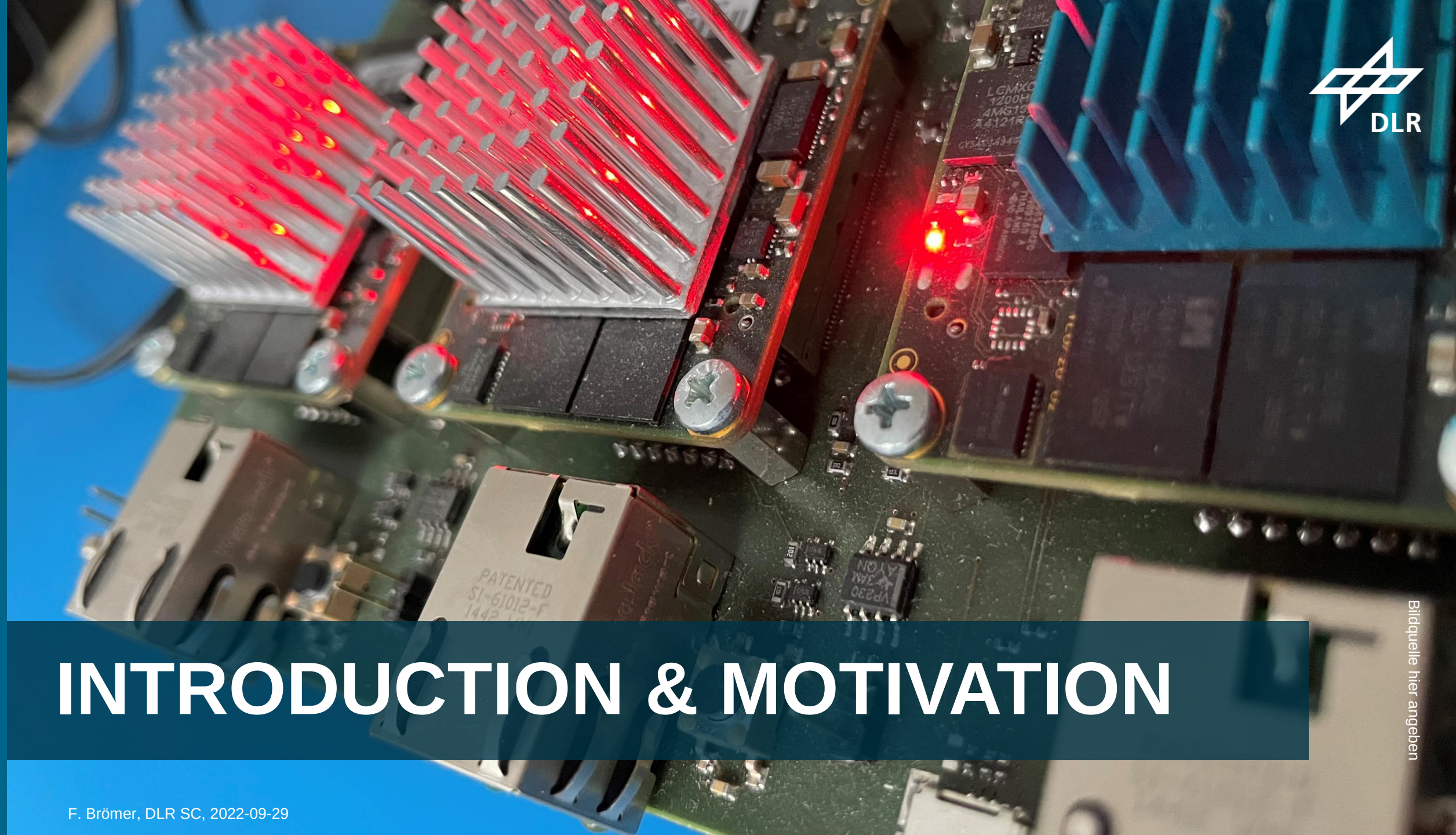
FPGA-BASED FAULT INJECTOR FOR SEU-ROBUSTNESS ANALYSIS OF SCOSA

DLRK 2022 Dresden

F. Brömer, P. Kenny, A. Lund, C. Gremzow , D. Lüdtke

Outline

- Introduction & Motivation
- The ScOSA Flight Experiment
- Fault Injection
- Fault Detection
- Scientific & technical goals
- Conclusion & Outlook

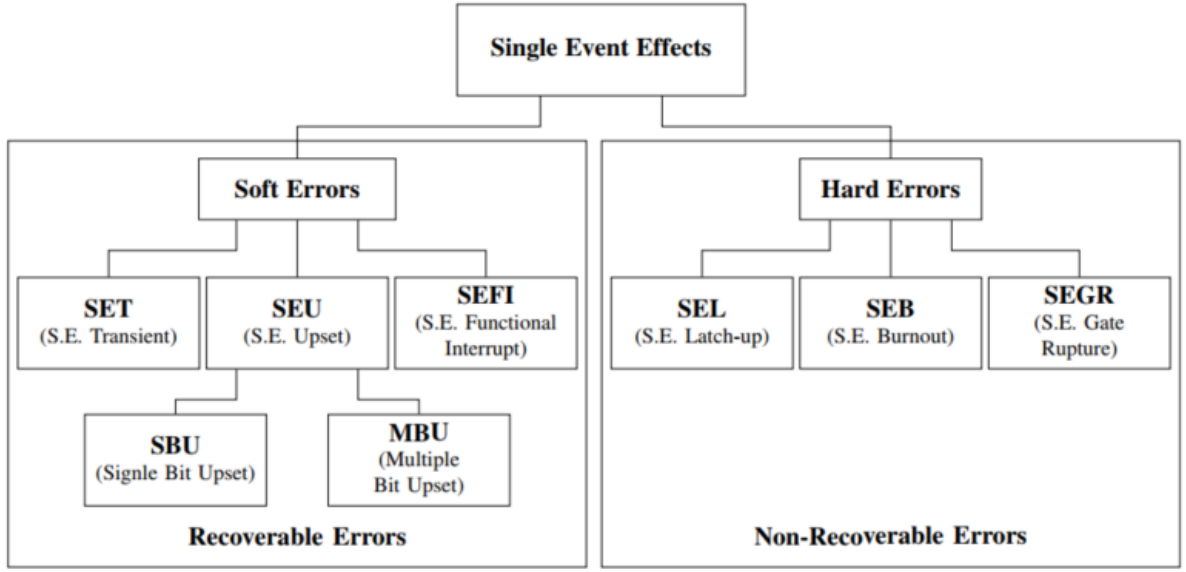


INTRODUCTION & MOTIVATION

Bildquelle hier angeben

Introduction & Motivation

- More Commercial-off-the-shelf (COTS) hardware
- Radiation in space and higher atmospheres → Single Event Upsets (SEU)
- Mitigation through software fault tolerance



Source: 10.1109/NORCHIP.2017.8124960

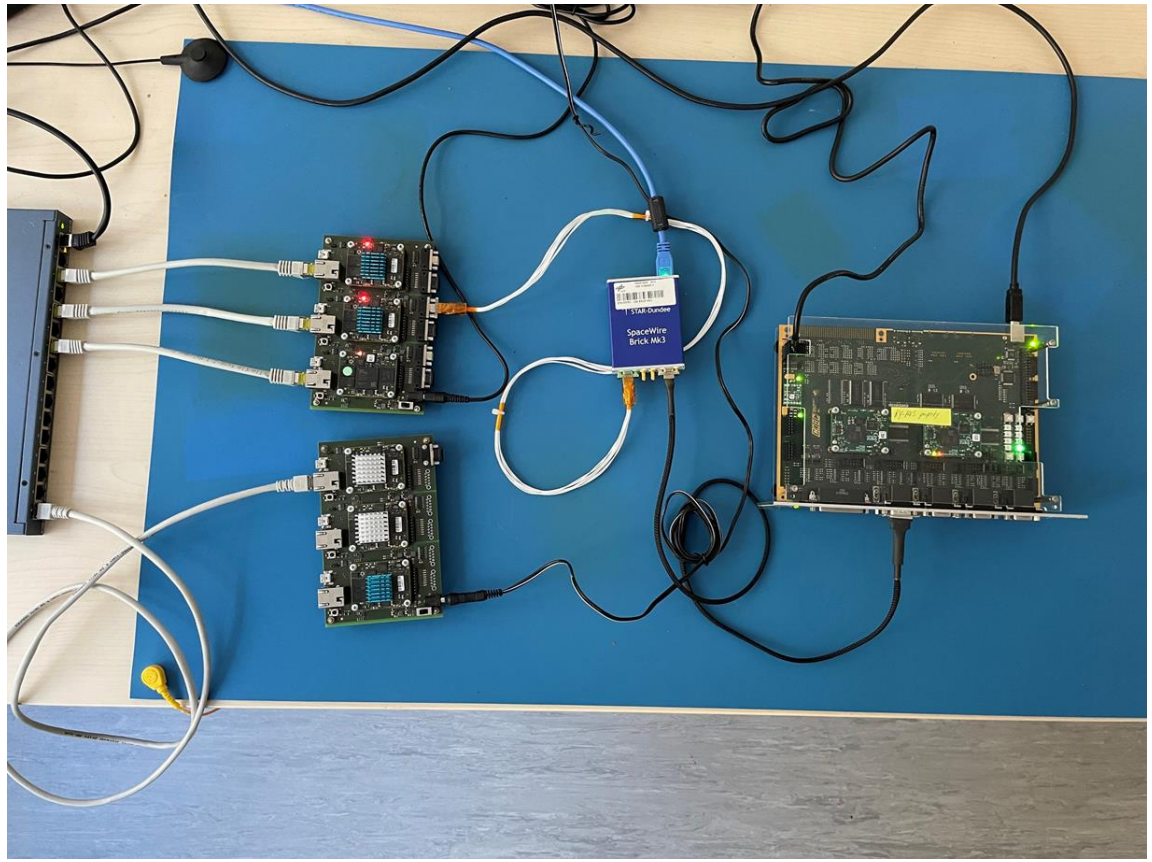
- How to test SEU tolerance?
  - Particle accelerators → expensive
  - Full physical simulation → complex
  - Fault injection

THE SCOSA FLIGHT EXPERIMENT

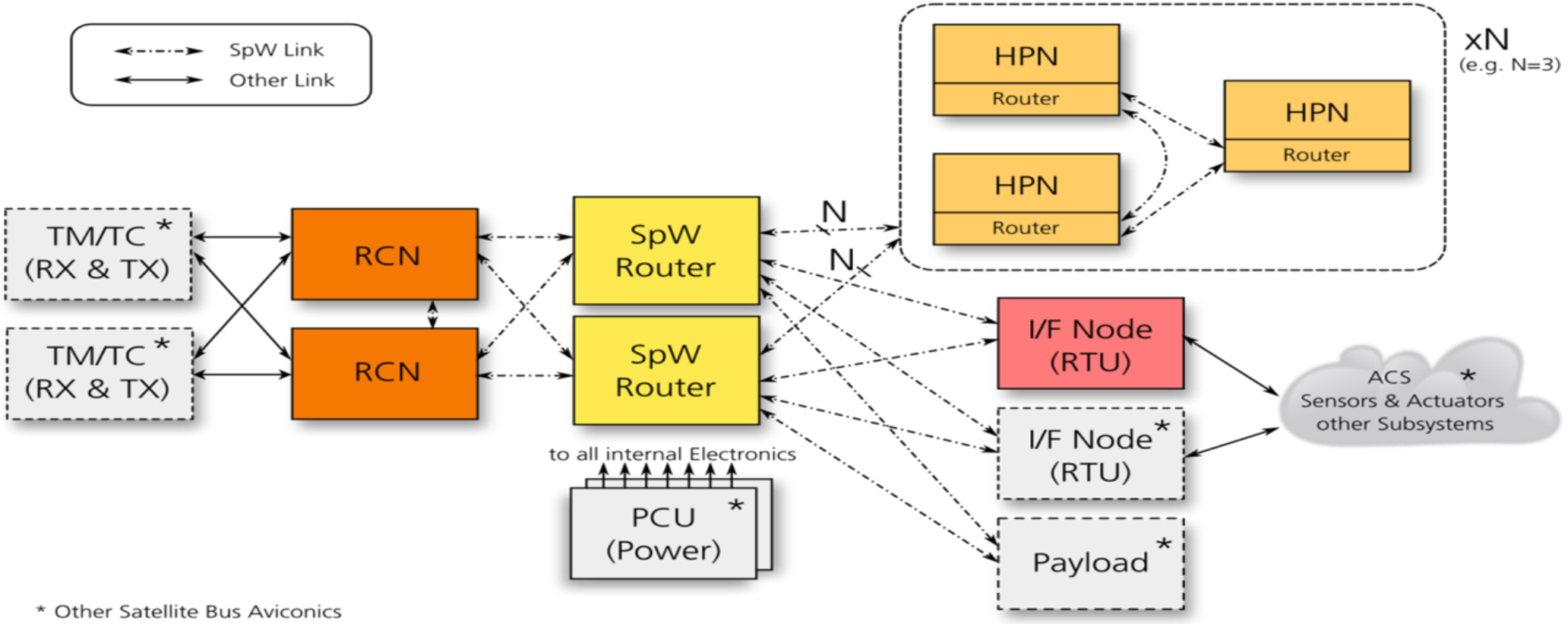
Bildquelle hier angeben

The ScOSA Flight Experiment

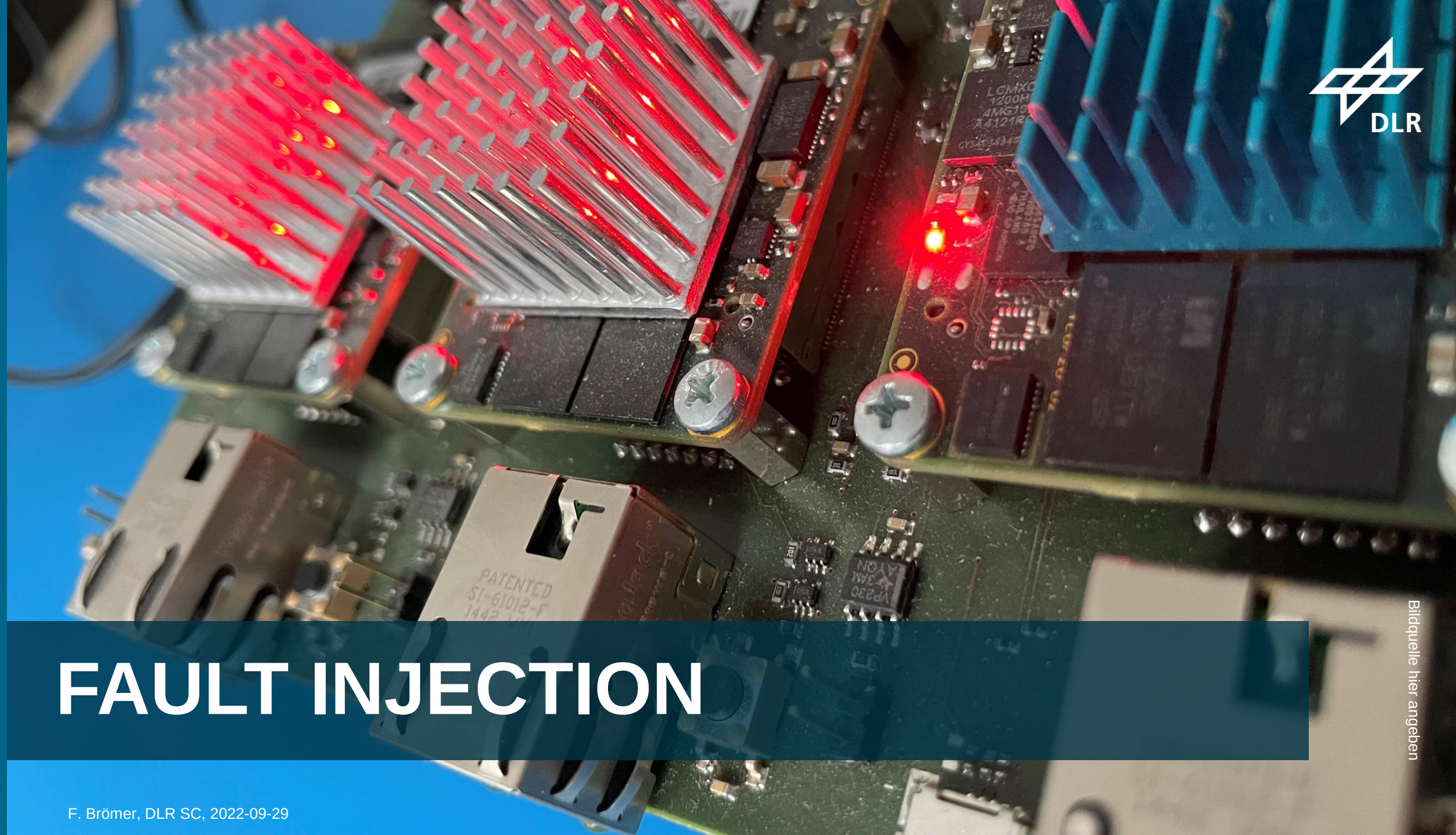
- DLR project
- **Sc**alable **O**n-board Computer for **S**pace **A**vionics
- Evaluating ScOSA OBC in flight mission
- ScOSA OBC:
  - Distributed, heterogeneous architecture
  - COTS & radiation-hard processors
  - Reconfiguration of tasks in case of faults
- Collection SEU rates in flight



The ScOSA Flight Experiment



Source: 10.1007/s12567-021-00371-7



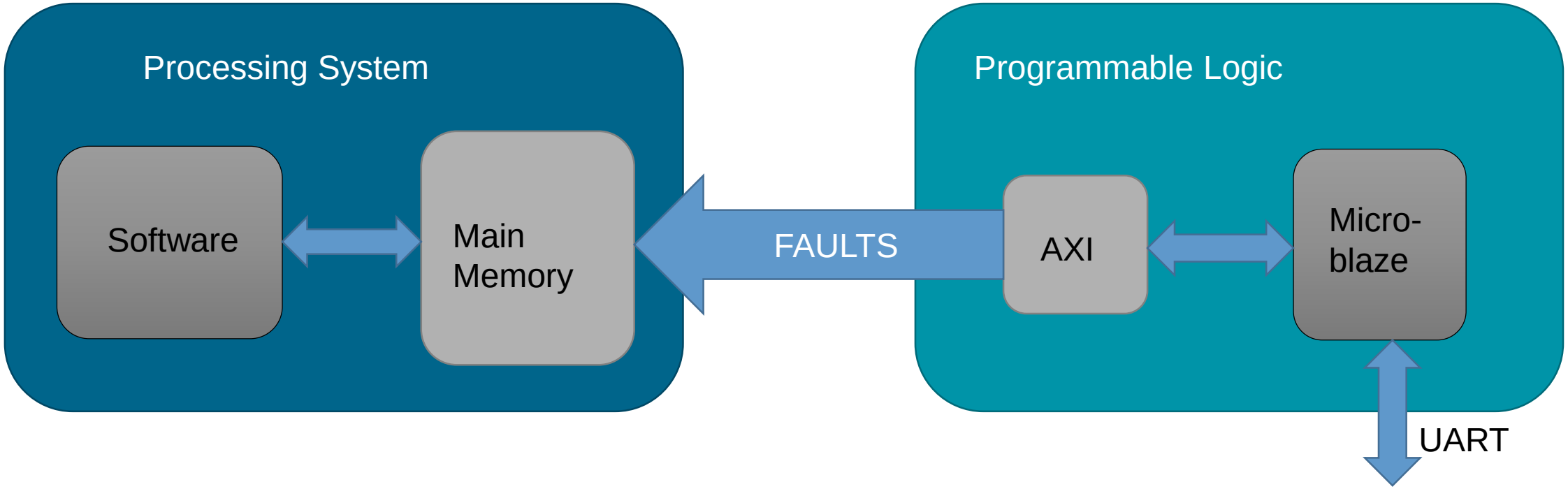
FAULT INJECTION

Bildquelle hier angeben

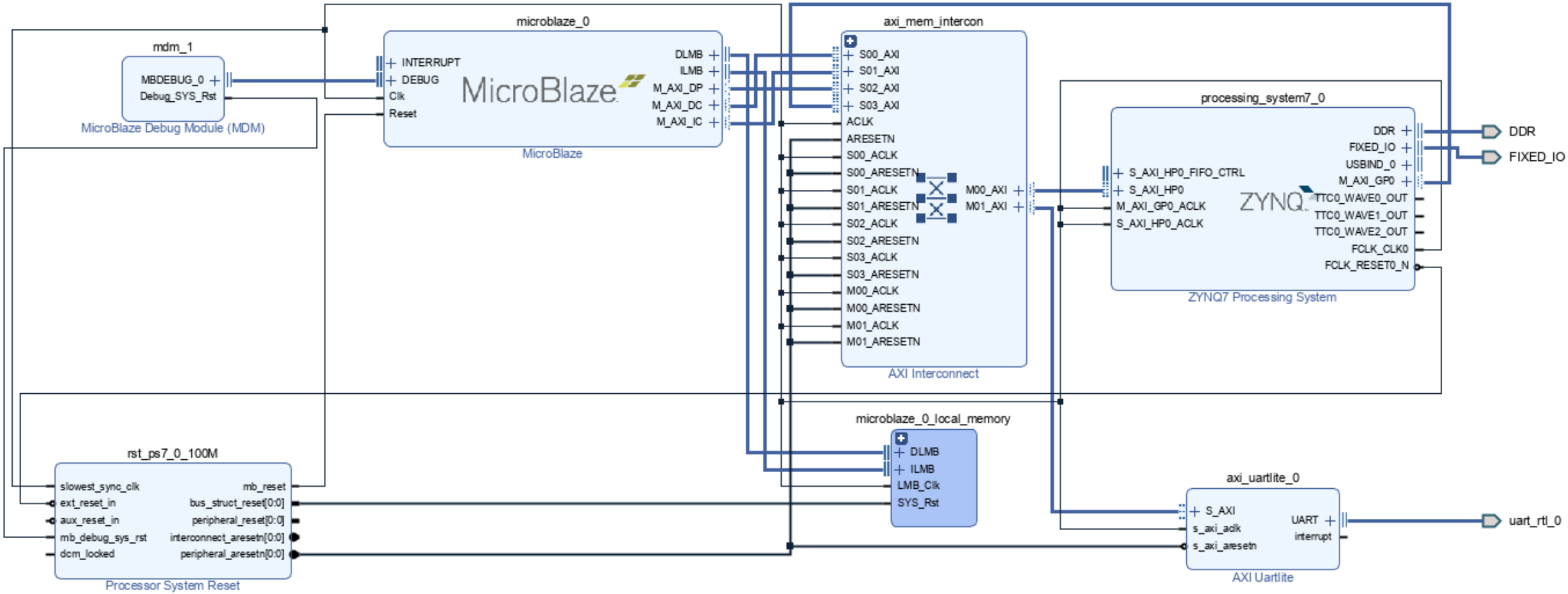
Fault injection

- How?
  - Software → influence to tested software?
  - External
    - Other computer
    - FPGA → Zynq 7020 (ScOSA)
- Fault injection into FPGA components also possible (e.g. SpaceWire router)
- Interface: AXI
  - High performance port
  - Accelerator port

Design


Processing System
Software
Main
Memory
FAULTS
Programmable Logic
AXI
Micro-
blaze
UART

Implementation - FPGA



Implementation - Microblaze

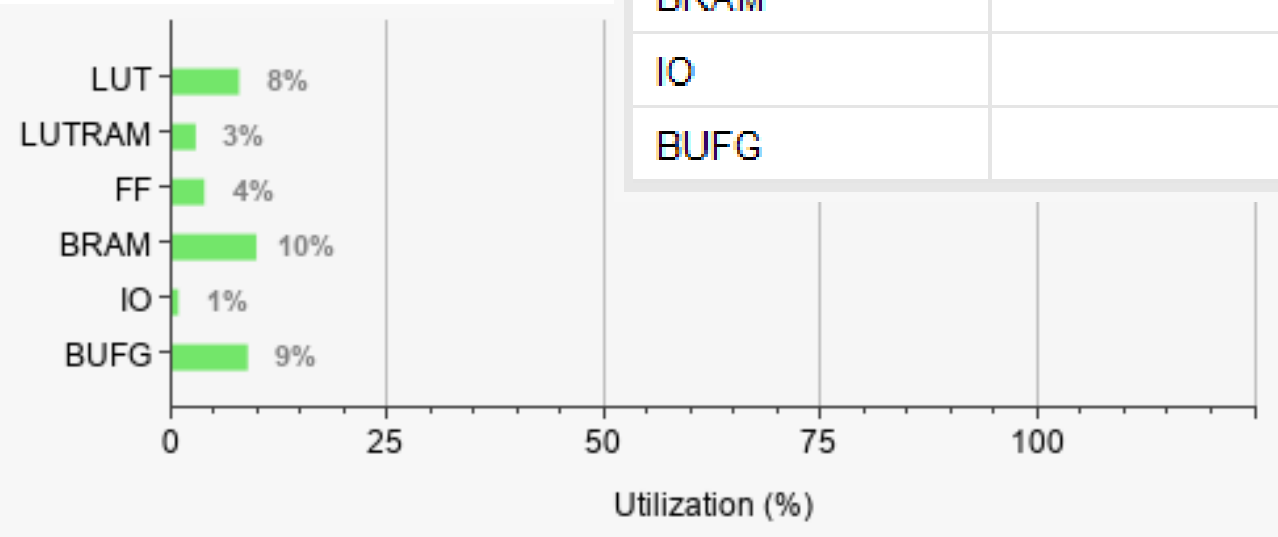
- Injection into random memory areas in a given range

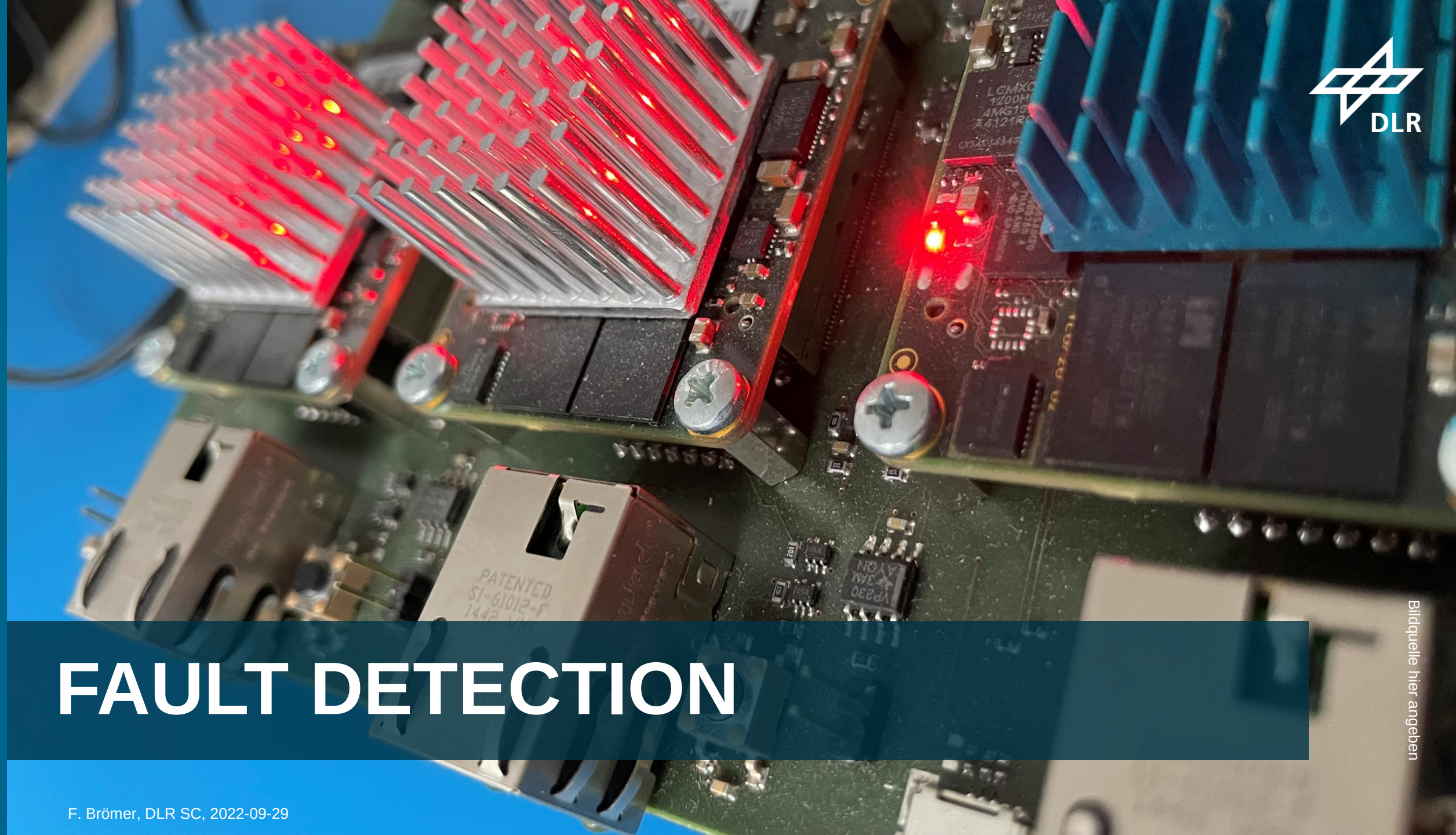
- Pseudo-randomness based on seed
  - Reproduceable

```
srand(rand_seed);
for(int i; i<num_faults; i++){
    random=rand();
    fault_address = start_address + (random % length);
    random=rand();
    fault_position = random % 8;
    bitflip(fault_address, fault_position);
}
```

Implementation – Ressource Utilization

Resource	Utilization	Available	Utilization %
LUT	3992	53200	7.50
LUTRAM	462	17400	2.66
FF	4333	106400	4.07
BRAM	14	140	10.00
IO	2	200	1.00
BUFG	3	32	9.38





FAULT DETECTION

Bildquelle hier angeben

Fault Detection

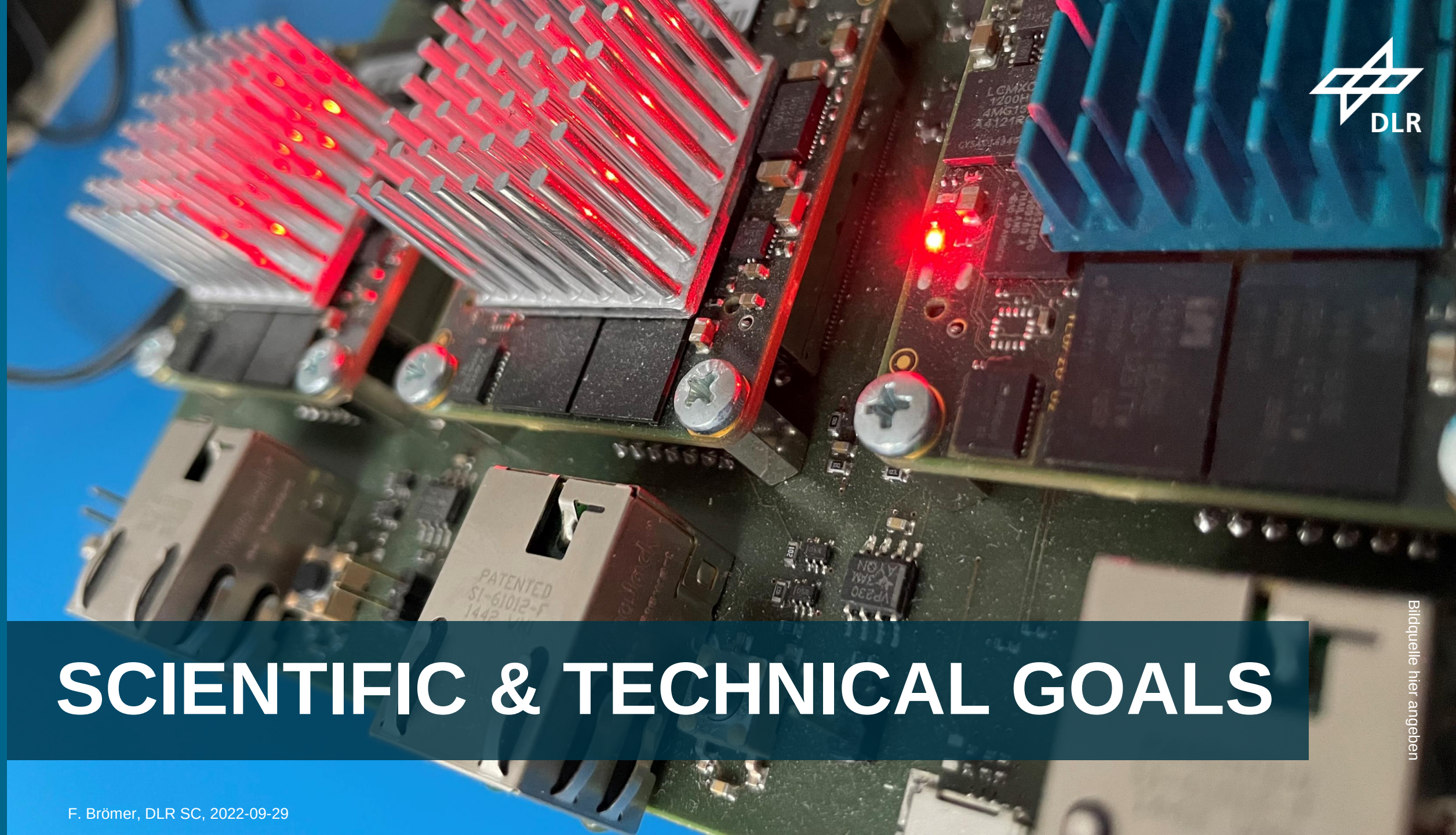
- Injection 

- Detection:
  - Part of ScOSA FE
  - Own software application
  - Monitoring of memory areas
- Characterization of Zynq 7020

```
void detectSeu() {
    monitoredMemory.fill(PATTERN_1);

    uint8_t* errorLocation = nullptr;
    const uint8_t compare = PATTERN_1;
    unsigned int counter = 0u;
    while(std::getchar() != 'q') {
        errorLocation = std::find_if_not(monitoredMemory.begin(),
        monitoredMemory.end(),
        [](uint8_t value){return value == PATTERN_1;});

        if (errorLocation != monitoredMemory.end()) {
            std::cout << "SEU detected at index " << std::dec << reinterpret_cast<uint32_t>(errorLocation) - reinterpret_cast<uint32_t>(&monitoredMemory)
                      << ". Found value 0x"   << std::hex << static_cast<int>(*errorLocation)
                      << ", expected 0x"      << std::hex << static_cast<int>(PATTERN_1)      << std::endl;
            monitoredMemory.fill(PATTERN_1);
        }
    }
}
```



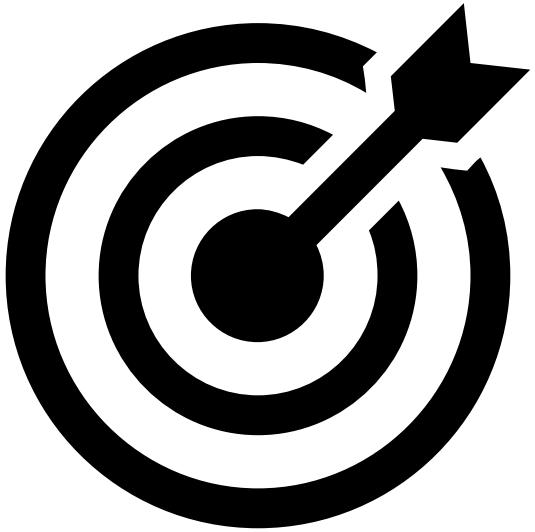
SCIENTIFIC & TECHNICAL GOALS

Bildquelle hier angeben

Goals

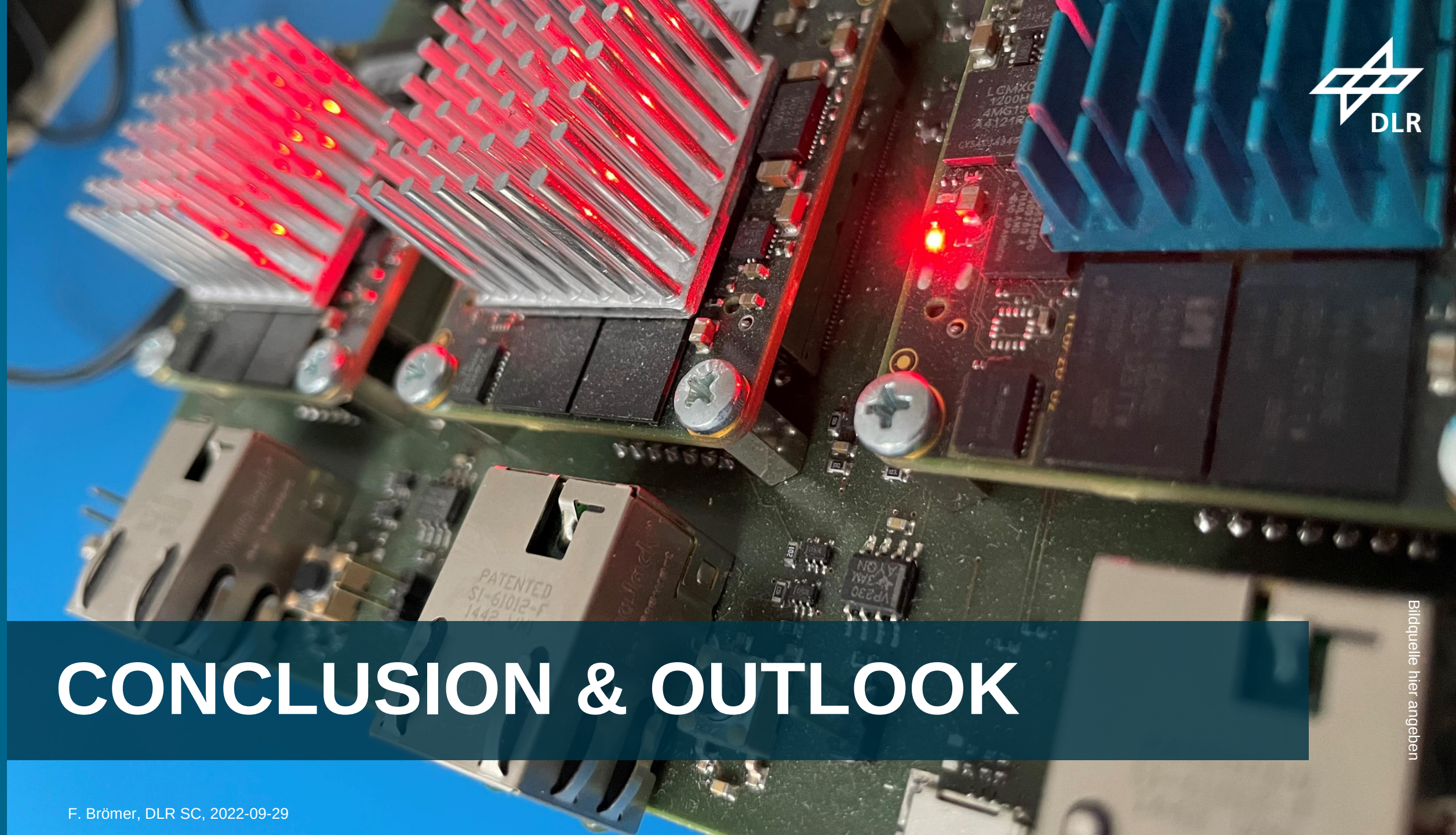
Scientific Goals

- Evaluation of fault injection
- Characterization of COTS hardware
- Collecting SEU rates in flight



Technical Goals

- Testing ScOSAs fault tolerance



CONCLUSION & OUTLOOK

Bildquelle hier angeben

Conclusion & Outlook

Conclusion

- Implementation of fault injector based on FPGA
  - Microblaze
- Fault detection

Outlook

- Integration of fault injection & detection
- Fault injection into FPGA components