AUTOMATED FAULT INJECTION FOR SYSTEMC & SYSTEMC AMS

Stephan Gerth, Fraunhofer IIS/EAS
Coside User Group Meeting 2016

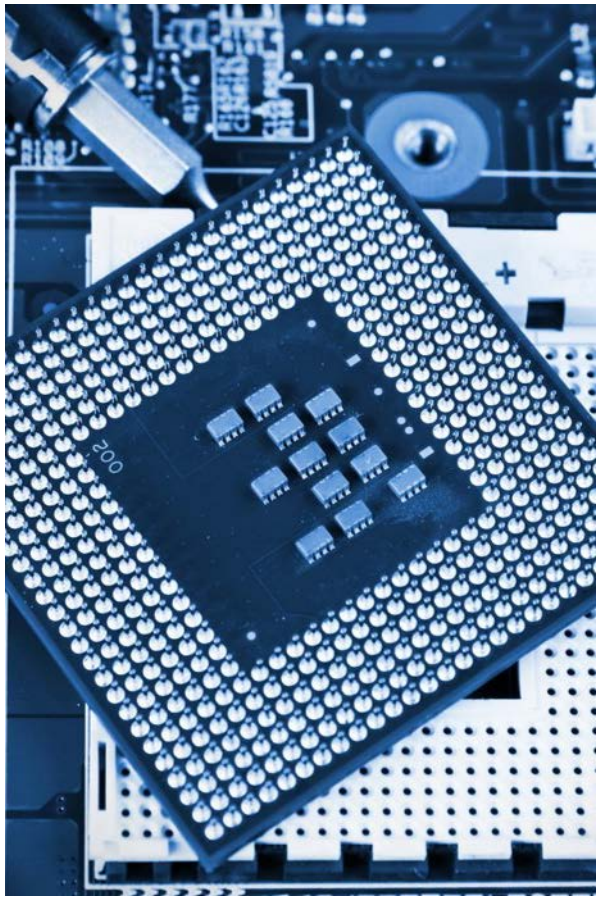
© Photo: Sergey Nivens / Fotolia

© Fraunhofer IIS/EAS

Fraunhofer IIS

Current state of fault injection
What limitations do we want to overcome?

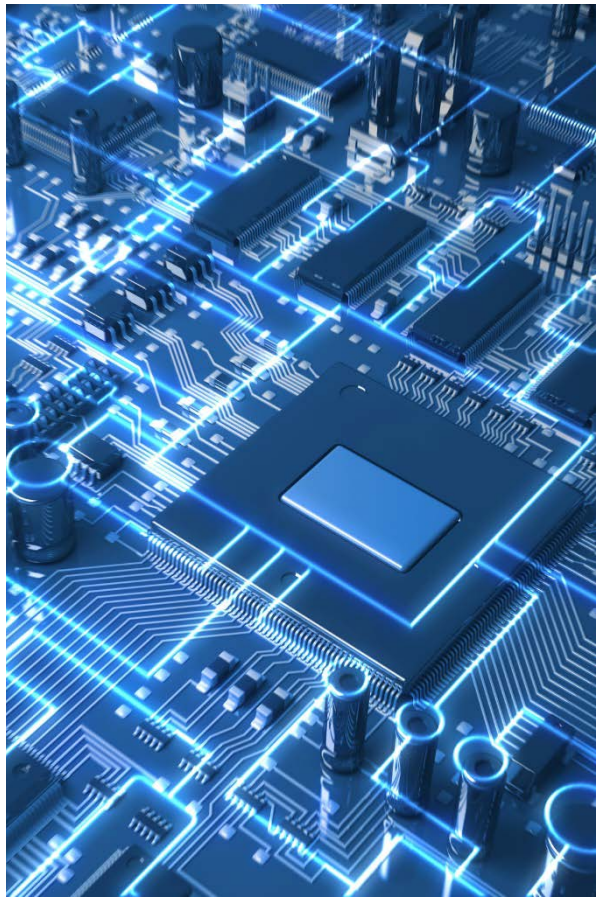
- Test a given DUT for e.g.
  - Diagnostic coverage
  - Functional safety
  - Fault tolerance and fault latency
- Solutions available mostly for basic single occurrence faults
  - Stuck-at-*
  - Bit-Flips



© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

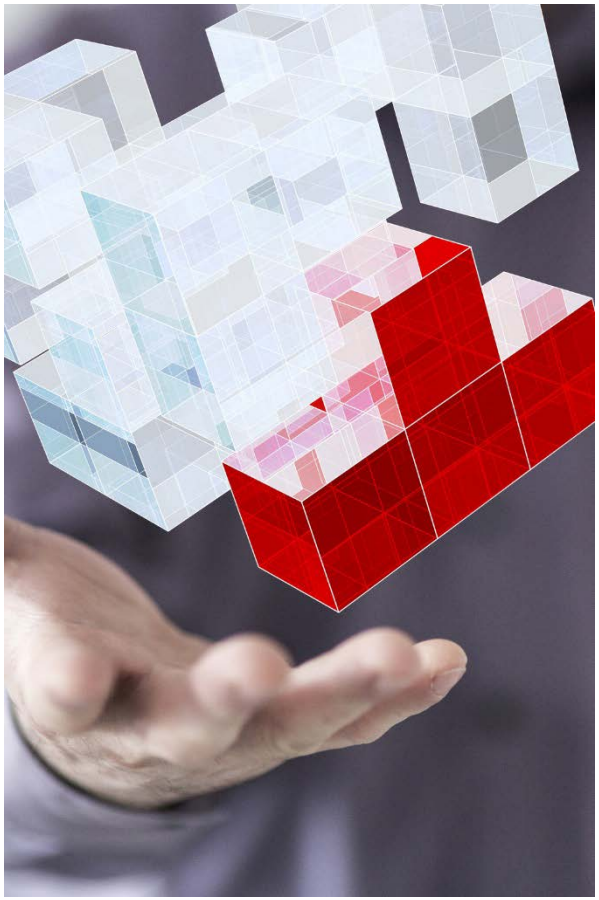
Challenges
What do we aim for?

- Large designs
- High-level faults
- Intermittent faults
- Multiple faults scattered throughout the design
- No test artefacts within designs
- Different MoCs



© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

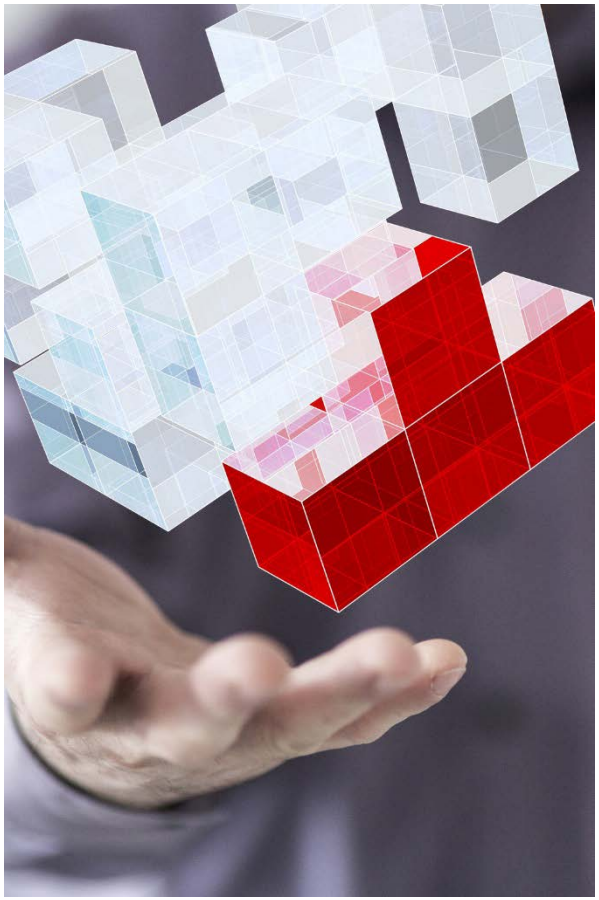
Approach
How do we want it?



- Create an allround solution for fault injection in SystemC & SystemC AMS
- Avoid DUT changes, obey separation of concerns
  - Keep the DUT strictly to its function
  - Don’t introduce artefacts for testing in the DUT code
  - Fault injection should only appear in tests
- Controlled distribution of faults over the DUT
  - Where does occur what kind of fault
  - At which rate does it occur (permanent, intermittent, …)

© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

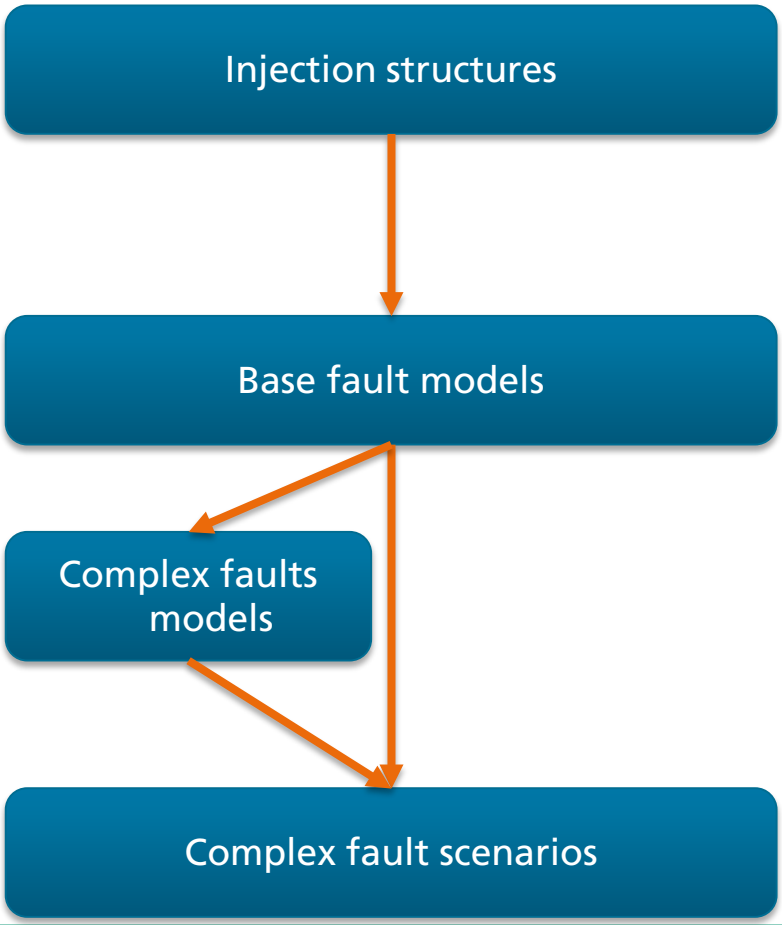
Solution
How to implement?



- Provide a library with
  - fault injection mechanisms
  - Base models
  - Support for different MoCs
  - Statistical configurability for occurrence rates and region
- Enable fault specializations for
  - high-level faults
  - User specific needs
- Insert faults only dynamically
  - Driven by the testbench
  - Keeping the DUT untouched

© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fault injection library
Structured approach open for extensions

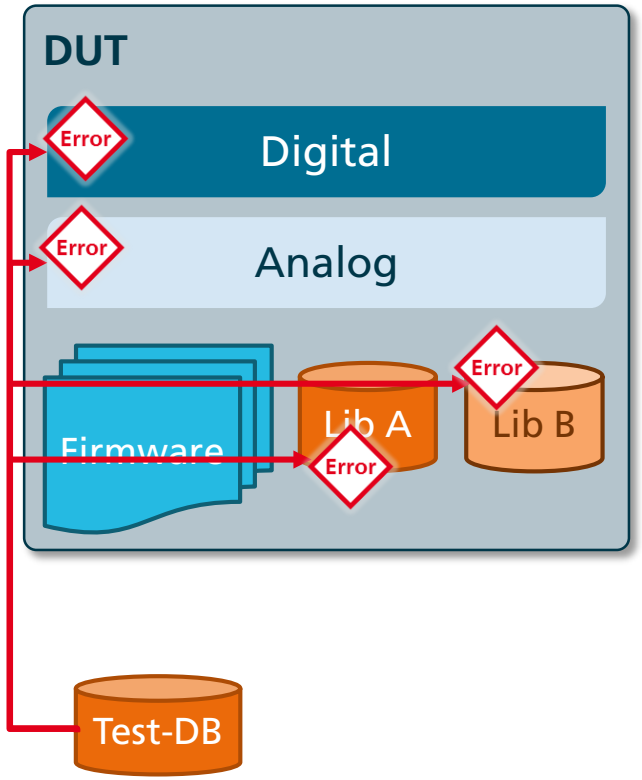


- Injection mechanisms for different MoCs
- Crosstalk, Delays, Glitching, Open, Stuck-At
- User specific detailing
- Combination & configuration for specific applications

© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Dynamic fault insertion
Don‘t touch the DUT code base!

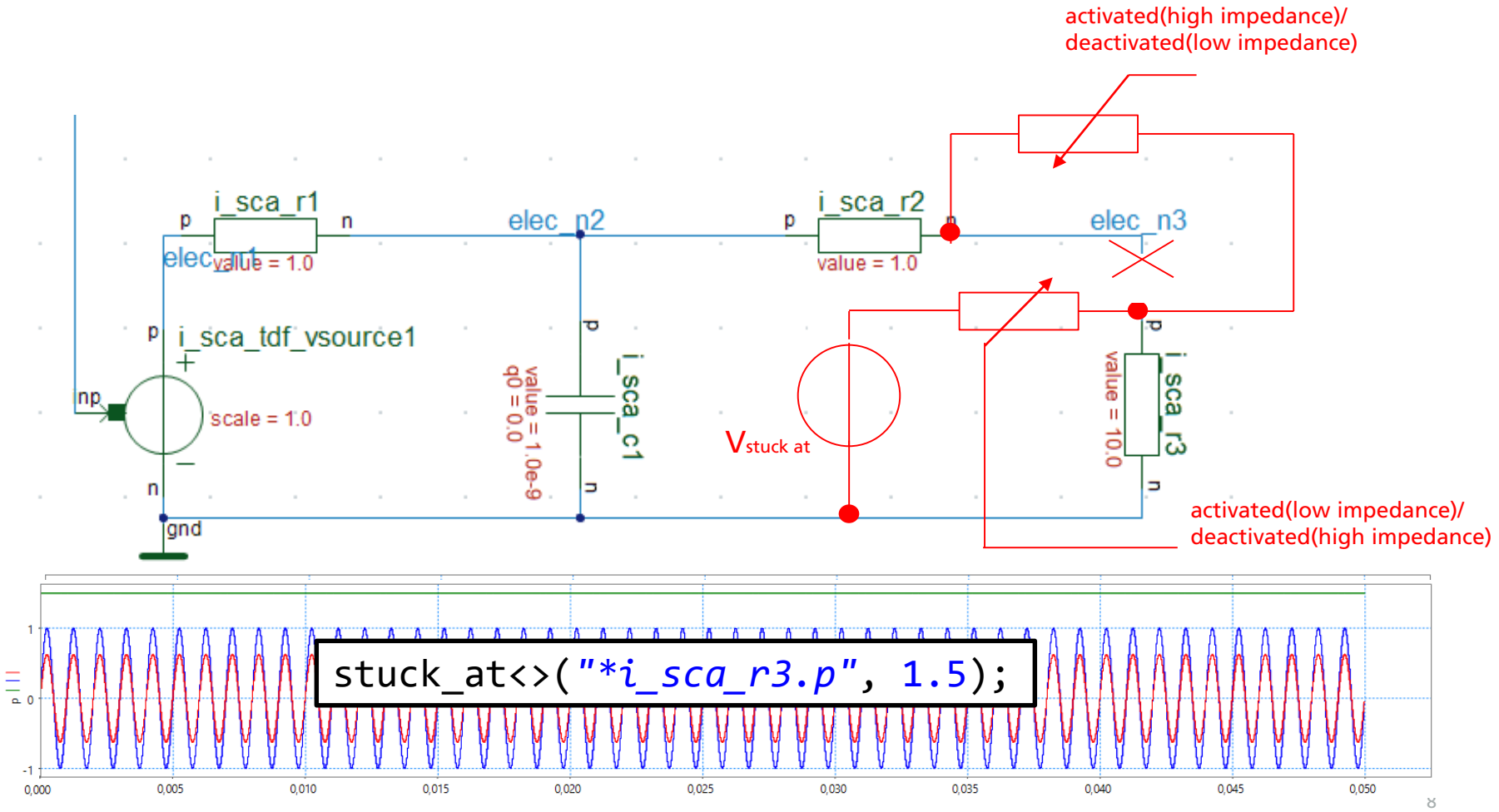
- Keep description of faults in tests
- Instrument the DUT during runtime
- Avoid bugs introduced by testing artefacts
- Faults can be active concurrently regardless of their MoC domain



© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fault Injection for Electrical Linear Network designs

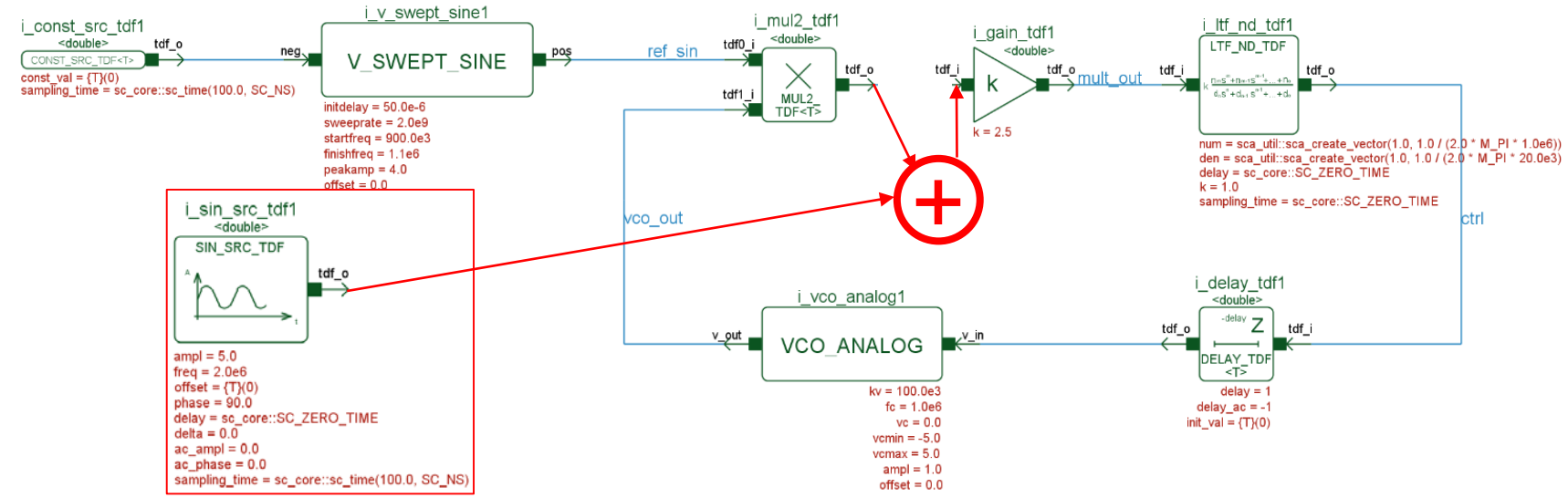
Stuck-at-value for basic filter

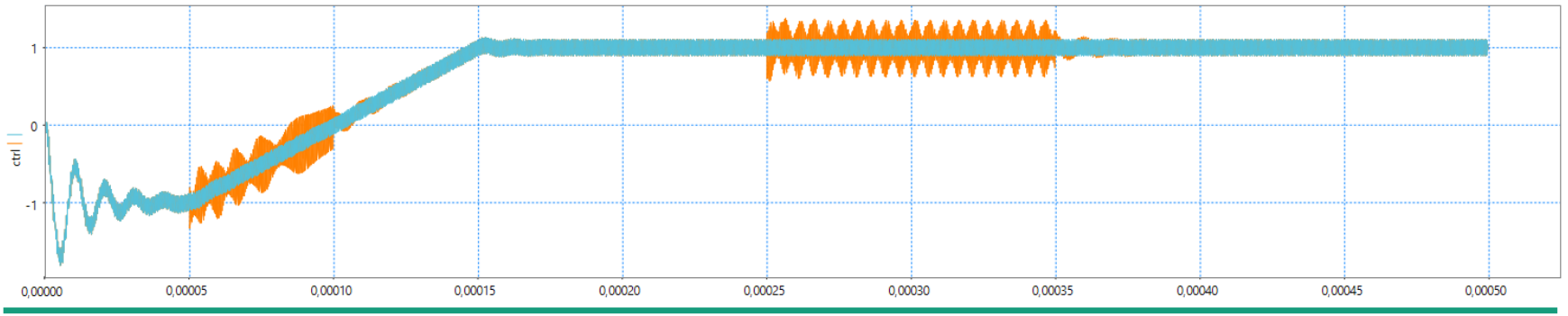


Fraunhofer IIS

Fault Injection for Timed Data Flow designs

Cross-talk for PLL model



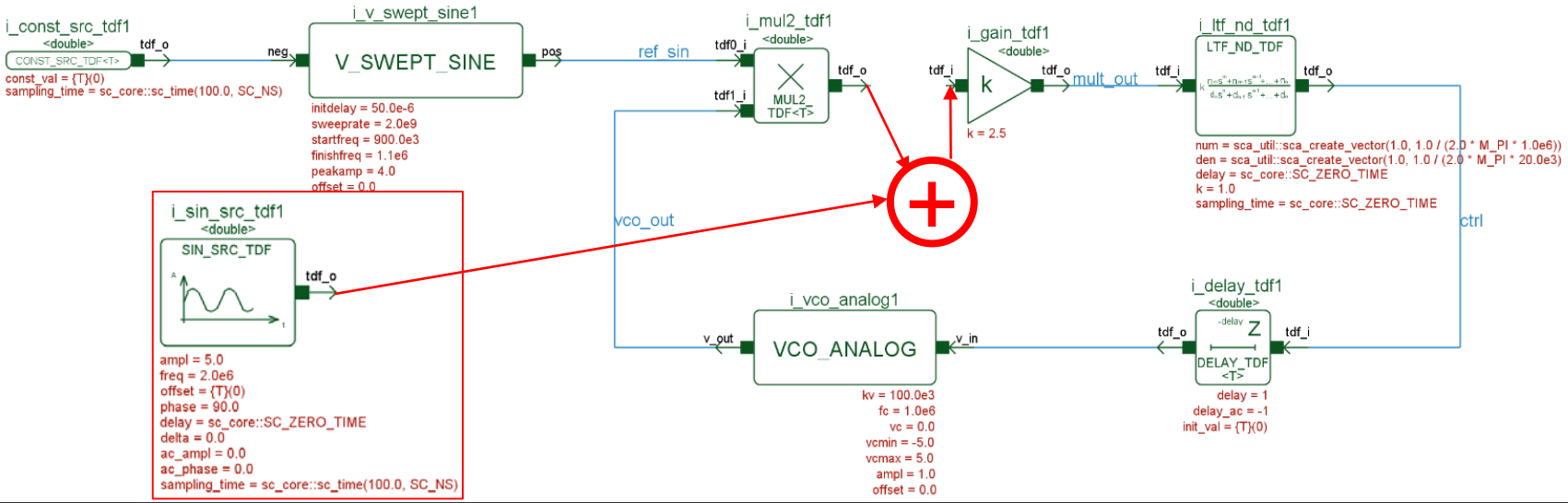


© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fraunhofer IIS

Fault Injection for Timed Data Flow designs

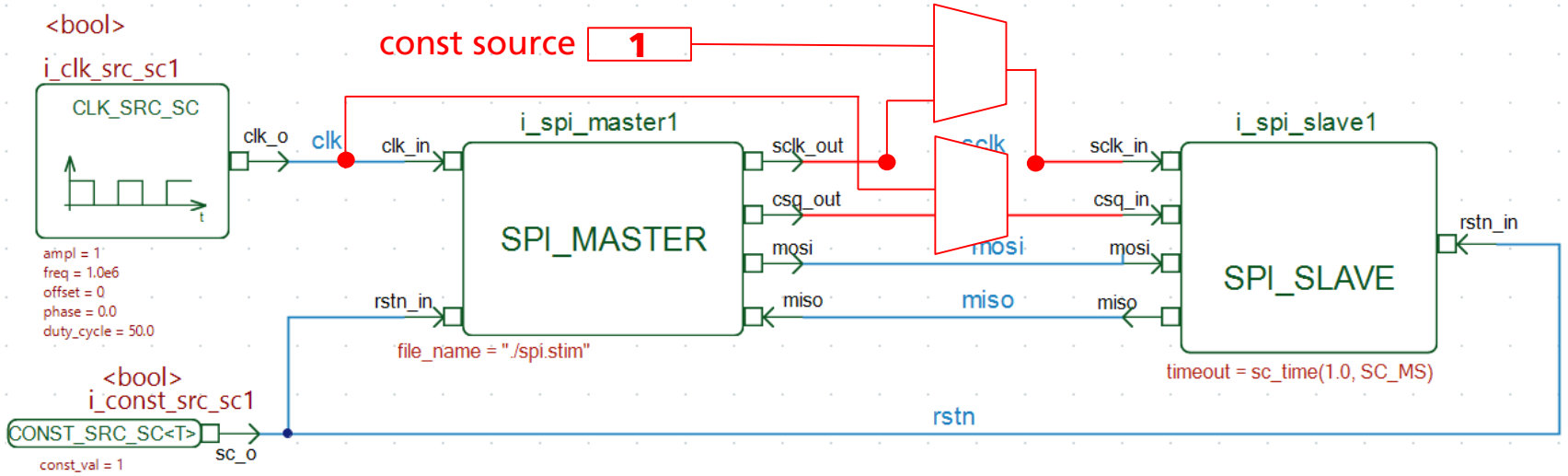
Cross-talk for PLL model

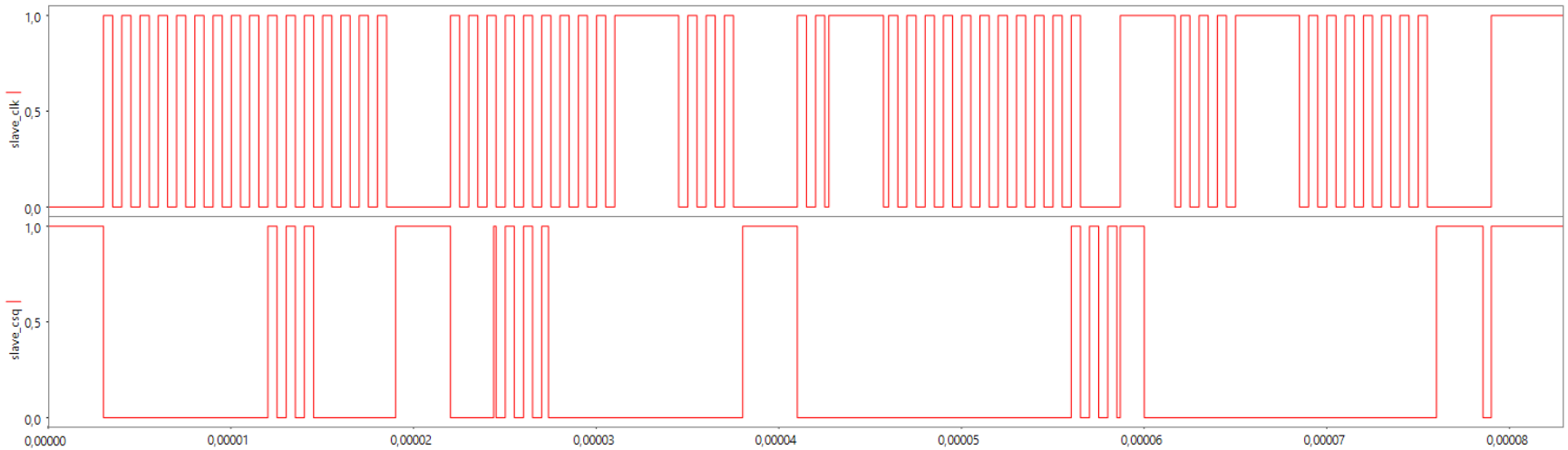



```
new sin_src_tdf<double>("i_sin_src_tdf1");
fault_scenario_template<double>({"*i_gain_tdf1.tdf_i","*i_sin_src_tdf1.tdf_o"}, FAULT_CROSSTALK, ADD) f;
...
wait(50.0, SC_US); f.activate();
wait(50.0, SC_US); f.deactivate();
wait(150.0, SC_US); f.activate();
wait(100.0, SC_US); f.deactivate();
```

Fault Injection for digital designs

Multiple faults at simple SPI transmissions



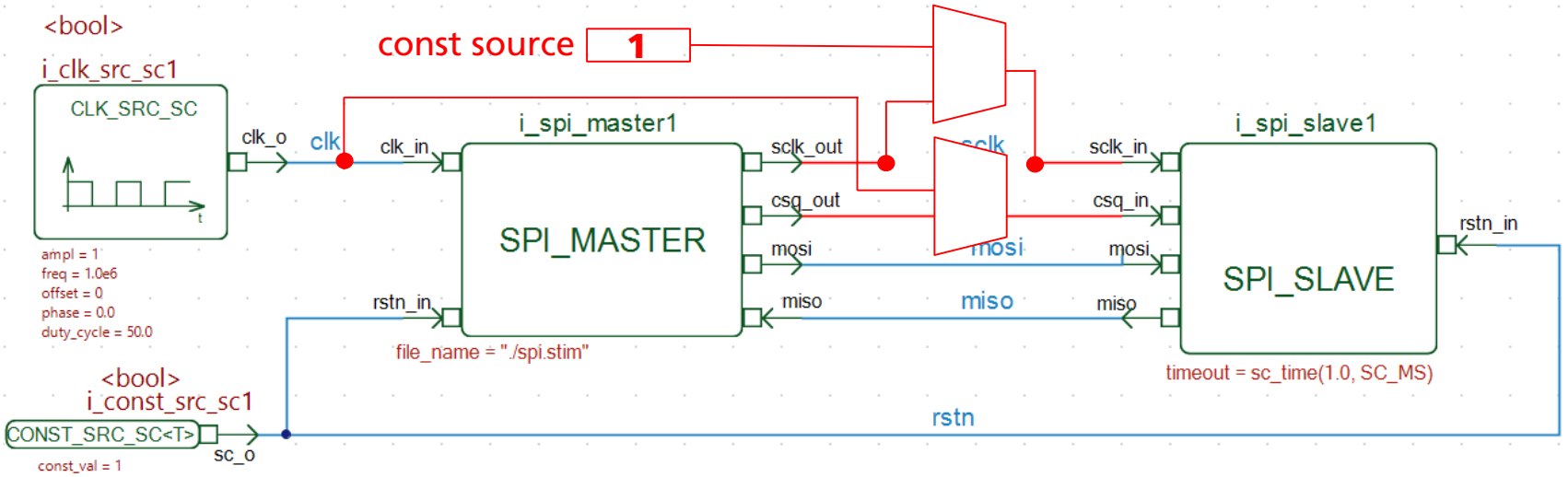


© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fraunhofer IIS

Fault Injection for digital designs

Multiple faults at simple SPI transmissions

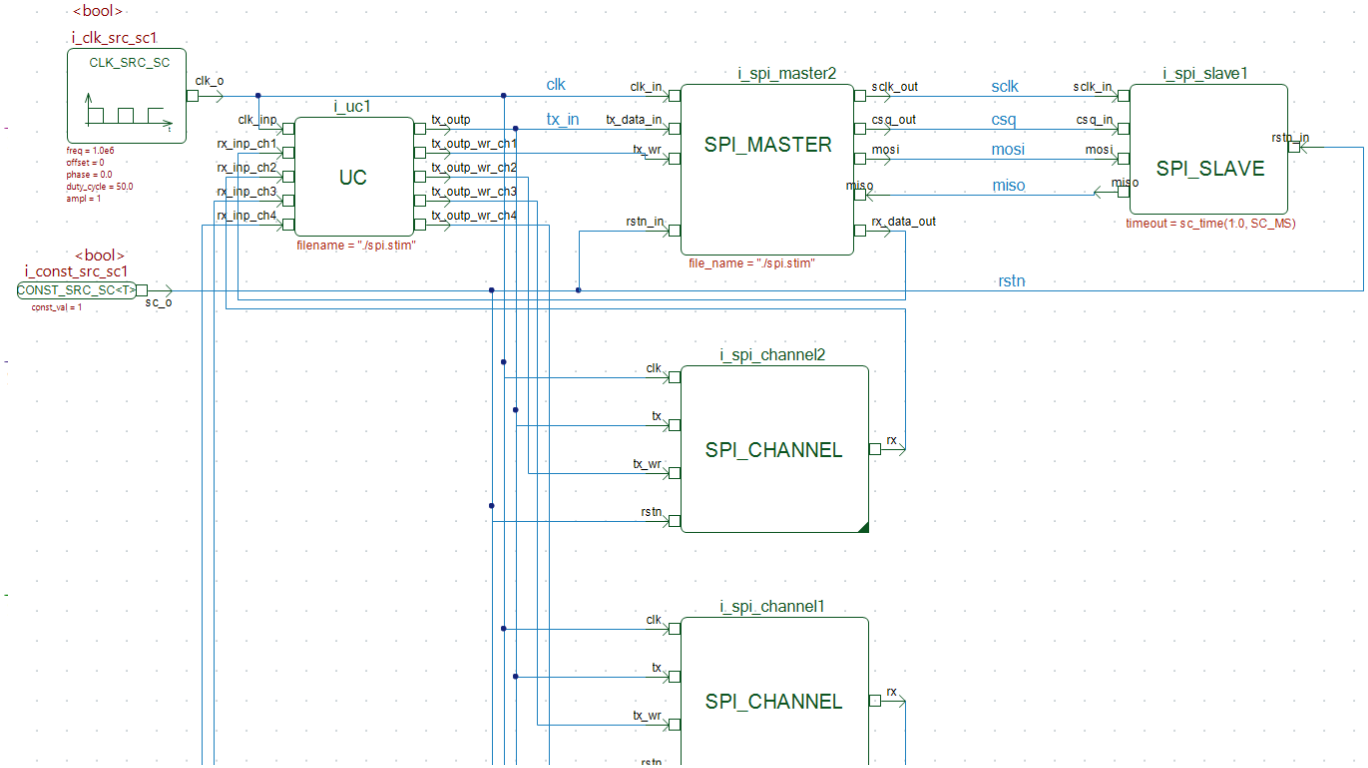


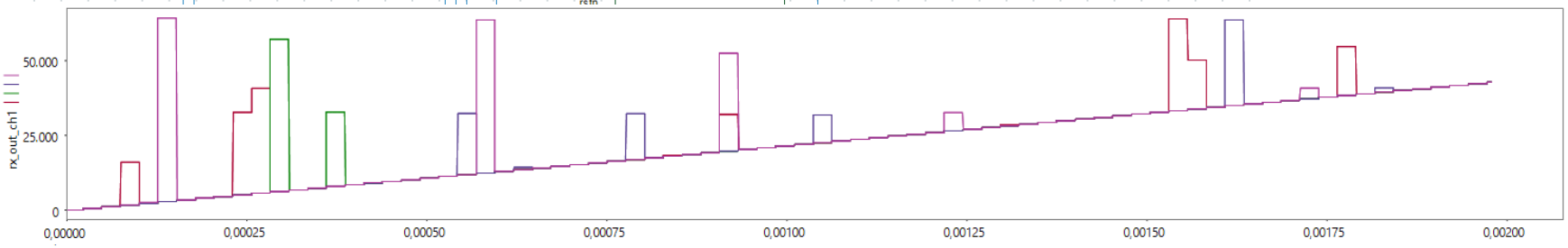

```
fault_scenario_template<bool>
    ({"*i_clk_src_sc1.clk_o","*i_spi_slave1.csq_in"}, FAULT_CROSSTALK, REPLACE);
fault_scenario_template<bool> ({"*i_spi_slave1.sclk_in"}, FAULT_STUCK_AT, true);
...
wait(stats::exponential(10.0e-6, 0.0));
fault_scenarios [ stats::exponential(fault_scenarios.size()) ]
    .activate(sc_time(3.0,SC_US)));
```


© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fault Injection for digital designs

Multiple faults at simple SPI transmissions (2)



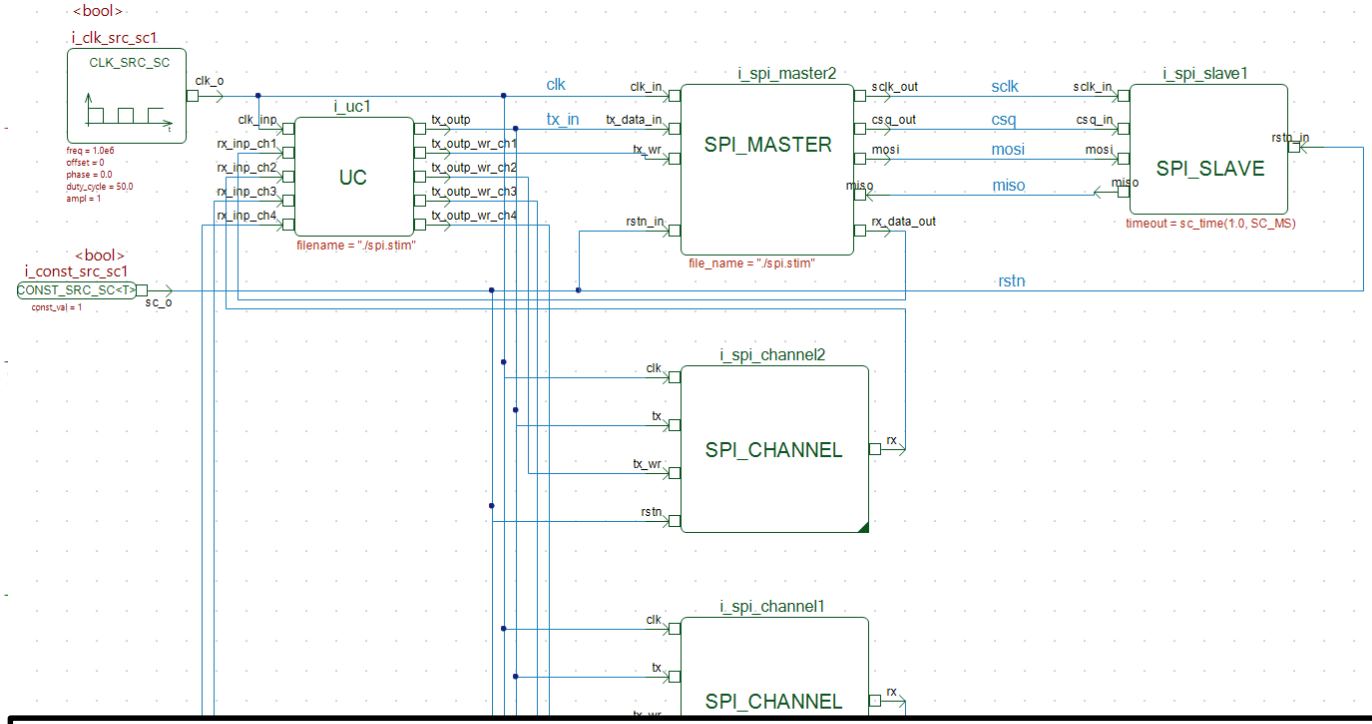


© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fraunhofer IIS

Fault Injection for digital designs

Multiple faults at simple SPI transmissions (2)



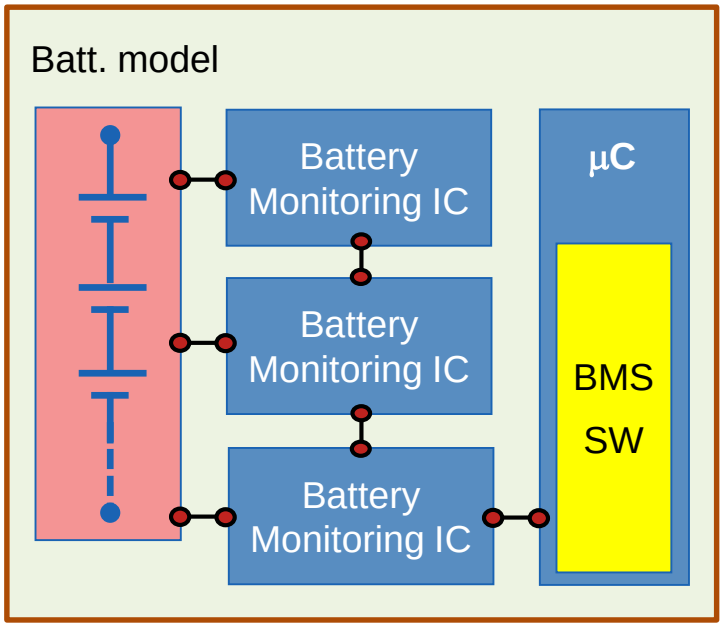

```
for (sc_object* obj: get_matching_objects("*mosi")) {
  ...
  fault_scenario_template<bool> ({*obj}, FAULT_STUCK_AT, true) f;
  f.configure (exponential, location_fct, sc_time(30.0, SC_US));
}
```


© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Project application (IKEBA): Battery management system

Handling of aging and faulty battery cells

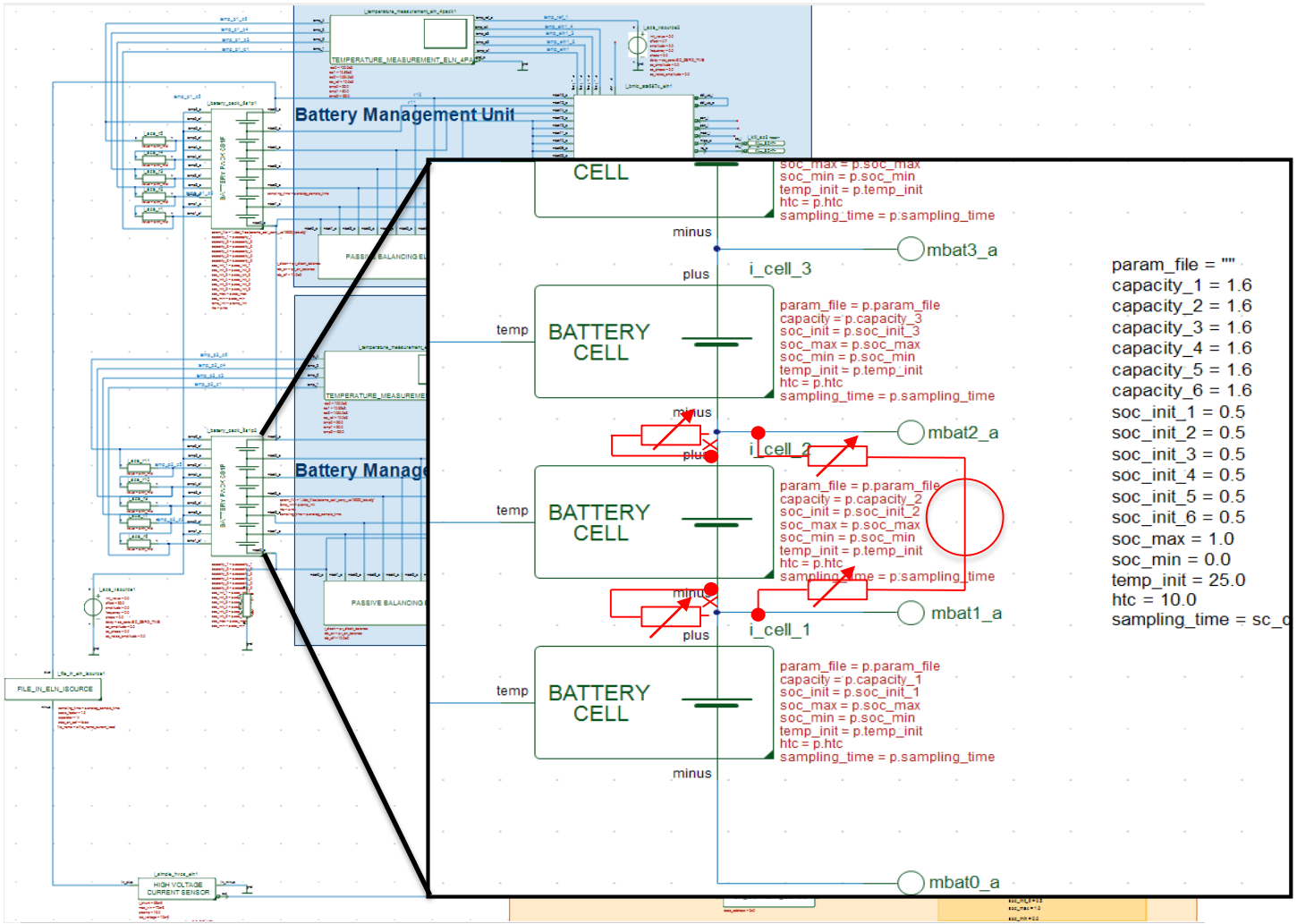
- Battery model consisting of several battery cells
- Daisy chained mointoring logic
- Controlled by programmable µC
- testing of
  - reliable handling of aging and faulty battery cells
  - battery monitoring ICs



© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Project application (IKEBA): Battery management system

Handling of aging/faulty battery cells

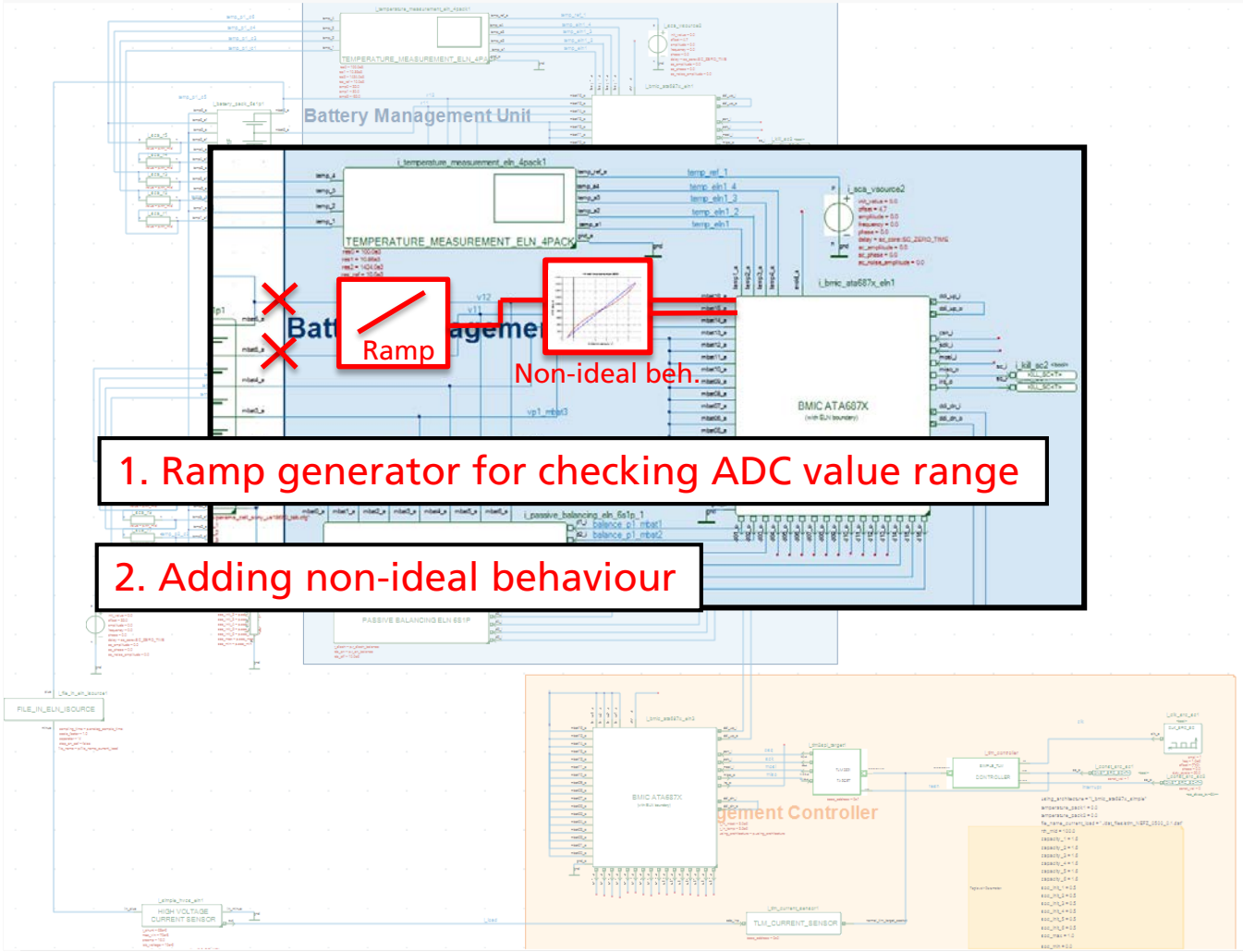


© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fraunhofer IIS

Project application (IKEBA): Battery management system

Battery monitoring IC testing

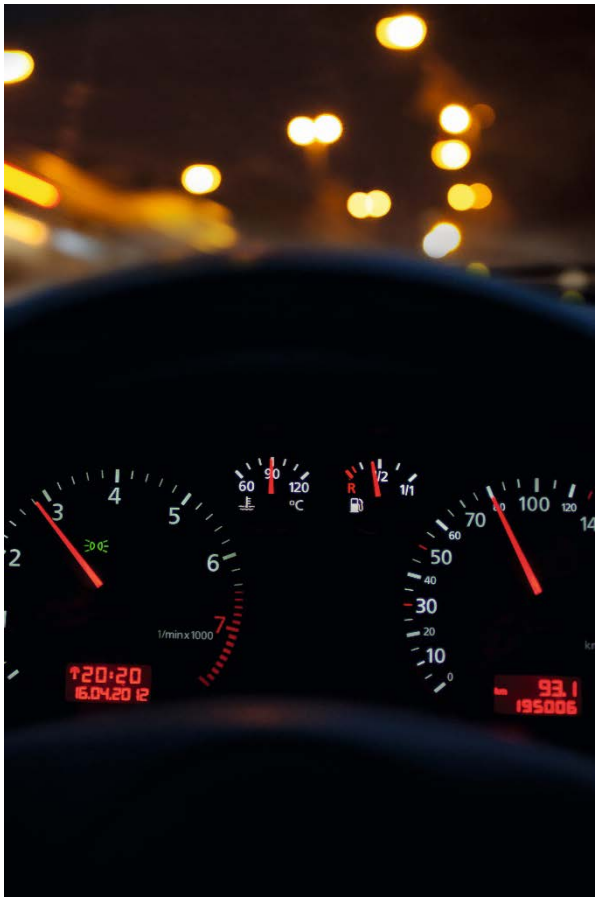


© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fraunhofer IIS

Further application areas
Other contexts

- ISO 26262
  - Low- & High-level hardware faults
  - Software safety on unreliable hardware
  - HiL testing
- Parallelisation
  - Improve coverage related testing
  - Almost for free



© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Outlook

- Extension of concurrent handling of different fault types
  - Currently some limitations apply
- Fault injection control via external input
  - Scripting
  - Coside Plug-In
- Adding more generic fault models

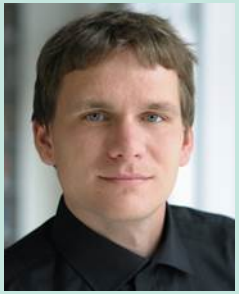
© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

THANK YOU FOR YOUR ATTENTION
YOUR CONTACTS

Roland Jancke

Head of Department
Design Methodology

✉ Roland.Jancke@eas.iis.fraunhofer.de
☎ +49 351 4640-747



Stephan Gerth

Group Manager
Functional Modeling and Verification

✉ Stephan.Gerth@eas.iis.fraunhofer.de
☎ +49 351 4640-847

Fraunhofer-Institute for Integrated Circuits IIS
Division Engineering of Adaptive Systems EAS
Zeunerstraße 38
01069 Dresden

www.eas.iis.fraunhofer.de

© Fraunhofer IIS/EAS | Stephan Gerth | 18.10.2016

Fraunhofer IIS