read-out boards will be operated in parallel, which requires a sophisticated time distribution and synchronization scheme. Also a much higher data throughput has to be handled than in small laboratory setups. The radiation level in the final experiment will be high enough to cause serious trouble for the electronics. Implementing all these features is a complex task that requires quite an amount of development time.

However, a working data read-out solution for laboratory setups needs to be available as early as possible to allow detector developers to analyze their detector data. The final experiment setup, on the other hand, is required much later, at the time when the experiment is constructed. The two scenarios constitute different requirements on the

transport logic. The final experiment setup requires a complex logic that supports all specified features, and in the early days a subset of all features is sufficient.

Table 4.1 lists the different requirements on the transport logic in early R'n'D and for the final detector setup.

The bottom line is that a lot of different firmwares are needed by different detector groups, but the firmwares only differ in certain aspects and much of the logic can be shared among those firmwares. A modular design of the firmware is the obvious approach.

In 2009, the first prototypes of the GET4 chips were available. At that time, the decision was made to split the existing monolithic nXYTER read-out firmware in two modules, one to interface the front-end electronics and one to handle data transport. The existing logic for data transport via Ethernet could be used to quickly develop a firmware for GET4 read-out. The read-out logic for the GET4 was urgently needed to test the first GET4 prototype and push its integration in the CBM read-out chain.

At the same time, the development of the CBMNet transport protocol reached a state mature enough to start the process of integration in the CBM read-out chain. While the GET4 was put into operation, the Optics module for integration of the CBMNet transport protocol could be developed in parallel. When the Optics module reached a mature state, it could be used as transport logic for both, GET4 and nXYTER.

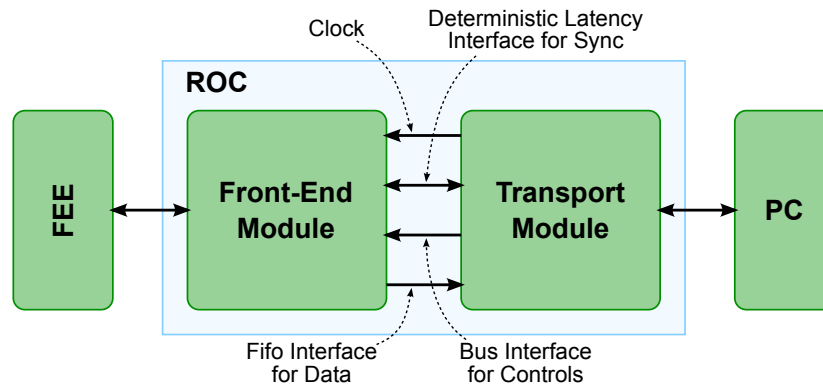
#### 4.1.2. Interface Requirements

Once the decision for a modular approach is made, a specification of the interface between the modules is needed. Figure 4.2 illustrates the separation of the firmware into transport and front-end modules and the interface between them. Naturally, such an interface should meet the requirements of the CBM experiment. The CBM requirements are:

- high data throughput
- reliable control
- precise time synchronization
- relaxed latency requirements

The interface clock is provided from the transport module to the front-end module. This allows to use the recovered clock from the optical connection, which is the global CBM clock that is distributed via CBMNet to all front-end electronics, as system clock. Together with CBMNet's deterministic latency interface this enables a precise time synchronization over multiple ROCs.

If the Ethernet logic is used as transport module, the local board clock is used and synchronization over multiple ROCs is not possible. This is not problematic as Ethernet



**Figure 4.2.:** Basic illustration of the CBM read-out controller firmware design based on two modules. The interface between the modules reflects the requirements of the CBM experiment.

is not intended to be used in complex setups with more than a few ROCs, and they can be synchronized as described in section 3.1.3 (“Poor Man’s Sync”, page 42).

A standard FIFO interface is chosen for the data path as it is easy to use, well known, and it does provide a high data throughput. The latency induced by a FIFO buffer is not relevant for CBM because the experiment is designed to operate without low-latency triggers.

The controls interface is designed as a bus structure, derived from the OPB Bus [IBM01] that was already in use in the early, Ethernet-based read-out logic for the nXYTER chip. Every read and every write of a register in the design is acknowledged, providing a robust controls channel.

The functionality of the different modules is briefly discussed in section 5.1.1, details of the modules and the interface are given in appendix B.

#### 4.1.3. Towards Radiation Tolerance

Radiation tolerance is not required for all use cases. Only the optical transport module and for the GET4 read-out module are planned to finally be operated on SRAM technology and in a radiation environment. Here, radiation mitigation techniques need to be implemented. Everything that is not part of the GET4 read-out via optical fibers is either for intermediate solutions, prototyping for ASIC solutions, or will be operated outside the radiation environment in the final setup, and thus does not need to address radiation effects.

## 4.2. Choice of Radiation Mitigation Techniques

Due to the high radiation level expected for CBM-ToF, it is clear that FPGAs can only be utilized when SEU countermeasures are implemented. Scrubbing is the obvious choice for a radiation mitigation technique. As laid out in section 3.2.3, scrubbing does not cover all aspects of radiation induced errors and requires additional design practices to protect dynamic content in the FPGA's memory.

This section summarizes the arguments for choosing or rejecting certain implementations for the various radiation mitigation techniques.

### 4.2.1. Blind Scrubbing

The choice of radiation mitigation techniques is restricted by the particular hardware for which it has to be implemented. Some techniques require support from the hardware. In case of the read-out controller for the GET4 ASIC, the hardware platform was the *SysCore Version 2* board, this board comes with a Xilinx Virtex-4 FPGA.

For the Virtex-4, two scrubbing strategies can be implemented (see section 3.2.3): blind scrubbing and read-back based scrubbing.

Compared to blind scrubbing, read-back based scrubbing brings the advantage of providing valuable information, for example the number of SEUs and even the position of the SEUs on the chip. However, it also comes with the drawback that one full scrubbing cycle lasts twice the time of a full scrubbing cycle in case of blind scrubbing.

For this thesis, blind scrubbing has been chosen since high SEU rates are expected. It is important to repair SEUs as quickly as possible to minimize the possibility for multi-bit upsets. For lower SEU rates, read-back based scrubbing might be the better choice.

It should also be noted that for blind scrubbing there was already a ready-to-use implementation available. A read-back based scrubbing implementation would have had to be developed first.

### 4.2.2. Selective TMR

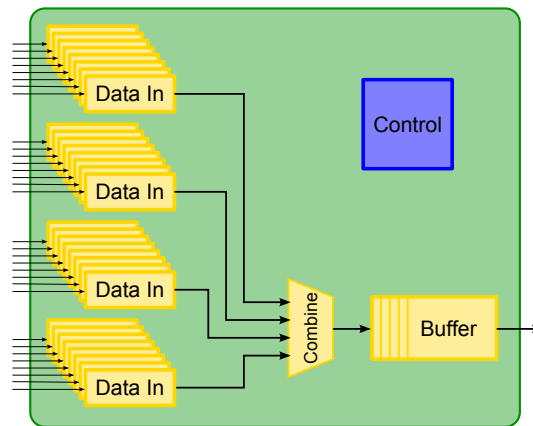
The most severe problem of the TMR approach is the massive overhead in resource consumption (see section 3.2.2). Up to six times the resources of the original design are required for a TMR'ed implementation of the same logic. An increase in resource consumption translates directly to an increase in cost, either larger FPGAs or a higher number of FPGAs are required to connect the same amount of input channels. Luckily, a characteristic feature in the requirements for CBM can be exploited such that a lot of the overhead from TMR can be avoided.

As the efficiency of the detectors is not perfect anyways, some additional data corruption can be accepted as long as its contribution remains reasonably small. The data related read-out logic does not need to operate 100% correctly at all times and errors can be allowed in rare cases [Mül13, slide 23]. Of course, erroneous data needs to be detected, but it is not required to correct the data. For data, a simple CRC checksum can

be used to detect possible corruption. This is easy to implement and requires only few resources. Only errors in the configuration logic and in the time-representing logic need special protection and must be implemented as TMR'ed logic. The logic components can be grouped into two categories:

- data path related → only error detection required → use CRC
- control related → error correction required → use TMR

Figure 4.3 shows the resource consumption of the data path logic and control logic in the GET4 read-out design from a conceptual point of view. A lot of input channels are connected, therefore, the data-related part of the read-out logic represents the major part of the overall logic. Control logic constitutes only a small part of the design.

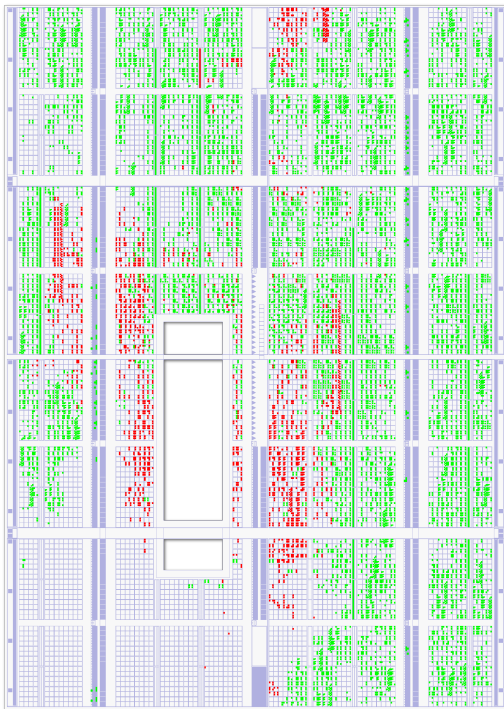


**Figure 4.3.:** Analysis of the GET4 read-out design. The combination of multiple input channels consumes the major part of the resources, the contribution of control logic is much less.

In the particular implementation for the GET4 read-out firmware control logic contributes about 10% of the full design, based on the MAP report from the synthesis of the not redundant design. Figure 4.4 shows an illustration of the given design with highlighted controls and data resources.

It is evident that avoiding costly TMR implementations in the data path saves the major part of the overhead of a full TMR implementation. Automated tools do not allow for fine-grain TMR replication and hence they are not used. For the critical logic parts, TMR is implemented manually using VHDL (see section 5.2.2). Results are presented in section 6.2.2.

**Hamming Coded FSMs** For hardening of finite state machines (FSMs), it was considered to use hamming coded state vectors instead of using a TMR approach. As already mentioned in paragraph *FSM Encoding* (page 52), contradictory statements about the performance of hamming coded FSMs compared to TMR'ed FSMs can be found. To clarify



**Figure 4.4:** Illustration of the resource consumption for the GET4 read-out firmware without redundancy, created with the Xilinx design tool *PlanAhead*. Resources that belong to control and timing logic are highlighted in red, resources that are part of data path logic are highlighted in green. It can be seen, that critical resources (red) contribute much less to the overall resource consumption than uncritical (green) resources. According to the numbers in the map report, the percentage of control and timing related resources is only 10% of the total number of utilized resources in this design. Only the red resources need TMR protection.

the situation for the design in question, a test has been performed that evaluates the performance of the two options. The test is described in section 5.2.2 and suggests that, also in the case of FSM design, a TMR approach should be chosen.

#### 4.2.3. Fault Tolerant Protocol

A decision for a soft radiation mitigation strategy that allows occasional failures can only succeed when higher layers of the system design can tolerate an error in the lower levels. This means that the impact of the radiation mitigation strategy is not restricted to components in the radiation zone, but has also some implications for design decisions of parts that are located outside the radiation zone.

An example is the protocol design for data transport. Given that the protocol design supports retransmission and that the retransmission is not carefully designed, a data word that gets corrupted in a buffer in the radiation zone can result in an infinite cycle of retransmissions. By retransmitting this one corrupted word over and over again, the data channel is completely blocked.

When communicating with unreliable components, such details have to be considered at higher design levels. A clever use of timeouts and watchdog functionality is required alongside an intelligent treatment of responses that are not compliant to the defined protocol.

Problems related to these higher levels only emerge when the higher levels are put into operation. This means, that the whole system needs to be put together to see whether all

possible design flaws are eliminated.

#### 4.2.4. Fault Injection Tests

As already described in section 3.2.4, fault injection tests (also known as *SEU injection tests*) can be used, albeit with some limitations, to emulate SEUs in a firmware design. Fault injection tests use the same hardware and interfaces as scrubbing does. A *XILINX Platform Cable USB* programmer and the *Xilinx iMPACT* software can be used to write the configuration bitstream from the PC to the FPGA via a JTAG interface. Another possibility is to use the faster *SelectMAP* interface. The *SysCore Version 2* connects the *SelectMAP* interface to the Actel configuration controller that can read a configuration bitstream from an on-board flash memory.

The SelectMAP interface has the advantage of being much faster than JTAG. However, the JTAG solution is more versatile, the bitstream can be modified by software running on the PC. Flexibility is important for SEU injection tests, therefore, the solution that was chosen for the tests described later (in sections 5.2.2 and 5.3.2) utilizes the JTAG interface.

**Weak Points** The SEU injection method is a great helper for the evaluation of SEU effects as it emulates a real high-radiation situation very closely. However, some differences remain.

The weak points for the particular SEU injection implementation for this thesis are:

- Only the static configuration memory is targeted for SEU injection. SEUs are not injected to the “dynamic” memory like flip-flops and BRAMs although these memory cells are also SRAM-based and will suffer from SEUs in the real experiment.
- The injection of an SEU via JTAG is rather slow, one injection run takes about seven seconds. Recording a statistically significant set of samples requires a long time. Therefore, the tests had to run over night. This could have been improved by replacing JTAG based injection with a hardware based injection using the on-board configuration controller. However, it would also add new logic to the system. New logic is always a likely candidate for new errors, and hence complicates debugging of the overall system.
- For some tests with tight time schedule it was required to compensate the slow recording of statistics. Then (e.g. in the test explained in section 5.3.2), not only one SEU but 20 SEUs have been injected per iteration. This allows for multi-bit upsets, and some radiation mitigation techniques might not perform optimal on multi bit upsets (see paragraph *TMR Needs Repair*, page 49).

### 4.3. In-Beam Tests

The evaluation of electronics behavior in radiation environments is important, especially when a new design approach is followed. In the presented case one aims for higher

particle rates than usual, for CBM-ToF it is expected to expose SRAM-based electronics to a fast hadron flux of up to  $10^4 s^{-1}cm^{-2}$ . However, CBM-ToF does not require full reliability but allows the system to fail in rare cases as long as it recovers quickly and not too much data is lost.

This design verification cannot be carried out by fault injection tests alone as they do not cover the full spectrum of SEU effects. Also with commercial off-the shelf electronics, the internals are not known in full detail, evaluation under conditions that are as realistic as possible are required to avoid unpleasant surprises later on.

#### 4.3.1. Test Setups

For this thesis two in-beam tests were carried out in the JESSICA cave of the *Cooler Synchrotron* (COSY) at Forschungszentrum Jülich/Germany. A first test in August 2012, and a second test in July 2013.

In both tests the COSY beam consisted of high energy hadrons: protons with an energy of about 2 GeV. Such a beam is well suited for SEU tests since high energy hadrons are also the dominant contributor to the expected SEU rate at CBM (see section 3.3). The particle rate can reach up to  $10^7 s^{-1}cm^{-2}$  which is sufficient for such tests (see section 5.3.2).

The basic concept of both tests is the same. Deterministic data is delivered by a data generator that is running outside the radiation zone. The firmware under test is running on a board that is mounted directly in the beam line. It receives the deterministic data, executes the read-out algorithms and then delivers the processed data to the DAQ PC. On the PC the data can then easily be analyzed for corruption because it is deterministic. The main motivation is to evaluate the efficiency of the scrubbing method under the condition that not all logic parts are TMR-protected. So quality of data taken when scrubbing is enabled is compared to that of data that is taken when scrubbing is disabled.

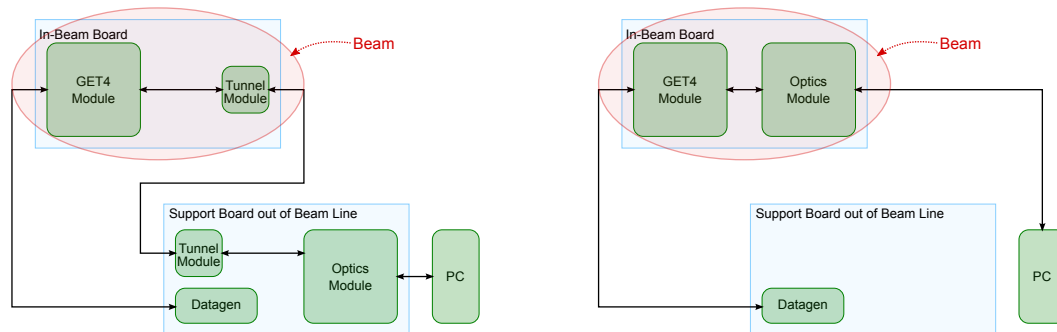
**Test Setup 2012** In 2012, only the GET4 module fully implemented the required radiation mitigation methods. The Optics module was not ready for use at time of the 2012 in-beam test, mostly due to difficulties with the preparation of an external core that is part of this module (see section 5.2.4).

For that reason, the optics module out was moved out of the beam line. Therefore, the firmware was split into two firmwares. The GET4 module was operated in-beam, while the Optics module was operated off-beam on a second board. Two dedicated modules were implemented, one on each board, that tunnel the interface signals of the modules between the two boards.

Figure 4.5(a) illustrates the 2012 test setup.

**Test Setup 2013** The good results of 2012 motivated a second test in 2013 to verify that the results are also valid when the full firmware, including the Optics module, is exposed to beam particles. Figure 4.5(b) illustrates the 2013 test setup.





- (a) The test setup 2012. Because of trouble provided by an external core in the Optics module, only the GET4 module was mounted in-beam. The Optics module ran on a second board along the deterministic data generator outside the radiation zone. Both boards are connected via dedicated tunnel interface modules.
- (b) The test setup 2013. The full firmware logic is operated in-beam, directly delivering data to the DAQ PC. The second board is only required as deterministic data generator.

**Figure 4.5.:** Illustration of the two setups for the in-beam tests in 2012 and 2013 respectively. In 2012, only half of the design was ready for operation in a radiation environment, but the part that was ready performed very well (see section 6.3). In 2013 the promising results of 2012 could be repeated with the complete firmware exposed to radiation

### 4.3.2. Beam Diagnostics

As already mentioned before, COSY at the Forschungszentrum Jülich is very well suited for single event upsets experiments. However, there are some obstacles when it comes to beam diagnostics.

There are two online status monitors on the COSY website, one showing the current view of the oscilloscope used for measuring the number of particles in the storage ring, and another one showing a history of about one day of the number of particles per bunch. The information of those online monitors can be used for rough approximations (see section 5.3.3), but it is not meant to be used for beam diagnostics. For example, they do not take into account the loss of rate efficiency during extraction of the beam. Therefore, particle rate measurement at the experiment area in JESSICA cave is not provided by local staff. It is the responsibility of the experimenters to add proper beam diagnostics system to the setup.

### Standard Beam Diagnostics

Well-established methods for beam diagnostics exist, yet the task is not trivial (e.g. see [Kno00]). A qualified expert on-site is required for getting a correct characterization of the beam. This is especially problematic when the person carrying out the experiment is not a physicist but a hardware designer.

For particle rates up to  $10^5\text{s}^{-1}$  it is common to use scintillation counters or fiber hodoscopes that are based on the same effect, but also give beam position information. For higher particle rates ionization chambers are a good choice.

Sven Löchner carried out a parallel experiment during the in-beam tests 2012 [LGWH13] and 2013 [LFG<sup>+</sup>14]. Part of his setup was an ionization chamber to measure the particle rate. With the help of his setup an estimate of the particle rate was obtained as reference.

### SEU Counter Approach

The SEU Counter approach comes with some advantages. First of all, and this point should not be underrated, it is a rather easy exercise for a hardware designer compared to particle rate measurement which is not trivial and requires an expert on-site.

Second, results based on SEU rates are (almost) directly comparable to results of fault injection tests (see section 3.2.4) that can be carried out in the lab. If new effects that did not occur during fault injection tests show up, they can be much easier identified as such.

Besides, the particle flux of the experiment specific particle beam does not represent the real radiation environment in which the device is later to be operated. Many different kinds of particles at a wide energy spectrum contribute to the radiation environment of a high energy physics experiment. A particle beam, on the other hand, consists of only one type of particle at a well defined and narrow energy range. To compare the effects measured at accelerated beam tests with the effects of the expected radiation environment at the CBM experiment, both have to be normalized to a common denominator. Often a calculated equivalent flux of 1 MeV neutrons ( $1\text{ MeVn}_{eq}$ ) is used for normalization.  $1\text{ MeVn}_{eq}$  is the equivalent rate of neutrons with a kinetic energy of 1 MeV that would cause the same radiation damage as the actual particle beam that was used for the test. Since for a given device, 1 MeV neutron equivalent flux is proportional to the SEU rate (see also figure 3.5 on page 56), it is also possible to normalize to SEU rate instead.

SEU counting comes with the advantage of being precise, and easy to implement. Uncertainty comes mainly from normalization to 1 MeV neutrons and the *device cross section*. For that reason, the SEU counting method was used during the in-beam tests.

**Problems with Blind Scrubbing** At first sight, it seems problematic to apply the SEU Counter approach on a setup in which the device under test is protected by blind scrubbing. Blind scrubbing corrects SEUs in the configuration memory without reporting their occurrence (see section 3.2.3), hence, SEUs cannot be counted in the device under test. As the in-beam tests that were carried out for this thesis use blind scrubbing, SEU counting could not be implemented straightforward. However, a setup involving two boards allows for SEU counting nevertheless (see section 5.3.3).

**Reflipping Bits** The situation that the same bit is flipped a second time, returning to its original value, can be a problem if the total amount of accumulated SEUs are recorded. However, if the number of configuration bits in the device is very high and the number

of SEUs accumulation during a common test run is comparably modest, such double bit-flips can be neglected. The Virtex-4 FX20 FPGA that was used in the tests for this thesis comprises a configuration memory of  $\sim 7 \cdot 10^6$  bits while the SEU rate was only in the order of a few bit-flips per second. For that reason, double bit-flips can be neglected.



## 5. Implementation

This chapter presents the details of the implementation of the radiation tolerant GET4 read-out controller and details concerning the setup to verify the efficiency of the radiation mitigation techniques. As before, it is organized in three sections.

Section 5.1 presents the various modules for the modular CBM read-out controller that have been implemented. It also addresses briefly the challenges that arise with the organization of the growing number of possible combinations of modules.

Section 5.2 presents the considerations for selecting and implementing radiation mitigation techniques. This includes an evaluation of two competing techniques to add redundancy to finite state machines - *Hamming coded FSM* and *TMR'ed FSMs*.

Verification through in-beam tests is addressed in section 5.3. It includes the preparation work for the tests as well as the details of the final setup, including beam diagnostics through SEU counting by using a second board.

### 5.1. Main FPGA Design

As described in section 4.1, the design of the firmware follows a modular approach. The modules "Ethernet" and "nXYTER" were implemented by Norbert Abel (see section 3.1.3), for the sake of completeness they are also mentioned here. Only a brief overview of the functionality of the modules presented since the functionality of the design itself is not in focus of this thesis. However, to get a better idea about the complexity of the firmware, the reader can find the documentation of the GET4 read-out firmware in appendix B.

#### 5.1.1. The Modules

By now, five modules have been developed, two front-end modules and three transport modules. The two front-end modules provide an interface to different front-end electronics, the nXYTER chip and the GET4 chip while the three transport modules reflect the different requirements of the different operation scenarios.

Ethernet and USB modules are intended for usage in smaller setups and do not provide features for synchronization of several boards. The optics module is more complex since it also features clock distribution and time synchronization mechanisms, which are required for the final CBM experiment setup.

All front-end modules can be combined with all transport modules, figure 5.1 illustrates the principle.

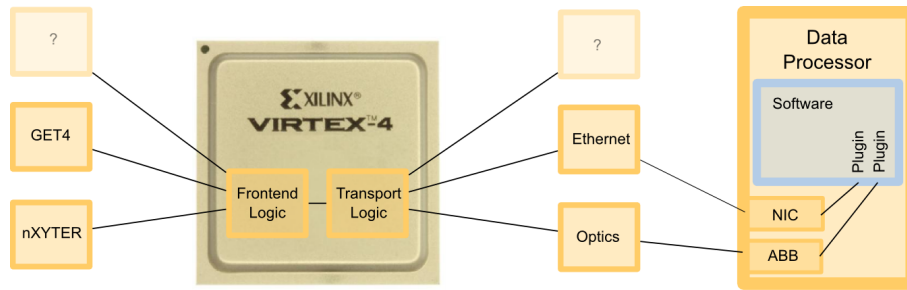


Figure 5.1.: The modular design of the CBM read-out controller firmware.

The combination of the five modules results in six firmwares: nXYTER-Ethernet / nXYTER-USB / nXYTER-Optics / GET4-Ethernet / GET4-USB / GET4-Optics. Except nXYTER-USB, all combinations have been implemented, but not every combination is available on each hardware platform.

### Transport Module: Ethernet

The Ethernet module was implemented by Norbert Abel in 2007/2008 as part of nXYTER read-out logic. The motivation then was to provide an intermediate solution to transport data from the ROC to the DAQ-PC. This was urgently needed since a CBMNet based solution was not available at the time and an immediate solution was required by scientists working on the detector design to be able to read out data from their detector prototypes. The read-out chain was set up by only two people, Norbert Abel providing the ROC firmware and Sergey Linev providing the corresponding software. The firmware is a PowerPC based “System on Chip”, implemented with the Xilinx EDK suite. It can be used with a commodity PC with a standard network interface card and does not require additional exotic hardware. Soon, many laboratory setups started to use this logic because of its early availability.

However it lacks some features which are important for the final setup. The rather poor throughput of only about 40 MB/s is not enough to interface a sufficient number of chips, the copper cables prevent an operation on different electrical ground levels, it does not provide an intrinsic clock distribution and synchronization method, and radiation tolerance is not addressed at all.

Nevertheless, nowadays it is still widely used in smaller setups because of the relaxed hardware requirements.

### Transport Module: Optics

When ZITI/Mannheim released their first version of CBMNet in 2009, a difficult integration process started. A significant amount of glue logic was written in the process.

The CBMNet based setup is more complex than the Ethernet based setup, it requires at least one extra FPGA board, the ABB, that receives the data from the optical links and

then provides it to the DAQ-PC via PCIe.

Five developers worked in parallel on the setup for this read-out chain, again Sergey Linev for the DAQ Software, Guillermo Markus provided the kernel driver for the ABB, Wenxue Gao the firmware for the ABB, Frank Lemke was in charge of CBMNet, and the author of this thesis provided the aforementioned integration into the ROC firmware. The Optics module was first used in a beamtest in 2010.

With the CBMNet based Optics module a transport logic was developed that fulfills the CBM requirements [LSB10]. A throughput of 220 MB/s per optical link is considered to be fast enough for CBM, it provides clock distribution and time synchronization mechanism that scales to the range of the CBM experiment and grounding issues are avoided by optical connections between the different elements of the read-out chain.

Later, in 2011, a second version of CBMNet was released and had to be integrated in the the ROC firmware as well. Since the CBMNet version 2 logic is going to be operated in an environment with harsh radiation, it was developed with the option to add radiation mitigation techniques.

The logic of this module is part of the firmware that was tested for radiation tolerance in the tests described later. Several issues have been detected, reported, and been resolved during the work for the radiation tolerance tests (see for example section 5.2.4).

### **Transport Module: USB**

The Ethernet core as a the lightweight solution for small laboratory setups has proven to be very useful for CBM community. However, the implementation of the Ethernet module depends on an outdated version of the Xilinx development tools (ISE 8.2i and EDK 8.2i) and cannot be build with an up-to-date tool chain. When the main developers, Norbert Abel and Sergey Linev, both left CBM collaboration the maintenance of the Ethernet module was reduced to minimal support of legacy systems. The SysCore Version 3 (details in section D.2.3) does not equip an Ethernet connector. Therefore, no lightweight solution for small laboratory setups was available for the SysCore Version 3. However, the SysCore Version 3 comes with an USB subsystem that allows for a comparable lightweight-solution.

Walter F.J. Müller already implemented an USB-based transport logic for a different project, but he was using the same Cypress USB chip that is also available on the SysCore Version 3 board. This logic was extended by the author of this thesis to suit the CBM requirements. No details are given here because this development is mainly engineering work and does not significantly contribute to the results of this thesis. It performs very stable and, at time of writing, no open issues are reported, although it is frequently used in CBM community.

### **Front-End Module: nXYTER Read-Out**

The self-triggered and data driven nXYTER chip became available in 2007 and was the first front-end ASIC for the free-running CBM read-out chain [STS<sup>+</sup>07]. The chip has 128

input channels and a mixed analog and digital interface towards the ROC. The nXYTER read-out logic, developed by Norbert Abel, was the first read-out logic that was developed for CBM. At time of writing, the nXYTER chip is still widely used in the CBM community, the same is also true for its read-out controller.

The read-out logic induced the 48 bit data format that has been adopted also for the GET4 read-out. A the data format is presented in figure B.4.

Since the nXYTER chip is a prototype and not intended to be used in the final experiment, the read-out logic for this chip will never be operated in a radiation environment. Thus, no measures to mitigate radiation effects have been implemented for the nXYTER read-out logic.

### Front-End Module: GET4 Read-Out

The *GSI Event-Driven TDC with 4 channels* (GET4) [DF09] was developed especially for the requirements of the CBM-ToF detector and is the second front-end chip supported by the Modular ROC. As already indicated by its name, the chip has four input channels. The focus lays on the resolution of its *time to digital conversion* (TDC). CBM-ToF has very tight requirements regarding time measurements, therefore the chip provides a very good time resolution of less than 15 ps, a double hit resolution of better than 5 ns [Har13]. A first prototype of the chip was released in 2009 and in 2010 it was used the first time in a beam test.

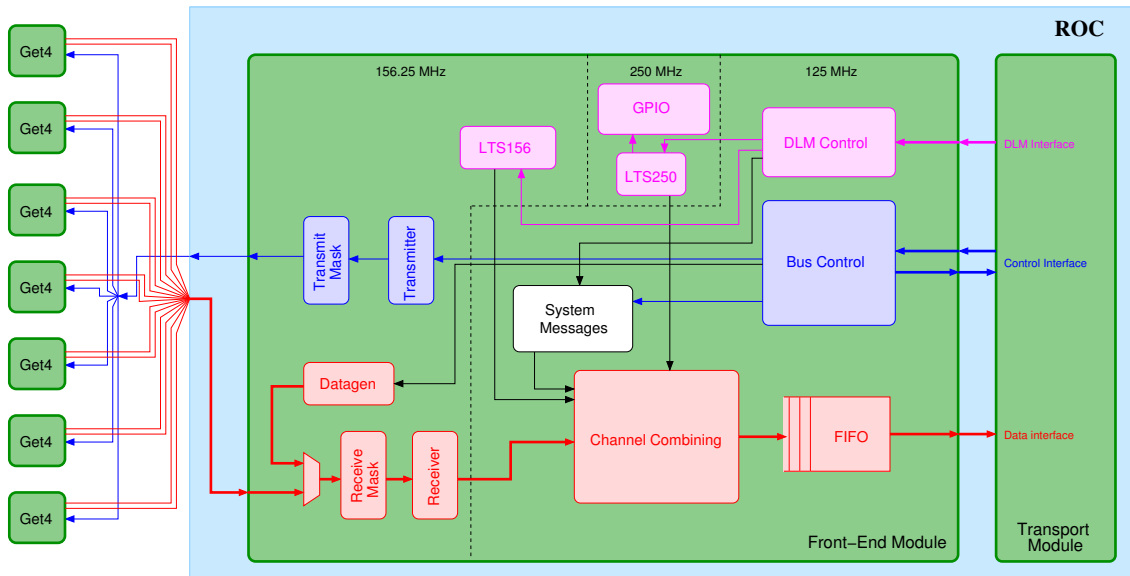
The front-end module for interfacing the GET4 chips provides all necessary functionality to connect to multiple GET4 ASICs.

- Control messages can be forwarded to the GET4 ASICs. A subset of ASICs can be addressed by configuring a mask register.
- Data send from the GET4 ASICs is sampled and combined to a single data stream. Stream multiplexing is implemented as round-robin algorithm.
- Data is reformatted to CBM-compliant 48 bit messages.
- Data is buffered on backpressure.
- Epoch Messages received from the GET4 ASICs are tested for correct synchronization with the local clock. In case of mismatch, an error flag in the message is set.
- The 21 bit wide time field in the Epoch Messages from the GET4 ASICs is extended to 32 bit.
- A System Message is inserted into the data stream on certain events, like a pulse on a specific input pin, after a FIFO reset command, etc.
- In case of buffers filling up, hit data is discarded first so that high priority messages (like System Messages or Epoch Messages) can still be send. All lost data events



are recorded in the data stream. Lost hit events are distinguished from more critical lost Epoch events.

- Data input lines from each GET4 ASICs can be masked to reduce noise in case of a broken chip.



**Figure 5.2.:** A schematic overview of the front-end module for the GET4 read-out. Taken from the documentation for the read-out controller for the GET4 chips, see appendix B

Figure 5.2 illustrates the basic functional blocks of the module. More details about the functionality of the read-out logic for the GET4 chip can be found in the documentation for the firmware that is provided in appendix B.

Several prototypes of the GET4 chip, with different behavior, have been developed, and all are supported by the read-out controller. In 2012, with the GET4 v1.0, a new read-out interface was introduced. The new interface is based on 32 bit wide words on both, data and control channel, hence called “32 bit”-interface. It allows for advanced configuration features of the chip and provides much better diagnostics functionality [Har13]. Fortunately, the old interface, based on 8 bit control words and 24 bit data words, is still integrated for backwards compatibility. The work on this thesis is based on the old interface, called “24 bit”-interface.

Nevertheless, to fully support the read-out of the GET4 ASIC, the existing firmware for the 24 bit interface was ported to support the 32 bit-interface in the scope of a diploma thesis [Leh13]. This was done by Johannes Lehrbach, a diploma student that was supervised by the author of this thesis.

All logic parts that were part of the efficiency tests, namely GET4 Read-Out Module and the Optics Module, were implemented with respect to the requirement of radiation

mitigation. The combination of those two blocks, the “GET4-Optics” firmware, was developed by the author of the thesis.<sup>1</sup> The “GET4-Optics” firmware constitutes the foundation for carrying out the efficiency tests of the radiation mitigation techniques that are presented later.

### 5.1.2. Handling the Multiplicity of Firmwares

Due to the different demands within the various detector groups, support for many combinations of the two front-end modules, the three transport modules, and four different hardware platforms was required. In total, 24 different firmwares are released.

Since a change in one module usually affects more than one firmware, it is not very practical to use the standard tool flow suggested by the vendor. This would require to maintain a separate project for each firmware that is managed within a graphical user interface (GUI) by manual interaction, heavily based on using a computer mouse. While this approach might be very convenient to initially set up a single project, it is not really compatible with our modular firmware approach.

Fortunately, the vendor tool suite provides command line access to the underlying build routines. This enables the implementation of a scripted build flow. To be able to maintain the multiplicity of different firmware, a Makefile-based build flow was developed that can be executed from the command line. After a change in the source, a simple make in the relevant project folder re-builds the project’s firmware. Such a command line-based build flow enables automated builds and is much more convenient than GUI-based building when dealing with multiple firmwares that share code.

The sources are maintained in a version control system, *subversion* in the early years and *git* from 2012 on. Every night, a fresh copy of the latest sources is checked out from the repository and all firmwares are build from scratch. This allows to maintain repository’s integrity and it also helps in the search for bugs. Especially if a bug is not directly detected but keeps lingering in the code base for a while, it still can be traced to commits of a single day. The “Nightly Builds” system was installed and maintained by the author of this thesis until 2014. Due to the general advantages of such an approach, the CBM collaboration decided in 2014 to install a central build server for all CBM-related firmwares at GSI<sup>2</sup>. GET4-related “Nightly Builds” are now managed there as well.

The Makefile-based build flow automatically writes the repository revision number and the time when the build was initiated into user-accessible firmware registers. This allows to easily trace a certain firmware after its release to a specific build.

---

<sup>1</sup>A part of the Optics Module is the “CBMNet” core which was developed by Frank Lemke and Sven Schatral at ZITI/Mannheim.

<sup>2</sup>The CBM build server can be found here: <https://cbm-firmware.gsi.de>

## 5.2. Radiation Mitigation for the FPGA

Radiation mitigation for SRAM-based FPGAs requires, besides the implementation of the according mitigation techniques (in our case *Scrubbing* and *Selective TMR*), also some general design considerations, on multiple design levels. Implementation details and crucial design considerations are now discussed in this section.

### 5.2.1. Scrubbing

A mature and ready-to use implementation for the SysCore Version 2 boards could be adapted from a previous project by Heiko Engel [Eng09]. Both, the firmware for the flash-based configuration controller (“Actel”) and the procedure for generating *partial bitfiles*<sup>3</sup> could be reused.

Adaptions required mainly spawned from the use case of a bigger FPGA in addition to the already supported SysCore Version 2.0. Additionally, a number of bugs have been fixed that happened to remain unnoticed before but became relevant for the in-beam test setups.

**Design Restrictions** Although the setup is in principal ready to run *Scrubbing*, certain design restrictions must be obeyed. There are some design restrictions for the actual firmware that has to be protected by scrubbing. If the firmware does not respect the following restrictions, it cannot be protected by scrubbing.

- The Virtex-4 FPGA comes with a feature to use look-up tables (LUTs) as shift registers or distributed memory. This feature must not be used. LUTs are refreshed by scrubbing and this would corrupt any dynamic context stored there when scrubbing is enabled.
- The design tool *xst* has to be called with the options
  - “-shift\_extract no” and
  - “-ram\_style block”,

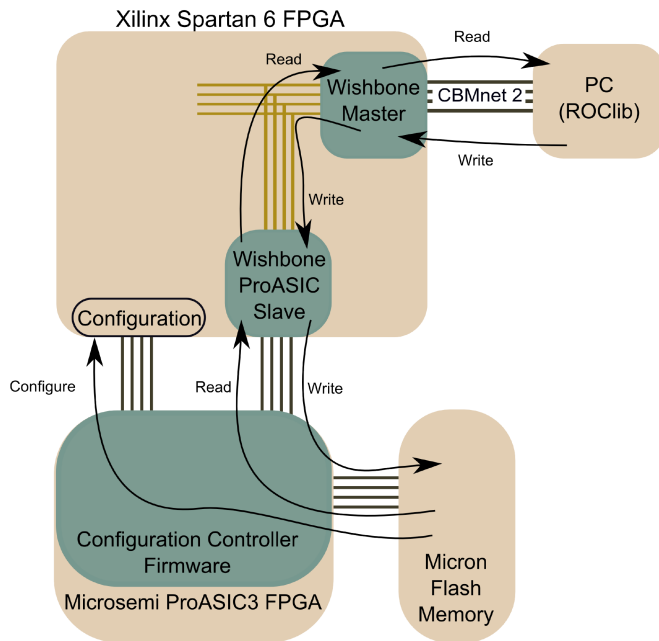
to not unintentionally infer the forbidden usage of LUTs for dynamic context information.

- *Bitgen* has to be called with the parameters
  - “-g DriveDone:Yes” and
  - “-g Persist:Yes”

to keep the SelectMAP interface active after the initial configuration of the FPGA.

---

<sup>3</sup>A *partial bitfile* is the part of the firmware that represents the static configuration bits (look-up tables and switch matrices). The dynamic configuration bits (flip-flops and memory) are stripped off.



**Figure 5.3:** A schematic overview of the configuration controller. Power-up configuration and scrubbing is accomplished by reading the configuration data from a Flash memory. The Flash memory is transparently mapped into the address space of the CBM slow control protocol. [FS13, page 89]

**Configuration Controller Firmware Reimplementation** (side note) It became clear, that the existing scrubbing controller implementation is sufficient for in-beam tests but does not meet all demands of the final CBM use case. In particular, it was missing compatibility with the controls protocol for integration in the CBM read-out chain environment.

To overcome the limitations of the existing solution for future projects, a reimplementation for the SysCore Version 3 board was carried out by Andrei Oancea in the scope of his diploma thesis. The new implementation provides

- power-up configuration,
- a scrubbing engine that implements *blind scrubbing*,
- a watchdog functionality, to be able to recover even in the rare events when scrubbing fails,
- a transparent integration of the address space in the CBM configuration scheme.

Figure 5.3 illustrates the basic functionality, all details about the implementation can be found in Andrei Oancea’s diploma thesis [Oan13].

The implementation by Andrei Oancea was carried out in close cooperation with Heiko Engel and the author of this thesis. However, since this implementation is restricted to the SysCore Version 3 board, it could not be used for the measurements in this thesis as they are based on the SysCore Version 2 board.

### 5.2.2. Redundancy

For some components, like the local timestamp counter, it was clear from the beginning to use TMR as radiation mitigation technique. For the implementation of finite state machines (FSMs), two competing techniques are available (see also paragraph *FSM Encoding*, page 52) and the best technique had to be identified first.

The results of this selection process are presented first, followed by implementation details of the selected TMR.

#### Hamming Coded versus TMR'ed State Machines

To compare the efficiency of hamming-coded FSMs with TMR'ed FSMs, an SEU-injection test was performed. The logic used in the 2012 in-beam test was chosen as basis. Two radiation mitigated firmwares were built, one that implements hamming-coded FSMs and another one that implements TMR'ed FSMs. In addition a firmware without redundancy was built for comparison.

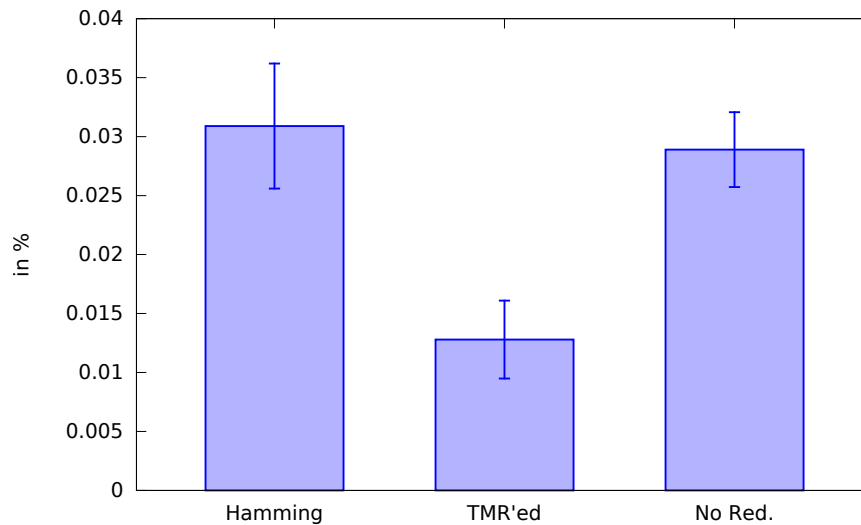
The test has a very low probability to show an error for two reasons:

- To keep the test simple, the functionality of only one FSM in the logic was tested. The FSM that was selected for the test is considered to be highly critical, however, the probability for an injected SEU to actually affect this particular FSM is still rather small.
- To avoid multi-bit upset effects, only one SEU was injected per iteration. A random bit in the partial bitfile was flipped and this modified bitstream was then loaded on the FPGA.

Due to that, the test had to run for a significant time, but this not a major obstacle, as the test could be performed over weekends and during other times when the hardware was not required for something else.

Figure 5.4 shows the results of the test. A positive SEU mitigation effect of hamming coded FSMs cannot be seen, whereas TMR'ed FSMs show a significant mitigation effect.

It might be criticized that the Hamming coded FSMs protect the state vector which is part of the “dynamic” memory and this is not affected by SEU injection tests at all. Therefore, one can argue that it is no surprise that no effect in favor of Hamming coded FSMs can be observed here. However, it has been shown that the reason for an SEU induced failure is more likely an error in the static memory than in the dynamic memory, simply because much more static memory contributes to the functionality of the design [Whi14, page 25]. Even though the FSM's state vector is not directly affected by SEU injection tests, the FSM's functionality is, and this is the relevant criterion to evaluate. Actually, this is probably even the reason why Hamming coded FSMs do not improve SEU resilience despite adding redundancy to the logic. Hamming codes only protect the dynamic state vector at cost of blowing up the static part. TMR increases the overall resource consumption but it mitigates both, static and dynamic part of the logic.



**Figure 5.4.:** Comparing the SEU susceptibility of Hamming Coded FSMs, TMR'ed FSMs, and FSMs without redundancy. Plotted is the percentage of all injected SEUs that caused erroneous design behavior. The error bars refer to the square root of the number of errors. TMR'ed FSMs show a clear improvement while Hamming Coded FSMs do not.

Hamming coded FSMs might perform well on Flash-based FPGAs and Antifuse FPGAs where static logic is not susceptible to radiation but they do not perform well on SRAM-based FPGAs.

For that reason, all critical FSMs that had to be protected in the scope of this thesis were implemented with TMR.

### Implementation of TMR

All critical parts of the design are implemented with TMR. The parts that are considered to be critical are:

- the control logic (e.g. the state machines implementing the GET/PUT protocol)
- the logic that handles time and synchronization (e.g. the local timestamp counters)

The basic idea of TMR as well as important design considerations have already been presented in section 3.2.2.

Figure 5.5 shows a VHDL example that implements a TMR'ed flip-flop, with synchronous set, synchronous reset, clock enable and also the voters for the outputs. The combinational logic that is the source for the inputs of such a flip-flop needs to be tripled as well.

It is best practice to connect the three output signals of one TMR entity to the corresponding three input signals of the next entity. In reality, this is not always possible, for

```

47  -- to output
48  Q_out_tmr0 <= Q_voted_tmr0;
49  Q_out_tmr1 <= Q_voted_tmr1;
50  Q_out_tmr2 <= Q_voted_tmr2;
51
52  -- voters
53  Q_voted_tmr0 <= ((Q_tmr0 and Q_tmr1) or
54                  (Q_tmr0 and Q_tmr2) or
55                  (Q_tmr1 and Q_tmr2));
56  Q_voted_tmr1 <= ((Q_tmr0 and Q_tmr1) or
57                  (Q_tmr0 and Q_tmr2) or
58                  (Q_tmr1 and Q_tmr2));
59  Q_voted_tmr2 <= ((Q_tmr0 and Q_tmr1) or
60                  (Q_tmr0 and Q_tmr2) or
61                  (Q_tmr1 and Q_tmr2));
62
63  -- register 0
64  process_tmr0: process(clk)
65  begin
66  if rising_edge(clk) then
67  if set_tmr0='1' then
68  Q_tmr0 <= '1';
69  elsif reset_tmr0='1' then
70  Q_tmr0 <= '0';
71  elsif ce_tmr0='1' then
72  Q_tmr0 <= D_tmr0;
73  else
74  Q_tmr0 <= Q_voted_tmr0;
75  end if;
76  end if;
77  end process;
78
79  -- register 1
80  process_tmr1: process(clk)
81  begin
82  if rising_edge(clk) then
83  if set_tmr1='1' then
84  Q_tmr1 <= '1';
85  elsif reset_tmr1='1' then
86  Q_tmr1 <= '0';
87  elsif ce_tmr1='1' then
88  Q_tmr1 <= D_tmr1;
89  else
90  Q_tmr1 <= Q_voted_tmr1;
91  end if;
92  end if;
93  end process;
94
95  -- register 2
96  process_tmr2: process(clk)
97  begin
98  if rising_edge(clk) then
99  if set_tmr2='1' then
100 Q_tmr2 <= '1';
101 elsif reset_tmr2='1' then
102 Q_tmr2 <= '0';
103 elsif ce_tmr2='1' then
104 Q_tmr2 <= D_tmr2;
105 else
106 Q_tmr2 <= Q_voted_tmr2;
107 end if;
108 end if;
109 end process;
110

```

**Figure 5.5.:** VHDL implementation of a TMR'ed flip-flop with synchronous set, synchronous reset, and clock enable (ce) inputs and data outputs (Q\_out). The data validity on the outputs is hardened by majority voters. Not shown here is the triplication of the combinational logic that sources the tripled inputs (D) of the TMR'ed flip-flop.

example when a communication between two different clock domains has to be implemented. A situation that occurs frequently with *Selective TMR* is that the next entity is not hardened by TMR and only accepts one input signal. In such situations, only one voter was used. The voter as single point of failure is tolerated.

**Design Restriction** As previously with *scrubbing*, an additional design restriction has to be taken into consideration in order to allow for a TMR'ed firmware implementation:

- The design tool *xst* has to be called with the options

– “-equivalent\_register\_removal no”

to not inadvertently remove the TMR feature in the optimization process.

**Detection of Data Corruption with CRC** Since the logic that handles data readout and data transport (the green resources in figure 4.4) is not protected by TMR, data corruption is possible and cannot be neglected. To detect such corruption, a CRC checksum is calculated at the very first input stage, in parallel to the deserialization of the signal received from the GET4. CRC calculation can be implemented very efficiently in hardware, figure 5.6 gives the complete source for calculating the 8 bit wide CRC checksum that is used to detect data corruption. The checksum implementation is based on the generator polynomial  $x^8 + x^2 + x + 1$  (ITU-T conform CRC-8 [Int13]).

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity crc8 is
5   port ( clk : in  std_logic;
6         rst : in  std_logic;
7         data : in  std_logic;
8         crc  : out std_logic_vector(7 downto 0)
9       );
10 end crc8;
11
12 architecture behavioral of crc8 is
13
14   signal crc_int : std_logic_vector (7 downto 0);
15   signal xor0, xor1, xor2, xor8 : std_logic;
16
17   begin
18
19     xor8 <= crc_int(7) xor data;
20     xor2 <= xor8 xor crc_int(1);
21     xor1 <= xor8 xor crc_int(0);
22     xor0 <= xor8;
23
24     crc  <= crc_int;
25
26     process (clk)
27     begin
28       if (clk'event and clk='1') then
29         if (rst='1') then
30           crc_int <= x"00";
31         else
32           crc_int <= crc_int(6 downto 2) &
33                   xor2 & xor1 & xor0;
34         end if;
35       end if;
36     end process;
37
38 end behavioral;
```

**Figure 5.6:** VHDL implementation of the CRC calculation that is used to detect whether data was corrupted by SEU effects. It implements the ITU-T conform CRC-8, with the generator polynomial  $x^8 + x^2 + x + 1$ . The 38 lines of code shown here represent a full VHDL entity that is synthesizable. The CRC calculation uses 8 flip-flops and 2 LUTs per input channel. Although one CRC entity has to be instantiated for every input channel, it is not much overhead to the overall design.

### 5.2.3. Fault Tolerance in Higher Design Levels

As also the higher levels in system design needs to sustain unexpected behavior of the components in the radiation environment. On all levels, it is important to not rely on correct behavior but to have contingency plans for cases out-of-specification.

- A watchdog functionality is implemented in the configuration controller for the main FPGA. For the SysCore Version 3 configuration controller, this was developed in [Oan13]. Such a hardware-based approach is intended for the final setup where many components are utilized. For the in-beam tests, watchdog functionality was implemented in software. The design of the test procedure for the in-beam tests implements a mechanism to recover from permanent device failure based on periodic sanity checks in software (see figure 5.9).
- The internal configuration bus is based on IBM's *On-Chip Peripheral Bus* (OPB) [IBM01]. Many features of the original specification are not implemented to keep the design as simple as possible. Without radiation effects, timeout feature of the OPB could have been omitted as well. However, radiation induced errors can lead to incomplete bus transactions. The timeout feature ensures, that such situations do not result in a stuck system. Instead, the error is reported after the timeout and the system remains responsive. For that reason, the timeout feature of the OPB is absolutely required and cannot omitted.
- The internal FSMs are designed in a way that they will not get stuck even on unexpected behavior of the logic. In addition it is assured, that in the case one of the three FSMs of a TMR-set gets disturbed, all FSMs will eventually resynchronize at



some point. Depending on the underlying functionality of the FSMs, the measures to assure continuation of operation and resynchronization are different. For example, for the FSM implementing the controls bus master a timeout was implemented to prevent to get stuck on a missing acknowledge response of a slave. Other FSMs are designed to naturally return to the idle state.

- All FSMs are designed in a way that they subsequently transit into an error recovery state upon entering an illegal state (due to an SEU). This error recovery state has to be reachable from a legal state to not be removed by the synthesis tools during the optimization process. This can be achieved by defining a “recovery state” as shown in figure 5.7.

```

12 architecture Behavioral of FSM is
13
14   -- state-attributes
15   attribute safe_recovery_state : string;
16   attribute safe_implementation : string;
17
18   type states is (st_idle, st_fifo_rst, st_wait, st_error);
19   signal state, next_state : states := st_idle;
20
21   attribute safe_implementation of state : signal is SAFE_STATEMACHINES;
22   attribute safe_recovery_state of state : signal is "st_error";
23

```

**Figure 5.7.:** VHDL code snippet showing how to declare the state attributes and define one state as “recovery state” (here called `st_error`). This way, XST implements the described FSM with recovery state functionality.

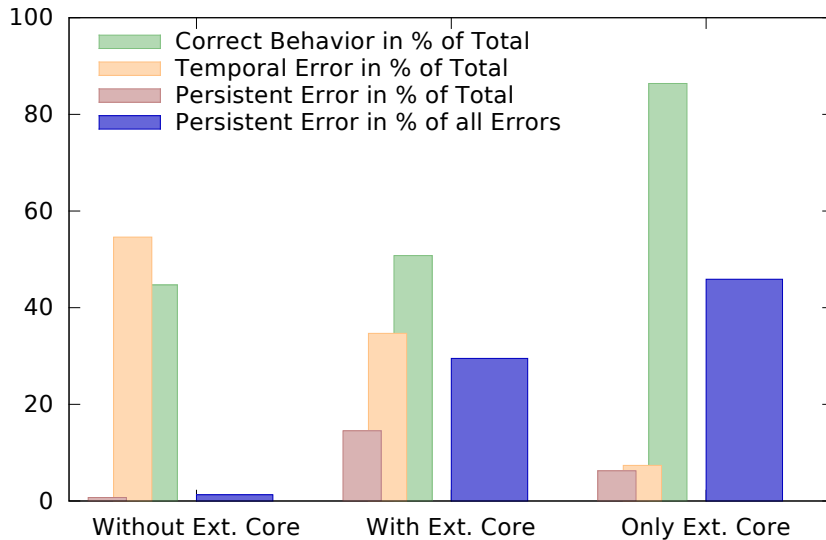
#### 5.2.4. Identification of Critical Components

With SEU injection it is possible to test existing logic components for their behavior with respect to SEUs. Since SEU injection only covers a subset of possible SEUs (those in static configuration memory) it is still mandatory to carry out in-beam tests for a valid evaluation of the efficiency of the radiation mitigation techniques. Nevertheless, with help of SEU injection tests, many problems can be identified and fixed prior to the in-beam test. An exemplary case for identification and resolving of an SEU-critical component is presented here.

For the 2013 in-beam test besides the self-written logic also an external core was part of the main firmware design. The external core is developed and provided by another group of the CBM collaboration. As the design that performed very well in the 2012 in-beam test did not utilize this core, major implications were expected when adding the core for the 2013 test. To evaluate the implications of adding the external core to the design, SEU injection tests were performed prior to the in-beam test.

SEU injection tests showed that the design including the external core did not perform well compared to the design without it. However, the problem could also spawn from new developments since the 2012 in-beam test and associated logic. To identify the problem, further investigations were undertaken and a third design was implemented. It consists of only the external core with minimal support logic to enable a successful oper-

ation with the rest of the read-out chain. This design was then also tested using the SEU injection method.



**Figure 5.8.:** Evaluation of three designs with the SEU injection method. The bars in green show the percentage of SEU injection cycles where no functional error was observed. The orange bars refer to the cycles where a functional error was observed but the design recovered after the injected SEUs are repaired with scrubbing. The red bar represents errors that persisted even after the SEUs are repaired. The blue bars give the percentage of SEU injection cycles with persistent errors, not taking into account cycles without error. The blue bars are independent from the SEU cross section of a specific design, and hence they are suitable for comparison of the three.

Figure 5.8 shows the result of the tests for all three designs, the one used in the 2012 in-beam test (without the external core), the one being prepared for being used in the 2013 in-beam test (with the external core), and the one consisting almost entirely of the external core.

Due to different resource utilization, and hence different SEU cross sections of the three designs, the measurements with respect to the total number of iterations (bars in red, orange and green) cannot directly be compared. However, the blue bars represent the fraction of error-cycles where errors are persistent, which is a value independent from device SEU cross section. The results of these SEU injection test show that the external core contributed significantly to the performance loss of the design.

Because of these SEU injection tests, a critical problem was identified prior to the in-beam test. The particular implementation of data retransmission for the transport protocol could lead to an infinite loop when data in the send buffers is corrupted. Together with the developers of the external core, the problem could then be identified and corrected before the in-beam test started.

Without the SEU injection method, it would not have been possible to identify the

problem beforehand, resulting in a high probability of the in-beam test to fail.

## 5.3. In-Beam Tests

An in-beam test has to be well prepared, the following points should be addressed before the test starts: The test procedure has to be designed properly in order to exploit the interesting features of the design under test. Furthermore, the expected failure rate need to be known in advance to apply for a beam time were the particle rate appropriate.<sup>4</sup> Another problem for in-beam tests is correct beam diagnostics. These issues will now be addressed in this section.

### 5.3.1. The Experiment Setups

The efficiency of applied radiation mitigation techniques, that were implemented in the course of this thesis, was evaluated through two in-beam tests at Cooler Synchrotron (COSY) in Jülich/Germany, one in 2012 and one in 2013. The particle accelerator provided protons at  $\sim 2\text{ GeV}$  and a maximum beam flux in the order of  $10^7\text{ s}^{-1}\text{ cm}^{-2}$  in both tests.

Both setups consisted of the device under test that was mounted in the beam line and a support board was mounted out of the beam line. Data of 28 GET4s was generated on the support board, emulating the GET4 message protocol. The generated data was deterministic and therefore could easily be analyzed for corruption after processing on the device under test. Using real GET4 ASICs as data source would have been possible but it would also have unnecessarily increased the complexity of the setup and make data analysis more difficult.

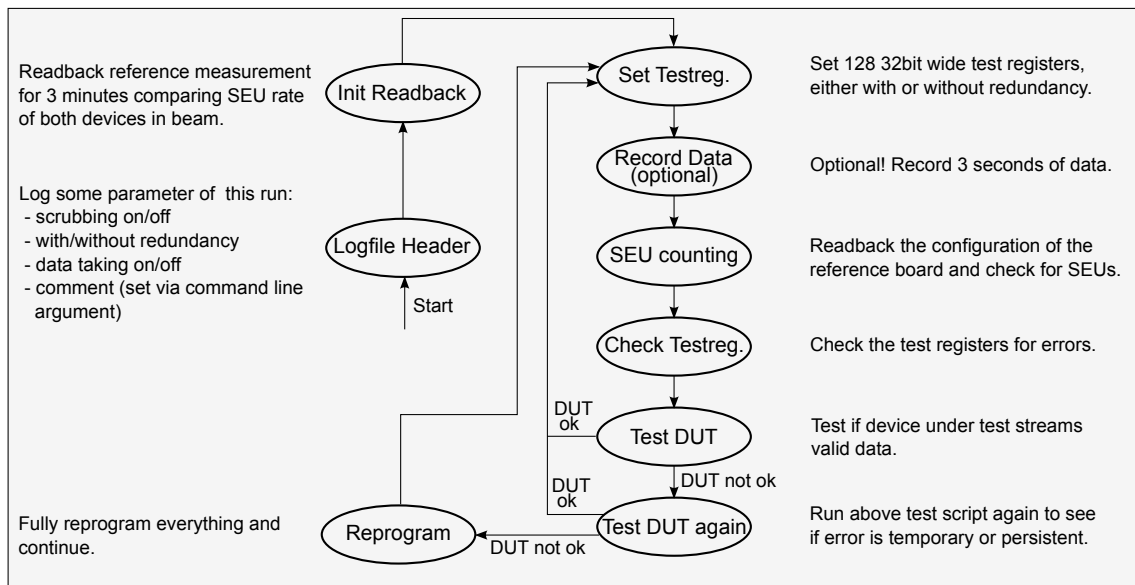
The firmware implementing the read-out and radiation mitigation techniques was running on a SysCore version 2 board. The on-board configuration controller was configurable to either execute blind scrubbing or to remain idle.

The test procedure illustrated in figure 5.9 is specially designed to evaluate the efficiency of scrubbing. During the execution of the test procedure scrubbing was either enabled, which means continuously refreshing the configuration memory of the FPGA, or scrubbing was disabled, then SEUs can accumulate over time. The test procedure is divided into the following steps:

- In the first step (`LogFileHeader`) important information is recorded and stored in the header of the logfile to help for subsequent analysis and archiving of the data.
- Step `InitReadback` is required to calibrate the board counting SEUs. The SEU Counter board is explained later in section 5.3.3 in more detail.

---

<sup>4</sup>It is not uncommon to share beam time among several experimenters. This becomes problematic when the setup of one group cannot sustain high particle rates but another group requires high particle rates.



**Figure 5.9.:** Illustration of the test procedure performed during the 2012 in-beam test to evaluate the efficiency of the applied radiation mitigation techniques. If enabled, scrubbing is running continuously in the background during all steps except Logfile Header and Init Readback. The key aspect is to test twice for correct operation which gives scrubbing time to repair the device. Without data taking, a regular loop lasts about 8 seconds. If Test DUT again is reached and the device did recover, it lasts about 12 seconds. A cycle with full reprogramming takes 18 seconds. The same procedure was also used in 2013 but without steps Set Testreg and Check Testreg.

The procedure then enters the main loop. During the runs with scrubbing enabled, the scrubbing engine is turned on at this point and continuously refreshes the FPGA configuration memory in the background.

- `Set Testreg` and `Check Testreg` are only included in the 2012 test. Here, a set of registers is tested for errors. Two firmware versions were created, one implementing TMR for these registers and one without redundancy.
- In `Record Data` data is recorded to hard disk for three seconds ( $\sim 15MB$ ) for subsequent offline analysis. This step is optional to not unnecessarily fill the hard disk, e.g. during debugging or at reference runs without beam. The timescale of three seconds is comparable to the time to recovery one can expect when board failures are detected and reconfigured from an entity outside the radiation zone.
- In SEU counting the current number of accumulated SEUs is recorded.
- The key idea of the algorithm is to check twice for an operational device, in `Test DUT` and in `Test DUT again`. The functional status is determined based on the online analysis of 2000 data samples. If not all data samples are valid the first time, the test is repeated to allow for scrubbing to repair the device. A single test takes much longer (2 s) than a full scrubbing cycle (80 ms).
- If the second test fails, the complete setup is fully reset (`Reprogram`).

The key aspect is to repeat `Test DUT` in case of an error, this gives scrubbing the opportunity to repair the device in the meantime. Only in case of two consecutive errors, the device is considered to be “permanently” corrupted. If the first test fails but the second test runs flawlessly, the error is counted as “temporary”.

It should be noted that the test procedure is mainly designed for the evaluation of the effect of scrubbing on the recorded data. The decisive tests, `Test DUT` and `Test DUT again`, operate on data only and leave aside any test of control register validity. For that reason, the positive effect of *Selective TMR* is not directly visible in the results (see section 6.3). With the steps `Set Testreg` and `Check Testreg` the redundancy effect is measured nevertheless, but the exploited test registers are specially added for the sole purpose of this test and are not part of the logic of the original GET4 read-out firmware.

**Test Setup 2012** The experiment took place from the 6<sup>th</sup> to the 9<sup>th</sup> of August in 2012. For the reasons described in section 4.3.1, only the GET4 module was installed in the beam line. The Optics module was operated on a supporting board, the same board on which also the GET4-data generator was running (see also figure 4.5).

The main tests for a functional device (`Test DUT` and `Test DUT again`) are based on data consistency. With *Selective TMR*, however, the data path is not protected by TMR. A comparison of a firmware implementing *Selective TMR* to a plain firmware without redundancy will not yield meaningful results. To nevertheless measure the redundancy effect,

128 registers (32 bit wide) were integrated in the DUT firmware. Two DUT firmwares were synthesized, one with TMR'ed test registers and one with non-redundant test registers.

The main intention for the test, however, is the offline analysis of the recorded data.

**Test Setup 2013** The experiment took place from the 2<sup>nd</sup> to the 4<sup>th</sup> of July in 2013.

This time the full GET4 read-out controller firmware was exposed to the particle beam. The supporting board was only required to execute the logic for deterministic generation of GET4 data.

The registers for testing redundancy are not implemented for the 2013 firmware. As they were implemented as part of the tunnel modules it would have been extra effort to integrate them into the 2013 firmware and the only gain would be a replay of the 2012 test. The main goal of the in-beam test is to operate a complete read-out controller firmware in a very high radiation environment and not to repeat the TMR test of 2012. Therefore, these registers are omitted in the 2013 test.

### 5.3.2. Preparation for In-Beam Tests

SEU injection tests already yielded valuable results during preparation for the in-beam tests, the efficiency of Hamming coded FSMs was evaluated in section 5.2.2 and a critical component could be identified in section 5.2.4.

In addition, such SEU injection tests also help to estimate the outcome of the in-beam tests for given particle fluxes. Knowing the expected failure rate is crucial when planning in-beam tests, e.g. when applying for particular particle fluxes.

For electronics SEU sustainability tests it is usually good to have a high particle flux because this results in more SEUs. More SEUs result in a higher failure rate, and hence in better statistics. However, there are upper limits for particle flux, either because the accelerator facility cannot provide more or because equipment of other experiments would be damaged. Knowledge of the functional failure rate is a key value when selecting an accelerator facility and also when deciding weather it is reasonable to join other groups for an in-beam test or not.

Since most FPGA configuration bits, especially routing resources, are unused within particular implementations [Xil14b, page 27], an SEU does not necessarily cause a functional failure. Functional failures of the design are also known by the term "SEFI", an abbreviation of *Single Event Functional Interrupt*).

Knowing the device cross section,<sup>5</sup> it is straightforward to calculate the expected SEU rate:

$$\text{SEU rate} \left[ \frac{1}{s} \right] = \text{part. flux} \left[ \frac{1}{s \text{ cm}^2} \right] \cdot \text{dev. cross sect.} [cm^2] \quad (5.1)$$

---

<sup>5</sup>The device cross section depends on the particle type and energy, it does not vary much for high energy hadrons (see figure 2.7).

From equation 5.1 one can then estimate the SEFI rate by a multiplication with the probability for an SEU resulting in a SEFI. In other words, one needs to know the average relation between the number of SEFIs to the number of SEUs.

$$\text{SEFI rate} = \text{SEU rate} \cdot \frac{\text{no. of SEFIs}}{\text{no. of SEUs}} \quad (5.2)$$

$$= \text{particle flux} \cdot \text{device cross section} \cdot \frac{\text{no. of SEFIs}}{\text{no. of SEUs}} \quad (5.3)$$

While the values required for device cross section are published by Xilinx ([Xil14b, page 27]), the SEU-SEFI relation is not known. An educated guess can be used for this value, but it might not be accurate. The value depends on design specific characteristics, like fabric resource consumption and applied radiation mitigation techniques, and therefore is not easily guessed. SEU injection tests allow for a more accurate SEU-SEFI estimate because the average number of SEUs required for a functional failure can directly be measured on a design specific level.

**Device Cross Section Calculation** The average device cross section for a Xilinx Virtex-4 FX20 FPGA as used in the tests is calculated from the amount of memory and the cross sections specific to the type of memory:

- The Virtex-4 FX20 has  $7.20 \cdot 10^6$  configuration bits in total [Xil09, page 87/88],  $6.02 \cdot 10^6$  bit of configuration memory and  $1.18 \cdot 10^6$  bit of BRAM memory [Xil10b, page 2].
- The bit cross section of the configuration memory is published by Xilinx ([Xil14b, page 27]). Configuration memory bits have a cross section of  $1.55 \cdot 10^{-14} \text{cm}^2$  while BRAM bits have a cross section of  $2.74 \cdot 10^{-14} \text{cm}^2$

The average bit cross section is then:

$$1.55 \cdot 10^{-14} \text{cm}^2 \cdot \frac{6.02}{7.20} + 2.74 \cdot 10^{-14} \text{cm}^2 \cdot \frac{1.18}{7.20} = 1.75 \cdot 10^{-14} \text{cm}^2 \quad (5.4)$$

The average device cross section is then:

$$\text{device cross section} = \text{number of configuration bits} \cdot \text{bit cross section} \quad (5.5)$$

$$= 7.20 \cdot 10^6 \text{ bit} \cdot 1.75 \cdot 10^{-14} \text{cm}^2/\text{bit} \quad (5.6)$$

$$= 1.26 \cdot 10^{-7} \text{cm}^2 \quad (5.7)$$

**Determination of SEU-SEFI Relation** The missing factor in equation 5.3 is the SEU-SEFI relation. It is estimated in a test employing the SEU injection method. A series of SEU injections with 20 injected SEUs per cycle revealed a failure rate of approximately one out of three cycles. From the cycles with errors, only one in 75 showed persistent errors that could not be repaired by removing the SEUs with scrubbing.

The SEU-SEFI ratio can now be estimated: About every 60<sup>th</sup> SEU causes a SEFI if scrubbing is not performed. If scrubbing is enabled, about every  $60 \cdot 75 = 4500^{\text{th}}$  SEU causes a SEFI. The SEU-SEFI relation is therefore:

$$\frac{\text{no. of SEFIs}}{\text{no. of SEUs}} \approx \frac{1}{60} \quad (\text{without scrubbing}) \quad (5.8)$$

$$\frac{\text{no. of SEFIs}}{\text{no. of SEUs}} \approx \frac{1}{4500} \quad (\text{with scrubbing}) \quad (5.9)$$

Note: As already mentioned, this is only an estimate. It does not take multi-bit upsets into account which are possible when 20 bits are injected simultaneously. Also the estimate for the number of SEFIs when scrubbing is enabled is based on the loose and optimistic assumption that only “persistent” errors will show when scrubbing is enabled. However, for this use case it is perfectly fine to not be very precise. The achieved values are still more accurate than a general “educated guess”.

Knowing the device cross section and SEU-SEFI relation, time to failure can be estimated:

Assumed Particle Flux	Expected Time to Failure			
	Without Scrubbing		With Scrubbing	
$10^5 \text{ 1/s.cm}^2$	4800 s	$\approx$ 80 minutes	354 000 s	$\approx$ 4 days
$10^6 \text{ 1/s.cm}^2$	480 s	$\approx$ 8 minutes	35 800 s	$\approx$ 10 hours
$10^7 \text{ 1/s.cm}^2$	48 s	$\approx$ 1 minute	3 580 s	$\approx$ 1 hour
$10^8 \text{ 1/s.cm}^2$	4.8 s	$\approx$ 5 seconds	358 s	$\approx$ 6 minutes

It can be seen that, even at a particle flux of  $10^6 \text{ 1/s.cm}^2$ , several in-beam days are required to measure reliable statistics with scrubbing enabled.

With these results it is possible to plan in-beam tests. It is evident that it makes no sense to carry out these measurements as parasitic experiment during in-beam tests for detector-prototype characterization, where high particle fluxes are not foreseen. A dedicated beam time for electronics tests over several days with particle fluxes of at least  $10^6 \text{ 1/s.cm}^2$  is required instead. The maximum possible particle flux delivered at COSY was in the order of  $10^7 \text{ 1/s.cm}^2$  (see section 6.3).

### 5.3.3. Beam Diagnostics

In this section common methods for beam diagnostics and their qualification for COSY beam tests are briefly discussed.

#### Scintillation Counter

A very common device to measure particle rates is the scintillating counter. A particle passing through the active volume of the scintillator generates a small amount of light



that can be converted to an electrical signal, amplified, and the resulting signal can then be counted.

However, the following issues prevent the usage of scintillation counters for our single event upset tests.

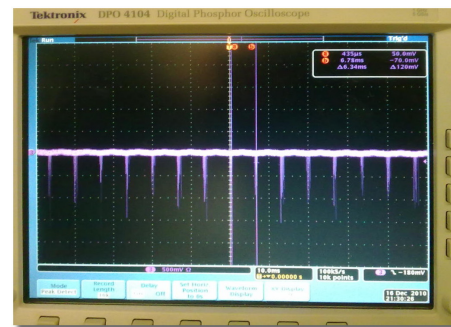
**High Rates** Scintillation counters cannot be used at particle rates of above  $10^5 - 10^6$  particles per second and per active area of the scintillator. At such high particle rates the dead time of the scintillation counter is longer than the time between two particles. The scintillation counter then approaches saturation and the counting rate reaches a plateau. The measured particle rate is then less than the real particle rate.

To be able to achieve enough statistics, we aim for rather high particle rates of not less than  $10^6 \text{ s}^{-1} \text{ cm}^{-2}$  (see section 5.3.2). This is already in the area, or even slightly above, where scintillators start to become unreliable.

Not only will they become unreliable, the scintillator will also show degradation effects caused by the high particle rate. Eventually, the device will be damaged.

**Pulsed Beam** On top of the required high rates comes another effect that is considerably vivid at COSY. Within one spill, the beam intensity is not constant but forms bunches. Figure 5.10 shows the response of a scintillator mounted in COSY beam. The bunched structure of the beam is clearly visible.

During  $\sim 5/6$  of the time there is no beam, and in consequence, during the remaining  $\sim 1/6$  the beam intensity is six times higher than it would be with a constant intensity beam. With such a temporal beam structure it is impossible to use a scintillator, already at much lower rates than we require for single event upset tests. A scintillator would measure no beam for  $\sim 5/6$  of the time and operate completely in saturation during the remaining  $\sim 1/6$  of the time period. This is especially delicate because the counting rate is usually interpreted as the average over a much longer time period than the cycle time of the bunch structure. This misleadingly suggests that the scintillator is not yet operated in saturation which would mean the measured particle rate is reliable. In reality however, during the relevant time window when particles are actually measured, the scintillator is operated completely in saturation, and the measurement is not reliable at all.



**Figure 5.10.:** Oscilloscope measuring the response of an in-beam scintillator at COSY. The bunched beam structure is clearly visible, the cycle time is about 6.5 ms.

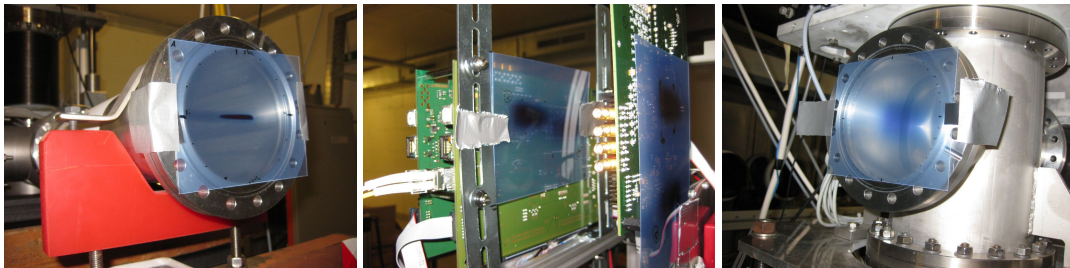
In consequence, a scintillator as particle rate monitor is not an option for single event upset tests at COSY.

### Self-Developing Dosimetry Film

A method for dosimetry, especially popular in biological and in medical science, is to install self-developing dosimetry film [Int] in the radiation exposed environment. The film darkens when exposed to radiation. The level of darkening can be used to determine the amount of radiation it was exposed to. Knowing the time of exposition, the average particle rate can be calculated.

Because there is no online information available and because it is cumbersome to determine the rate information from the darkening self-developing was not used for dosimetry film for the purpose of rate measurement.

However, it was very useful to determine position, opening angle, and shape of the beam by installing the film on the electronics in-beam. Figure 5.11 shows how self-developing film is used during the 2012 beam test at COSY. It proved to be a very straightforward, convenient and reliable method to “aim” the beam directly on the target and to choose a distance from the exit window where the beam is wide enough to fully cover the electronics to be tested, while it is still is focused enough to hit the target with high intensity.



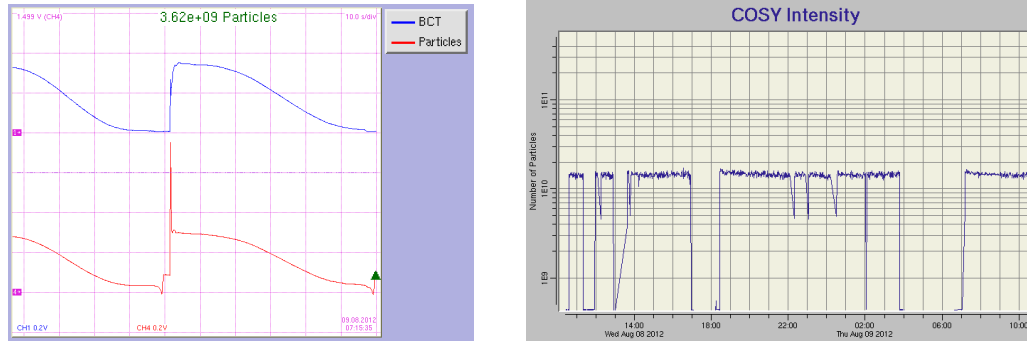
**Figure 5.11.:** Usage of self-developing dosimetry film during 2012 beam test. The picture on the left shows the exit window of the beam pipe in JESSICA cave. The picture in the middle shows the film directly mounted on the electronics setup. The picture on the right shows the widened beam at the end of the beam line.

### Estimates from COSY Data

As already mentioned in section 4.3.2, COSY provides online information of the synchrotron status on their homepage. Figure 5.12 shows exemplary screenshots of the two relevant plots.

Figure 5.12(a) illustrates, in blue, the measured beam-current transformer signal and, in red, the calculated number of particles per spill. Shown in figure 5.12(b) is the history of approximately one day of “beam intensity”. “Beam intensity” means the number of particles in the synchrotron, sampled once per spill. The trigger time is determined by Beam Position Monitors (BPMs) [BBG<sup>+</sup>94].

Knowing the number of particles per spill and the duration of extraction, the particle rate can easily be calculated:



- (a) The “Beam Current Transformer” (BCT) signal in blue, and, derived from BCT and synchrotron frequency, the number of particles in red. The green value at the top refers to the sample value at position of the green triangle on the red curve.
- (b) One day history of “intensity” (particles in the storage ring). The samples are taken from the red curve in figure 5.12(a) at time the Beam Position Monitors (BPMs) [BBG<sup>+</sup>94] trigger, resulting in one sample per spill.

**Figure 5.12.:** Information of beam characteristics provided by COSY on their homepage: <http://donald.cc.kfa-juelich.de/world>. Figure 5.12(a) has been color-inverted and slightly optimized for better readability in print.

$$\text{particle rate} \left[ \frac{1}{s} \right] = \text{number of particles} / \text{extraction time} [s]$$

This approach is not very accurate because it only takes into account the particles in the synchrotron and not the extracted particles reaching the experiment site. Potential particle loss during extraction procedure is completely neglected. During the in-beam tests situations were observed that show signatures possibly related to particle loss during extraction. The particle rate at our experiment (measured by the SEU Counter approach described in section 3.3.1) continuously decreased over time while the measured number of particles from BCT signals remained constant. This effect might also be caused by a shift of the beam position.

However, COSY homepage information allows a valuable estimation of the particle rate (upper bound). This is e.g. be used to check for plausibility of the values measured by other beam diagnostics instruments.

In addition, the one-day-history of the intensity gives a very convenient overview of beam condition during the experiment. Interruptions of the beam are common, e.g. to grant access to the cave for adjusting the experimental setup.

## Ionization Chamber

In higher radiation environments, a very common device for particle rate measurements is the ionization chamber. Unlike with scintillators, not a single signal per particle is detected but a flow of electric charge. A traversing particle ionizes the gas in the active

volume of the detector, the created ions are collected by a condenser and the resulting electric current is measured. The higher is the particle rate, the more ion pairs are created and the higher is the measured current.

Sven Löchner from GSI electronics department brought an elaborate setup for beam diagnostics to the COSY in-beam tests 2012 [LGWH13] and 2013 [LFG<sup>+</sup>14]. One amongst several components is an ionization chamber.

In 2013, the ionization chamber measured particle fluxes between  $6 \cdot 10^7 \text{cm}^{-2} \cdot \text{s}^{-1}$  and  $6 \cdot 10^7 \text{cm}^{-2} \cdot \text{s}^{-1}$ . Since the spill repetition rate is 22 seconds, but particles are extracted in only 7 of the 22 seconds and the ionization chamber only takes the 7 seconds into account, this translates to an average rate of  $\sim (2 - 3) \cdot 10^7 \text{cm}^{-2} \cdot \text{s}^{-1}$ .

### SEU Counter Approach

As already mentioned in section 4.3.2, the SEU Counter approach was used for particle rate measurements during the in-beam tests. A problem occurs when evaluating (blind) scrubbing. Scrubbing corrects SEUs and thus the number of SEUs does not accumulate. As a result, a test that is comparing the situation when scrubbing is enabled to the situation when scrubbing is disabled is not directly possible.

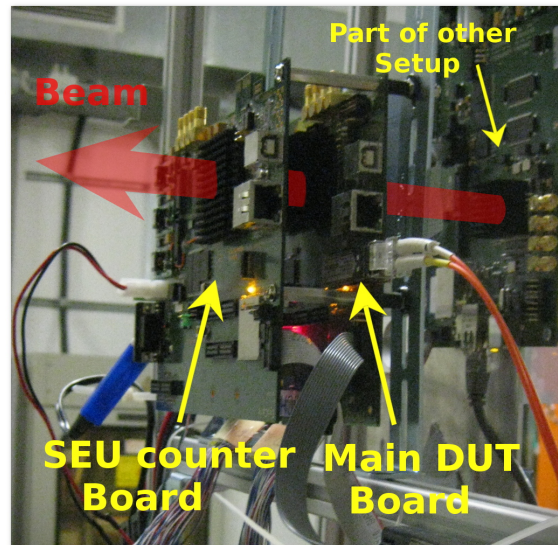
One solution is to perform a readback of the design before scrubbing the correct bits. Thereby the number of SEUs can be counted. This has been done in [RBK<sup>+</sup>12]. The drawback is that the scrubbing cycle time is thereby more or less doubled, because it can be assumed that readback of the configuration lasts as long as writing the configuration. For low SEU rates, this is not a problem. However, for high SEU rates, as expected (and required) for in-beam tests, this becomes relevant. In section 5.3.2 a SEFI was estimated every minute (for  $10^7 \text{particles/s}$ ), according to equation 5.8 this means one SEU every second. A scrubbing cycle is approximately 80 milliseconds, only one order of magnitude faster as the SEU rate. When too many SEUs accumulate during a single scrubbing cycle, the chance for multi-bit upsets increases. For that reason readback of the FPGA configuration was not performed. Instead, blind scrubbing was used.

The setup for the in-beam test solves the problem of missing SEU counts by mounting a second board with an identical FPGA into the beam line. The setup combines the advantages of fast scrubbing on the main board with precise measurement of SEUs on the counter board. Figure 5.13 shows the two-board setup.

It has to be cross checked if both boards indeed show the same SEU rate. The SEU rates on both boards may differ for the obvious reason that they might not be perfectly aligned in the beam line, i.e. one board might not be hit by full beam intensity, and hence shows less SEUs than the other.

There are also more complex reasons, like secondary particles that are created when beam particles pass through the first board then contribute to the particle rate in the second board. On the other hand, the material of the first board in the beam line might scatter a significant amount of beam particles out of the beam line, this would reduce the particle rate in the second board.

**Figure 5.13:** Two board setup used for the in-beam tests of section 6.3, here the 2012 installation. With this trick it is possible to perform SEU counting on one board while on the other board (DUT) the efficiency of blind scrubbing is evaluated. If executed on the same board, blind scrubbing would interfere with SEU counting.



Therefore, a comparison of SEU rates has been measured at the beginning of each run, see step `Init Readback` in figure 5.9. Both boards were left exposed to the beam for three minutes and the numbers of SEUs collected during this period in each board are recorded in the logfile.

In the main loop of the test procedure, only the SEU Counter board collected SEUs while the DUT board can safely execute scrubbing.



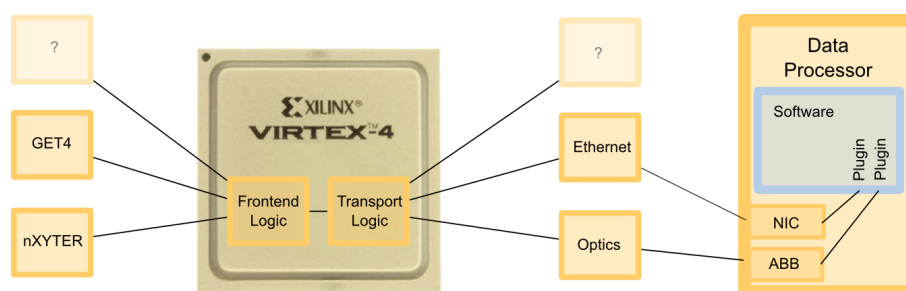
## 6. Results

This chapter presents the results that are achieved within the scope of the thesis. For the sake of continuity, it is also organized in three sections covering the same topics as the chapters before, although the third section – results of the in-beam tests – is clearly dominating this chapter.

First, in section 6.1, the modular approach for the implementation of the GET4 read-out controller is presented briefly. Then, section 6.2 covers the results of the radiation mitigation efforts that were not obtained during the in-beam tests. A particularly important achievement that is presented in this section is the very modest increase of resource consumption due to *Selective TMR*. However, since most of the results concerning radiation mitigation were achieved during in-beam tests, they are presented in section 6.3.

### 6.1. Modular ROC Usage

An important requirement for the Modular ROC firmware was the continuous support in development and maintenance, keeping up with the constantly increasing complexity of the read-out chain. Therefore, along with each important step in the development of the CBM read-out chain, compatible ROC firmwares were provided in a timely manner. Three generations of GET4 ASIC prototypes, two generations of optical transport protocols, two different types of transport solutions for smaller setups, and also the integration in the *roclib* software was supported during the years since 2009. This thesis does not focus on such achievements in engineering, nevertheless, they shall not remain unmentioned.



**Figure 6.1.:** The modular design of the read-out controller firmware.

Figure 6.1 illustrates the modular design of the ROC firmware with its separation into two modules, the *front-end module* and the *transport module*. The two modules are connected by a well defined interface that meets the requirements of the CBM readout chain.

For both use cases, the nXYTER read-out and the GET4 read-out, the same transport logic can be used.

For smaller setups, the Ethernet transport module provides a very convenient solution that was already available in early R'n'D phase. In bigger setups, that emerged in later R'n'D phase, the Optics transport module provided the missing features.

At all times the compatibility with the DAQ software [AMEKL10a] was maintained. For receiving data, and also for slow control, the main software can transparently use one of the two plugins referring to the according transport module.

It could be argued that the design with only two modules and also the choice of the interface in-between the modules is not the best choice. The approach with only two modules was not granular enough to prevent reimplementing of logic in all cases and the OPB/FIFO/DLM interface could seem to be an unnecessary re-specification of what the CBMNet interface already provides. However, when considering the situation in which the design decisions were taken it turns out to be a very successful approach that allowed for continuously providing a working read-out solution while improving the overall design in parallel.

During the time the radiation mitigation efficiency experiments have been prepared and carried out, development on the GET4 read-out firmware has been continued. Nowadays the design is more granular, based on three modules instead of just two. However, the design that was used for the in-beam tests, and hence on which the results of this thesis are based upon, is the design with two modules and this one is therefore referred to in this chapter.

### 6.1.1. Operative Firmware

The fixed schedule of several in-beam experiments induced sharp deadlines for providing operative firmwares.

Two of those in-beam experiments are described in detail later in this chapter because their outcome is a major part of the results of this thesis (see section 6.3). In addition, many in-beam experiments have been carried out by other CBM members to test their detector prototypes.

For all in-beam experiments listed in table 6.1, a variant of the *Modular ROC* was used for data read-out from the referring front-end electronics. For all in-beam tests, increasingly complex firmwares were provided, always in time and always working as intended. Although there have been bugs in the read-out chain, responsible for irrecoverably corrupted data, not a single cause for significant data corruption was traced to a failure of the *Modular ROC* firmware. The modular design of the firmware helped a lot in meeting the deadlines while still providing reliable logic, especially in the early days (2010/2011) when the optical transport was not in a very stable state yet.



Beamtime	Date	Frontend Chip	Data Transport
GSI/Darmstadt	2008, Sep.	nXYTER	Ethernet
GSI/Darmstadt	2009, Aug./Sep.	nXYTER	Ethernet
CERN (PS/T10)/Geneva	2010, Nov.	nXYTER	Ethernet
COSY/Jülich	2010, Nov.	GET4	Optics/Ethernet
COSY/Jülich	2010, Dec.	GET4/nXYTER	Optics
COSY/Jülich	2011, Mar.	GET4	Ethernet
CERN (PS/T9)/Geneva	2011, Oct.	nXYTER	Ethernet
COSY/Jülich	2011, Nov.	GET4/nXYTER	Optics
COSY/Jülich	2012, Jan,	nXYTER	Optics
COSY/Jülich	2012, Jul./Aug.	GET4	Optics
GSI/Darmstadt	2012, Nov.	GET4	Optics
CERN (PS/T9)/Geneva	2012, Oct./Nov.	nXYTER	Optics
COSY/Jülich	2013, Jun./Jul.	GET4	Optics/Ethernet
COSY/Jülich	2013, Dec.	nXYTER	Optics
CERN (PS/T9)/Geneva	2014, Nov.	nXYTER	Optics
COSY/Jülich	2014, Dec.	nXYTER	Optics
CERN (SPS)	2015, Feb./Mar.	GET4	Optics

**Table 6.1.:** A list of all in-beam tests where a *Modular ROC* firmware was used in at least one sub-system to read out data from the front-end electronics.

### 6.1.2. Software Integration

In addition to firmware development, integration in the existing *roclib* software<sup>1</sup> was essential for a successful operation of the read-out chain. Without software integration, the firmware cannot be interfaced from the CBM framework. The required procedures for decoding GET4 data messages were integrated in the referring GSI software repository, and the graphical user interface of the software was upgraded to provide an intuitive interface to control the GET4 chip. A screenshot of the GUI is shown in figure 6.2.

## 6.2. Radiation Mitigation Techniques

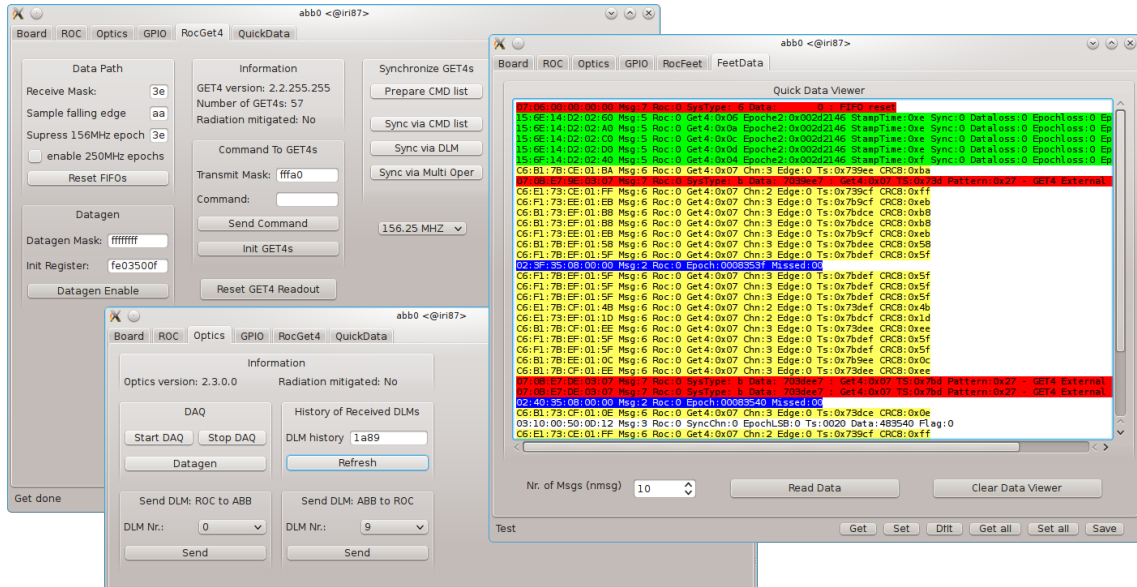
This section remains short as the major part of the radiation mitigation results are presented in section 6.3. Nevertheless, the very modest increase of fabric resource consumption of *Selective TMR* that is presented here is important as it is the reason for not implementing “full TMR” in the first place.

### 6.2.1. Blind Scrubbing

The configuration of the FPGA was permanently refreshed in the background, while the firmware executed its main task, the read-out of GET4 data, in parallel.

Scrubbing was implemented as *blind scrubbing*, the integrity of the FPGA configuration was not monitored. To do so, read-back of the FPGA configuration would be required.

<sup>1</sup>The *roclib* software is organized in a subversion repository at <https://subversion.gsi.de/cbm/ROC> (GSI Account required).



**Figure 6.2.:** The *roclib* software comes with a graphical user interface called *rocGui*. This figure shows the “tabs” that were added to the existing software as part of the work for this thesis. Those “tabs” can be used to intuitively control the GET4 chip and provide a well-arranged and color-highlighted view of the received data.

This is unnecessary (no harm is done when refreshing a valid configuration) and would double the time to repair.

*Blind scrubbing* showed great results during in-beam tests that are described later, in section 6.3.

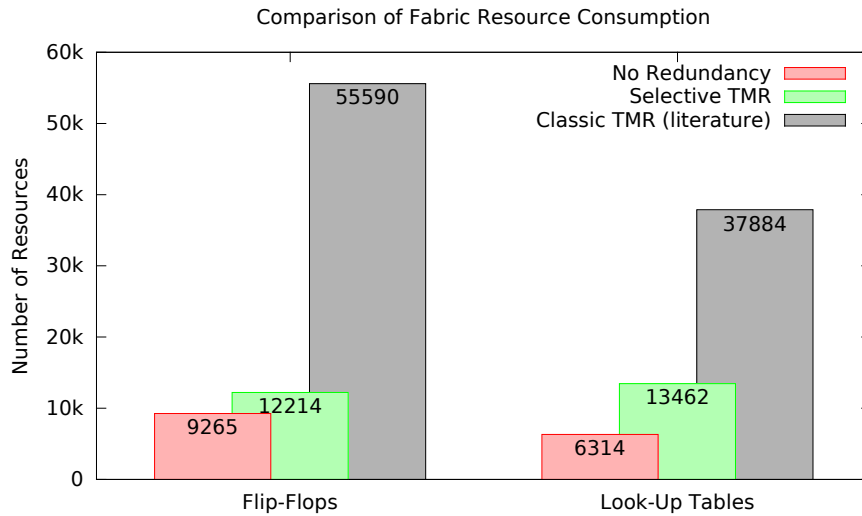
### 6.2.2. Fabric Resource Consumption

The main reason to avoid automated tools for implementing the necessary redundancy in the firmware is the enormous overhead in terms of fabric resource consumption that such an approach entails (see section 3.2.2).

The additional usage of fabric resources required for implementing TMR’ed logic could be significantly reduced with the introduction of *Selective TMR* (see section 4.2.2).

*Selective TMR* exploits the fortunate coincidence that the logic for the data path represents the major part of the whole firmware, and furthermore, corruption of a (very) small fraction of the data is not a huge issue for a high energy physics experiment. Therefore, we can omit costly redundancy for the whole data path related logic and use inexpensive CRC checksums to detect data corruption. TMR is only required for the control logic which makes just a small part of a typical FPGA design. In case of the GET4 Read-Out Controller it contributes to the firmware with about 10%.

To determine the extend of resource usage overhead of *Selective TMR*, the GET4 read-out firmware was synthesized with and without redundant logic. Figure 6.3 shows a comparison of the resource usage as given by the Xilinx MAP reports.



**Figure 6.3.:** Comparison of fabric resource consumption (Flip-Flops and Look-Up Tables), as reported in the Xilinx MAP report for the GET4 read-out firmware that was used for the 2012 in-beam test. The boxes in red represent the numbers of the original design, without redundant logic, and the boxes in green give the numbers of the design with *Selective TMR*. The gray boxes are estimates for the resource consumption of traditional Triple Module Redundancy (TMR) and are added for comparison. As suggested by literature, six times the numbers of the original design is used to estimate the numbers for traditional TMR.

The resource usage of the design that implements *Selective TMR* grew only to about 150 % (FFs) / 200 % (LUTs) compared to the original design without redundancy. This is fairly modest considering the  $\sim 600\%$  that the overhead of full TMR entails according to literature [WRGC03b], [WRGC03a], [MMPW07].

### 6.3. In-Beam Tests

As already mentioned in section 4.3.1, the firmware was tested in two in-beam tests. Due to tight schedule for the in-beam tests in 2012, only parts of the firmware could be prepared for the test in time. The complete front-end module for interfacing the GET4 ASIC was extracted for stand-alone tests in-beam and showed big improvements in radiation tolerance while keeping the overhead manageable. Finally, in 2013 the rest of the design was ready as well and the promising results of 2012 could be repeated with the full design in-beam. The two tests show the feasibility of operating complex detector read-out logic in an FPGA in a high radiation environment if radiation mitigation is applied properly.

The results of both experiments are presented in the following.

### 6.3.1. Test Setup 2012

The board that executed the logic under test was mounted in the beam line while data generation and communication to the DAQ PC was implemented on an auxiliary board out of the radiation zone. The accumulation of SEUs was monitored in parallel by a third board that was also mounted in the beam line. The basic setup is explained in more detail in section 4.3.1, the SEU Counter approach in section 5.3.3.

The test consisted of several runs, alternately testing the various combinations of scrubbing on/off and with/without *Selective TMR*.

Figure 6.4 shows the exemplary data recorded during the experiment. These plots give a very direct impression of the behavior of the design when scrubbing is disabled (figure 6.4(a)) and when scrubbing is enabled (figure 6.4(b)). Each of the diagrams shows the results of a three hours run. The plotted data directly represents the values were measured during the tests.

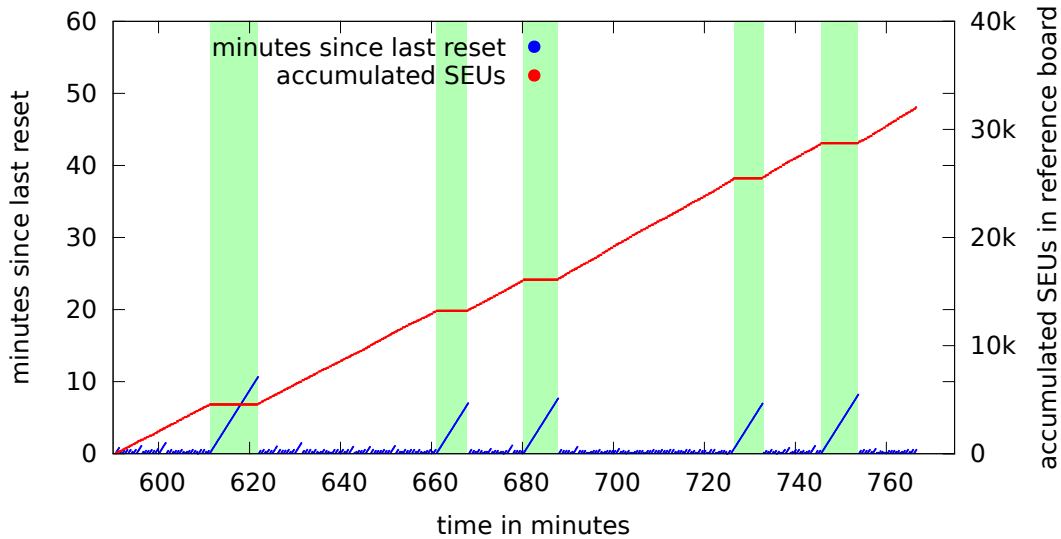
The red plot represents the number of SEUs that have accumulated in the SEU Counter board since start of the run. Plotted in blue is the time since the setup had to be reprogrammed the last time (step *Reprogram* in the test procedure, figure 5.9). Every return-to-zero of the blue plot refers to a system reset.

From the blue plot one can see, that without scrubbing the design had to be reset within about one minute (the blue plot crawls at the bottom of the diagram) while when scrubbing is enabled, the system can survive for a much longer time (the blue plot rises to much higher values). This does not imply that no error occurred for several minutes when scrubbing is enabled but only that if an error did occur the system could recover to correct operation.

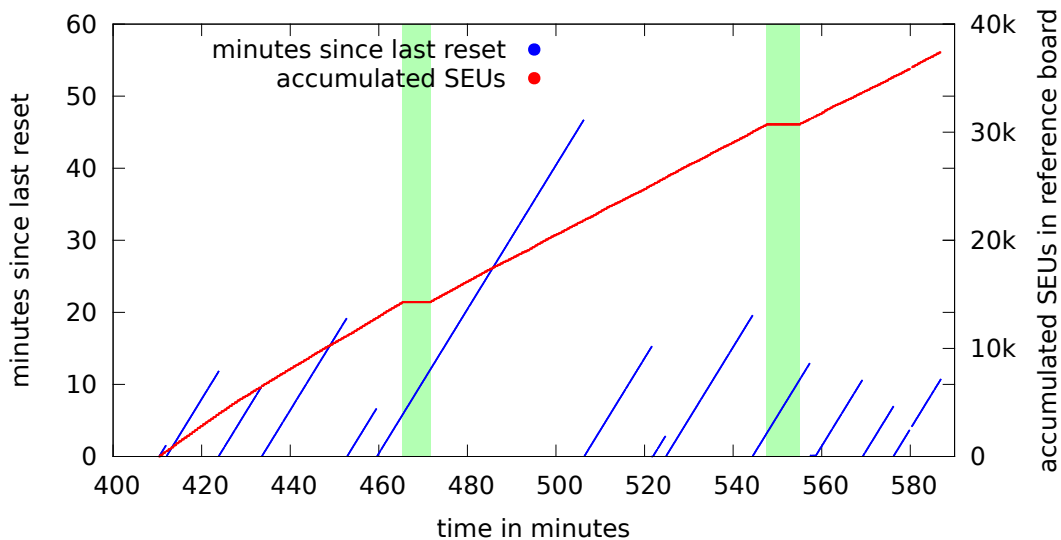
There are some time slots highlighted in green during which no beam was available. At the time the beam was shut down because the neutron level in the synchrotron was exceeding the safety threshold. During these time slots no SEUs are collected in the SEU Counter board, the red plot forms a plateau. The blue plot shows that no system reset was required during these times as well, which is especially visible in figure 6.4(a) where scrubbing is disabled and system uptime is usually significantly lower.

The three hours runs (the data is shown in figure 6.4) delivered data for the firmware without redundancy only, during the measurements with a firmware based on *Selective TMR* the beam was interrupted for more than half of the time. No data could be stored for offline analysis in both cases. As the particle beam showed to become more and more unstable, the time of a single run was reduced from 3 hours to 30 minutes and the synchrotron was set to a lower particle rate. The following analysis is therefore based on data from the shorter runs, and at a slightly smaller SEU rate (2 SEUs/s instead of 3 SEUs/s). Nevertheless, all the runs (scrubbing on and off, firmwares with and without redundancy) were operated under comparable conditions.

**Statistical Analysis** While figure 6.4 is provided to give a direct impression on the system behavior during the experiment, figure 6.5 shows more details in a statistical anal-



(a) Scrubbing is disabled. Full reset of the setup required in less than a minute. The setup is only stable when beam turned off.



(b) Scrubbing is enabled. The setup runs stably for several minutes.

**Figure 6.4.:** The red plot refers to the number of SEUs collected in the reference board while the blue plot shows the time since the last full reset of the setup. Every return to zero of the blue plot refers to an unrecoverable failure of the setup caused by radiation. During the time slots highlighted in green, the beam was shut down for technical reasons.

ysis. It compares the results of four different runs. The four runs were recorded with two different firmwares, one implementing *Selective TMR* and one without redundancy. In both cases, the firmware was operated for one run with scrubbing disabled and for another run with scrubbing enabled. The results of the firmware that was synthesized without redundancy are shown in figure 6.5(b) and the results based on the firmware that implements *Selective TMR* in figure 6.5(c).

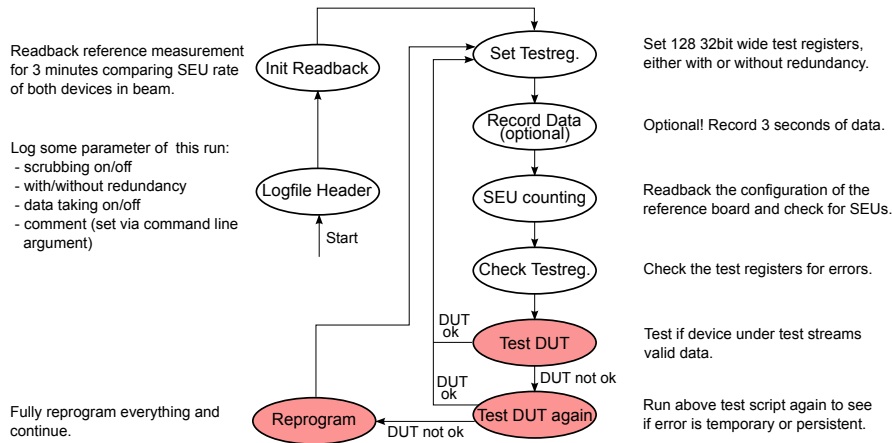
The green bars refer to iterations in the test procedure where no error was detected in step Test DUT (figure 5.9). The orange bars represent the iterations in which an error was detected in step Test DUT, but the error was temporary and the system could recover so that step Test DUT again showed a functional device. The iterations in which an error was detected in both steps, in Test DUT and also in Test DUT again, are illustrated by the red bars.

A first observation is that, within the statistical uncertainty, no difference can be determined between the design without redundancy and the design that implements *Selective TMR*. This leads to the question why *Selective TMR* should be implemented at all, given it shows no difference in the error counts. The reason for *Selective TMR* not showing any effect is, that the validation of the device's functionality is based only on data and not on control registers. *Selective TMR*, however, adds redundancy only for control logic and not on the data path. The cross section of the control logic is much smaller than the cross section of the data path' logic. The number of errors based on SEU effects in the control logic is smaller than the uncertainty (or "noise") of the number of errors based on SEU effects in the data path logic. The mitigation effects of *Selective TMR* are therefore not visible here. Nevertheless, TMR effects are evaluated in a separate measurement that does not take into account errors in the data path and is discussed later in figure 6.7.

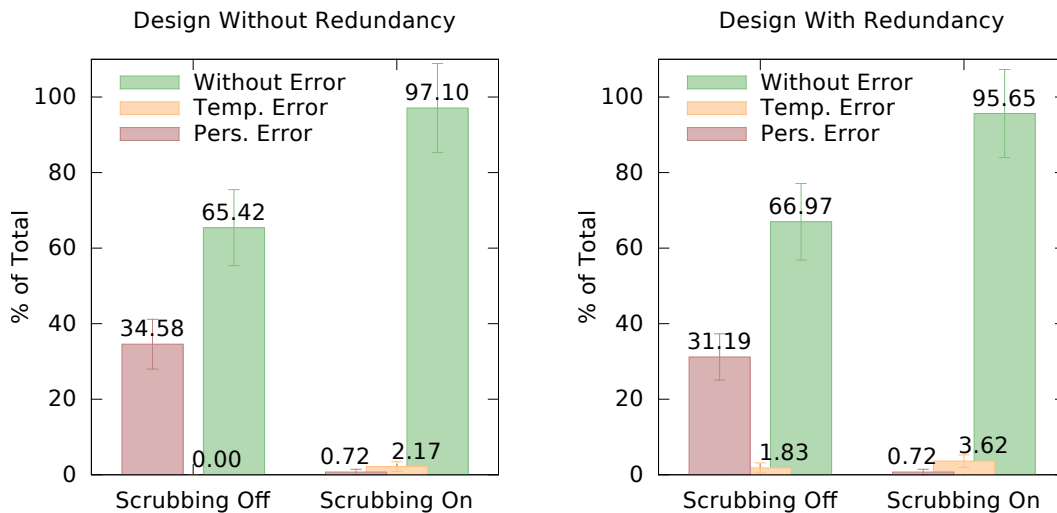
The second observation is that with scrubbing enabled, the number of iterations with persistent error (when Test DUT failed twice) drops significantly, by a factor of almost 50. This means, that 98 % of the errors could be repaired by scrubbing.

Third, it can be observed that a few persistent errors remain, even with scrubbing enabled. This is due to the following reasons. An error is considered to be "persistent" if two consecutive tests fail (see figure 5.9). However, the second test fails with the same probability as the first one does. So, those two consecutive tests can both fail independently, without causal relationship. The probability for both tests failing independently is  $p_{\text{test 1 and 2}} = p_{\text{test 1}} \cdot p_{\text{test 2}} = p_{\text{test 1}}^2$  which results in about 0.1 to 0.2 % false "persistent errors" in case of figure 6.5, which is already within the error bars. An SEU in the clock manager for the design could also lead to an "persistent error" and there is also the (unlikely) possibility of a multi-bit upset in the controls logic.

As a fourth observation, it can be noted that even when scrubbing is disabled (and SEUs are *not* repaired) a few temporary errors appear. This can happen, because the data path is not protected by TMR. An SEU that does not change static FPGA configuration (routing or LUT) but the dynamic part (flip-flop or BRAM) can corrupt a data word in a buffer. After read-out of the faulty data word the error is no longer present in the device, also in the case when scrubbing is disabled.



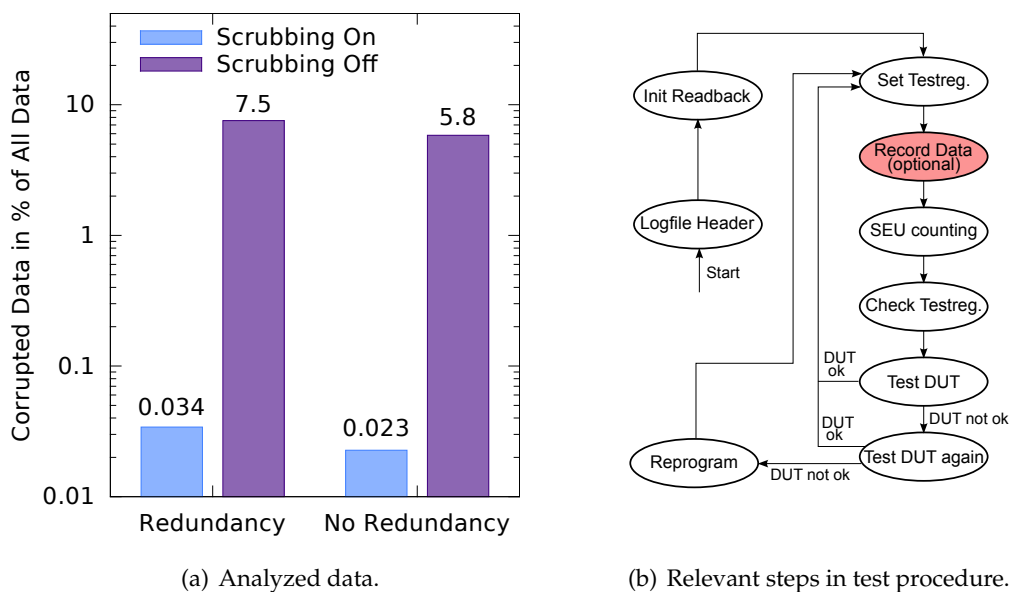
(a) Relevant steps in test procedure (see figure 5.9, page 88).

(b) Results of the design without redundancy. (c) Results of the design implementing *Selective TMR*.

**Figure 6.5.:** This plots shows the percentage of iterations without errors (green), iterations that show errors once but recovered to correct behavior afterwards (orange), and iterations that suffered from permanent errors (red). The data was recorded during the 2012 in-beam tests in four comparable runs (SEU rate:  $\sim 2 s^{-1}$ ). Scrubbing significantly improves the in-beam behavior of the design while no difference was measured for *Selective TMR* (due to the nature of the test, see text). For TMR effects, see figure 6.7. The error bars are calculated by assuming a Poisson distribution for the absolute values.

**Analysis of Recorded Data** More important than the statistical analysis of the steps Test DUT and Test DUT again is the offline analysis of the data that was recorded in step Take Data. This gives an impression of the data quality, the actually important property for the CBM experiment.

Figure 6.6 shows the results of the offline analysis of the obtained data. The bars represent the fraction of corrupted data with respect to all data that is recorded. For the same reasons as above, there is no significant difference between the firmware without redundancy and the firmware that implements *Selective TMR*.



**Figure 6.6.:** Analysis of the quality of recorded data. Note the logarithmic scale on the y-axis. During the same four runs that are presented in figure 6.5, data was recorded for 3 seconds ( $\sim 15$  MB) in every iteration. The bars show the percentage of corrupted data in the recordings. In this scenario, scrubbing improved the system by two orders of magnitude. Error bars are not given here for reasons explained in text.

Then again, the effect of scrubbing on data quality is enormous, the values differ by two orders of magnitude. A logarithmic scale is required to compare data taken when scrubbing is enabled with data taken when scrubbing is disabled in the same plot. Scrubbing could reduce the percentage of corrupted data by a factor of 200.

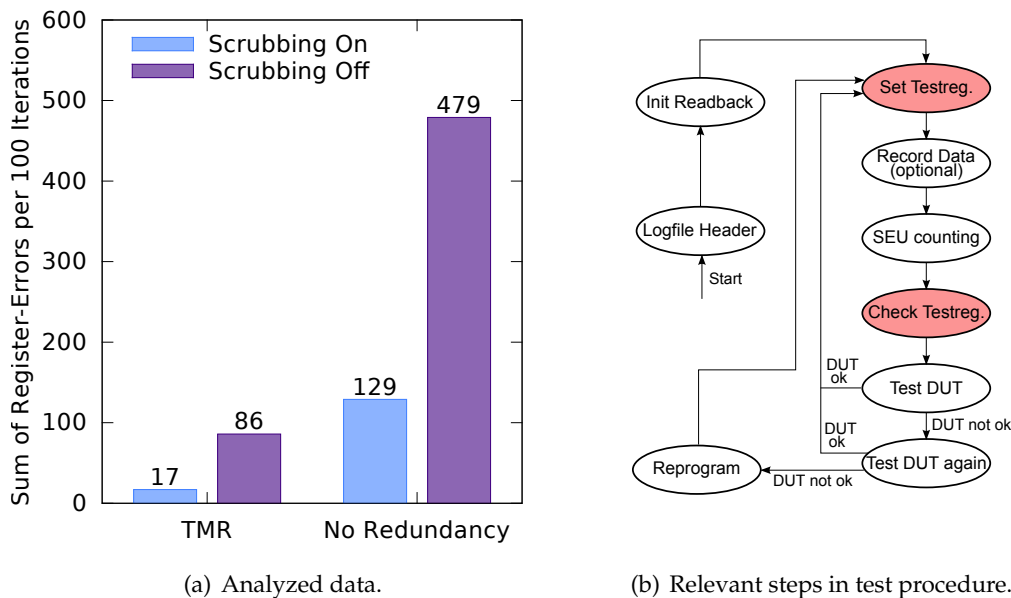
No error bars are given here since errors occur in bursts, the probability for a data message to be corrupted is not independent from corruption of previous data messages. In addition, there is a systematic dependency between the values measured without scrubbing and the time for which data is recorded. Without scrubbing, the device can only recover at the end of the test procedure iteration. The longer data is taken during one iteration, the higher is the percentage of corrupted data in the case of disabled scrubbing. When scrubbing is enabled, however, the percentage of corrupted data is mostly



independent from this effect because scrubbing can repair most errors in the device on-the-fly.

During the in-beam tests, data was recorded for 3 seconds per test procedure iteration. This is comparable to the reaction latency one can expect when board failures are detected and corrected from an external entity, as it is planned for the final CBM setup.

**Test Register Measurements** To exploit the effect of TMR, a set of 128 test registers (32 bit wide) was implemented in the firmware and checked in steps Set Testreg and Check Testreg of the test procedure. Figure 6.7 shows the results of the evaluation of the test registers.



**Figure 6.7.:** Analysis of TMR efficiency. As the main test does not exploit the effect of TMR, an array of test registers was added to the design that was exposed to beam particles. This array does not serve any functional purpose, however, it provided enough statistics to unveil the positive effect of TMR. It can also be seen, that *scrubbing* helps as well.

Since these measurements are not overlaid by errors in the data path, the effect of TMR unveils. In contrast to the data path centric tests presented in figures 6.5 and 6.6, here a clear improvement can be seen when TMR is implemented. TMR alone reduces the errors in the test registers by factor of about 5 to 7, in combination with scrubbing the factor is around 20.

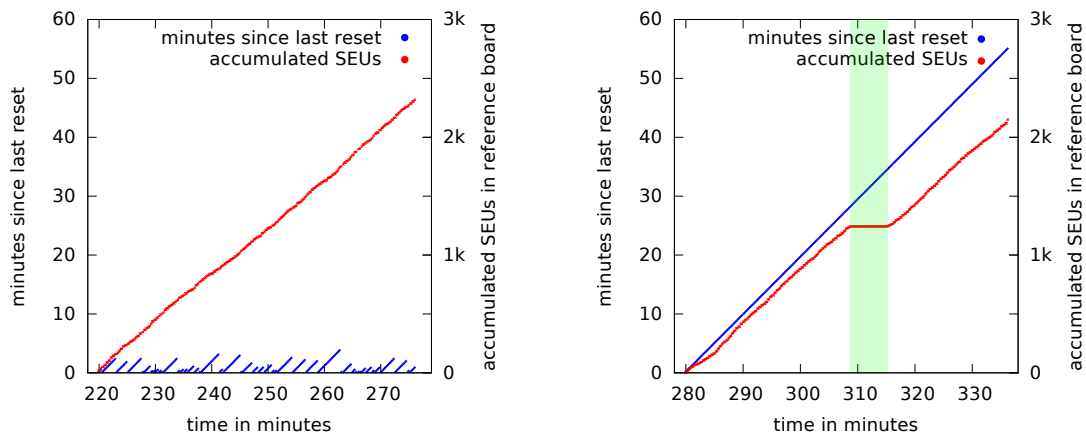
The fact that despite TMR and scrubbing still some errors are measured is probably due to multi-bit upsets. The very high data rate during the experiment clearly favors multi-bit upsets. For the operation scenario at CBM, a much more relaxed upset rate is

expected while the time to repair remains the same. Hence, the probability for multi-bit upsets is much lower.

### 6.3.2. Test Setup 2013

The 2012 test showed very promising results for the frontend logic interfacing the GET4 ASIC. However, a significant part (the transport logic connecting to the DAQ PC) was operated out of the beam line for reasons described in section 4.3.1. Therefore the 2012 tests are not complete. So, in 2013 the tests have been repeated with the full GET4 read-out firmware mounted in the beam line.

Contrary to 2012, no actions have been taken to synthesize a firmware without redundancy. The 2012 tests already showed that the major measurements do not change significantly without redundancy. Furthermore, the 2012 *Test Register Measurements* are omitted. They are based on an additional piece of logic that is not part of the GET4 read-out controller, repeating these tests would not gain any new results.



(a) Scrubbing is disabled. Same behavior as 2012, a full reset of the setup required in less than a minute. (b) Scrubbing is enabled. The setup runs stably for the whole run (one hour).

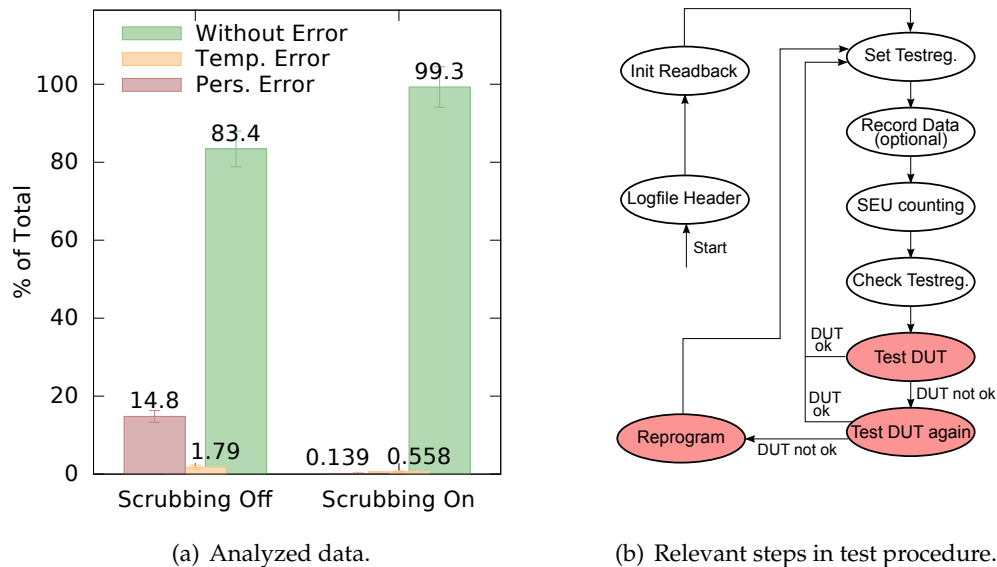
**Figure 6.8.:** The same plots as in figure 6.4, here based on data from the 2013 in-beam test. For this test the full GET4 read-out logic was exposed to the beam particles. Further differences to 2012 are that the two runs only lasted for one hour each (three hours in 2012), and the SEU rate is about  $1/4$  of the rate from 2012. However, the main result could be repeated, scrubbing significantly improves the system uptime.

Figure 6.8 shows the same plots for the 2013 test as figure 6.4 does for 2012. The intention for these plots is to give a direct impression on the behavior of the system during the experiment. The plotted values were directly recorded like that, no recalibration or other calculation was made for the plots.

The proton rate ( $5 \cdot 10^6 \text{ cm}^{-2} \text{ s}^{-1}$ ) was less than 2012, so the upset rate was lower, resulting in longer system runtimes also without scrubbing enabled. However, the basic

observation is the same, scrubbing significantly improves the system uptime.

**Statistical Analysis** A more detailed statistical analysis of 2013 data is shown in figure 6.9 (as done for 2012 data in figure 6.5). Again, the basic conclusion is the same, the number of iterations with persistent error (when Test DUT failed twice) drops significantly.



**Figure 6.9.:** The same plot as in figure 6.5, here based on data from the 2013 in-beam test with the whole design exposed to beam particles. The beam intensity was lower than 2012 (SEU rate:  $\sim 0.6 s^{-1}$  compared to  $\sim 2 - 3 s^{-1}$  in 2012), however, the promising results with respect to scrubbing could be repeated.

The gaining factor is better this time, about 100 which is better than the factor of  $\sim 50$  that was measured in 2012. The reason for this might be that in 2013 the synchrotron could not deliver the same high intensity beam as 2012. The beam particle rate was lower and hence less multi-bit upsets occurred.

However, it should also be mentioned, that the value of 0.139% shown in figure 6.9(a) is based on only one event. One event is not enough for a statement of statistical significance, the factor of 100 should not be overrated.

Nevertheless, the improvement when enabling *scrubbing* is still clearly visible.

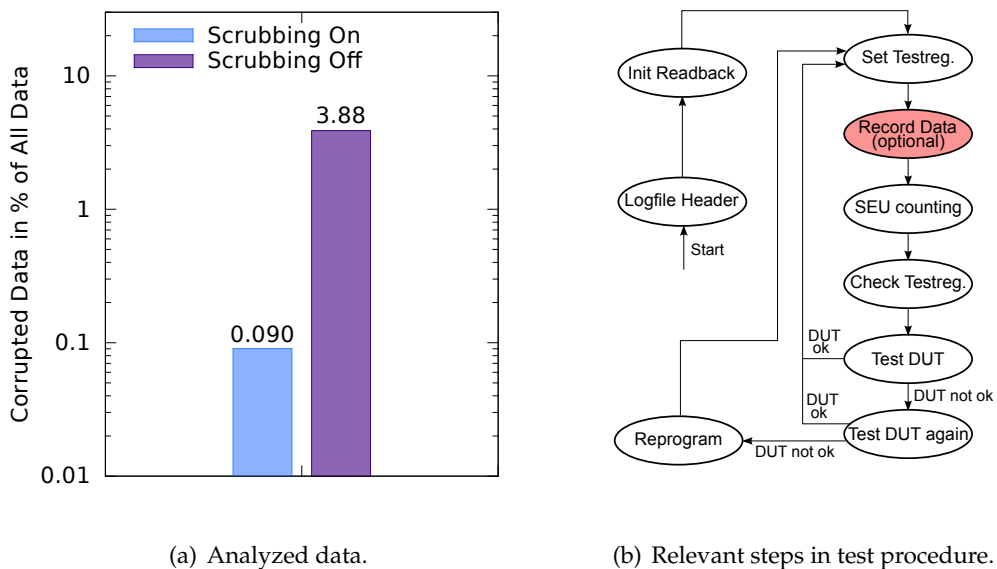
**Analysis of Recorded Data** Offline analysis of recorded 2013 data is presented in figure 6.10. As for the analysis of 2012 data (figure 6.6) again a logarithmic scale is chosen for the plot. The effect of scrubbing is not as large as it was in 2012, but it still differs by a factor of  $\sim 40$ , or about one and a half orders of magnitude in the chart.

One reason why the factor is lower than in 2012 is the missing step *Set Testreg* in the test procedure of 2013. In 2012, this step was executed before the data was recorded and

lasted approximately one second. Any SEU that causes a failure during this second also corrupts the data that is recorded afterwards. In 2013, data was recorded right after the device was reprogrammed. Without scrubbing, errors remain in the device and the probability to record corrupted data is higher the longer the system is already running. Data that is recorded from second 2 to 4 (the 2012 case) shows more corruption than data that is recorded from second 1 to 3 (the 2013 case). However, this is not true when scrubbing is enabled. Most errors are repaired and the uptime of the system is less relevant.

It should also be mentioned that a direct comparison between the two measurements (2012 and 2013) is not as straightforward as it seems. In 2013, the particle rate was lower than 2012, but on the other hand, the cross section of the firmware was higher (“full firmware” vs. “half of the firmware”). In addition, it should be noted again, that erroneous data occurs in bursts and the accuracy of the numbers should not be overrated.

However, the qualitative analysis of the measurement does not change, the mitigation effect of scrubbing is very high.



**Figure 6.10.:** The same plot as in figure 6.6, here based on data from the 2013 in-beam test with the whole design exposed to beam particles. The beam intensity was lower than 2012 (SEU rate:  $\sim 0.6 s^{-1}$ ), however, the great results with respect to scrubbing could be repeated. Note the logarithmic scale on the y-axis.

### 6.3.3. Beam Diagnostics

The approach of directly measuring SEUs instead of calculating them from an otherwise measured particle rate has proven to be very comfortable. It is discussed here briefly.

## SEU Counter Approach

In the beginning of each run, in step `Init Readback`, SEUs were collected in both boards, the SEU Counter and the device under test (DUT) (see section 5.3.3). Comparing the SEU rates of the SEU Counter board and DUT shows, that indeed the SEU rates of the two boards differ slightly. In 2012 the DUT was the first board in beam direction and the SEU Counter board was mounted behind, in 2013 the setup was the other way around.

In 2012, the SEU rate in the DUT board was slightly lower (around 90%) than the SEU rate in the SEU Counter board. In 2013, it was the same, despite the different order of the boards. This time, the DUT showed an SEU rate significantly lower (around 70%) than the one of SEU Counter board. This was probably due to a small misalignment, the two boards might not have been perfectly aligned in the beam line. Anyway, when using the measured SEU values, they have to be corrected for these factors.

From a qualitative viewpoint, the counting of SEUs during the experiment worked very well, the red curves in figures 6.4 and 6.8 represent the number of SEUs accumulated over time in the SEU Counter board. As expected, the number of SEUs continuously grows showing a constant slope indicating stable beam conditions. The plateaus that show in the plots refer to times during which the beam was stopped, either for technical reasons, or because access to the cave was requested.

In 2013, the particle rate calculated from SEUs is significantly smaller than the particle rate of  $\sim (2 - 3) \cdot 10^7 \text{ cm}^{-2} \cdot \text{s}^{-1}$  that was measured by the ionization chamber described in section 5.3.3. The SEU rate translates to a particle rate in the order of  $5 \cdot 10^6 \text{ cm}^{-2} \cdot \text{s}^{-1}$ , a factor four or five off.

Probably both boards were not centrally hit by the beam. Without SEU counting and only relying on the values from the ionization chamber, this would be a severe problem. The results would refer to a higher particle rate than the boards were actually exposed to, and hence SEU induced failures appear to occur less often than they really do. However, because of the SEU counting method was implemented, this is not a problem at all, the number of SEUs are accurately recorded.



## 7. Discussion

### 7.1. Applying Results on the CBM Use Case

Fast hadron flux is the relevant parameter for SEUs (see *Soft Errors*, page 27). It can be assumed that effective SEU cross sections of hadrons above a certain energy do not differ significantly. The expected flux of fast hadrons at CBM is known from FLUKA simulations [Sen11]. Hence, simulated fast hadron flux can be used to give a rough estimate of the expected SEU rate at CBM.

Beam particles at the in-beam tests were also fast hadrons (2 GeV protons) and the SEU rate was measured. Nevertheless, the results from the in-beam tests do not directly translate to the expected behavior of the electronics in the operation scenario at CBM for the following reasons.

1. A much lower particle rate is expected for CBM. The fast hadron flux at the inner part of the ToF wall<sup>1</sup> is in the order of  $10^4 s^{-1} cm^{-2}$  [Sen11]. This is a factor of 500 less than the particle rate during the in-beam tests.
2. The above effect is somehow counterbalanced by the fact that the number of ROCs to be installed at CBM-ToF is around 300 to 400 (ToF-TDR assumes 336 [TOF, p. 59]), a factor in the same order of magnitude.

In consequence, the SEU rate measured during the in-beam tests with only one board more or less translates directly to the SEU rate expected for the whole CBM experiment with several hundreds of boards. However, there is more to consider:

3. The FPGAs to be used in the final experiment are likely Xilinx Series 7 (or later) devices. Due to smaller process technology, those devices have a slightly smaller bit cross section ( $\sim 7 \cdot 10^{-15} cm^2$ ) compared to the Virtex-4 FPGAs used for the in-beam tests of this thesis ( $\sim 1$  to  $2 \cdot 10^{-14} cm^2$ ). The numbers are from [Xil14b, page 27].
4. An SEU effect at CBM is not as severe concerning data quality as an SEU effect during the in-beam test was. During the in-beam tests, only one board was used. If data was corrupted this effected 100% of the boards (one out of one). The ToF-TDR assumes 336 boards to be operated simultaneously [TOF, p. 59]. If an SEU causes

---

<sup>1</sup>The inner part is the area with the highest radiation level of ToF. Unfortunately also the density of read-out channels is highest in this part of the detector, and hence cabling cost can be reduced best when read-out controllers can operate as close as possible to the inner region.

data corruption it will only affect one out of some hundred boards, the remaining boards will still deliver valid data.

5. The very high beam particle rate caused a much higher SEU rate **per device** than expected in the operation scenario of CBM. However, the time to repair the device remains the same. In consequence, the probability for multi-bit upsets in the device at CBM is much lower than it was during the in-beam tests.

While 1 and 2 more or less out balance each other, 3, 4, and 5 attenuate the expected radiation effects. In the following, only 3 and 4 are taken into account and the attenuation effect of 5 is reserved as safety margin.

## 7.2. Detector Dead Time Estimations Based on Parameters Measured at In-Beam Tests

Based on these numbers the implications on electronics at the ToF wall can be estimated, here for the inner region where the radiation level is highest.

**Expected SEU Rate at CBM-ToF** First the expected SEU rate at CBM-ToF is calculated, it is given by the following relation:

$$\text{SEU rate} = \text{device cross section} \cdot \text{fast hadron flux} \quad (7.1)$$

$$\text{SEU rate} = \overbrace{\text{No. of configuration bits} \cdot \text{bit cross section}} \cdot \text{fast hadron flux} \quad (7.2)$$

The required parameters are all available:

- *Number of configuration bits:* With a current GET4 read-out implementation for a Spartan-6 LX150T FPGA that is supporting 57 GET4 ASICs, the “number of occupied slices” is reported to be 36% of all available slices. To reduce cost, the size of the final ToF-ROC’s FPGA will most likely be chosen such that not much fabric resources remain unused, 80% occupied slices should be a realistic value.

A Spartan-6 LX150T firmware used to “full capacity” (80 %) would therefore support:  $57 \text{ GET4s} \cdot 80/36 \approx 126 \text{ GET4s}$ . The firmware for the Spartan-6 is about 4 MB or  $3.2 \cdot 10^7 \text{ bit}$ , this translates to  $2.5 \cdot 10^5 \text{ bit}$  per GET4.

The ToF-TDR describes a detector layout with 26 592 GET4s connected to 336 ROCs [TOF, p. 59], i.e. about 80 GET4s per ROC.

The number of configuration bits in the proposed design is therefore  $2 \cdot 10^7 \text{ bit}$  per ROC and  $6.8 \cdot 10^9 \text{ bit} \approx 7 \cdot 10^9 \text{ bit}$  for the whole experiment.

- *Bit cross section:* The bit cross section of Xilinx Series 7 devices is published in [Xil14b, page 27]. The numbers are  $6.99 \cdot 10^{-15} \text{ cm}^2$  (configuration memory) and  $6.32 \cdot 10^{-15} \text{ cm}^2$  (BRAM memory), it should be safe to assume  $7 \cdot 10^{-15} \text{ cm}^2$  for both.



- *Fast hadron flux*: FLUKA simulations prefigure a fast hadron flux at the inner region of the ToF wall of about  $10^4 \text{ cm}^{-2}\text{s}^{-1}$  [Sen11].

With these numbers, the expected SEU rate at CBM-ToF can now be calculated according to equation 7.2.

$$\text{SEU rate}_{(\text{per ROC})} = 2 \cdot 10^7 \cdot 7 \cdot 10^{-15} \text{ cm}^2 \cdot 10^4 \text{ s}^{-1} \text{ cm}^{-2} = 0.0014 \text{ s}^{-1} \quad (7.3)$$

$$\implies \text{every 12 minutes in a single ROC}$$

$$\text{SEU rate}_{(\text{whole ToF})} = 7 \cdot 10^9 \cdot 7 \cdot 10^{-15} \text{ cm}^2 \cdot 10^4 \text{ s}^{-1} \text{ cm}^{-2} = 0.49 \text{ s}^{-1} \quad (7.4)$$

$$\implies \text{every 2 seconds in one of the ROCs of the detector}$$

**Expected Error Rate at CBM-ToF** Not every SEU has an effect on the running hardware. With the results from the in-beam tests we can now estimate the error rate.

It should be noted that such factors can also be achieved with fault injection test (see e.g. section 5.3.2). However, the results of fault injection tests only cover SEUs in the static part of the configuration memory (PSMs, LUTs) but not in the dynamic part (FFs, Memory). Therefore, in-beam tests are mandatory to analyze the full spectrum of SEU effects. This is especially true for commercial off-the-shelf electronics where the internals are not known in full detail.

The 2012 in-beam test has better statistics since the test lasted longer and the particle rate was higher. However, the problem with the 2012 test is that only a stripped down version of the firmware was exposed to beam particles. For that reason the following considerations are based on the numbers from the 2013 in-beam test despite its fewer statistics. After all, the 2013 measurements are consistent with the results of 2012.

First we consider the case without scrubbing. We can see from figure 6.9, that without scrubbing, the average number of test procedure iterations until a system error is  $1/14.8\% = 1/0.148 \approx 6.8$ . One iteration takes about 7.5 seconds, the average lifetime can therefore be estimated to 51 seconds. In 51 seconds and with 0.61 SEUs/s, 31 SEUs are collected in the SEU Counter board. As described in section 6.3.3, the SEU rate in the DUT was only 70% of the SEU rate in the SEU Counter board. This means that in 51 seconds 21 SEUs have accumulated in the DUT.

Without scrubbing, the expected error rate at CBM-ToF is therefore:

$$\text{error rate}_{(\text{per ROC})}^{(\text{no scrubbing})} = \text{SEU rate}_{(\text{per ROC})} / 21 = 6.7 \cdot 10^{-5} \text{ s}^{-1} \quad (7.5)$$

$$\implies \text{every 4 hours in a single ROC}$$

$$\text{error rate}_{(\text{whole ToF})}^{(\text{no scrubbing})} = \text{SEU rate}_{(\text{whole ToF})} / 21 = 0.023 \text{ s}^{-1} \quad (7.6)$$

$$\implies \text{every 43 seconds in one of the ROCs of the detector}$$

When scrubbing is implemented, two error rates have to be distinguished, temporary errors and permanent errors. Note that in case of scrubbing enabled, data from figure 6.9 cannot be used to determine the rate of temporary errors. Figure 6.9 does not show

those errors that have already been fixed silently by scrubbing during early steps of the test procedure before Test DUT is reached. However, it can safely be assumed that with scrubbing enabled, temporary errors occur at the same probability as the error rate that is calculated above for the case without scrubbing.

$$\begin{aligned} \text{temporary-error rate}_{(\text{per ROC})}^{(\text{scrubbing})} &= \text{SEU rate}_{(\text{per ROC})} / 21 = 6.7 \cdot 10^{-5} \text{ s}^{-1} & (7.7) \\ &\implies \text{every 4 hours in a single ROC} \end{aligned}$$

$$\begin{aligned} \text{temporary-error rate}_{(\text{whole ToF})}^{(\text{scrubbing})} &= \text{SEU rate}_{(\text{whole ToF})} / 21 = 0.023 \text{ s}^{-1} & (7.8) \\ &\implies \text{every 43 seconds in one of the ROCs of the detector} \end{aligned}$$

Permanent errors, on the contrary, can be deduced from figure 6.9 as they are not removed by scrubbing. With 0.139 % of the iterations showing permanent errors, the average number of iterations until a permanent error occurs is about 720. This means the DUT sustained an average of about 2300 SEUs until it failed. When scrubbing is enabled, the expected rate of permanent errors at CBM-ToF is therefore:

$$\begin{aligned} \text{permanent-error rate}_{(\text{per ROC})}^{(\text{scrubbing})} &= \text{SEU rate}_{(\text{per ROC})} / 2300 = 6.1 \cdot 10^{-7} \text{ s}^{-1} & (7.9) \\ &\implies \text{every 19 days in a single ROC} \end{aligned}$$

$$\begin{aligned} \text{permanent-error rate}_{(\text{whole ToF})}^{(\text{scrubbing})} &= \text{SEU rate}_{(\text{whole ToF})} / 2300 = 2.1 \cdot 10^{-4} \text{ s}^{-1} & (7.10) \\ &\implies \text{every 80 minutes in one of the ROCs of the detector} \end{aligned}$$

Unfortunately, the factor of 2300 is based on only one single event in the 2013 in-beam test, when a permanent error was measured with scrubbing enabled. The uncertainty of the factor is accordingly high.

**Expected Radiation Induced Dead Time of CBM-ToF Electronics** To determine the fraction of time in which an error is disturbing the setup one needs to divide the time required to repair an error by the average time between the occurrence of two errors. A temporary error will exist only for a short time (in the order of 100 ms) until it is corrected by *scrubbing* whereas a permanent error persists until the device is reset externally (probably for a few seconds if an intelligent error detection mechanism is implemented).

$$\begin{aligned} \text{dead time (without scrubbing)} &= 3 \text{ seconds every 43 seconds} \\ &\implies \text{in total: 7\%} \\ \text{dead time (with scrubbing)} &= 100 \text{ milliseconds every 43 seconds and} \\ &\quad 3 \text{ seconds every 80 minutes} \\ &\implies \text{in total: 0.3\%} \end{aligned}$$

These values are based on the assumption, that the whole detector is down when a single error occurs. This is not the case as an error affects only one out of 336 boards

(see point 4 in section 7.1). For physics cases that do not require a complete picture of the detector these values can be further reduced by a factor of 336. Dead time without scrubbing is then  $\sim 0.02\%$  and dead time with scrubbing  $\sim 0.0009\%$ .

**Expected Radiation Induced Data Corruption at CBM-ToF** The expected data quality for CBM-ToF would be very much the same as shown in figure 6.10 because points 1 and 2 in section 7.1 counterbalance each other.

$$\begin{aligned}\text{corrupted data (without scrubbing)} &= 3 - 4\% \\ \text{corrupted data (with scrubbing)} &= 0.1\%\end{aligned}$$

As before, corrupted data from a single ROC does not necessarily render the data from the other ROCs useless. Therefore, depending on the physics case, the expected percentage of corrupted data has to be further reduced by a factor of 336 (the number of installed ROCs). This results in about  $0.01\%$  of the data to be corrupted without scrubbing, and only about  $0.0003\%$  with scrubbing.

**Impact on CBM-ToF Strategy** The conceptual design for CBM-ToF consists of six different modules, organized in an “inner wall” ( $4.3\text{ m} \times 3\text{ m}$ ), modules M1 to M3) and an “outer wall” (rest of the  $12\text{ m} \times 9\text{ m}$  wall, modules M4 to M6) [DHA<sup>+</sup>14]. The results look promising and suggest to put FPGAs close to the inner region. The ToF-TDR foresees SRAM-based read-out electronics directly on the modules of the “outer wall”. For the read-out of the “inner wall”, however, SRAM-based electronic will be placed outside the area of the “inner wall” [TOF, p. 18]. The results of this thesis already provided important input to this decision. The ToF-TDR [TOF] references [Mül14] in this context, and the information from [Mül14] is based on assessments from the present work. Using FPGAs in such an harsh radiation environment would possibly have been rejected without this research.

As a backup strategy, the radiation level can be lowered further by a factor of 10 when moving the electronics one or two meters farther away [Sen11]. Of course, this would come with the drawback of higher cost due to more cabling. However, given the results of the present work, it seems unlikely that the backup strategy is required. One might rather consider the usage of FPGAs in zones with even higher radiation levels instead.



## 8. Conclusion

This chapter summarizes the achievements of this thesis and gives an outlook for future work that remains to be done before an FPGA based CBM-ToF read-out chain can be put into service.

### 8.1. Summary

The present work addresses three closely related topics that are now summarized individually.

#### 8.1.1. Implementation of the CBM-ToF Read-Out Controller Firmware

A read-out controller firmware for the GET4 ASIC was implemented at the beginning of this thesis and was continuously enhanced throughout the years. Five different prototypes of the GET4 ASIC with an increasingly complex feature set were successfully read out. The read-out firmware was not a pure prove-of-concept implementation for a laboratory setup but it was also operationally used by the ToF detector group during several in-beam tests to characterize their detector prototypes.

#### 8.1.2. Evaluation and Implementation of Radiation Mitigation Techniques

When evaluating existing radiation mitigation techniques, a compromise between flexibility, reliability and expenses had to be found. After ruling out some strategies, such as temporal redundancy or Hamming-coded finite state machines, and choosing TMR as one of the main pillars for radiation mitigation, the major problem that remained was the huge resource consumption of TMR.

The special circumstance that 100% reliability cannot be guaranteed due to very high radiation levels but on the other hand 100% reliability is also not required (as long as contributions to detector dead time are not significant) led to the idea of *Selective TMR*. With *Selective TMR*, only the control logic is implemented with triple modular redundancy (TMR). Logic for handling data, which comprises the major part of the firmware, is not TMR protected but data integrity is protected by a CRC checksum instead. The huge demand of fabric resources entailed by conventional full TMR can be attenuated very well with this approach. The resource consumption of the radiation mitigated firmware grew by a factor between 150% and 200% compared to the unmitigated firmware (depending on the resources, LUTs or flip-flops) instead of a factor of  $\sim 600\%$  that is found in literature for full TMR.

### 8.1.3. In-Beam Verification of the Implemented Radiation Mitigation Techniques

It was not clear, if the implemented radiation mitigation techniques perform sufficiently well, despite the abovementioned compromises in the implementation. Also challenging are the complexity of the firmware and the expected high particle rate at CBM. The efficiency of the implementation was therefore shown in two in-beam verified by mounting the FPGA directly into a beam of accelerated protons.

The results of these tests provided valuable information to better estimate the failure rate of SRAM-based FPGAs for the case that they are used in the early stage of the ToF detector read-out chain. It can now be assumed, that the usage of SRAM-based FPGAs without mitigation would increase detector dead time in the order of a few percent. This is already a significant contribution to the detector dead time and cannot be tolerated.

Fortunately, the usage of *scrubbing* performs very well, even at a very high particle flux, on a highly complex design, and with only part of the design is covered by TMR. In the CBM-ToF use case, contributions of radiation caused electronics failure to detector dead time can be reduced by more than one order of magnitude.

However, when expecting erroneous behavior, even if only in rare occasions, one must not forget to design higher system levels (e.g. communication protocols) in a way that they can tolerate temporal erroneous behavior of radiation-exposed devices.

In the end, if the firmware is designed carefully and *scrubbing* is used, the usage of SRAM-based FPGAs at CBM-ToF is possible. Therefore, based on the assessments from the present work, the current plans for CBM-ToF are foresee SRAM-based FPGAs in the early stage of the read-out chain.

## 8.2. Outlook

The work on this thesis is based on a fully functional detector read-out chain. However, the final setup that will be implemented at CBM eventually will look different than the setup of today and even more different than the setup as it existed at the time the work for this thesis started. This section gives an overview of tasks that have to be addressed in future work.

### 8.2.1. Fault Tolerant Communication Module

The current work is based on CBMNet technology. However, with the redesign of the CBM magnet, more space for STS electronics became available. This has an impact on all CBM read-out electronics. CERN's more mature *GBT* project [MMK07] will be used as a drop-in replacement for CBMNet, future CBM read-out chains will be based on *GBT* technology rather than on CBMNet technology. The transition from CBMNet to *GBT* technology requires a lot of reimplementation of logic and even redefinition of protocols.

### 8.2.2. SEU Mitigation in Xilinx Series 7 FPGAs

Because *scrubbing* is such a successful strategy, Xilinx recently introduced in their series 7 devices an on-chip *scrubbing* controller as a new feature [Xil14a]. If not more than one SEU is present per frame<sup>1</sup>, it can be corrected automatically by the chip itself. As the correction of a single bit error is based on error correction codes, no additional memory is required for that task. External action is only required in case of a multi-bit error, the chip then indicates the error and the referring frame number via a configuration interface and waits for external reconfiguration of the corrupted frame. Only for correction of multi-bit upsets, which are much less frequent than single bit errors, a memory device that stores the original configuration is required.

In conjunction with the JTAG feature of the GBT-SCA technology, this allows to perform *scrubbing* on a much more elaborate level. As external action is only required for the rare event of a multi-bit upset, the action can be executed from outside the radiation zone. Without on-chip *scrubbing*, the performance of GBT-SCA JTAG would be too slow for efficient *scrubbing*, it would then not be feasible to scrub over the long distance from outside the CBM cave.

From CBM perspective, the major advantage of on-chip *scrubbing* is the missing requirement for on-board Flash memory. The usage of Flash technology in the radiation zone was one of the greatest concerns regarding the configuration system used to perform *scrubbing* so far. Flash memory is known to suffer severely from total ionizing dose effects (see section 2.2.1 and paragraph “Cumulative Radiation Effects” in 2.4.1), they will eventually stop working after they have been operated in a radiation environment too long.

The Xilinx series 7 on-chip *scrubbing* feature solves this problem. External reconfiguration is required very rarely and can now be executed from outside the CBM cave over the GBT-SCA JTAG feature. In consequence, the memory holding the original FPGA configuration is not exposed to radiation at all.

First efforts of putting on-chip *scrubbing* of a Xilinx series 7 device into service has already been started by group member Andrei Oancea, and he will also continue his work on this topic.

### 8.2.3. Resilience

Device failures are rare events but we cannot fully prevent them. Therefore it is important to define the situations under which a full device reset is necessary. The test procedure used during the in-beam tests includes a software based data integrity test (see figure 5.9). If the test fails twice, the setup is reset. For the in-beam test setup, this was sufficient. However, for the final CBM experiment with several hundreds of read-out controller boards, data integrity checks cannot be done in software. For that purpose it is required to have a monitor entity outside the radiation zone (e.g. in the DPB). This monitor entity

<sup>1</sup>The configuration memory of Xilinx FPGAs is logically organized in smaller units, called *frames*.

decides whether a device is working properly or not. Therefore, it needs to analyze the data quality, implement watchdog functionality, etc.

A further problem occurs when the design finally has been reset. It then needs to be resynchronized to the global time value of the running system. The current concept that is used by the CBM-ToF group for read-out of detector prototypes and that was used as well for the in-beam tests in the present work depends on a global, simultaneous reset of all time counters in all devices. This concept works sufficiently well for setups with few boards, but it does not scale to several hundreds of boards. For the final setup at CBM, we need a concept for re-synchronization of a single board into the running experiment setup. Basic ideas have already been considered in the current system, the time stamp counters in GET4 ASICS can be set via slow control to arbitrary values and will then start counting upon reception of a signal that is synchronously and periodically distributed on a global scale. For his diploma thesis Johannes Lehrbach has already implemented a proof of concept design that automatically synchronizes the GET4 time to the ROC time [Leh13]. Such a concept needs to be implemented for the ROC time stamp counters as well. The global, synchronous signal can be implemented with so called “deterministic latency messages” or DLMs (see section B.2.3) in case of CBMNet based setups. Similar functionality is also available on GBT based systems where it goes by the name “Timing Trigger and Control” or TTC [MMK07].



## 9. Acknowledgments

I want to close this thesis by giving credit where credit is due. It is not possible to carry out a project at the scale of this thesis without support from other people, I would like to thank everybody who, directly or indirectly, supported me in the process.

First, I would like to thank Prof. Dr. Udo Keschull for giving me the opportunity to work on this exciting project and for sharing his valuable ideas. He managed to find the right balance between guiding me and giving me all the freedom I needed to realize my own ideas.

I also want to thank Andrei Oancea and Johannes Lehrbach who were brave enough to carry out their diploma theses in the field of the CBM-ToF read-out controller. In doing so, both contributed significantly to my work. Heiko Engel did some very valuable preparatory work before I started and helped me a lot with his expertise later on. Furthermore, I also want to thank the rest of my colleagues of the IRI working group for the great working atmosphere and all the valuable feedback. Some of my colleagues are not only great companions at work but real friends.

For the great work atmosphere in the collaboration, I also want to thank all of my colleagues at CBM, namely Walter Müller, Dirk Hutter, Jan de Cuveland, Sven Löchner, Jochen Frühauf, Pierre-Alain Loizeau, Christian Simon, Ingo Deppner, Frank Lemke, Sven Schatral, and many more. I learned a lot from working as part of a collaboration consisting of so many highly skilled people.

Special thanks also to Frederik Grüll, Andrei Oancea, Dirk Hutter, Heiko Engel, Stefan Boettger, Norbert Abel, Cruz Garcia, Hanna Zatschler, Julia Zatschler, Harf Zatschler, and especially Stefan Kirsch for proofreading.

Additional thanks go to Norbert Abel and Jano Gebelein for organizing so much of the administrative tasks that are - unfortunately - necessary for the completion of this kind of work, I appreciate their support very much.

The support from my family shall not remain unmentioned, however, I do not need to spend more words, they know they are great.

And last but definitely not least, Julia, thanks for being as great as you are.



# Appendix A.

## List of Publications

### A.1. As Main Author

#### A.1.1. Peer-Reviewed

- MANZ, S., ABEL, N., GEBELEIN, J. and KEBSCHULL, U. “*A Universal Read-Out Controller*”, Journal of Instrumentation, Proceedings of TWEPP Conference, Aachen, **2010**
- MANZ, S., GEBELEIN, J., OANCEA, A., ENGEL, H. and KEBSCHULL, U. “*Radiation Mitigation Efficiency of Scrubbing on the FPGA Based CBM-ToF Read-Out Controller*”, Proceedings of FPL Conference, Porto, **2013**

#### A.1.2. At Conferences

- MANZ, S. “*Status of the Read Out Controller for the FEET Boards*”, 13<sup>th</sup> CBM Collaboration Meeting, Darmstadt, Germany, **March 2009**
- MANZ, S. “*Status of the Readout Controller for the GET4 Chip*”, 14<sup>th</sup> CBM Collaboration Meeting, Split, Croatia, **October 2009**
- MANZ, S. and KEBSCHULL, U. “*Design and Implementation of the Read-Out-Controller for the GET4 chips*”, DPG Conference, (HK 48.6), Bonn, Germany, **March 2010**
- MANZ, S. “*A Modular Read-Out Controller*”, 16<sup>th</sup> CBM Collaboration Meeting, Mamaia, Romania, **September 2010**
- MANZ, S. and KEBSCHULL, U. “*The Universal Read-Out Controller for CBM at FAIR*”, DPG Conference, (HK 20.6), Münster, Germany, **March 2011**
- MANZ, S. “*The ToF read-out controller operating in a radiation environment - Results of the beamtimes at COSY in 2010*”, 17<sup>th</sup> CBM Collaboration Meeting, Dresden-Rossendorf, Germany, **April 2011**
- MANZ, S. “*Open Issues in Dealing with Radiation Effects in the CBM Readout Chain*”, 18<sup>th</sup> CBM Collaboration Meeting, Beijing, China, **September 2011**

- MANZ, S., ERDMANN, R., SCHATRAL, S., OANCEA A. and LÖCHNER, S. “*Front-End Electronics Radiation Tests 2012*”, 20<sup>th</sup> CBM Collaboration Meeting, Kolkata, India, **September 2012**
- MANZ, S. and KEBSCHULL, U. “*Modular CBM-ROC Firmware - Was bisher geschah und wie es weitergeht*”, DPG Conference, (HK 34.6), Dresden, Germany, **March 2013**
- MANZ, S. “*Modular CBM-ROC Firmware*”, 21<sup>th</sup> CBM Collaboration Meeting, Darmstadt, Germany, **April 2013**
- MANZ, S. “*Updates on the GET4 Read-out Firmware*”, 22<sup>th</sup> CBM Collaboration Meeting, Dubna, Russia, **September 2013**
- MANZ, S. “*GET4-ROC on SysCore-V3*”, 23<sup>th</sup> CBM Collaboration Meeting, Darmstadt, Germany, **April 2014**
- MANZ, S. “*Status ROC Firmwares for Upcoming Beamtimes*”, 24<sup>th</sup> CBM Collaboration Meeting, Krakow, Poland, **September 2014**
- MANZ, S. and KEBSCHULL, U. “*Abschätzung der strahlungsbedingten Fehlerrate in der CBM-ToF Frontendelektronik*”, DPG Conference, Heidelberg, Germany, **March 2015**

### A.1.3. Others

- MANZ, S. and KEBSCHULL, U. “*Design and Implementation of the Read Out Controller for the GET4 TDC of the CBM ToF Wall Prototype*”, CBM Progress Report 2009, **2010**
- MANZ, S. and KEBSCHULL, U. “*Design and Implementation of the Read Out Controller for the GET4 TDC of the CBM ToF Wall Prototype*”, GSI Scientific Report 2009, **2010**
- MANZ, S., GEBELEIN, J., OANCEA, A., ENGEL, H. and U. KEBSCHULL “*ToF-ROC FPGA Irradiation Tests 2012*”, CBM Progress Report 2012, **2013**
- MANZ, S., GEBELEIN, J., OANCEA, A., ENGEL, H. and U. KEBSCHULL “*ToF-ROC FPGA Irradiation Tests 2012*”, GSI Scientific Report 2012, **2013**
- MANZ, S., OANCEA A., GEBELEIN, J., SCHATRAL, S. and KEBSCHULL, U. “*GET4-ROC - Research and Development in 2013*”, CBM Progress Report 2013, **2014**
- MANZ, S. and KEBSCHULL, U. “*Maintenance of read-out controller firmwares for GET4 and NXYTER chips*”, CBM Progress Report 2014, **April 2015**

## A.2. As Coauthor

### A.2.1. Notable

- ABEL, N., MANZ, S., GRÜLL, F. and KEBSCHULL, U. *“Increasing Design Changeability Using Dynamical Partial Reconfiguration”*, IEEE Transactions on Nuclear Science, Proceedings of RT conference, Beijing, China, **2009**
- GEBELEIN, J., MANZ, S., ENGEL H., ABEL, N. and KEBSCHULL, U. *“Smart Module Redundancy - approaching cost efficient radiation tolerance”*, to appear in *“Transforming Reconfigurable Systems: A Festschrift Celebrating the 60th Birthday of Professor Peter Cheung”*, Imperial College Press, **2015**

### A.2.2. Others

- ABEL, N., MANZ, S. and KEBSCHULL, U. *“Design and Implementation of an Universal Read Out Controller”*, CBM Progress Report 2009, **2010**
- ABEL, N., MANZ, S. and KEBSCHULL, U. *“Design and Implementation of an Universal Read Out Controller”*, GSI Scientific Report 2009, **2010**
- LEMKE, F., MANZ, S. and GAO, W. *“Time synchronization and measurements of a hierarchical DAQ network”*, DPG Conference, Bonn, Germany, **March 2010**
- LOIZEAU, P.-A., HERRMANN, N., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KOCH, K., LINEV, S., MANZ, S. and MÜLLER, W. *“A demonstrator for the CBM Time Of Flight wall electronic readout chain”*, DPG Conference, Bonn, Germany, **March 2010**
- LOIZEAU, P.-A., HERRMANN, N., DEPPNER, I., WISNIEWSKI, K., XIANG, C., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KOCH, K. and MANZ, S. *“Status of the analysis chain for CBM-TOF demonstrator data”*, CBM Progress Report 2010, **2011**
- FRÜHAUF, J., HERRMANN, N., DEPPNER, I., LOIZEAU, P.-A., XIANG, C., WISNIEWSKI, K., CIOBANU, M., FLEMMING, H., DEPPE, H., KOCH, K. and MANZ, S. *“Status of the CBM-TOF-Readout-Chain”*, CBM Progress Report 2010, **2011**
- ABEL, N., ENGEL, H., GEBELEIN, J., GOTTSCHALK, D. MANZ, S., OANCEA, A. and KEBSCHULL, U. *“Radiation Tolerance of the Universal Read Out Controller”*, CBM Progress Report 2010, **2011**
- ABEL, N., ENGEL, H., GEBELEIN, J., GOTTSCHALK, D. MANZ, S., OANCEA, A. and KEBSCHULL, U. *“Radiation Tolerance of the Universal Read Out Controller”*, GSI Scientific Report 2010, **2011**

- LOIZEAU, P.-A., HERRMANN, N., DEPPNER, I., WISNIEWSKI, K., XIANG, C., AD-AMCZEWSKI-MUSCH, J., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KOCH, K., LINEV, S. and MANZ, S. *"In beam test of CBM-TOF electronics chain"*, CBM Progress Report 2011, **2012**
- XIANG, C., HERRMANN, N., DEPPNER, I., LOIZEAU, P.-A., WISNIEWSKI, K., ZHANG, Y., FRÜHAUF, J., LINEV, S. and MANZ, S. *"FPGA based control and monitor for CBM-TOF readout"*, CBM Progress Report 2011, **2012**
- ABEL, N., GARCIA, C., GEBELEIN, J., MANZ, S. and KEBSCHULL, U. *"SysCore3 - a new board for the Universal ROC"*, CBM Progress Report 2011, **2012**
- LOIZEAU, P.-A., HERRMANN, N., DEPPNER, I., WISNIEWSKI, K., XIANG, C., AD-AMCZEWSKI-MUSCH, J., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KOCH, K., LINEV, S. and MANZ, S. *"In beam test of CBM-TOF electronics chain"*, GSI Scientific Report 2011, **2012**
- XIANG, C., HERRMANN, N., DEPPNER, I., LOIZEAU, P.-A., WISNIEWSKI, K., ZHANG, Y., FRÜHAUF, J., LINEV, S. and MANZ, S. *"FPGA based control and monitor for CBM-TOF readout"*, GSI Scientific Report 2011, **2012**
- ABEL, N., GARCIA, C., GEBELEIN, J., MANZ, S. and KEBSCHULL, U. *"SysCore3 - a new board for the Universal ROC"*, GSI Scientific Report 2011, **2012**
- LOIZEAU, P.-A., HERRMANN, N., DEPPNER, I., SIMON, C., XIANG, C., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KIŠ, M., KOCH, K., LINEV, S., MANZ, S. and the CBM TOF WORKING GROUP *"Status of the CBM TOF free streaming electronics chain"*, CBM Progress Report 2012, **2013**
- XIANG, C., HERRMANN, N., LOIZEAU, P.-A., DEPPNER, I., MANZ, S., FRÜHAUF, J. and LINEV, S. *"The online data pre-processing for CBM-TOF"*, CBM Progress Report 2012, **2013**
- DEY, M. ROY, A., CHATTOPADHYAY, S., ABEL, N., MANZ, S. and GOTTSCHALK, D. *"Development and production of the ROC SysCore board V2.2"*, CBM Progress Report 2012, **2013**
- LOIZEAU, P.-A., HERRMANN, N., DEPPNER, I., SIMON, C., XIANG, C., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KIŠ, M., KOCH, K., LINEV, S., MANZ, S. and the CBM TOF WORKING GROUP *"Status of the CBM TOF free streaming electronics chain"*, GSI Scientific Report 2012, **2013**
- XIANG, C., HERRMANN, N., LOIZEAU, P.-A., DEPPNER, I., MANZ, S., FRÜHAUF, J. and LINEV, S. *"The online data pre-processing for CBM-TOF"*, GSI Scientific Report 2012, **2013**

- OANCEA, A., MANZ, S., ENGEL, H., GEBELEIN, J. and KEBSCHULL, U. "*Entwicklung einer fehlertoleranten Konfigurations- und Scrubbing-Kette für den CBM Read-Out Controller (ROC)*", DPG Conference,(HK 34.5), Dresden, Germany, **March 2013**
- LOIZEAU, P.-A., HERRMANN, N., DEPPNER, I., SIMON, C., XIANG, C., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KIŠ, M., KOCH, K., LINEV, S., MANZ, S. and the CBM TOF WORKING GROUP "*Characterization of the GET4 v1.0 TDC ASIC with detector signals*", CBM Progress Report 2013, **2014**
- OANCEA, A., GEBELEIN, J., MANZ, S. and KEBSCHULL, U. "*Firmware Development for the SysCore v3.1 Configuration Controller*", CBM Progress Report 2013, **2014**
- XIANG, C. Z., HERRMANN, N., DEPPNER, I., LOIZEAU, P.-A., WISNIEWSKI, K., ZHOU, D. C., ZHANG, Y. P., FRÜHAUF, J., LINEV, S., MANZ, S. and MÜLLER, W. F. J. "*The online data pre-processing for CBM-TOF*", Journal of Instrumentation (JINST), Volume 8, **February 2013**
- LÖCHNER, S., FRÜHAUF, J., GRAF, H., HOFFMANN, J., KOCZON, P., MANZ, S. and WITTHAUS, M. "*CBM proton beam test for electronic components*", CBM Progress Report 2013, **2014**
- OANCEA, A., GEBELEIN, J. and MANZ, S. "*Electronics Irradiation Tests 2014 at COSY*", 24<sup>th</sup> CBM Collaboration Meeting, Krakow, Poland, **September 2014**
- LOIZEAU, P.-A., HERRMANN, N., DEPPNER, I., SIMON, C., XIANG, C., CIOBANU, M., DEPPE, H., FLEMMING, H., FRÜHAUF, J., KIŠ, M., KOCH, K., LINEV, S., MANZ, S. and the CBM TOF WORKING GROUP "*Characterization of the GET4 v1.0 TDC ASIC with detector signals*", GSI Scientific Report 2013, **2014**
- DEPPNER, I. et al. "*The CBM Time-of-Flight wall - a conceptual design*", Journal of Instrumentation (JINST), Volume 9 **October 2014**





## Appendix B.

# Documentation for the GET4 Read-Out Controller

This chapter is part of the documentation for the GET4 read-out controller firmware that was written during the development of the firmware. It describes some basic design concepts and the functionality of the *Modular-ROC* firmware.

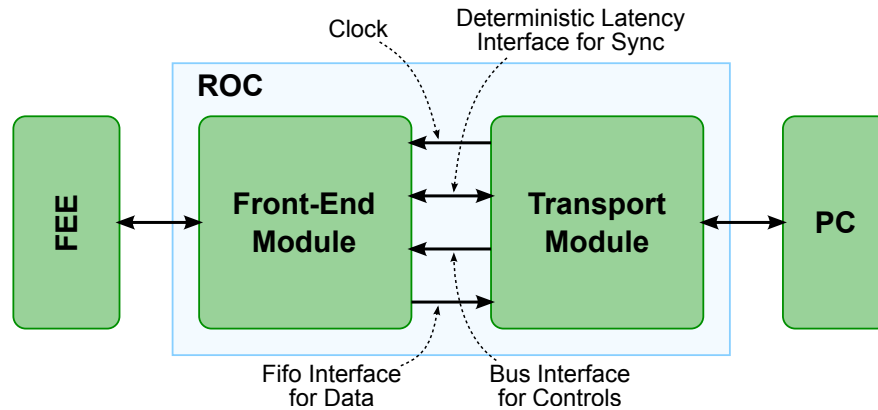
The firmwares that have been used during the in-beam experiments are derived from an early version of the *Modular-ROC* firmware that is based on a two-modules concept. Since the most important results of this thesis came out of those in-beam experiments the documentation chosen to be shown here has been taken from this early *Modular-ROC* firmware. As the development on the GET4 read-out controller continued, it is already slightly outdated at time of writing. Today, a version that implements three modules is used in the CBM community and for the future an even finer modularization is planned.

### B.1. Introduction

The CBM experiment at FAIR requires the readout of multiple detector front end electronics. This is usually done with special readout controller (ROC) boards where each detector assembles different frontend electronics, requiring different ROCs. However, the interface in the other direction of the readout chain is usually quite the same. This led to the idea of a modular ROC design, which separates the readout logic from the transport logic. Much functionality can be reused and only the interface to the frontend electronics needs to be exchanged. In addition, this modular approach allows an efficient development of the firmware by more than one designer. The frontend and the transport module can be developed separately by different developers.

This document is intended to describe the firmware of the ROCs. For a closer look at the actual hardware please refer to <http://cbm-wiki.gsi.de/cgi-bin/view/NXYTER/SysCoreV2>.

In chapter B.2 we will describe some basic design concepts of the ROC firmware and functionality which is relevant for *all* combinations of the modules. Then in chapter B.3 the module-specific details are explained. Chapter C is intended as a reference work and



**Figure B.1.:** Schematic overview of the modular approach for the ROC design.

provides an overview of all available slow control addresses, including a short explanation for each address.

## B.2. Basic Functionality

Some ROC functionality is crucial for the design of the CBM readout chain, this functionality needs to be provided by all firmwares for the CBM-ROC. Most important, the modules have to share a common interface.

The interface is separated in three different classes similar to the CBMNet traffic classes: a bus interface to transport slow control commands, a FIFO interface for data transport and a deterministic latency interface for synchronization using so-called DLM messages.

The common functionality that is important for all modules shall be described in this chapter. The chapter is thereby structured according to the three different classes.

### B.2.1. Slow Control

#### GET/PUT Commands and Multi Oper Lists

The ROC can be configured with GET and PUT commands, GET commands are 32 bit wide, PUT commands consist of 64 bit. Those commands are part of what is known as *Multi-Oper-List protocol* which defines seven commands in total:

CMD	Hex code	Bin code	Comment
NOP	0x00	00000000	(32 bit, not acked)
N	0x01	00000001	(32 bit, not acked)
PUT	0x02	00000010	(64 bit)
GET	0x04	00000100	(32 bit)
ACK-PUT	0x0A	00001010	(64 bit)
ACK-GET	0x0C	00001100	(64 bit)
NACK	0x10	00010000	(64 bit)

They are either 32 bit or 64 bit wide, the meaning of the bits is illustrated in figure B.2.

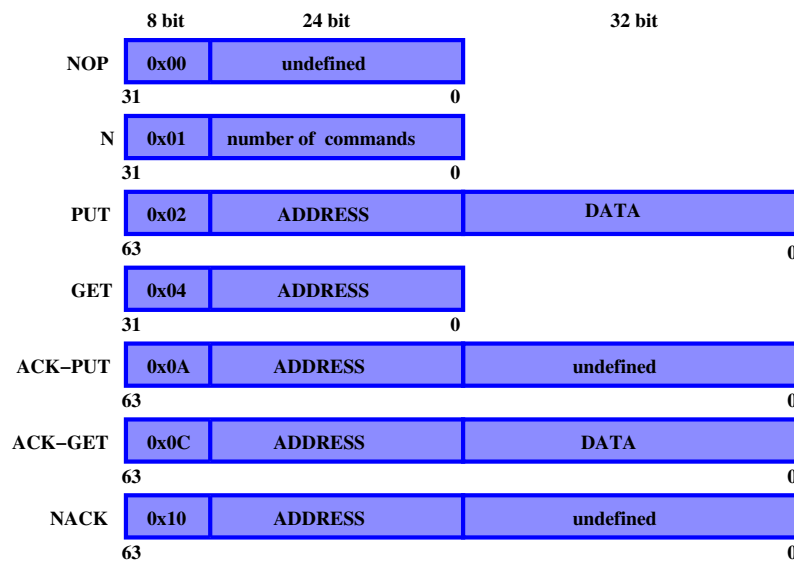


Figure B.2.: Bit organization of the GET/PUT commands.

The multi oper lists are build by using these commands. The following rules apply:

- The first command of a multi oper list is N. N defines the number of commands (GET/PUT/NOP) that will follow in the list.
- A multi oper list is always transmitted in one *CBMNet* packet and therefore, due to the 64 byte limitation of *CBMNet*, a maximum of 7 commands per multi oper list is possible.<sup>1</sup>
- Since *CBMNet* requires the granularity on the control path to be 64 bit, there is a 32-bit NOP padded at the end if necessary. Note that the NOP also counts for the referring N command.

<sup>1</sup>Note that in software you can issue multi oper lists with more than 7 commands. But they will be split to multiple multi oper lists by the *ABBdaemon* (the driver for the PCI express board). So if you need to ensure that the commands in your multi oper list are executed within a very short time period, do not use more than 7 commands or use the *cmd-lists* feature.

- All commands of a multi oper list are executed and all the referring ACKs/NACKs are send back in one packet.
- Since the number of expected ACKs/NACKs is known, the replying packet does not contain the preceding N command.
- Since the ACKs/NACKs are all 64 bit, it does not contain a NOP either.
- A NOP will not cause a ACK or NACK.
- If one command fails, a NACK will be replied. All subsequent commands of the multi oper list will not be executed and a NACK will be replied for them.
- A single GET or PUT command is just a multi oper list with  $N = 1$ .

### Slow Control Addresses

The address space of the *Modular ROC* is divided in several address subspaces. One subspace for each module and one subspace for common functionality (see section C for all addresses).

The following address subspaces are defined:

- 0x0XXXXX: common functionality
- 0x1XXXXX: Ethernet specific functionality
- 0x2XXXXX: Optics specific functionality
- 0x3XXXXX: reserved
- 0x4XXXXX: nXYTER specific functionality
- 0x5XXXXX: FEET/GET4 specific functionality

**Hardware Identification** To identify which combination of frontend module and transport module is present, address 0x 00 00 00 returns a 32 bit value. The upper 16 bit indicate the type of frontend module and the lower 16 bit the type of transport module that is used.

The following values are defined:

- Frontend Module:
  - 0x0001: The nXYTER readout.
  - 0x0002: The FEET readout (old GET4s). Support dropped.
  - 0x0003: The GET4 v1.x readout.
- Transport Module:
  - 0x0001: Transport via Optics
  - 0x0002: Transport via Ethernet (for the Virtex4-FX20 FPGA)

- 0x0003: Transport via Ethernet (for the Virtex4-FX40 FPGA)
- 0x0004: Transport via Ethernet (for the Virtex4-FX60 FPGA)
- 0x0005: Transport via USB

So, for example the firmware for the readout of the GET4 v1.x chips via Optics would return: 0x0003 0001.

### Hardware Versioning

The mostly independent development of different modules requires an independent versioning of the different modules. In each module, address 0x X0 00 00 must return it's hardware version. An exception is the address subspace of the common functionality since here 0x 00 00 00 is already used to identify the combination of frontend module and transport module. 0x 00 00 04 is used for the version of the interface between the two modules.

### Command Lists

Eight programmable *Command Lists* are available on the ROC. They can be used for command sequences that need to be repeated frequently, and for time critical command sequences. Time critical means that the commands in the list are executed within a few clock cycles but also that execution of the lists can be started by DLMS, i.e. on multiple ROCs simultaneously.

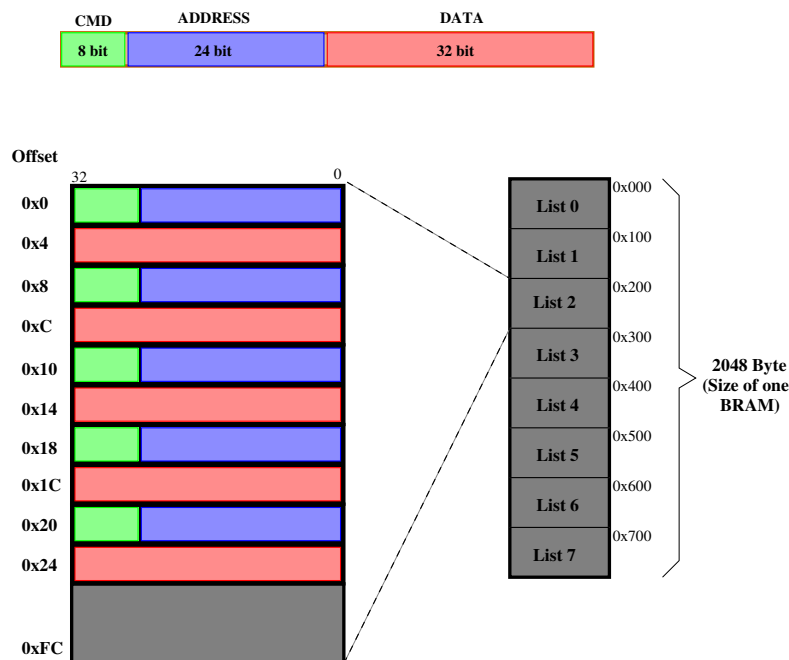


Figure B.3.: CMD-Lists memory (2kB) is mapped into PUT/GET-address-space

- There are 8 Lists available, each list provides a maximum of 32 commands
- Execution of a list can be issued by a PUT command, alternatively, DLMs 8-15 trigger list 0-7 respectively.
- GET commands can also be programmed but are of limited use as ACKs/NACKs of list-commands are not stored
- In case of GET, proceed with 64 bit alignment after the GET command.
- The last CMD has to be the command to deactivate list processing ...
- ... except if you want to chain lists (careful)
- Predefined list commands:
  - List 0: Start DAQ, then deactivate list processing
  - List 1: Stop DAQ, then deactivate list processing
  - Lists 2-7: just deactivate list processing

### Example

rocutil-command	what happens
<b>Example sequence (initialize a GET4 chip and send back a DLM):</b>	
put 0x501000 0x8	enable readout from port 4 (8="1000", 4th bit enables 4th port)
put 0x510000 0x0	Send command to GET4: <i>activate dll and ro-rst</i>
put 0x510000 0x10	Send command to GET4: <i>enable GET4 readout</i>
put 0x201000 0xA	Generate a DLM with pattern "1010" (=0xA)

### Commands to store the above sequence in CMD-List No. 5:

put 0x021500 0x02501000	Store CMD and ADR of 1st cmd
put 0x021504 0x8	Store DATA of 1st cmd
put 0x021508 0x02510000	Store CMD and ADR of 2nd cmd
put 0x02150C 0x0	Store DATA of 2nd cmd
put 0x021510 0x02510000	Store CMD and ADR of 3rd cmd
put 0x021514 0x10	Store DATA of 3rd cmd
put 0x021518 0x02201000	Store CMD and ADR of 4th cmd
put 0x02151C 0xA	Store DATA of 4th cmd
put 0x021520 0x02020004	Store CMD and ADR of "deactivate list-processing"-cmd
put 0x021524 0x0	Store DATA of "deactivate list-processing"-cmd

### Select list No. 5 and activate processing:

put 0x020000 0x5            Activate list No. 5

(An incoming DLM ("1101") will trigger the processing of list No. 5 as well.)

---

### B.2.2. Data Readout

Besides hit data, some other important messages are send over the 48 bit wide data path. Those messages are Epoch Messages for bandwidth efficient time representation, Sync/Aux Messages to merge external events like beam-on/beam-off in the data stream, and System Messages to merge internal events like FIFO reset in the data stream.

Figure B.4 shows the bit arrangement of the different messages.

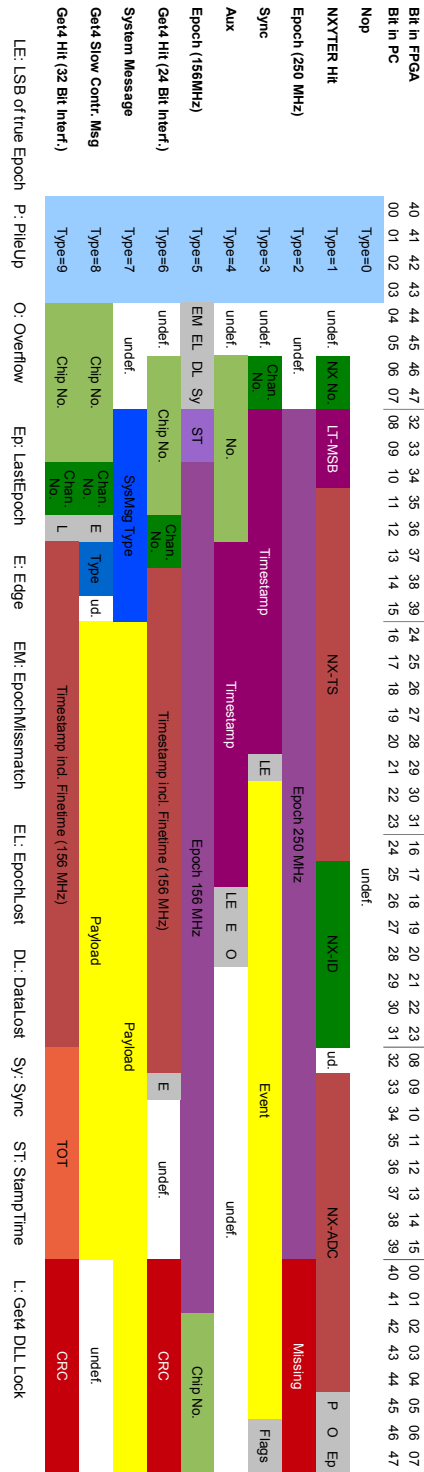
#### Hit Messages

The hit data coming from the front-end electronics is embedded in a 48 bit wide data message. Those messages carry the timestamp of the hit, time over threshold, the chip number and the channel number.

#### System Messages

Since the data acquisition at CBM is designed to be timestamped and free running (no global trigger) the latency from data generation in the detector and data interpretation in the computing node can be quite high. It is important to have a mechanism to insert certain marker events in the data stream (e.g. when a FIFO was reset). Such marker events are called System Messages. The following System Messages are defined:

SysMsg Type	Name	Comment
0x01 1	DAQ-START	first message when DAQ is started
0x02 2	DAQ-STOP	last message before DAQ is stopped
0x03 3	NX-PARITY	<i>nXYTER specific message</i>
0x04 4	SYNC-PARITY	<i>legacy functionality</i>
0x05 5	DAQ-RESUME	<i>legacy functionality</i>
0x06 6	FIFO-RESET	first message after a FIFO reset
0x07 7	USER / ADDSYSMSG / SW_DEF_SYSMMSG	can be inserted via slow control or DLM 4, the message carries a 32 bit wide field that can be set via slow control prior to insertion
0x09 8	PCTIME	<i>legacy functionality</i>
0x09 9	ADC	<i>nXYTER specific message</i>
0x0A 10	PACKETLOST	<i>legacy functionality</i>
0x0B 11	GET4_EVENT	control event from the GET4 chips
0x0C 12	CLOSYSYNC_ERROR	inserted if the internal timestamp counter is not conform to the sync strobe received from CLOSY
0x0D 13	TS156_SYNC	inserted if the internal timestamp counter is reset



**Figure B.4.:** Data path messages: bit correlation between firmware and PC (bit 0 is always the LSB).



## Epochs

The time resolution of the detectors is in the order of nanoseconds or even picoseconds. Transporting the full, unique timestamp of every hit would result in a unnecessary high bandwidth. Therefore the data is organized in time intervals, so called epochs. The timestamp information tagged to the hit data only gives the time that elapsed since the beginning of the epoch. When a new epoch begins, an epoch marker is inserted in the data stream.

## Sync/Aux

The ROC comes with some GPIO pins (a.k.a. Sync/Aux pins). When a rising or falling edge is detected at such a pin, a message - tagged with the 250 MHz timestamp - is inserted in the data stream. This allows to insert time critical information of external events into the data stream.

### B.2.3. Deterministic Latency Messages (DLMs)

The 4 bit wide deterministic latency messages (DLM) are implemented for the synchronization of all the CBM-ROCs. Besides their deterministic latency they provide as well a broadcast functionality, every DLM is received by all ROCs.

The following DLMs are defined:

DLM-No.	Name	Comment
0x00 0	DLM_PING	reply with DLM 0, required for CBMNet link initialization
0x01 1	DLM_SYNC	reset timestamps
0x02 2	undef.	
0x03 3	undef.	
0x04 4	DLM_SW_DEF_SYSMSG	insert system message into data stream
0x05 5	undef.	
0x06 6	undef.	
0x07 7	undef.	
0x08 8	DLM_CMDLST_1	execute Start-DAQ sequence
0x09 9	DLM_CMDLST_2	execute Stop-DAQ sequence
0x0A 10	DLM_CMDLST_3	
0x0B 11	DLM_CMDLST_4	
0x0C 12	DLM_CMDLST_5	
0x0D 13	DLM_CMDLST_6	
0x0E 14	DLM_CMDLST_7	
0x0F 15	DLM_CMDLST_8	

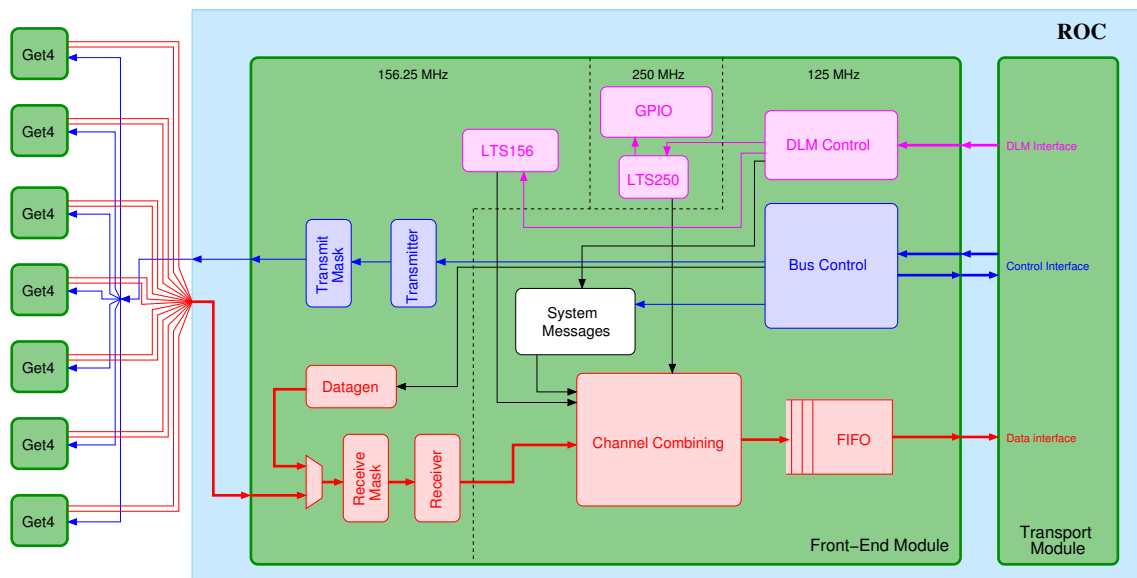
Multiple ROCs can be synchronized by sending the DLM number 1 (DLM\_SYNC). This sets the local 250 MHz timestamp counter in the ROCs to zero.

The DLMs number 8 to 15 trigger a configurable sequence of up to 32 PUT commands stored in the ROC (see section B.2.1). Those lists can be predefined and then simultaneously triggered on all ROCs by the referring DLM. The convention is to use list 1 for Start-DAQ sequence and list 2 for Stop-DAQ sequence.

## B.3. The Modules

### B.3.1. Frontend Readout: GET4

This module is designed to interface multiple GET4 chips. It gathers data from the different GET4 chips and combines it into one data path and provides an interface to configure the GET4 chips.



**Figure B.5.:** A schematic overview of the front-end module for the GET4 read-out.

The important blocks for the GET4 read-out are shown in Figure B.5 and described in the following list.

- **Bus Control / DLM Control:** Those blocks receive the slow control commands and deterministic latency messages (DLMs) respectively and invoke the according action.
- **GPIO:** The GPIO block generates the SYNC and AUX Messages mentioned in section B.2.2. It is required to merge information from external events into the data stream.
- **System Messages:** The System Messages block generates the System Messages described in section B.2.2.

- **Mask Registers:** There are two different mask registers, one for masking the data path from (Receive Mask) and one for masking the control path to (Transmit Mask) the connected GET4 chips. The Receive Mask is needed to disable all unconnected ports because otherwise those LVDS lines may produce ghost hits. It can also be used to suppress data from a malfunctioning GET4.

The Transmit Mask is needed if a command is to be send to only a subset of the connected GET4s.

- **Receiver:** Each GET4 chip transports its data via a serial protocol over an LVDS data line.

Data sampling can be configured to happen at rising or at falling edge of the clock. This enables to choose stable data signal.

Every data line is connected to a deserializer and a small FIFO buffer.

While receiving the data, a CRC checksum is calculated (generator polynomial:  $x^8 + x^2 + x + 1$ ) and padded to the data in case of a hit data.

- **Channel Combining:** Here the data from the different sources is combined into a single data stream. A source can be data from one of the GET4s, a Sync/Aux message, a System Message or an Epoch Message. It also arranges the data in a way that it fits the 48 bit message format shown in figure B.4. The data from GET4s is then read out round-robin.

When data has to be thrown away due to backpressure, a bit in the next epoch message is set to indicate the data loss. Data is thrown away first, even if there is still some little space left in the buffer FIFO. This ensures that epoch and system messages are not lost when data acquisition is running (bandwidth should be high enough to transport these rare messages). If an epoch message is lost due to backpressure (e.g. when data acquisition is stopped), a second flag is set in the next epoch message.

The ROC can be configured to suppress the epoch messages if there was no hit message transmitted since the last epoch message.

- **Local Timestamps:** There are two different timestamp counters in the ROC, one in the 156.25 MHz and one in the 250 MHz clock domain. However those two clock domains are phase locked.

The 250 MHz timestamp counter is needed for the GPIO block and to generate the 250 MHz epoch messages.

The 156.25 MHz timestamp counter is needed to tag the epoch number to the Epoch Messages. In the future, this will be implemented directly on the GET4.

- **The Data Generator:** For debugging purpose, on the very beginning of the data path (even before the deserializer and the mask register) data can be inserted by a

data generator. Data is generated by a circular shift register shifted with the serial clock. The LSB of this register is used as input instead of the real input pin. The content of the shift register is configurable via slow control.

### Synchronize ROC and GET4s

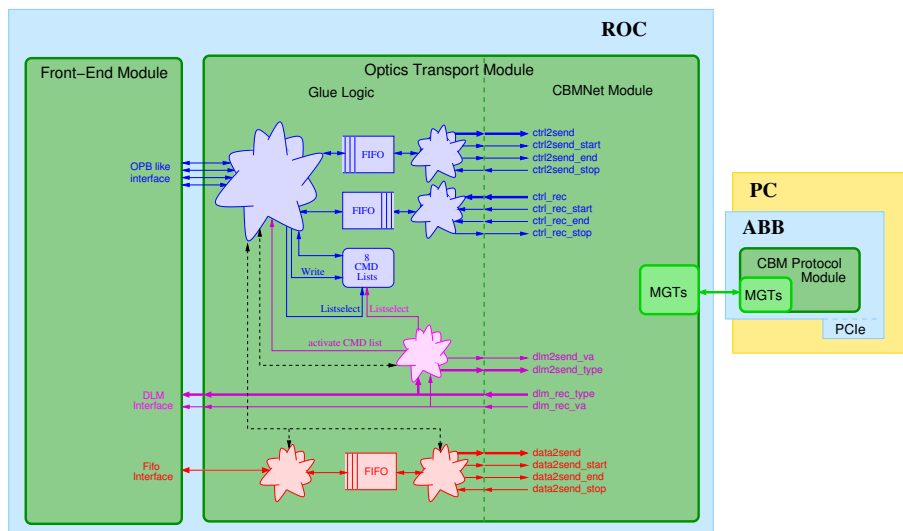
Two PUT commands are required to reset the timestamps on ROC and GET4. Both devices will not immediately continue to count but wait for the next CLOSY sync pulse and then both start synchronous to this pulse.

The two commands to reset the timestamps need to be issued within a very short time frame. This can be achieved in two ways, by sending both PUT commands in one Multi-Oper-List (see section B.2.1), or by programming a command list (see section B.2.1).

If more than one ROC is part of the setup, the command lists approach is mandatory since only here DLMs are used.

### B.3.2. Transport: Optics

The optics module basically consists of two parts, the *CBMNet* module and some glue logic.



**The CBMNet Module** is developed in Mannheim by Frank Lemke and Sven Schenk under the supervision of Ulrich Brüning. It comes with a standardized interface, providing the three traffic classes (Control, Data, DLM). Each class comes with special functionality, like retransmission for Control or deterministic latency for DLM. The yellow LED on the SysCore Board indicates an established optical connection. For more *CBMNet* details, contact Frank Lemke (lemke@uni-hd.de).

**The Glue Logic** translates the *CBMNet* interface to the internal ROC-module interface (OPB, FIFO, DLM). Additionally it provides some functionality needed for the CBM read-out chain:

- registers for the ROCID and the hardware version
- Start-DAQ / Stop-DAQ functionality
- a data generator (see section B.3.2)
- the command lists as described in section B.2.1
- interpreting and executing the following DLMs: DML\_PING and DLM\_CMDLIST\_<nr> (DLMs are described in section B.2.3)
- a register holding the history of received DLMs

### *CBMNet* Encapsulation

The *CBMNet* interface takes the data on the control path and the data path in 16 bit wide words. However the width of the transported data units on these channels is wider than 16 bits. So these data units have to be split in multiple 16 bit words. In this chapter we want to specify how we use the 16 bit wide *CBMNet* interface. To avoid confusion, the following convention shall be used:

- A **word** is the 16 bit wide data that is sampled by *CBMNet* in one cycle.
- A **message** is the 48 bit wide data on the data path, representing a hit event, epoch marker, system message, etc.
- A **command** is 32 or 64 bit wide data unit representing a GET, PUT, ACK, NACK or NOP on the control path.
- A **packet** is a number of words that are consecutively send over the data or control channel of *CBMNet*. By design, the minimum size of a packet is 4 words (8 byte), the maximum is 32 words (64 byte).

**Data Path:** The data messages (hit-data, epoch markers, system messages, ...) are all 48 bit wide, so they are split in three 16 bit wide data words and send in three consecutive clock cycles. For the bit organization of the 48 bit messages refer to figure B.4. The very first word that is send over the data path is the 16 bit ROCID. Then the next three words represent the first 48 bit message, beginning with the lower 16 bit (see figure B.6). If enough data is available, the number of messages in one packet is ten, the maximum that fits into a valid *CBMNet* packet. When there are less than ten messages available, they are send after a short timeout in a shorter *CBMNet* packet. On the ABB side the 48 bit messages are recovered and additionally the ROCID is padded to each message so that they all become 64 bit wide.

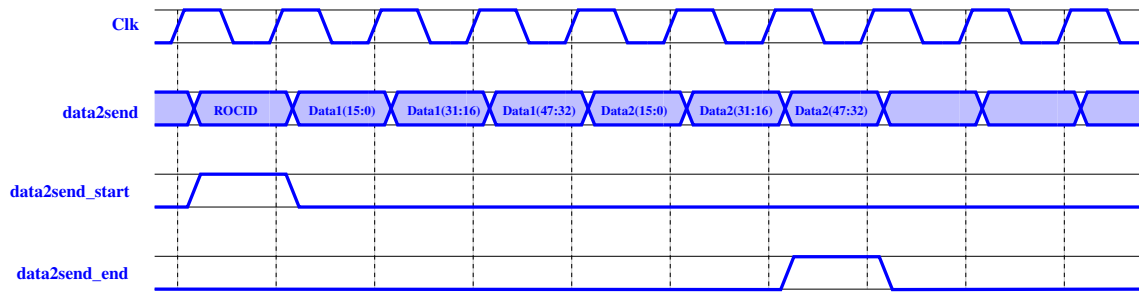
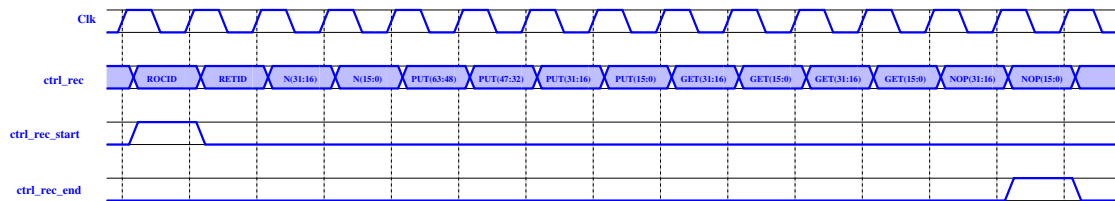
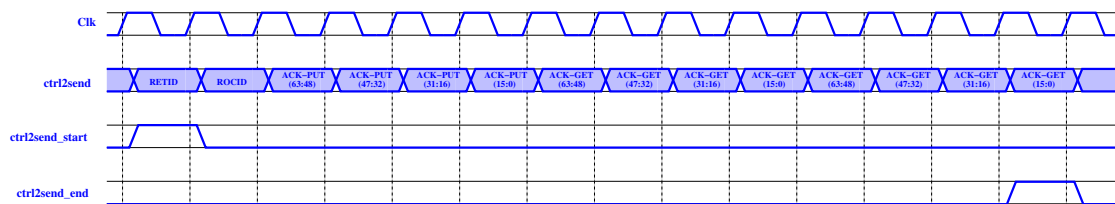


Figure B.6.: A *CBMNet* data path packet with two messages.

**Control Path:** On the control path we implement a GET/PUT protocol based on so called "multi oper lists" (see section B.2.1). The commands defined for the *multi oper lists* protocol are either 32 bit or 64 bit wide and are therefore divided in two or four 16 bit words. Contrary to data messages, control commands are send with the more significant 16 bit first (see figure B.7). The first two words received by the ROC over the control path of *CBMNet* are by definition rocid and retid, in that order. The first two words of a control packet that is send back by the ROC are the same rocid and retid but in the reverse order. Since *CBMNet* requires an alignment to 64 bit, a NOP command is inserted at the end when necessary.



(a) Receiving three PUT/GET commands (one PUT and two GETs) in one packet.



(b) The control path reply for the above packet (figure B.7(a)).

Figure B.7.: The protocol on the *CBMNet* control path.

**DLMs:** DLMs cannot accept the delay induced by a handshaking protocol, they are send and received using just a valid flag (see figure B.8).

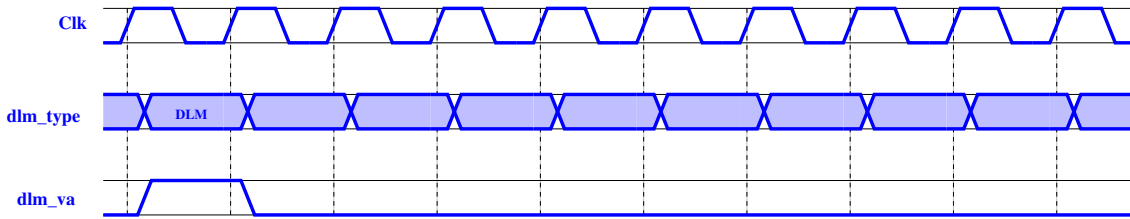


Figure B.8.: Data arrangement for the *CBMNet* DLMs.

### The Optics Data Generator

The optics module includes a fake-data generator. The data generator is a 16-bit counter, so the resulting data stream is a pattern of consecutive 16-bit values. If the values are interpreted as 48 bit data messages they will show up as following:

```

0x 0701 0000 0000 (System Message: Start-DAQ)
0x 0003 0002 0001 (counting up from here on, no useful interpretation)
0x 0006 0005 0004
0x 0009 0008 0007
0x 000C 000B 000A
0x 000F 000E 000D
0x 0012 0011 0010
0x 0015 0014 0013
0x 0018 0017 0016
0x 001B 001A 0019
...

```





## Appendix C.

# Registers in the GET4 Read-Out Controller

This chapter is part of the documentation for the GET4 Read-Out Controller firmware that was written during the development of the firmware. It is intended as a reference work and provides an overview of all available slow control addresses, including a short explanation for each address.

The firmwares that have been used during the in-beam experiments are derived from an early version of the *Modular-ROC* firmware. Since the most important results of this thesis are based on those in-beam experiments the documentation chosen to be shown here describes the early ROC-firmware, and hence is already slightly outdated at time of writing.

### C.1. Addresses in the *Common* Address Space (0x 00 00 00 - 0x 0F FF FC)

#### ROC\_TYPE: 0x 00 00 00

32 bit, readonly. This address indicates which frontend module and which transport module is available in the firmware. The higher 16 bit determine the frontend module, the lower 16 bit determine the transport module. The following values are defined:

- Frontend Module:
  - 0x0001: The nXYTER readout.
  - 0x0002: Support dropped. The FEET readout (old GET4s).
  - 0x0003: The GET4 v1.0 readout.
- Transport Module:
  - 0x0001: Transport via Optics
  - 0x0002: Transport via Ethernet (for the Virtex4-FX20 FPGA)
  - 0x0003: Transport via Ethernet (for the Virtex4-FX40 FPGA)
  - 0x0004: Transport via Ethernet (for the Virtex4-FX60 FPGA)
  - 0x0005: Transport via USB

So, for example the firmware for the readout of the GET4 chips via Optics would return: 0x0003 0001.

### **ROC\_HWV: 0x 00 00 04**

32 bit, readonly. This address returns the (hardware) version of the interface between the frontend module and the transport module.

### **ROC\_FPGA: 0x 00 00 08**

8 bit, readonly. This address returns the FPGA-TARGET the firmware is synthesized for as defined by the following list:

- 0x01: xc4vfx20-ff672-10 (SysCoreV2.0: Virtex-4 FX20, Speedgrade -10, Package ff672)
- 0x02: xc4vfx40-ff672-10 (SysCoreV2.2: Virtex-4 FX40, Speedgrade -10, Package ff672)
- 0x03: xc4vfx60-ff672-10 (SysCoreV2.1: Virtex-4 FX60, Speedgrade -10, Package ff672)
- 0x04: xc6slx45t-3-fgg484 (SP605: Spartan-6 LX45T, Speedgrade -3, Package fgg484)
- 0x05: xc6slx150t-3-fgg900 (SysCoreV3.x: Spartan-6 LX150T, Speedgrade -3, Package fgg900)

### **ROC\_ROCID: 0x 00 00 10**

16 bit, read/write. The ROCID register is intended to be the unique identifier of the ROC. However, this register has to be set in the initialization phase. The messages on the data path are tagged with the value from this register. In the final setup, it should be set according to the routing of the optical network. Contact Frank Lemke (<mailto:lemke@uni-hd.de>) or Sven Schatral (<mailto:schatral@uni-hd.de>) for details concerning the routing of the optical network.

### **ROC\_SVN\_REVISION: 0x 00 00 14**

16 bit, readonly. When synthesized from a clean subversion checkout and the Makeflow is used for synthesis, this address returns the revision number of the subversion repository. Otherwise it defaults to "0x0000".

### **ROC\_BUILD\_TIME: 0x 00 00 18**

32 bit, readonly. When using the Makeflow this address returns the time (number of seconds since 1970-01-01 00:00h) the firmware was synthesized. The value is determined by the command "date +%s" on the PC executing the Makefile, it can be converted to human readable form by executing the command "date -d @VALUE".

**ROC\_SYSTEM\_RESET: 0x000100**

1 bit, writeonly. When writing a 1 to this register, if the Actel is connected (ROC's CON41 and CON2 are set), the Actel will reconfigure the FPGA.

**ROC\_ADDSYSMSG: 0x000200**

32 bit, writeonly. When writing to this address, a System Message carrying the written 32 bit pattern is inserted in the data stream.

**reserved: 0x000600**

0 bit, undefined. This address is used as ROC\_HARDWARE\_VERSION in outdated versions of the firmware. Those firmware versions (version 1.x.x.x) are incompatible to the current address mapping (version 2.x.x.x). The software uses this address to check which firmware version is in use. The address has to be undefined for 2.x.x.x versions.

**ROC\_GPIO\_CONFIG: 0x010000**

32 bit, read/write. This register configures the functionality of the GPIO pins.

**ROC\_SYNC1\_BAUD\_START / ROC\_SYNC2\_BAUD\_START /  
ROC\_SYNC3\_BAUD\_START: 0x010100 / 0x010200 / 0x010300**

8 bit, read/write. This register configures communication parameters for the *poor man's synchronization* (see chapter 3.1.3).

**ROC\_SYNC1\_BAUD1 / ROC\_SYNC2\_BAUD1 / ROC\_SYNC3\_BAUD1:  
0x010104 / 0x010204 / 0x010304**

8 bit, read/write. This register configures communication parameters for the *poor man's synchronization* (see chapter 3.1.3).

**ROC\_SYNC1\_BAUD2 / ROC\_SYNC2\_BAUD2 / ROC\_SYNC3\_BAUD2:  
0x010108 / 0x010208 / 0x010308**

8 bit, read/write. This register configures communication parameters for the *poor man's synchronization* (see chapter 3.1.3).

**ROC\_SYNC1\_SCALEDOWN: 0x01010C (a.k.a.  
ROC\_SYNC1\_M\_DELAY)**

8 bit, read/write. This register sets the time interval between two synchronization events, in case of *poor man's synchronization* (see chapter 3.1.3). These synchronization events then happen every  $2^{\text{ROC\_SYNC1\_SCALEDOWN}}$  epochs.

**ROC\_CMD\_LST\_NR: 0x 02 00 00**

3 bit, writeonly. This will activate a command list. See chapter B.2.1 for details.

**ROC\_CMD\_LST\_ACTIVE: 0x 02 00 04**

1 bit, writeonly. This will (de-)activate the command list processing. This is available for internal use, to deactivate the command list processing at the end of the list. However one can also use it to activate list processing, then the previously processed list is activated again. See chapter B.2.1 for details.

**ROC\_CMD\_LST\_MEM: 0x 02 10 00 - 0x 02 17 FC**

32 bit words, 2k address space, read/write. This address space translates directly to the memory where the command lists are stored. Only 4-byte aligned addresses (last two bits equal 0) are valid. See chapter B.2.1 for details.

**C.2. Addresses in the Module: Optics (0x 20 00 00 - 0x 2F FF FC)**

**ROC\_OPTICS\_HWV: 0x 20 00 00**

32 bit, readonly. This address returns the hardware version of the optics transport module.

**ROC\_OPTICS\_RADTOL: 0x 20 00 04**

1 bit, readonly. This address returns "1" if the module is build with radiation mitigation techniques (redundancy) and "0" otherwise.

**ROC\_START\_DAQ: 0x 20 01 00**

1 bit, writeonly. Writing any value to this address will start the readout of data. Before the first data packet a system message (SYS\_MSG\_DAQ\_START) is send out.

**ROC\_STOP\_DAQ: 0x 20 01 04**

1 bit, writeonly. Writing any value to this address will stop the readout of data. If sending of an CBMNet data packet currently in process it will not be interrupted. A system message (SYS\_MSG\_DAQ\_STOP) is send out as the last data packet. If there is still data in the FIFO of the optical module, this data will reside there. However, no more data will be read out from the frontend module when DAQ is stopped.

**ROC\_OPTICS\_DATAGEN: 0x200300**

1 bit, writeonly. The optics module includes a fake-data generator. This data generator can be enabled by writing a 1 to this address and it can then be disabled again by writing a 0. The data generator is a 16-bit counter, so the resulting data stream is a pattern of consecutive 16-bit values.

**ROC\_OPTICS\_DLMGEN: 0x201000**

4 bit, writeonly. The lower 4 bit that are written to this address will be send as a DLM to the ABB.

**ROC\_OPTICS\_DLM\_HISTORY: 0x201004**

32 bit, readonly. This register stores the last 8 received DLMS. The four lowest bits hold the value of the most recently received DLM.

### C.3. Addresses in the Module: GET4 (0x500000 - 0x5FFFFC)

**ROC\_GET4\_HWV: 0x500000**

32 bit, readonly. This address returns the hardware version of the GET4 frontend module.

**ROC\_GET4\_RADTOL: 0x500004**

1 bit, readonly. This address returns "1" if the module is build with radiation mitigation techniques (redundancy) and "0" otherwise.

**ROC\_GET4\_NR\_OF\_GET4S: 0x500008**

8 bit, readonly. This address returns the number of ports available in the firmware for connecting GET4 chips.

**ROC\_GET4\_RESET: 0x500010**

0 bit, writeonly. This will reset the control logic in the GET4 frontend module.

**ROC\_GET4\_FIFO\_RESET: 0x500014**

1 bit, writeonly. This will reset the data path logic in the GET4 frontend module. Note that there might be some FIFOs in the communication module that are not reset. A clean FIFO reset requires the ROC to be in the state "DAQ-Stopped".

**ROC\_GET4\_TS\_RESET: 0x 50 00 18**

0 bit, writeonly. This will reset the time stamp counter (156 MHz and 250 MHz) in the GET4 frontend module.

**ROC\_GET4\_BURST1/ROC\_GET4\_BURST2/ROC\_GET4\_BURST3:  
0x 50 01 00/0x 50 01 04/0x 50 01 08**

32 bit, readonly. With this commands the 48 bit wide data FIFO can be read out over the control path. The control path has only 32 bit wide data words. So for the readout of two 48 bit words there are three read commands specified.

**ROC\_GET4\_SAMPLE\_FALLING\_EDGE: 0x 50 02 00**

32 bit, read/write. For each GET4 chip one can specify if the serial data will be sampled on the rising edge or on the falling edge of the serial clock. The default is sampling on the rising edge.

**ROC\_GET4\_EPOCH250\_EN: 0x 50 02 04**

1 bit, read/write. If this bit is set, there are epoch markers derived from the 250 MHz clock inserted in the datastream. This is important when synchronizing the GET4 data to data from non-GET4-frontends. Per default this register is set.

**ROC\_GET4\_SUPPRESS\_EPOCHS: 0x 50 02 08**

32 bit, read/write. For each channel one can specify whether 156MHz epoch markers should be withheld if no hit was detected during the corresponding epoch. Suppressing markers of empty epochs might drastically reduce the data volume in low rate experiments, however full epoch information might be useful for debugging and consistency checks. By default epoch marker suppression is turned off on all channels.

**ROC\_GET4\_RECEIVE\_CLK\_CFG: 0x 50 02 0C**

2 bit, read/write. The 2 bit register is used to configure the incoming data rate. This should match the configuration of the GET4s (see GET4 documentation). The following values are available:

"11"	156.25 MHz
"10"	78.13 MHz
"01"	39.06 MHz
"00"	19.53 MHz (default)

#### **ROC\_GET4\_RECEIVE\_MASK: 0x501000**

32 bit, read/write. The 32 bit register is used to unmask incoming data from the GET4 chips. If this register holds the value 0x00000001, only data from the first GET4 will be sampled. A value of 0xFFFFFFFF will sample on every data port. One should only activate ports with a connected GET4 chip since unconnected LVDS lines tend to toggle unpredictably. The default is 0x00000000, all ports are disabled.

#### **ROC\_GET4\_TRANSMIT\_MASK: 0x501004**

32 bit, read/write. The 32 bit register is used to unmask outgoing control messages to the GET4 chips. If this register holds the value 0x00000001, the commands send to ROC\_GET4\_CMD\_TO\_GET4 will only be passed to GET4 number 1. A value of 0xFFFFFFFF can be used to address to every connected GET4 simultaneously, this is also the default.

#### **ROC\_GET4\_DATAGEN\_MASK: 0x501008**

32 bit, read/write. The 32 bit register is used to unmask the outputs of the data generator. If this register holds the value 0x00000001, data will be available on data generator output 0. A value of 0xFFFFFFFF can be used to generate data on all outputs. If ROC\_GET4\_DATAGEN\_EN is set, these outputs will be connected to the data receivers instead of data from the GET4s.

#### **ROC\_GET4\_DATAGEN\_INIT: 0x50100C**

32 bit, read/write. The 32 bit register can be used to shape the data that is generated by the data generator. When the data generator is enabled, the register is shifted left and the MSB is used as LSB (circular shift register). So every 32 serial data clock cycles the pattern is repeated. The MSB is connected to the (unmasked) outputs of the data generator.

#### **ROC\_GET4\_DATAGEN\_EN: 0x501010**

1 bit, writeonly. This enables/disables the data generator. When the data generator is enabled, ROC\_GET4\_DATAGEN\_INIT and ROC\_GET4\_DATAGEN\_MASK cannot be updated but the old values are used until the data generator is disabled.

#### **ROC\_GET4\_CMD\_TO\_GET4: 0x510000**

32 bit, writeonly. The GET4 chip has a 32 bit wide control interface. The 32 bit that are written to this address will be passed to the GET4s that are enabled according to the ROC\_GET4\_TRANSMIT\_MASK register. For a description of the available commands, refer to the GET4 documentation or contact Holger Flemming (h.flemming@gsi.de).





## Appendix D.

### The Read-Out Controller Hardware Platform

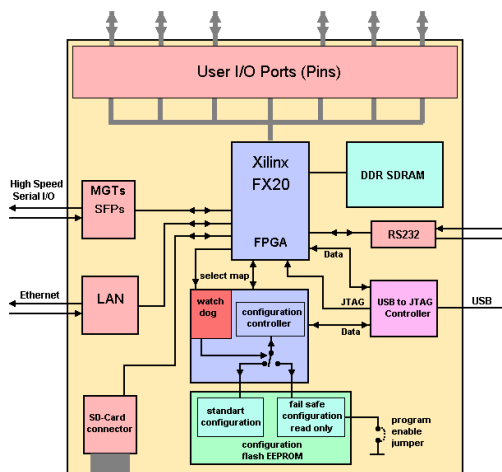
This chapter presents the hardware that was used for carrying out the work for this thesis, the SysCore Boards. The first section explains the abstract set of rules which constitute the theoretical basis for design and realization of the actual SysCore Boards. The second section describes the three different generations of SysCore Boards that have been realized so far. The third section gives an outlook on how the final ToF read-out board might look like.

#### D.1. SysCore Architecture

The *SysCore Architecture* is a set of design rules for FPGA boards. The rules specify certain features that are commonly required in multiple applications, particularly in (but not limited to) high-energy physics detector read-out controllers [Keb09].

The required features are:

- capable of running Linux
- various possibilities to configure the FPGA
  - from Flash memory
  - via commodity USB cable
  - via Xilinx-JTAG programmer
- configuration scrubbing to mitigate radiation effects
- high speed communication to a PC
- maximize I/O flexibility



**Figure D.1.:** Illustration of the *SysCore Architecture*, taken from [Got08]

The list is a superset of the features required by the target applications, i.e. a single target application might not require the full list of features, but the requirements of every target application should be covered by the listed features. The configuration scrubber to mitigate radiation effects requires FPGAs that are capable of Dynamic Partial Reconfiguration (DPR) (see section 3.2.3). At

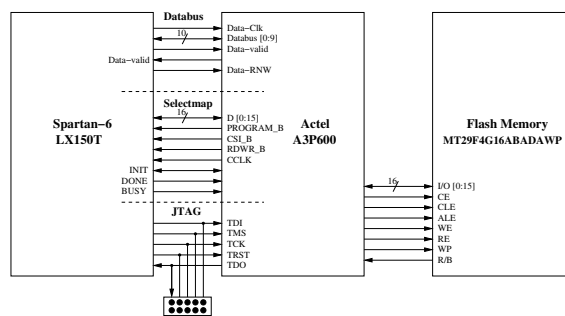
the time the *SysCore Architecture* was formulated, the only FPGAs allowing DPR were FPGAs produced by Xilinx.

Based on this abstract formulation of the *SysCore Architecture* a series of boards was implemented by the CBM community, namely the *SysCore Boards* from version 1 to 3. Sometimes they are also referred to as ROC (Read-Out Controller) version 1 to 3. Since the *SysCore Architecture*, amongst other features, also emphasizes flexibility, the *SysCore Boards* could easily be used for different applications, e.g. read-out of different detector front-end electronics, as Data Combiner Board, as prototyping platform for ASIC designs, etc.

**Configuration System** The heart of the *SysCore Architecture* is an SRAM-based FPGA which needs to be initially configured after every power cycle. Normally, this is done by special on-board flash memories that are connected to the SelectMAP interface of the FPGA [Xil09].

However, in a radiation environment, the SRAM cells of the main FPGA have a relatively high probability to suffer from SEUs. Key element to mitigate SEU effects is the configuration controller which supports configuration scrubbing to avoid build-up of bit errors in the configuration memory of the SRAM-FPGA (see section 3.2.3). With very little overhead, this controller can be used to perform the initial power-up configuration of the SRAM-FPGA as well, an additional SelectMAP-capable flash memory is not required.

The configuration controller of the *SysCore Boards* is implemented in a Flash-based FPGA and the configuration data is stored in commodity flash memory. Figure D.2 shows a high-level overview of the configuration controller design on the *SysCore Board Version 3*.



**Figure D.2.:** Overview of the *SysCore Board Version 3* configuration system. Image from the SysCore V3 specification document [AGK<sup>+</sup>11]

The flash memory can hold more than one set of SRAM-FPGA configuration data and also some additional information, e.g. which configuration set to use by default and if scrubbing should be performed after initial configuration. Further functionality is possible as well, e.g. a watchdog that loads a failsafe configuration set in case it becomes active. However, the main feature of this setup is the possibility to perform *scrubbing*.

## D.2. SysCore Boards

The three generations of SysCore boards realized so far are presented in this section. All three boards implement the *SysCore Architecture*.

### D.2.1. SysCore Board Version 1

The *SysCore Board Version 1*, as pictured in figure D.3, is the pre-prototype of the CBM Read-Out Controller (ROC). It is based on an Xilinx Virtex-4 FX20 FPGA and its development was the first step towards a common read-out controller board for CBM. It was used to develop a very first read-out chain for the nXYTER chip, the *nXYTER Kludge Kit*<sup>1</sup>. First research projects regarding radiation mitigation, an implementation of a fault tolerant Soft-Core CPU, were also carried out on this platform [Eng09, page 61/62]. Being a pre-prototype and the first implementation of its kind, the board naturally had some teething problems. Therefore, only five boards were produced and it was quickly superseded by the *SysCore Board Version 2*.

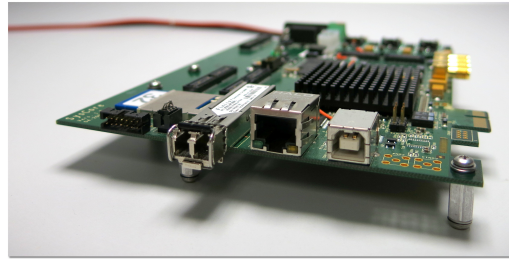


Figure D.3.: The *SysCore Board Version 1*

The *SysCore Board Version 1* item list<sup>2</sup>:

- Xilinx Virtex-4 XC4VFX20, in Speedgrade -11 for boards with PCIe, and in Speedgrade -10 for boards without PCIe
- Actel ProASIC3 Flash FPGA (type A3P125\_FGG144) as on-board configuration controller for power-on configuration for the Xilinx FPGA and also acting as radiation mitigation engine
- Two 64 Mbit Flash memories (type Macronix MX29LV640B) to store the configuration bits for the Xilinx FPGA
- FTDI FT2232C to externally configure the Xilinx FPGA and to communicate with a PC (USB 1.1)
- Two 128 MByte DDR1 SDRAM with 16 bit interface to Xilinx FPGA
- 2.5 Gb optical link, via Virtex-4 MGTs and any SFP compatible transceiver
- 10/100 MBit Ethernet (Virtex-4 on-chip MAC, Intel LXT971ALC PHY, and HALO HFJ11-2450E-L12 MAG)

<sup>1</sup>see <https://cbm-wiki.gsi.de/foswiki/bin/view/DAQ/N-XYTER-KludgeKit>

<sup>2</sup>for detailed technical specification, see <https://cbm-wiki.gsi.de/foswiki/bin/view/DAQ/SysCoreV1>

- two 80 pin connectors, each with 31 LVDS capable pairs as main user IO interface
- 1-lane PCIexpress, only on two boards
- SD card connector, RS232, JTAG, ...

### D.2.2. SysCore Boards Version 2.X

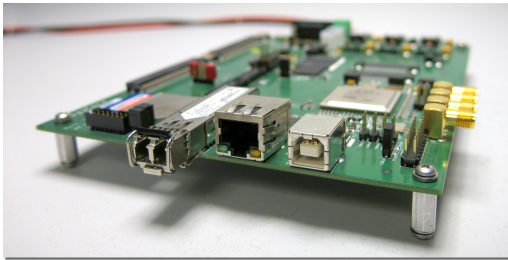


Figure D.4.: The *SysCore Board Version 2.2*

The *SysCore Board Version 2.0* is the second iteration of the CBM Read-Out Controller prototype, based very closely on the hardware and layout of its predecessor, the *SysCore Board Version 1*. The problems that could be identified with the *SysCore Board Version 1* were fixed, some minor design optimizations were implemented (e.g. different form factor for the user IO connectors), and the focus was narrowed on those features that were really significant

for CBM use cases at the time, i.e. support of PCIe was dropped.

In 2008, the board was used the first time as part of the *nXYTER Starter Kit*<sup>3</sup>. It soon became a well established resource in CBM detector test experiments, especially when *nXYTER* or *GET4* chips were involved. One board could interface up to four *nXYTER* and up to 16 *GET4* chips, either be read out via Ethernet or via CBMNet (see chapter 4.1).

Soon the demand exceeded the number of available boards. To overcome this shortcoming, a new series of boards was ordered. Thereby, the opportunity was used to utilize the new boards with a bigger FPGA and a pinout that is better optimized for the *nXYTER* front-end boards. First, two boards were fabricated to test the changes in the layout (*SysCore Board Version 2.1*). They were utilized with *XC4VFX60* FPGAs that happened to be available in the collaboration at the time as left over of a different project. Then the the major series with *XC4VFX40* FPGAs were produced (*SysCore Board Version 2.2*) [DRC<sup>+</sup>13].

The *SysCore Board Version 2.X* item list reads as follows<sup>4</sup>:

- Xilinx Virtex-4, *XC4VFX20* (Version 2.0), *XC4VFX60* (Version 2.1), *XC4VFX40* (Version 2.2), all in Speedgrade -10
- Actel ProASIC3 Flash FPGA (type *A3P125\_FGG144*) as on-board configuration controller for power-on configuration for the Xilinx FPGA and also acting as radiation mitigation engine
- Two 64 Mbit Flash memories (type *Macronix MX29LV640B*) to store the configuration bits for the Xilinx FPGA

<sup>3</sup>see <https://cbm-wiki.gsi.de/foswiki/bin/view/NXYTER/NXYTER-StarterKit>

<sup>4</sup>for more details, see <https://cbm-wiki.gsi.de/foswiki/bin/view/NXYTER/SysCoreV2>

- FTDI FT2232C to externally configure the Xilinx FPGA and to communicate with a PC (USB 1.1)
- Two 128 MByte DDR1 SDRAM with 16 bit interface to Xilinx FPGA
- 2.5 Gb optical link, via Virtex-4 MGTs and any SFP compatible transceiver
- 10/100 MBit Ethernet (Virtex-4 on-chip MAC, Intel LXT971ALC PHY, and HALO HFJ11-2450E-L12 MAG)
- Two 80 pin connectors, each with 32 LVDS capable pairs, as main user IO interface
- SD card connector, RS232, JTAG, ...

Most of firmware development and all in-beam tests described in this thesis are carried out using these version 2.X boards.

### D.2.3. SysCore Board Version 3

The *SysCore Board Version 3* is a new implementation of the *SysCore Architecture* that became available in 2013. Only some parts of the layout of the *SysCore Board Version 2*, e.g. the Actel ProASIC based configuration controller, were reused. The motivation for this re-implementation is cost. With the Series 6 FPGAs, Xilinx started to equip also their low budget Spartan-FPGAs with all required features for implementing the *SysCore Architecture*, especially Dynamic Partial Reconfiguration for scrubbing and high speed interfaces for fast data transport [AGM11].

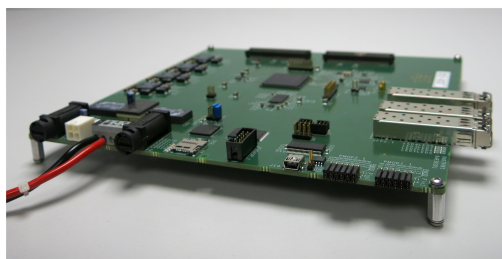


Figure D.5.: The *SysCore Board Version 3*

The flexibility of the *SysCore Board Version 3* allows for its usage as prototyping platform in a number of subsystems. Of course, the use cases of the *SysCore Board Version 2*, the read-out of the nXYTER and the GET4 chips, remain. Additional use cases are the read-out of FEELink interface based ASICs, CBMNet HUB chip development, and DPB development [GGMK13].

With a big Spartan FPGA, about five times more fabric resources are available compared to the Virtex-4 FX-40, however, those resources are slower. High clock frequencies of up to 250 MHz as used in the Virtex-4 are not realistic for Spartan-6 designs. Therefore, the existing 250 MHz logic was adapted to 125 MHz clock frequency. On the other hand, the bigger size of the Spartan-6 allowed to fully utilize the higher connectivity of the *SysCore Board Version 3*. For example, the *SysCore Board Version 3* based firmware for the GET4 read-out can interface up to 57 GET4 chips, while the *SysCore Board Version 2* firmware only supported 16 GET4 chips.

The *SysCore Board Version 3* item list<sup>5</sup>:

- Xilinx Spartan 6 LX150T in Speedgrade -3
- Actel ProASIC3 Flash FPGA (type A3P600\_FGG144) as on-board configuration controller for power-on configuration for the Xilinx FPGA and also acting as radiation mitigation engine
- 512 MB Flash memory (type Micro MT29F4G16ABADAWP) to store the configuration bits of the Xilinx FPGA
- Cypress CY7C68013A-56 to externally configure the Xilinx FPGA and to communicate with a PC (USB 2.0)
- 128 MB DDR3 SDRAM (type MT41J64M16B-187E) with 16 bit interface to Xilinx FPGA
- 3x 2.5 Gb optical link, via Virtex-4 MGTs and any SFP compatible transceiver
- National LMK03200 jitter cleaner
- two FMC-HPC connectors as main user IO interface
- SD card connector, PMOD connectors, JTAG, ...

### **D.3. Final CBM-ToF ROC Board**

The results that could be achieved with the *SysCore Boards* are very important for design decisions of the final read-out chain. Especially the proof, that scrubbing can also efficiently mitigate radiation effects for complex FPGA-designs that are only partially redundant. However, the final read-out chain for CBM-ToF will be very different than those based on the *SysCore Boards*, and this is due to two reasons, both unrelated to the work of this thesis.

One reason is evolution of technology. New Xilinx FPGAs, as of series 7, come already with an integrated in-chip configuration scrubber [Xil14a]. The proven to be efficient technique of configuration scrubbing can be applied without Flash-based components on-board, thereby avoiding the Flash-TID problem (see section 2.2.1). The other reason is the massive change in the design of the read-out chain, mostly driven by the new layout of the CBM magnet. With the new layout, additional space for CBM-STS read-out electronics becomes available. This voids the need for a very dense data aggregator and opto-coupler component, the main reason for developing the CBMNet based HUB-ASIC. Without the space constraint, the HUB-ASIC can be replaced by more mature technology, CERNs GBTx (see section 8.2.1).

---

<sup>5</sup>for detailed technical specification, see <https://cbm-wiki.gsi.de/foswiki/bin/view/DAQ/SysCoreV3>

Nevertheless, also in the new read-out chain design, FPGAs are planned to be operated on-detector. This decision is encouraged by the promising results of this thesis. Using FPGAs so close to the detector would possibly have been rejected without this research.





## Bibliography

- [AAB<sup>+</sup>13] ALME, J., ALT, T., BRATRUD, L., CHRISTIANSEN, P., COSTA, F., DAVID, E., GUNJI, T., KISS, T., LANGØY, R., LIEN, J., LIPPMANN, C., OSKARSSON, A., REHMAN, A. U., RØED, K., RÖHRICH, D., TARANTOLA, A., TORGERSEN, C., TORSVIK, I. N., ULLALAND, K., VELURE, A., YANG, S., ZHAO, C., APPELSHÄUSER, H. and ÖSTERMAN, L., **2013**. *RCU2 - The ALICE TPC readout electronics consolidation for Run2*. *Journal of Instrumentation*, 8(12):C12032. URL <http://stacks.iop.org/1748-0221/8/i=12/a=C12032>
- [Abe07] ABEL, N., **2007**. *nNXYTER meets SysCore*. In *10<sup>th</sup> CBM Collaboration Meeting, 2007, Dresden, Germany*.
- [Abe11] ABEL, N., **02 2011**. *Design and Implementation of an Object-Oriented Framework for Dynamic Partial Reconfiguration*. Ph.D. thesis, Heidelberg University.
- [Act10] Actel, **April 2010**. *Radiation-Tolerant ProASIC3 FPGAs Radiation Effects*.
- [AEKL08] ADAMCZEWSKI, J., ESSEL, H., KURZ, N. and LINEV, S., **Feb 2008**. *Data Acquisition Backbone Core DABC*. *Nuclear Science, IEEE Transactions on*, 55(1):251–255. ISSN 0018-9499. doi:10.1109/TNS.2007.913938.
- [AGK<sup>+</sup>11] ABEL, N., GOTTSCHALK, D., KEBSCHULL, U., MANZ, S. and MÜLLER, W. F., **12 2011**. *SysCore V3 Specification*. Internal Document.
- [AGM11] ABEL, N., GOTTSCHALK, D. and MÜLLER, W. F., **2011**. *SysCore V3 - Firmware Aspects*. In *18<sup>th</sup> CBM Collaboration Meeting, Beijing, China*.
- [Ale13] ALESSIO, F., **2013**. *Trigger-less readout architecture for the upgrade of the LHCb experiment at CERN*. *Journal of Instrumentation*, 8(12):C12019. URL <http://stacks.iop.org/1748-0221/8/i=12/a=C12019>
- [AMEKL10a] ADAMCZEWSKI-MUSCH, J., ESSEL, H., KURZ, N. and LINEV, S., **2010**. *Dataflow Engine in DAQ Backbone DABC*. *Nuclear Science, IEEE Transactions on*, 57(2):614–617. ISSN 0018-9499. doi:10.1109/TNS.2010.2042174.
- [AMEKL10b] ADAMCZEWSKI-MUSCH, J., ESSEL, H. G., KURZ, N. and LINEV, S., **2010**. *Data Acquisition Backbone Core DABC release v1.0*. *Journal of Physics: Conference Series*, 219(2):022007. URL <http://stacks.iop.org/1742-6596/219/i=2/a=022007>

- [AMEL11] ADAMCZEWSKI-MUSCH, J., ESSEL, H. and LINEV, S., **Aug 2011**. *Online Object Monitoring With Go4 V4.4*. *Nuclear Science, IEEE Transactions on*, 58(4):1477–1481. ISSN 0018-9499. doi:10.1109/TNS.2011.2149541.
- [AMZE03] ASADI, G., MIREMADI, S., ZARANDI, H.-R. and EJLALI, A., **Dec 2003**. *Fault injection into SRAM-based FPGAs for the analysis of SEU effects*. In *Field-Programmable Technology (FPT), 2003. Proceedings. 2003 IEEE International Conference on*, pp. 428–430. doi:10.1109/FPT.2003.1275794.
- [Aug06a] AUGUSTIN, I., **12 2006**. *FAIR Baseline Technical Report - Volume 1: Executive Summary*. <https://www-alt.gsi.de/documents/DOC-2006-Dec-94.html>.
- [Aug06b] AUGUSTIN, I., **3 2006**. *FAIR Baseline Technical Report - Volume 3A: Experiment Proposals on QCD Physics*. <https://www-alt.gsi.de/documents/DOC-2006-Jul-41.html>.
- [Bau05] BAUMANN, R., **Sept 2005**. *Radiation-induced soft errors in advanced semiconductor technologies*. *Device and Materials Reliability, IEEE Transactions on*, 5(3):305–316. ISSN 1530-4388. doi:10.1109/TDMR.2005.853449.
- [BBG<sup>+</sup>94] BIRI, J., BLASOVSKY, M., GIGLER, J., MOHOS, I., SOMLAI, K., ZHHDI, Z., BOJOWALD, J., DIETRICH, J., HACKER, U. and MAIER, R., **1994**. *Beam Position Monitor Electronics at the Cooler Synchrotron COSY JÜLICH*. *IEEE Transactions on Nuclear Science*, 41(1).
- [BCMS12] BRUGGER, M., CALVIANI, M., MEKKI, J. and SPIEZIA, G., **2012**. *R2E - experience and outlook for 2012*. *3rd Evian Workshop on LHC beam operation*.
- [BCT08] BRIDGFORD, B., CARMICHAEL, C. and TSENG, C. W., **March 2008**. *Single-Event Upset Mitigation Selection Guide (XAPP987)*. Xilinx, v1.0 ed.
- [Ber14] BERG, M., **May 2014**. *An Analysis of Heavy-Ion Single Event Effects for a Variety of Finite State-Machine Mitigation Strategies*.
- [BSV11] BATTEZZATI, N., STERPONE, L. and VIOLANTE, M., **2011**. *Reconfigurable Field Programmable Gate Arrays for Mission-Critical Applications*. Springer. Doi:10.1007/978-1-4419-7595-9.
- [Car06] CARMICHAEL, C., **July 2006**. *Triple Module Redundancy Design Techniques for Virtex FPGAs (XAPP197)*. Xilinx, v1.0.1 ed.
- [CCS00] CARMICHAEL, C., CAFFREY, M. and SALAZAR, A., **June 2000**. *Correcting Single-Event Upsets Through Virtex Partial Configuration (XAPP216)*. Xilinx, v1.0 ed.

- [Cha10] CHAPMAN, K., **April 2010**. *SEU Strategies for Virtex-5 Devices (XAPP864)*. Xilinx, v2.0 ed.
- [CHH<sup>+</sup>13] CIOBANU, M., HERRMANN, N., HILDENBRAND, K. D., KIŠ, M., FLEMING, A. S. H., H. DEPPE, FRÜHAUF, J., DEPPNER, I., TRÄGER, M. and LOIZEAU, P. A., **2013**. *PADI-6 and PADI-7, new ASIC prototypes for CBM ToF*, vol. 2013-1 of *GSI Report*, p. 291. GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt.  
URL <http://repository.gsi.de/record/52186>
- [Con09] Configurable Computing Lab, Brigham Young University, **September 2009**. *BYU-LANL Triple Modular Redundancy User Guide*, v0.5.2 ed.
- [CT09] CARMICHAEL, C. and TSENG, C. W., **October 2009**. *Correcting Single-Event Upsets in Virtex-4 FPGA Configuration Memory (XAPP1088)*. Xilinx, v1.0 ed.
- [dC09] DE CUVELAND, J., **09 2009**. *A Track Reconstructing Low-latency Trigger Processor for High-energy Physics*. Ph.D. thesis, Heidelberg University.
- [Dep13] DEPPNER, I., **11 2013**. *Development of a fully differential Multi-gap Resistive Plate Chamber for the CBM Experiment*. Ph.D. thesis, Heidelberg University.  
URL <http://www.ub.uni-heidelberg.de/archiv/15771>
- [DF09] DEPPE, H. and FLEMMING, H., **2009**. *The GSI event-driven TDC with 4 channels GET4*. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, pp. 295–298. ISSN 1095-7863. doi:10.1109/NSSMIC.2009.5401741.
- [DFLH08] DEJONG, J., FABULA, J., LESEA, A. and HSIEH, W., **2008**. *The Total Ionizing Dose Performance of Deep Submicron CMOS Processes*. In *MAPLD Conference*.
- [DHA<sup>+</sup>14] DEPPNER, I., HERRMANN, N., AKINDINOV, A., BARTOS, D., BALACEANU, A., BELOGUROV, S., CAO, P., CARAGHEORGHEOPOL, G., CHEN, H., CHENG, J., CIOBANU, M., CONSTANTIN, F., DENG, Z., DEPPE, H., DUTA, V., FAN, H., FLEMMING, H., FRÜHAUF, J., GEBELEIN, J., HEIDEL, K., HILDENBRAND, K., KEBSCHULL, U., KIŠ, M., KISELEV, S., KOCH, K., KOCZON, P., KOTTE, R., GARCIA, A. L., LEHRBACH, J., LI, C., LI, Y., LOIZEAU, P.-A., LV, P., MALKEVICH, D., MANZ, S., NAUMANN, L., NEDOSEKIN, A., NIEBUR, W., OANCEA, A., PETRIS, M., PETROVICI, M., PLOTNIKOV, V., PROKUDIN, M., RADULESCU, L., SHAO, M., SIMION, V., SIMON, C., SULTANOV, R., SUN, Y., TANG, Z., WANG, Y., WÜSTENFELD, J., XIANG, C., XU, N., ZHANG, Y., ZHOU, D. and ZHU, X., **2014**. *The CBM Time-of-Flight wall - a conceptual design*. *Journal of Instrumentation*, 9(10):C10014.  
URL <http://stacks.iop.org/1748-0221/9/i=10/a=C10014>
- [DRC<sup>+</sup>13] DEY, M., ROY, A., CHATTOPADHYAY, S., ABEL, N., MANZ, S. and GOTTSCHALK, D., **2013**. *Development and production of the ROC SysCore board V2.2*. CBM Progress Report 2012. Page 86.

- [EMWG06] ENGEL, J. D., MORGAN, K. S., WIRTHLIN, M. J. and GRAHAM, P. S., **2006**. *Predicting On-Orbit Static Single Event Upset Rates in Xilinx Virtex FPGAs*. Los Alamos National Laboratory. <http://hdl.lib.byu.edu/1877/431>.
- [Eng09] ENGEL, H., **06 2009**. *Development of a Fault Tolerant Softcore CPU for SRAM based FPGAs*. Master's thesis, Heidelberg University. <https://www-alt.gsi.de/documents/DOC-2010-Oct-91.html>.
- [Eto07] ETO, E., **December 2007**. *Difference-Based Partial Reconfiguration (XAPP290)*. Xilinx, v2.0 ed.
- [Fac99] FACCIO, F., **1999**. *Radiation effects in the electronics for CMS*. CERN.
- [FHK<sup>+</sup>11] FRIMANN, B., HOEHNE, C., KNOLL, J., LEUPOLD, S., RANDRUP, J., RAPP, R. and SENGER, P., editors, **2011**. *The CBM Physics Book - Compressed Baryonic Matter in Laboratory Experiments*, vol. 814 of *Lecture Notes in Physics*. Springer. Doi:10.1007/978-3-642-13293-3.
- [FS13] FRIESE, V. and STURM, C., **04 2013**. *CBM Progress Report 2013*. <https://www-alt.gsi.de/documents/DOC-2014-Mar-16.html>.
- [FUW<sup>+</sup>14] FÄRBER, C., UWER, U., WIEDNER, D., LEVERINGTON, B. and EKELHOF, R., **2014**. *Radiation tolerance tests of SRAM-based FPGAs for the potential usage in the readout electronics for the LHCb experiment*. *Journal of Instrumentation*, 9(02):C02028.  
URL <http://stacks.iop.org/1748-0221/9/i=02/a=C02028>
- [FUWL14] FÄRBER, C., UWER, U., WIEDNER, D. and LEVERINGTON, B., **March 2014**. *Proton irradiation test of an Altera SRAM-based FPGA for the possible usage in the readout electronics of the LHCb experiment*.
- [Gao12] GAO, W., **03 2012**. *Active Buffer Development in CBM Experiment*. Ph.D. thesis, Heidelberg University.  
URL <http://www.ub.uni-heidelberg.de/archiv/13313>
- [Gar11] GARCÍA CHÁVEZ, C. d. J., **09 2011**. *First steps towards feature extraction in TRD*. 18<sup>th</sup> CBM Collaboration Meeting, 2011, Beijing, China.
- [Gar14] GARCÍA CHÁVEZ, C. d. J., **03 2014**. *Development of a Feature Extraction Pre-Processing Stage for the CBM-TRD*. DPG Spring Meeting (HK 56.5), Frankfurt, Germany. <https://www-alt.gsi.de/documents/DOC-2014-Aug-63.html>.
- [Garss] GARCÍA CHÁVEZ, C. d. J., **(work in progress)**. *title to be defined*. Ph.D. thesis, Frankfurt University.

- [Geb12] GEBELEIN, J., **09 2012**. *Investigation of SRAM FPGA based Hamming FSM encoding in beam test*. In *Radiation and its Effects on Components and Systems - RADECS, Biarritz, France*. <https://www-alt.gsi.de/documents/DOC-2012-Dec-43.html>.
- [GGMK13] GEBELEIN, J., GOTTSCHALK, D., MAY, G. and KEBSCHULL, U., **2013**. *SysCore3 - A universal Read-Out Controller and Data Processing Board*. CBM Progress Report 2012. Page 87.
- [GKS<sup>+</sup>06] GEORGE, J., KOGA, R., SWIFT, O., ALLEN, G., CARMICHAEL, C. and TSENG, C., **July 2006**. *Single Event Upsets in Xilinx Virtex-4 FPGA Devices*. In *Radiation Effects Data Workshop, 2006 IEEE*, pp. 109–114. doi:10.1109/REDW.2006.295477.
- [GKW<sup>+</sup>09] GAO, W., KUGEL, A., WURZ, A., MARCUS, G. and MÄNNER, R., **2009**. *Active buffer for DAQ in CBM experiment*. In *Real Time Conference, 2009. RT '09. 16th IEEE-NPSS*, pp. 527–531. doi:10.1109/RTC.2009.5321680.
- [Got08] GOTTSCHALK, D., **2008**. *SysCore for N-XYTER, version 1.04*. Kirchhoff-Institute for Physics, Heidelberg.  
URL <https://cbm-wiki.gsi.de/foswiki/bin/view/NXYTER/SysCoreV2>
- [Gra14] GRASSI, T., **March 2014**. *FPGAs in the CMS HCAL electronics*.
- [HAB14] HENRIK ÅKERSTEDT, S. M. and BOHM, C., **March 2014**. *FPGAs in ATLAS Front-End Electronics*.
- [Ham50] HAMMING, R. W., **April 1950**. *Error detecting and error correcting codes*. Bell System Technical Journal (Volume 29).
- [Har13] Harald Deppe and Holger Flemming, **January 2013**. *The GSI Event driven TDC with 4 Channels GET4*, v1.2 ed.
- [HDFC<sup>+</sup>11] HOEFFGEN, S., DURANTE, M., FERLET-CAVROIS, V., HARBOE-SORENSEN, R., LENNARTZ, W., KUENDGEN, T., KUHNHENN, J., LATESSA, C., MATHES, M., MENICUCCI, A., METZGER, S., NIEMINEN, P., PLESKAC, R., POIVEY, C., SCHARDT, D. and WEINAND, U., **Sept 2011**. *Investigations of single event effects with heavy ions of energies up to 1.5 GeV/n*. In *Radiation and Its Effects on Components and Systems (RADECS), 2011 12th European Conference on*, pp. 711–715. ISSN 0379-6566. doi:10.1109/RADECS.2011.6131368.
- [HSPG<sup>+</sup>08] HARBOE-SORENSEN, R., POIVEY, C., GUERRE, F.-X., ROSENG, A., LOCHON, F., BERGER, G., HAJDAS, W., VIRTANEN, A., KETTUNEN, H. and DUZELLIER, S., **Dec 2008**. *From the Reference SEU Monitor to the Technology Demonstration Module On-Board PROBA-II*. *Nuclear Science, IEEE Transactions on*, 55(6):3082–3087. ISSN 0018-9499. doi:10.1109/TNS.2008.2006896.

- [Hutss] HUTTER, D., (**work in progress**). *title to be defined*. Ph.D. thesis, Frankfurt University.
- [IBM01] IBM, **April 2001**. *On-Chip Peripheral Bus - Architecture Specifications*, v2.1 ed. SA-14-2528-02.
- [Int] International Specialty Products. *Gafchromic EBT2 Film for a contemporary RT environment*.  
URL <http://www.ashland.com/Ashland/Static/Documents/ASI/Advanced%20Materials/ebt2.pdf>
- [Int13] International Telecommunication Union (ITU), **January 2013**. *Integrated Services Digital Network*, itu-t recommendation i.432.1 ed.
- [JA14] JOHAN ALME, f. t. A. T. C., **March 2014**. *Experience from using SRAM based FPGAs in the ALICE TPC Detector and Future Plans*.
- [JED06] JEDEC Solid State Technology Association, **October 2006**. *Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices (JESD89A)*.
- [KBD<sup>+</sup>12] KAVATSYUK, M., BREMER, D., DREXLER, P., EISSNER, T., HEVINGA, M., KUSKE, T., LEMMENS, P., MOEINI, H., NISHIZAWA, T., SCHAKEL, P., SCHREUDER, F., SPEELMAN, R., TAMBAVE, G. and LOHNER, H., **Oct 2012**. *Trigger-less readout of the PANDA electromagnetic calorimeter*. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, pp. 1796–1801. ISSN 1082-3654. doi:10.1109/NSSMIC.2012.6551420.
- [Keb09] KEBSCHULL, U., **03 2009**. *ROC Status and Future Plans [KIP, Heidelberg University]*. <https://www-alt.gsi.de/documents/DOC-2009-Mar-96.html>.
- [Kir07] KIRSCH, S., **06 2007**. *Development of the Supermodule Unit for the ALICE Transition Radiation Detector at the LHC (CERN)*. Master's thesis, Heidelberg University.
- [Kno00] KNOLL, G. F., **1 2000**. *Radiation Detection and Measurement*. Wiley, 3 ed. ISBN 9780471073383.
- [Koc09] KOCH, K., **Oct 2009**. *CLOSYS: A very precise clock generation for timing measurements and synchronization of the CBM ToF wall*. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, pp. 324–326. ISSN 1095-7863. doi:10.1109/NSSMIC.2009.5401720.
- [Kum12] KUMAR, V., **2012**. *Texas Instruments FRAM MCUs for Dummies*. Mouser Electronics, Inc.

- [KZ04] KUMAR, N. and ZACHER, D., **2004**. *Automated FSM Error Correction for Single Event Upsets*. In *Military and Aerospace Programmable Logic Devices (MAPLD) International Conference, Washington DC*.
- [LCF<sup>+</sup>01] LIMA, F., CARMICHAEL, C., FABULA, J., PADOVANI, R. and REIS, R., **Sept 2001**. *A fault injection analysis of Virtex FPGA TMR design methodology*. In *Radiation and Its Effects on Components and Systems, 2001. 6th European Conference on*, pp. 275–282. doi:10.1109/RADECS.2001.1159293.
- [LDF<sup>+</sup>05] LESEA, A., DRIMER, S., FABULA, J., CARMICHAEL, C. and ALFKE, P., **Sept 2005**. *The rosetta experiment: atmospheric soft error rate testing in differing technology FPGAs*. *Device and Materials Reliability, IEEE Transactions on*, 5(3):317–328. ISSN 1530-4388. doi:10.1109/TDMR.2005.854207.
- [Leh13] LEHRBACH, J., **2013**. *Entwurf und Implementierung der Auslesefirmware für das 32 Bit Interface des Get4 Chips im Rahmen des CBM Experiments*. Master's thesis, Frankfurt University. (in German).
- [Lem12] LEMKE, F., **09 2012**. *Unified Synchronized Data Acquisition Networks*. Ph.D. thesis, Mannheim University.
- [Leo94] LEO, W., **1994**. *Techniques for Nuclear and Particle Physics Experiments: A How-To Approach*. U.S. Government Printing Office. ISBN 9783540572800.
- [LFG<sup>+</sup>14] LÖCHNER, S., FRÜHAUF, J., GRAF, H., HOFFMANN, J., P. KOCZON, S. M. and WITTHAUS, M., **2014**. *CBM proton beam test for electronic components*. CBM Progress Report 2013. Page 94.
- [LGWH13] LÖCHNER, S., GRAF, H. and WALASEK-HÖHNE, B., **2013**. *Single Event Effects and beam diagnostic studies at the CBM proton test beam*. CBM Progress Report 2012. Page 83.
- [Loi14] LOIZEAU, P.-A., **06 2014**. *Development and test of a free-streaming readout chain for the CBM Time of Flight Wall*. Ph.D. thesis, Heidelberg University. URL <http://www.ub.uni-heidelberg.de/archiv/17081>
- [LSB10] LEMKE, F., SCHENK, S. and BRUENING, U., **2010**. *Prototype Results of an Optical Communication Network for the CBM DAQ-System*. CBM Progress Report 2009. Page 54.
- [MAGK10] MANZ, S., ABEL, N., GEBELEIN, J. and KEBSCHULL, U., **2010**. *An universal read-out controller*. *Journal of Instrumentation*, 5(11):C11017. URL <http://stacks.iop.org/1748-0221/5/i=11/a=C11017>
- [Mar04] MARTENS, S., **2004**. *Strahlentest mit Mikrochips für das ALICE Experiment*. Master's thesis, Heidelberg University.

- [Mar11] MARCUS MARTÍNEZ, G. A., **01 2011**. *Acceleration of Astrophysical Simulations with Special Hardware*. Ph.D. thesis, Heidelberg University. URL <http://www.ub.uni-heidelberg.de/archiv/11757>
- [MGO<sup>+</sup>13a] MANZ, S., GEBELEIN, J., OANCEA, A., ENGEL, H. and KEBSCHULL, U., **2013**. *ToF-ROC FPGA irradiation tests 2012*. CBM Progress Report 2012. Page 69.
- [MGO<sup>+</sup>13b] MANZ, S., GEBELEIN, J., OANCEA, A., ENGEL, H. and KEBSCHULL, U., **2013**. *ToF-ROC FPGA irradiation tests 2012*. GSI Scientific Report 2012. Page 67.
- [Mic14] MicroSemi, **August 2014**. *IGLOO2 FPGAs Product Brief*, revision 8 ed.
- [MMK07] MOREIRA, P., MARCHIORO, A. and KLOUKINAS, K., **September 2007**. *The GBT, a Proposed Architecture for Multi-Gb/s Data Transmission in High Energy Physics*. In *Topical Workshop on Electronics for Particle Physics 2007, Prague, Czech Republic*, pp. 332–336.
- [MMPW07] MORGAN, K., MCMURTREY, D., PRATT, B. and WIRTHLIN, M., **Dec 2007**. *A Comparison of TMR With Alternative Fault-Tolerant Design Techniques for FPGAs*. *Nuclear Science, IEEE Transactions on*, 54(6):2065–2072. ISSN 0018-9499. doi:10.1109/TNS.2007.910871.
- [Mül13] MÜLLER, W., **11 2013**. *Towards CBNnet V3.0*. FLESDAQ Workgroup Meeting, GSI.
- [Mül14] MÜLLER, W., **2014**. *DAQ Coordination*. In *23<sup>th</sup> CBM Collaboration Meeting, 2014, Darmstadt, Germany*.
- [NAMH<sup>+</sup>13] NEISER, A., ADAMCZEWSKI-MUSCH, J., HOEK, M., KOENIG, W., KORCYL, G., LINEV, S., MAIER, L., MICHEL, J., PALKA, M., PENSCHUCK, M., TRAXLER, M., UĞUR, C. and ZINK, A., **2013**. *TRB3: a 264 channel high precision TDC platform and its applications*. *Journal of Instrumentation*, 8(12):C12043. URL <http://stacks.iop.org/1748-0221/8/i=12/a=C12043>
- [Oan13] OANCEA, A.-D., **2013**. *Design of a fault-tolerant configuration and scrubbing chain for the CBM ROC*. Master's thesis, Heidelberg University.
- [PCG<sup>+</sup>05] PRATT, B., CAFFREY, M., GRAHAM, P., MORGAN, K. and WIRTHLIN, M., **2005**. *Improving FPGA Design Robustness with Partial TMR*. In *Military and Aerospace Programmable Logic Devices (MAPLD) International Conference, Washington DC*.



- [RAF<sup>+</sup>11a] ROED, K., ALME, J., FEHLKER, D., LIPPMANN, C. and REHMAN, A., **2011**. *First measurement of single event upsets in the readout control FPGA of the ALICE TPC detector*. *J. Instrum.*, 6:C12022.
- [RAF<sup>+</sup>11b] ROED, K., ALME, J., FEHLKER, D., LIPPMANN, C. and REHMAN, A., **2011**. *First measurement of single event upsets in the readout control FPGA of the ALICE TPC detector*. *JINST*, 6:C12022. doi:10.1088/1748-0221/6/12/C12022.
- [RBK<sup>+</sup>12] ROED, K., BRUGGER, M., KRAMER, D., PERONNARD, P., PIGNARD, C., SPIEZIA, G. and THORNTON, A., **Aug 2012**. *Method for Measuring Mixed Field Radiation Levels Relevant for SEEs at the LHC*. *Nuclear Science, IEEE Transactions on*, 59(4):1040–1047. ISSN 0018-9499. doi:10.1109/TNS.2012.2183677.
- [Rø09] RØED, K., **09 2009**. *Single Event Upsets in SRAM FPGA based readout electronics for the Time Projection Chamber in the ALICE experiment*. Ph.D. thesis, University of Bergen.
- [Sch07] SCHMIDT, C., **2007**. *Test results on the n-XYTER ASIC, a self triggered, sparcifying readout ASIC*. *Topical Workshop on Electronics for Particle Physics , TWEPP-07*, p. 84.
- [Sch14] SCHATRAL, S., **03 2014**. *A high-speed serializer read-out ASIC prototype*. 23<sup>rd</sup> CBM Collaboration Meeting, 2014, Darmstadt, Germany.
- [Sen11] SENGER, A., **2011**. *FLUKA calculations for CBM - 35 AGeV Au+Au collisions*. In 18<sup>th</sup> CBM Collaboration Meeting, 2011, Beijing, China.
- [STS<sup>+</sup>07] SCHMIDT, C., TRUNK, U., SOLTVEIT, H. K., BROGNA, A., BUZZETTI, S., DABROWSKI, W., FIUTOWSKI, T., MODZEL, G., SOLVAG, K., SZCZYGIEL, R. and WIACEK, P., **2007**. *Test results on the n-XYTER ASIC, a self triggered, sparcifying readout ASIC*. In *Topical Workshop on Electronics for Particle Physics, TWEPP-07, Prague*.
- [TOF] TOF WORKGROUP. *Technical Design Report for the CBM Time-of-Flight System (TOF)*. Will be published.
- [UBKT12] UGUR, C., BAYER, E., KURZ, N. and TRAXLER, M., **2012**. *A 16 channel high resolution (<11 ps RMS) Time-to-Digital Converter in a Field Programmable Gate Array*. *Journal of Instrumentation*, 7(02):C02004.  
URL <http://stacks.iop.org/1748-0221/7/i=02/a=C02004>
- [Vog14] VOGEL, M., **June 2014**. *Nichtflüchtig gespeichert (in German)*. *Physik Journal*, 13:48–49.
- [Whi14] WHITE, D. M., **March 2014**. *Single Event Effects in Complex Components*.

- [Wir14] WIRTHLIN, M., **2014**. *BL-TMR and Mitigation Approaches for FPGAs*.
- [WRGC03a] WIRTHLIN, M., ROLLINS, N., GRAHAM, P. and CAFFREY, M., **2003**. *Evaluating TMR Techniques in the Presence of Single Event Upsets*. In *Military and Aerospace Programmable Logic Devices (MAPLD) International Conference, Washington DC*.
- [WRGC03b] WIRTHLIN, M., ROLLINS, N., GRAHAM, P. and CAFFREY, M., **2003**. *Hardness By Design Techniques for Field Programmable Gate Arrays*. In *11<sup>th</sup> Annual NASA Symposium on VLSI Design Coeur d'Alene, Idaho*.
- [XHD<sup>+</sup>13] XIANG, C. Z., HERRMANN, N., DEPPNER, I., LOIZEAU, P. A., WISNIEWSKI, K., ZHANG, Y. P., ZHOU, D. C., FRÜHAUF, J., LINEV, S., MANZ, S. and MÜLLER, W. F. J., **2013**. *The online data pre-processing for CBM-TOF*. *Journal of Instrumentation*, 8(02):P02002.  
URL <http://stacks.iop.org/1748-0221/8/i=02/a=P02002>
- [Xil] Xilinx. *Xilinx TMRTool Product Brief*.
- [Xil00] Xilinx, **September 2000**. *Virtex Series Configuration Architecture User Guide (XAPP151)*, v1.5 ed.
- [Xil08] Xilinx, **December 2008**. *Virtex-4 FPGA User Guide (UG070)*, v2.6 ed.
- [Xil09] Xilinx, **June 2009**. *Virtex-4 FPGA Configuration User Guide (UG071)*, v1.11 ed.
- [Xil10a] Xilinx, **November 2010**. *QPro Virtex 2.5V Radiation-Hardened FPGAs (DS028)*, v2.1 ed.
- [Xil10b] Xilinx, **August 2010**. *Virtex-4 Family Overview (DS112)*, v3.1 ed.
- [Xil13] Xilinx, **April 2013**. *Partial Reconfiguration User Guide (UG702)*, v14.5 ed.
- [Xil14a] Xilinx, **November 2014**. *7 Series FPGAs Configuration (UG470)*, v1.9 ed.
- [Xil14b] Xilinx, **August 2014**. *Device Reliability Report (UG116)*, v10.1 ed.
- [Xil14c] Xilinx, **November 2014**. *Radiation-Hardened, Space-Grade Virtex-5QV Family Overview (DS192)*, v1.4 ed.

## Anhang E.

# German Summary – Deutsche Zusammenfassung

### Analyse, Umsetzung und Überprüfung der Leistungsfähigkeit von Techniken zur Entschärfung von Strahlungseffekten in SRAM-basierten FPGAs im Kontext des CBM Experiments

#### Einleitung und Motivation

Die Detektoren moderner Experimente im Bereich der Hochenergiephysik generieren während des Betriebs enorme Datenraten. Die effiziente Auslese dieser Daten aus den Puffern der vordersten Elektronikstufen (englisch: “front-end electronics”), die sich zu meist noch direkt am Detektor befinden, ist ein kompliziertes Unterfangen. Verschiedene Experimente kämpfen dabei meistens mit unterschiedlichen Schwierigkeiten. Das *Compressed Baryonic Matter* (CBM) Experiment, welches derzeit an der *Facility for Antiproton and Ion Research* (FAIR) bei Darmstadt gebaut wird, sieht einen neuen Ansatz für die Datenauslese vor. In Gegensatz zu bisherigen, vergleichbaren Experimenten, deren Datenauslese auf globalen, hierarchischen Triggerentscheidungen beruhen, plant CBM freilaufende und selbstgetriggerte Elektronik einzusetzen. Die Daten werden in die nächste Verarbeitungsstufe hineingeschrieben anstatt sie, wie bisher, aus den Puffern der vorangehenden Verarbeitungsstufe auszulesen. Dieses neue Paradigma erfordert eine komplette Neuentwicklung der Ausleseelektronik.

Als Teil dieser Arbeit wurde eine Firmware für einen Auslesekontroller entwickelt, der als Schnittstelle für einen solchen freilaufenden und selbstgetriggerten ASIC<sup>1</sup>, den GET4 Chip, dient. Der GET4 Chip ist im *Time of Flight* (ToF) System des CBM Experiments für die Digitalisierung der analogen Signale aus dem Detektor vorgesehen. Seine Spezialität liegt in seiner besonders genauen Zeitauflösung.

Die angesprochene Firmware zum Auslesen des GET4 Chips wurde auf einem so genannten *Field Programmable Gate Array* (FPGA) entwickelt. FPGAs sind Microchips deren Verhalten nach Inbetriebnahme, also während des Einsatzes (englisch “in the field”, daher der Name) umprogrammiert werden kann. Dies bringt eine sehr hohe Flexibilität mit sich, Fehler im Design können behoben werden, ja sogar neue Funktionalität kann hinzu-

---

<sup>1</sup>ASIC steht für *Application Specific Integrated Circuit*

gefügt werden und das selbst nachdem die Hardware schon verbaut wurde. Der Einsatz von FPGAs für die Elektronik des CBM Experiments ist daher wünschenswert.

Leider sind FPGAs nicht in jedem Fall bedingungslos einsetzbar. Die einzigen FPGAs, die für das CBM Experiment in Betracht kommen sind kommerzielle Massenware, alle anderen Lösungen sind zu teuer. Diese FPGAs basieren entweder auf SRAM- oder auf Flash-Technologie und beide können nicht ohne weiteres in Umgebungen mit radioaktiver Strahlung eingesetzt werden. SRAM-basierte FPGAs werden durch so genannte *Single Event Upsets* (SEUs) erheblich gestört, Flash-basierte FPGAs werden zu schnell durch *Total Ionizing Dose* Effekte unbrauchbar.

Für SRAM-basierte FPGAs existiert eine Reihe an Techniken um Strahlungseffekte abzuschwächen, allerdings müssen diese für jeden speziellen Einsatzzweck sorgfältig neu analysiert werden.

Im Falle von CBM ist z.B. nicht klar, ob die höheren Kosten durch den zusätzlichen Ressourcenverbrauch redundant ausgelegter Logik nicht schon die generellen Vorteile des Einsatzes von FPGAs aufwiegen. Es ist nicht einmal klar, ob Techniken die schon erfolgreich in Weltraummissionen angewandt wurden auch bei den viel höheren Strahlungsniveaus des CBM Experiments noch effektiv funktionieren. In der vorliegenden Arbeit wurden deshalb die existierenden Techniken analysiert, geeignete Techniken für den oben angesprochenen Auslesekontroller ausgewählt und umgesetzt und schließlich deren Wirksamkeit in Strahltests überprüft.

## Stand der Technik

Es existieren verschiedene Ansätze um ungewollte Strahlungseffekte in Elektronik zu unterdrücken. Der dominierende Störfaktor beim Betrieb von SRAM-basierten FPGAs in einer Umgebung mit ionisierender Strahlung sind so genannte *Single Event Upsets* (SEUs). SEUs treten auf, wenn ionisierende Strahlung auf SRAM Zellen trifft. Der in der SRAM Zelle gespeicherte Wert kann dann durch die Strahlung verändert werden und man spricht von einem SEU oder Bitflip. Die wichtigsten Gegenmaßnahmen für FPGAs, die auch für den Einsatzzweck bei CBM in Betracht kommen sind *Triple Modular Redundancy* (TMR) in Kombination mit *Scrubbing*. Beide Techniken sollten kombiniert werden um einen relativ effektiven Schutz gegen SEUs zu erreichen.

Bei TMR wird die Logik dreifach instanziiert und die drei Ausgänge mit einer Mehrheitsentscheidung zusammengefasst. So kann einer der drei Logikteile beschädigt werden und die Ausgänge werden immer noch mit einer Mehrheit von 2:1 richtig gewertet. Der Nachteil liegt darin, dass die Schaltung mindestens um den Faktor drei größer wird und damit mindestens dreimal so viele Ressourcen verbraucht. Realistisch ist allerdings ein noch höherer Wert, in der Literatur wird oftmals der Faktor Sechs genannt [WRGC03b], [WRGC03a], [MMPW07], [Wir14, Seite 30]. Dieses Problem beschränkt sich nicht nur auf die Kosten für die zusätzlichen Ressourcen, sondern hat auch zur Konsequenz, dass TMR auf lange Sicht nicht ohne einen Mechanismus zur Reparatur von Fehlern funktioniert. Es kann sogar gezeigt werden, dass TMR ohne Reparaturmechanismus

---

die Fehleranfälligkeit eines Designs auf lange Sicht verschlechtert [Wir14].

Um die Reparaturmechanismus zu verstehen muss zuerst der Aufbau eines FPGAs verstanden sein. Die Funktionalität eines FPGAs wird zum großen Teil von Speicherzellen (SRAM) bestimmt, dabei sind dynamische und statische Speicherzellen zu unterscheiden. Dynamische Zellen speichern den derzeitigen Status der Logik, stellen also Register und integrierten Arbeitsspeicher dar. Sie ändern ihren Zustand während des Betriebs. Statische Zellen bestimmen die Funktionalität der kombinatorischen Logik, sie bilden Logikgatter und deren Verschaltung ab. Sie ändern ihren Zustand während des Betriebs nicht.

Die dynamischen Zellen können durch geschickte Rückkopplung der Resultate der erwähnten Mehrheitsentscheider korrigiert werden. Der Großteil der Speicherzellen eines FPGAs ist allerdings statischer Natur. Der üblicherweise eingesetzte Reparaturmechanismus für den statischen Teil der SRAM Zellen ist *Scrubbing*. Da die statischen Zellen ihren Wert während des Betriebs nicht verändern, kann man diesen Wert periodisch aktualisieren. Falls die Logik durch einen SEU beschädigt wurde, wird dieser wieder auf den korrekten Wert zurückgesetzt.

Xilinx FPGAs waren die ersten FPGAs, die solch eine Aktualisierung der statischen Speicherzellen im laufenden Betrieb erlauben. Das Ganze läuft im Hintergrund ab, ohne dass dabei das Design angehalten werden muss. Dazu muss ein Konfigurationskontroller entwickelt werden, der periodisch oder nach einem anderen Algorithmus, die statischen SRAM Zellen aktualisiert, aber die dynamischen Zellen nicht anfasst. Solch ein Kontroller war aus einem früheren Projekt der Arbeitsgruppe schon vorhanden [Eng09] und konnte mit recht geringem Aufwand wiederverwendet werden.

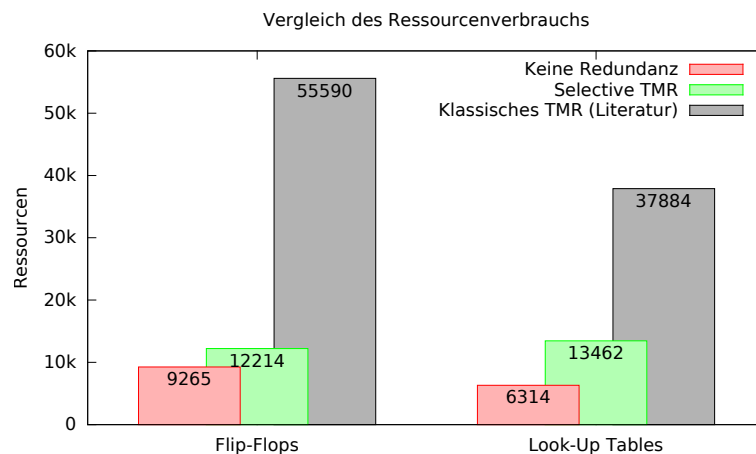
## Ansatz und Verwirklichung

Das größte Problem bei der Implementierung eines Strahlentoleranten Auslesekontrollers für CBM ist die vollständige Umsetzung von TMR. Eine Versechsfachung der Ressourcenbedarfs ist schon aus ökonomischer Sicht nicht realisierbar. Hinzu kommt, dass bei steigender Komplexität des Designs der Aufwand einer TMR Implementierung immens zunimmt. Zum Beispiel ist eine einfache Signalweitergabe in einen Logikbereich der mit einem anderen Takt betrieben wird nicht wirklich möglich wenn man TMR absolut vollständig implementieren möchte.

Eine Tatsache die der Entwicklung entgegenkommt ist allerdings, dass bei CBM keine absolut hundertprozentige Zuverlässigkeit erforderlich ist. Die Detektoren selbst arbeiten schon mit Effizienzen in der Größenordnung von nur etwa 95%, solange durch strahlungsbedingte Fehler in der Ausleseelektronik kein signifikanter Beitrag hinzukommt, können einzelne Fehler geduldet werden.

Für den CBM Auslesekontroller wurde deshalb entschieden, nur die wichtigsten Teile mit TMR zu schützen. Es wurde die Kontrollogik und ihre Statusregister mit TMR implementiert, der gesamte Datenpfad, der etwa 90% des Designs ausmacht, wurde ohne Redundanz implementiert. Dieser Ansatz wurde als *Selective TMR* bezeichnet. Zur Er-

kennung von Fehlern wurden die Daten mit einer CRC Checksumme versehen, korrupte Daten können also zwar nicht korrigiert, aber immerhin erkannt werden. Abbildung E.1 zeigt einen Vergleich des Ressourcenverbrauchs für ein Design ohne Redundanz, ein Design das mit *Selective TMR* implementiert wurde und eine Hochrechnung für ein Design mit vollständigem TMR.

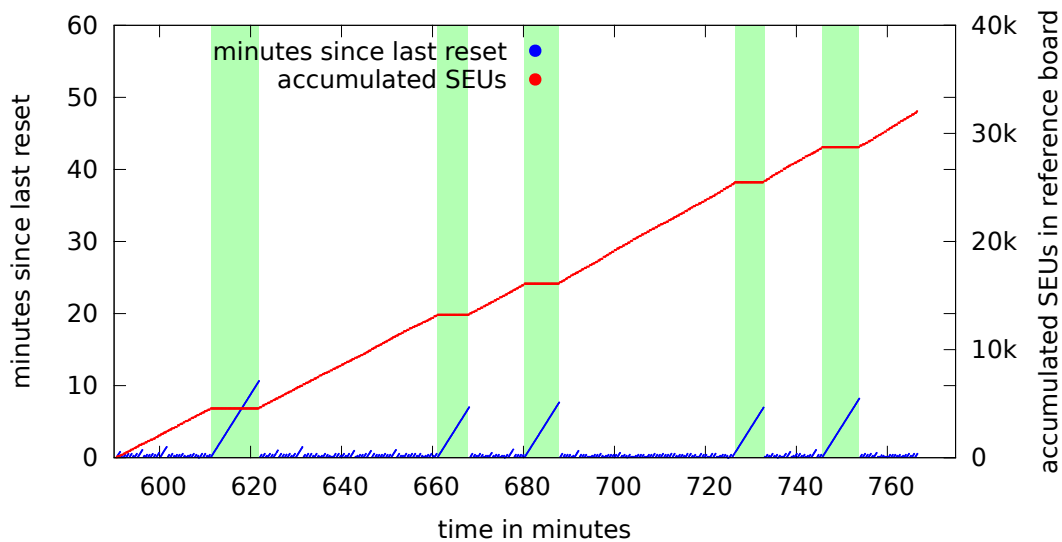


**Abbildung E.1.:** Ressourcenverbrauch (Flip-Flops und Look-Up Tables) der GET4 Auslesefirmware in der Version wie sie während des Strahltests 2012 verwendet wurde. Die Daten basieren auf den Zahlenwerten aus der Ausgabedatei des Xilinx Programms *MAP*. Der rote Balken stellt die verbrauchten Ressourcen der ursprünglichen Firmware ohne redundante Logik dar, die grünen Balken beziehen sich auf die Firmware deren Logik mit *Selective TMR* versehen wurde. Zum besseren Vergleich wurde auch eine Abschätzung des Verbrauchs bei Verwendung herkömmlicher TMR Lösungen als graue Balken hinzugefügt. Nach eingehender Literaturrecherche wurde als Abschätzung für die grauen Balken der sechsfache Ressourcenverbrauch der ursprünglichen Firmware ohne redundante Logik verwendet.

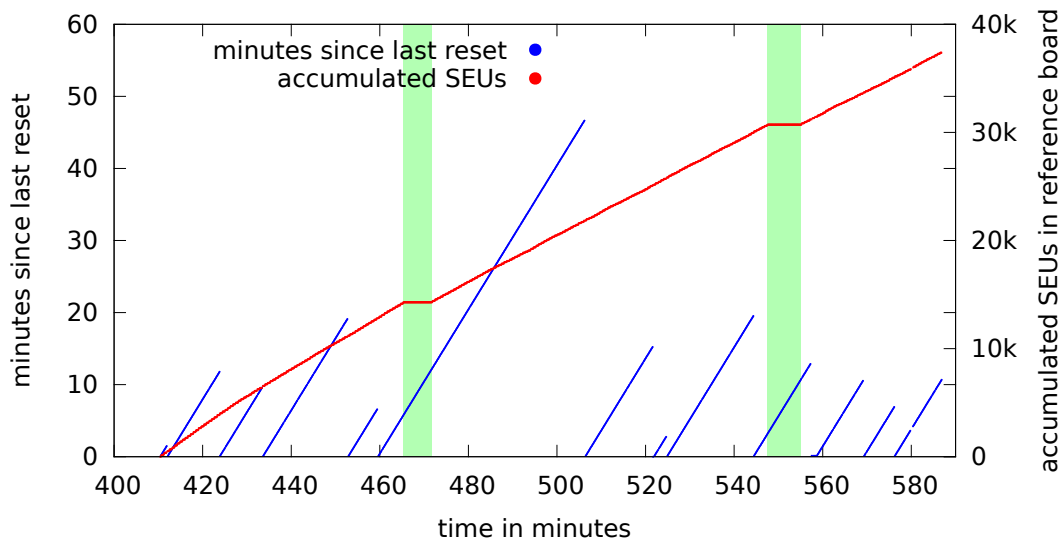
Ohne weitere Maßnahmen kann es durch die fehlende Redundanz an manchen Stellen des Designs vorkommen, dass bestimmte Logikteile, z.B. solche die als endlicher Zustandsautomat realisiert wurden, nicht mehr in ein korrektes Verhalten zurückfallen, selbst wenn die ursprüngliche Fehlerquelle (der SEU) korrigiert wurde. Diese Tatsache wurde schon während der Entwicklung berücksichtigt und die endlichen Zustandsautomaten und vergleichbare Logik so konzipiert, dass sie aus jedem beliebigen Zustand immer in ein korrektes Verhalten zurückfallen. Auch die höheren Designebenen wurden so realisiert, dass keine Verklemmung auftritt. Zum Beispiel warten die Kommunikationsprotokolle durch Einsatz von Timeout-Funktionalität, auch bei nicht konformem Verhalten der Gegenstelle, niemals unendlich lange auf eine korrekte Antwort.

## Ergebnisse

Da nicht ganz klar war, wie effizient der Strahlenschutz noch ist, wenn nicht alle Teile mit TMR implementiert wurden, wurde das Design am Forschungszentrum Jülich direkt



(a) Scrubbing ist deaktiviert. Das Setup muss innerhalb weniger als einer Minute komplett neu gestartet werden, nur innerhalb der Zeitfenster als kein Teilchenstrahl verfügbar war läuft es stabil.



(b) Scrubbing ist aktiviert. Das Setup läuft über mehrere Minuten stabil.

**Abbildung E.2.:** Der rote Graph zeigt die Anzahl an SEUs, die seit Beginn des Testlaufs im Referenzboard akkumuliert wurden, der blaue Graph repräsentiert die Zeit, die seit dem letzten Neustart des Setups vergangen ist. Jedes mal wenn der blaue Graph auf die Null zurückfällt wurde ein nicht behebbarer Fehler detektiert. Während den in grün unterlegten Zeitfenstern wurde vom Synchrotron aus technischen Gründen kein Teilchenstrahl geliefert. Dieses Bild wurde schon in [MGO<sup>+</sup>13a] und [MGO<sup>+</sup>13b] veröffentlicht.

in einen Protonenstrahl montiert und dort in Betrieb genommen. Die Ergebnisse sind in Abbildung E.2 zu sehen.

Mit Hilfe der Ergebnisse dieser Strahltests konnte die Effizienz der Strahlenschutz-Techniken überprüft werden. Es zeigte sich, dass die Totzeit des ToF-Detektors aufgrund strahlungsbedingter Elektronikfehler um mehr als den Faktor 20 verbessert werden kann. Der Anteil an korrupten Daten kann von etwa 3-4% auf 0.1% verbessert werden. Das bedeutet, dass SRAM-basierte FPGAs bei CBM-ToF eingesetzt werden können, wenn die entsprechenden Maßnahmen um Strahlungseffekten in Elektronik entgegenzuwirken so implementiert werden wie sie in der Arbeit beschrieben wurden.