DYNAMISCHE FEHLERINJEKTION FÜR DIE SYSTEMSIMULATION MITTELS SYSTEMC

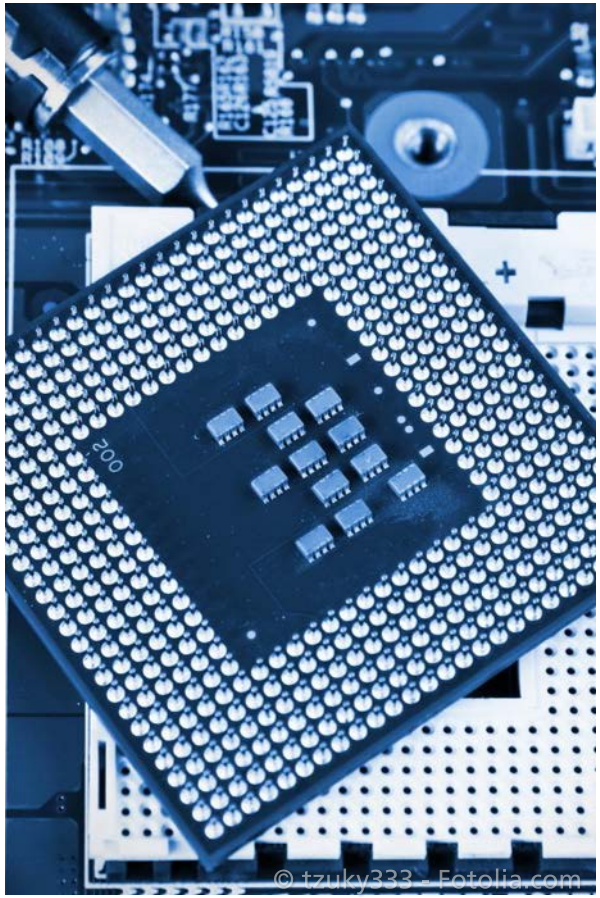
Thomas Markwirth, Fraunhofer IIS/EAS Dresden
ASIM/GI Fachgruppen-Treffen, Ulm 2017

Fraunhofer IIS

Herausforderungen beim Entwurf elektron. Systeme

Welche Aufgaben sind zu lösen?

- Untersuchung und Validierung des Gesamtsystems in vielen Fällen zwingend
  - Inkl. Software und nichtelektr. Effekte
- Sicherstellung der funktionalen Sicherheit
  - Für Nominal- und Fehlerfall
- Fehlersimulationen möglichst frühzeitig
  - Diagnosedeckungsgrad
  - Fehlerklassifizierung (ISO 26262)
  - Fehlertoleranz und Fehlerlatenz
- Fehlerinjektion
  - bisher meist Direktintegration von Fehlerverhalten in DUT

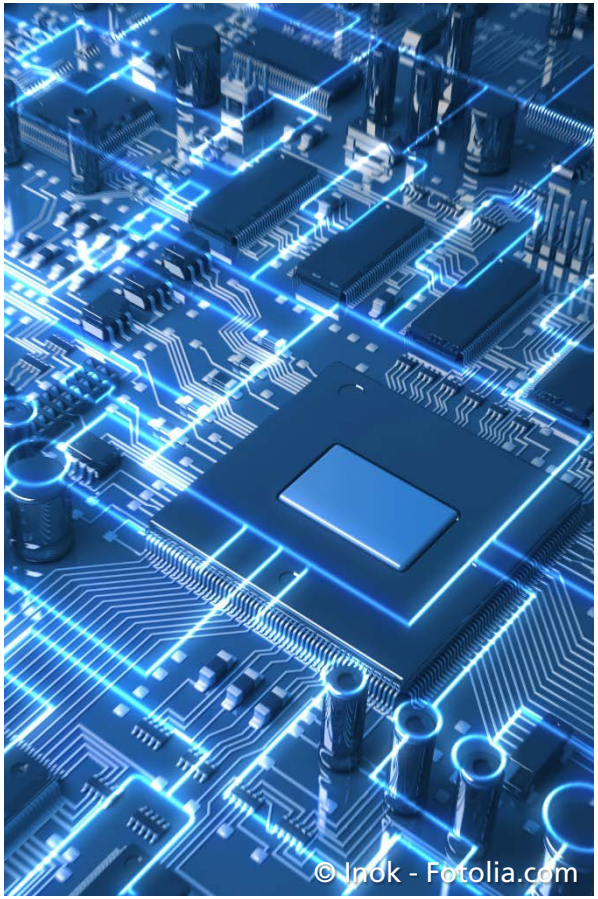


Erarbeitung eines Lösungsansatzes
Wofür und was soll erzielt werden?

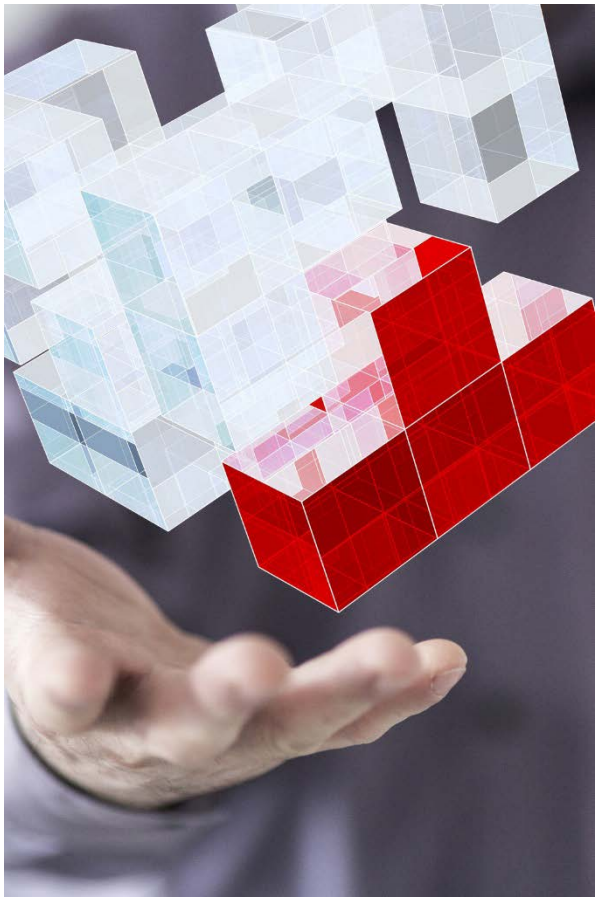
- Fokusiert im Designflow
  - Konzeptentwurf, Systemlevel-Verifikation
  - Unterstützung der Firmwareentwicklung
  - Referenz für die HW-Implementierung
- Fehlersimulation ermöglichen
  - Große Designs effizient handhabbar
  - High-level Fehler beschreibbar
  - Wiederholt auftretende Fehler konfigurierbar

SystemC/ SystemC AMS

ist die Lösung!

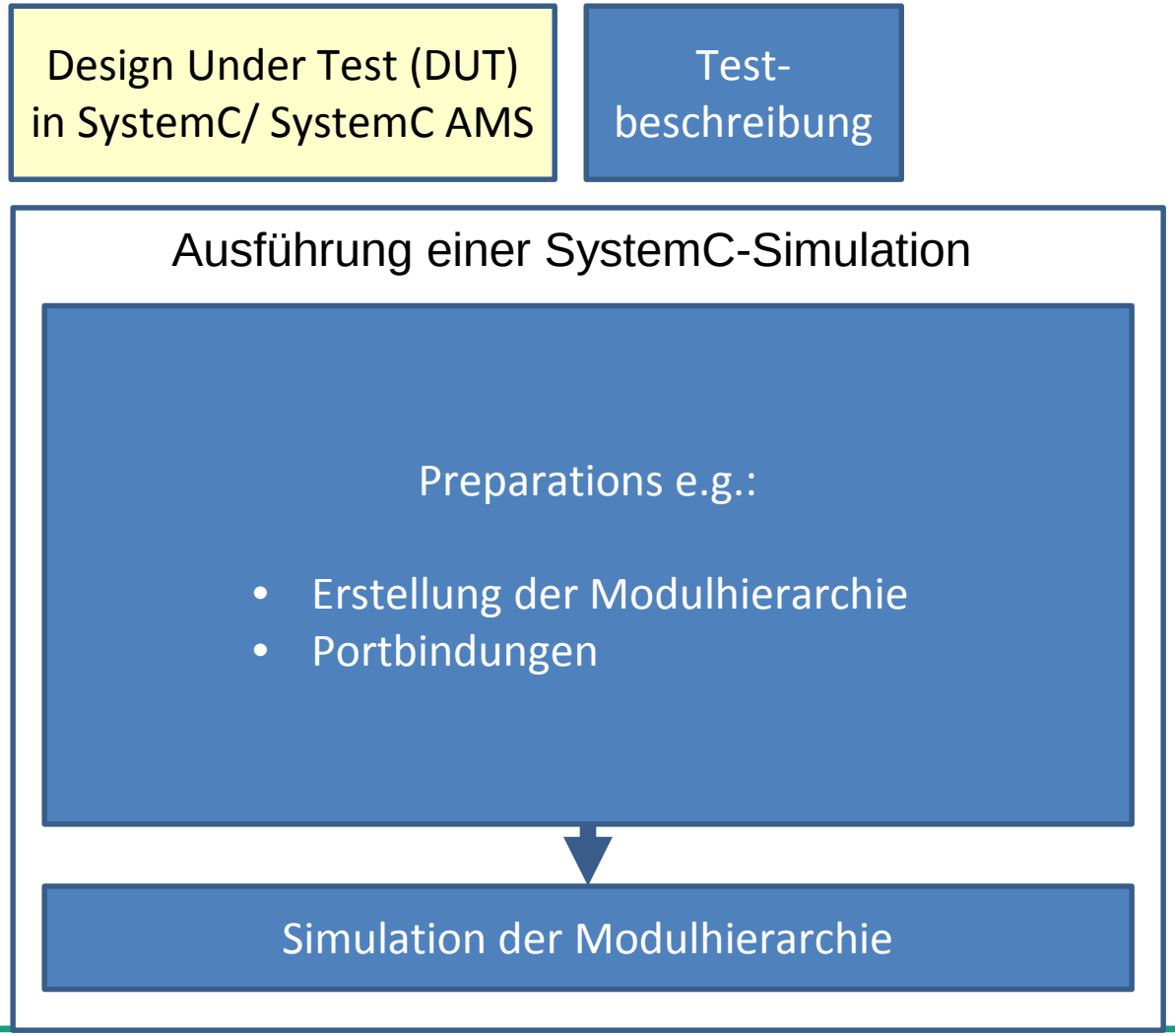


Ansatz
Wie soll dies gelöst werden?

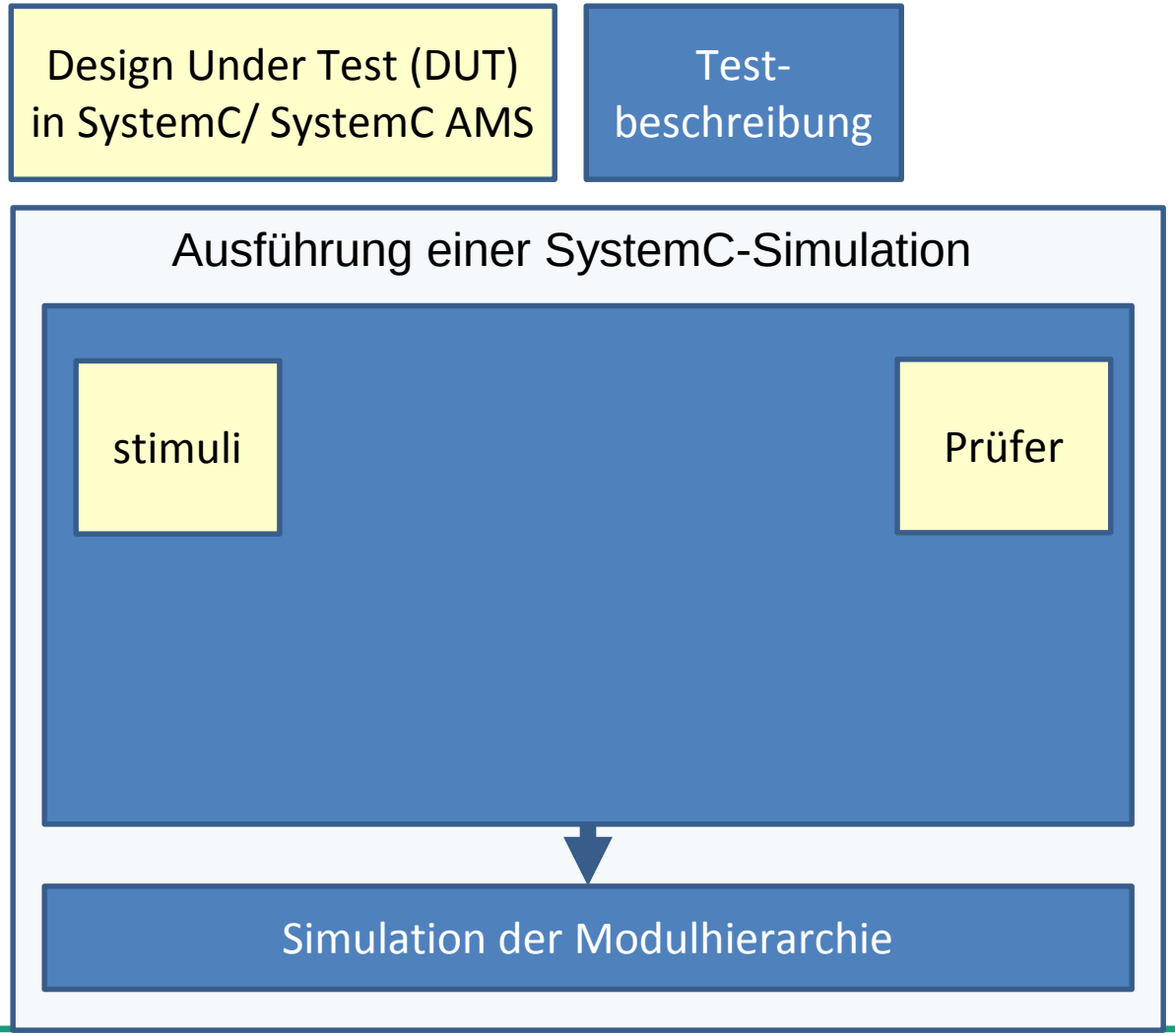


- Erstellung einer universellen Lösung zur Fehlerinjektion in SystemC & SystemC AMS
- Vermeidung von DUT-Änderungen, daraus ergibt sich:
  - DUT soll nur den Gutfall abbilden
  - keine Test-Artefakte im DUT- Code
  - Fehlerinjektionen nur während Tests realisieren
- Fehler beschrieben über Auftreten im DUT
  - Wo soll welcher Fehler auftreten
  - Mit welcher Rate soll dieser auftreten (permanent, periodisch, …)

Lösung
Prinzip der dynamischen Fehlerinjektion

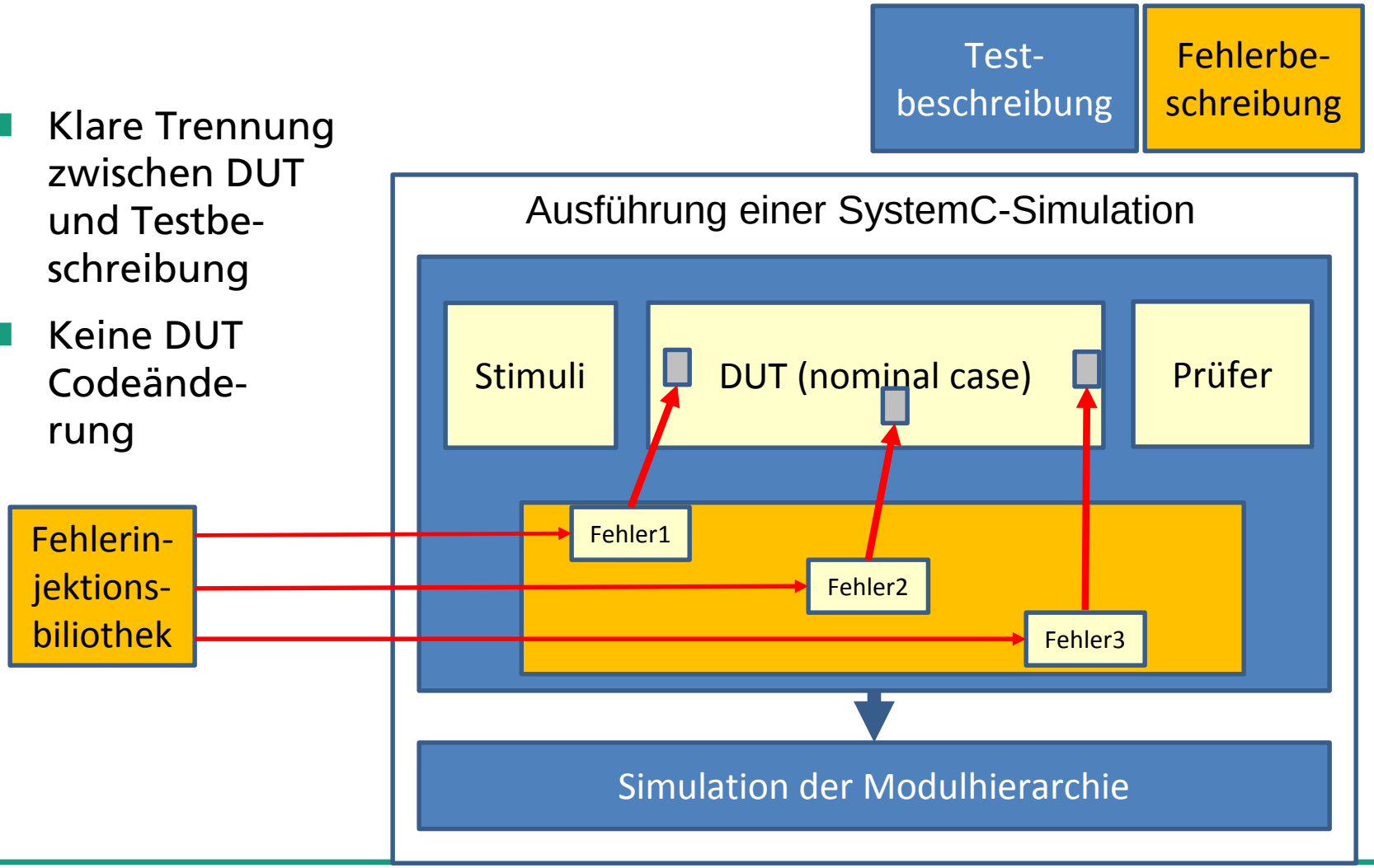


Lösung
Prinzip der dynamischen Fehlerinjektion



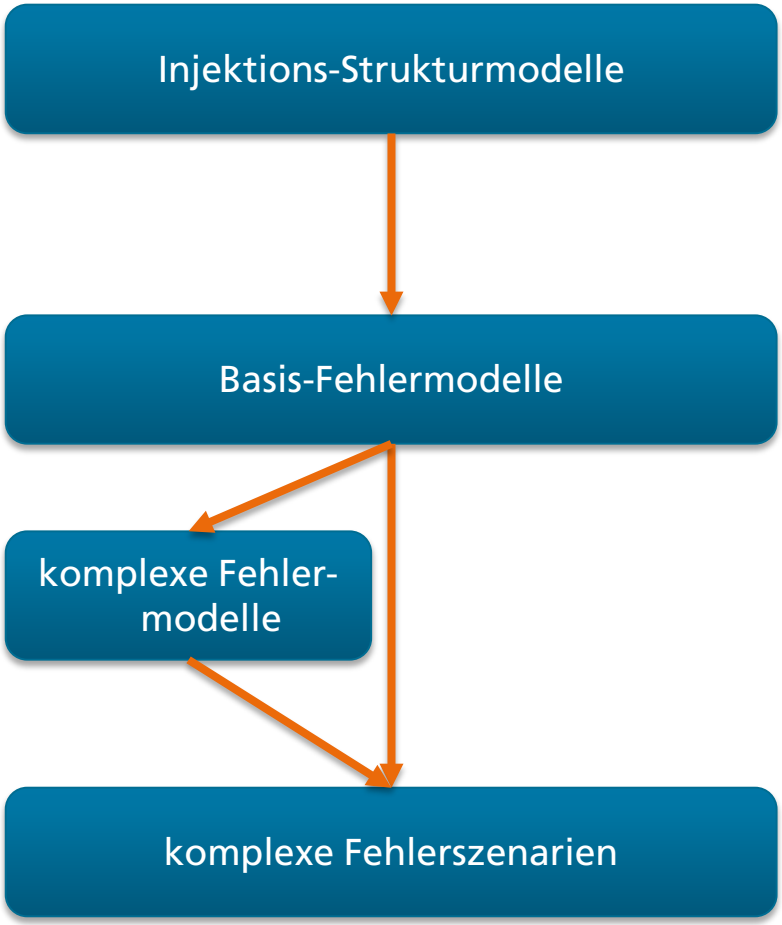
Lösung
Prinzip der dynamischen Fehlerinjektion

- Klare Trennung zwischen DUT und Testbeschreibung
- Keine DUT Codeänderung



Fehlerinjektionsbibliothek

Strukturierter Ansatz und offen für Erweiterungen

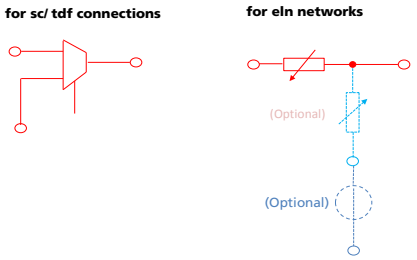


- Injektionsmechanismen für unterschiedliche MoCs
- z.B. Übersprechen (Crosstalk), Verzögerungen, Glitches, Unterbrechungen, Haftfehler (Stuck-At)
- anwenderspezifische Detailierung
- Kombination & Konfiguration von spezifischen Anwendungen

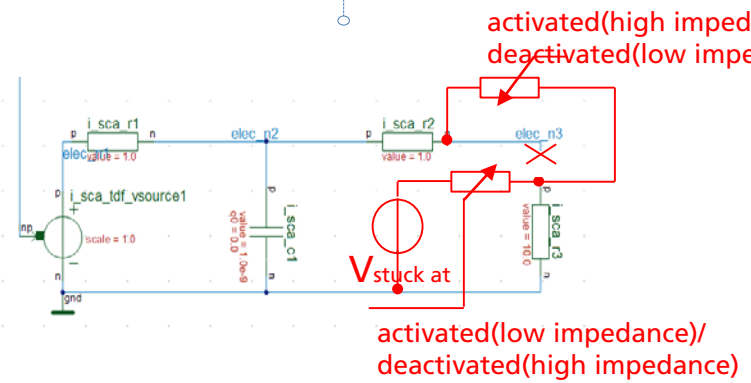
Fehlerinjektionsbibliothek

Strukturierter Ansatz und offen für Erweiterungen

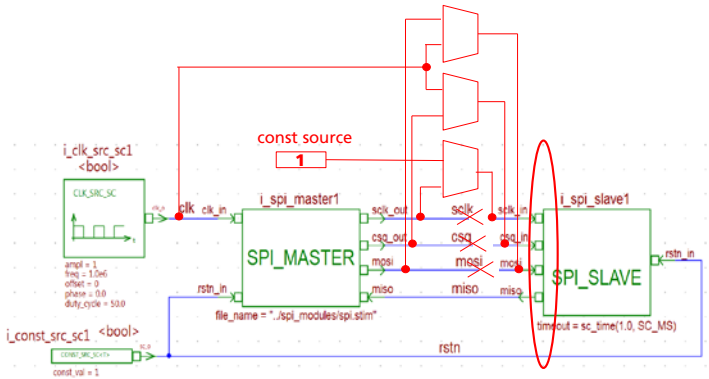
- Low level Fehlerstrukturmodelle
  - (MoC-abhängig)



- Auswahl Fehlermodelle
  - (stuck-at, crosstalk, open/short, delay, bridge, glitch)



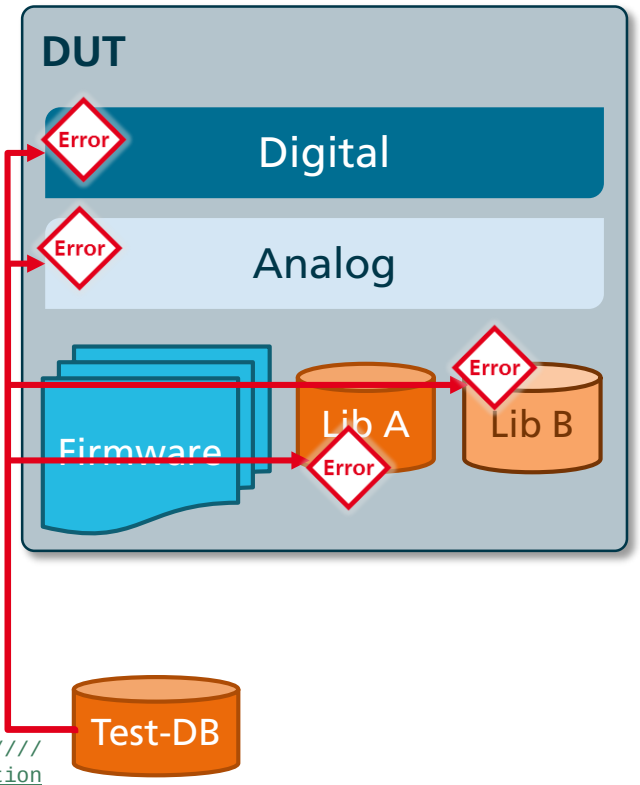
- Szenarien



Fraunhofer IIS

Dynamische Fehlerinjektion
Keine Änderung der DUT Codebasis!

- Manipulation des DUT nur während der Laufzeit
- Verhinderung von Bugs infolge eingefügter Testartifakte
- Fehler können gleichzeitig aktiv sein, MoC unabhängig
- Instanziierung von Fehlerszenarien auf Testebene mittels Stimulus-Modul

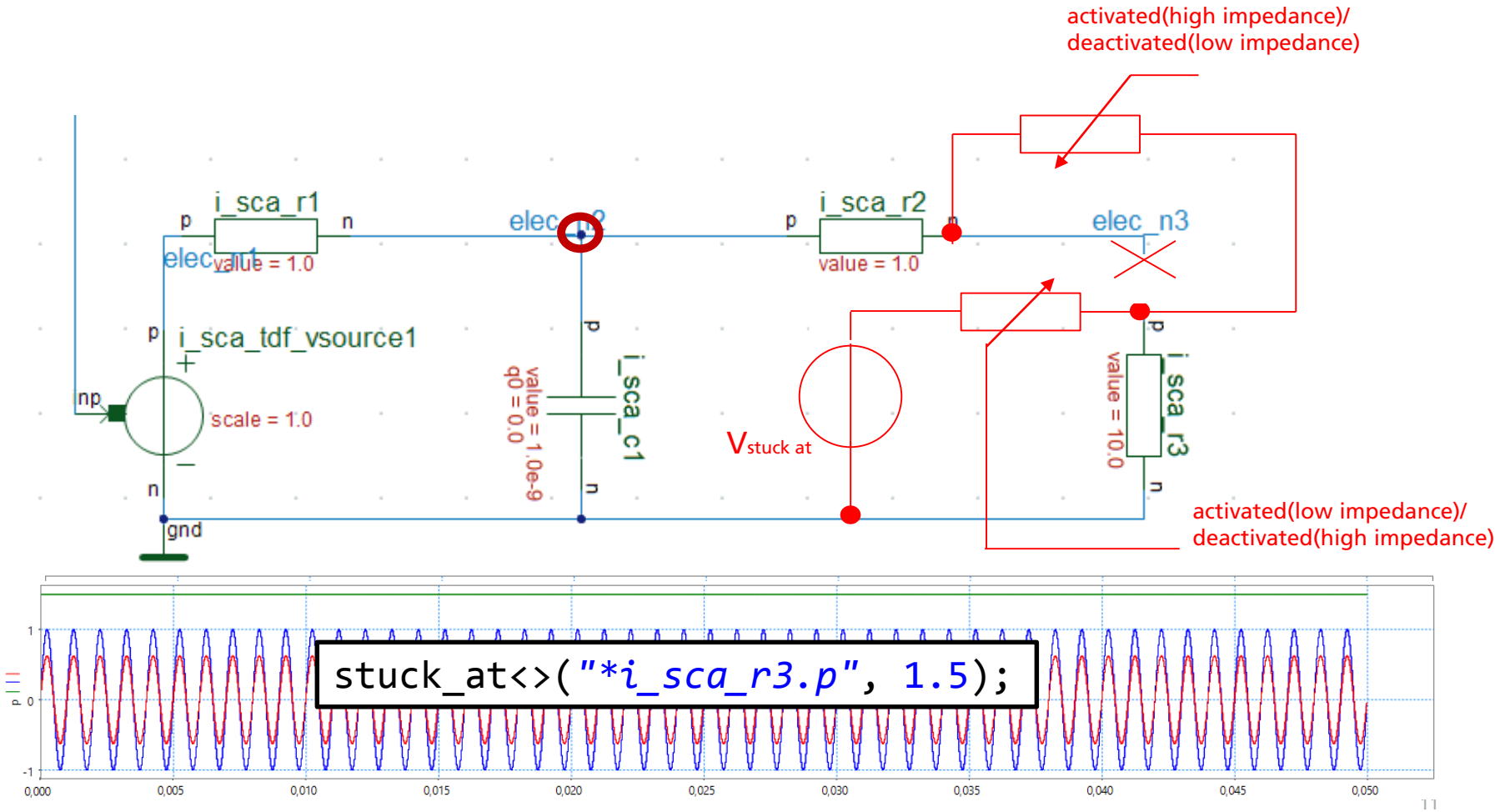



```
stimuli_fault_injection_tutorial_3(
       sc_core::sc_module_name nm, params pa = params() ) : p(pa){
////////////////////////////////////////////////////////////////////////////
/// Definition der Ziel- und optional Quellports bzw. -werte zur Fehlerinjektion
////////////////////////////////////////////////////////////////////////////
    port.push_back("*i_delay_tdf1.tdf_i");
    value.push_back(0.0);
////////////////////////////////////////////////////////////////////////////
/// Erstelle 1. Szenario: Stuck-At: Haftwert gesetzt auf 0.0 Volt
////////////////////////////////////////////////////////////////////////////
    stuck_at_fault1 = new fault_scenario_template<double > std::make_pair(port, port_b), fault_injection_base::FAULT_STUCK_AT);
}

fault_scenario_template<double >*           stuck_at_fault1;
```

Fehlerinjektion in lineare elektrische Netzwerke

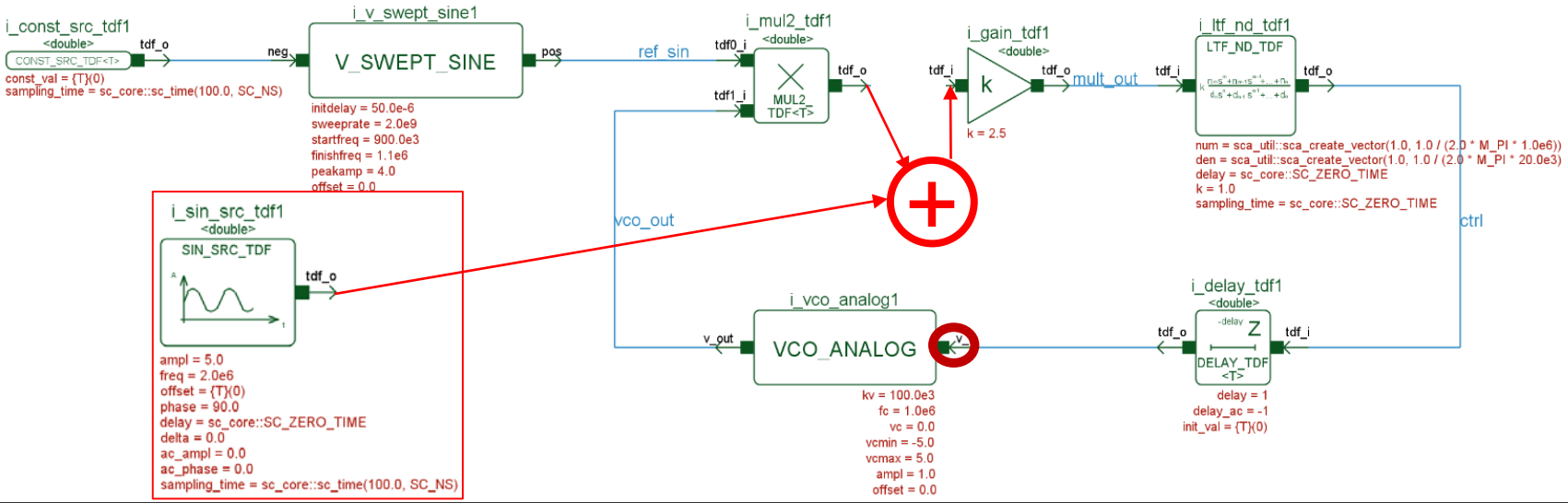
Haftfehler (Stuck-at-value) in Filter



Fraunhofer IIS

Fehlerinjektion für Timed Data Flow Designs

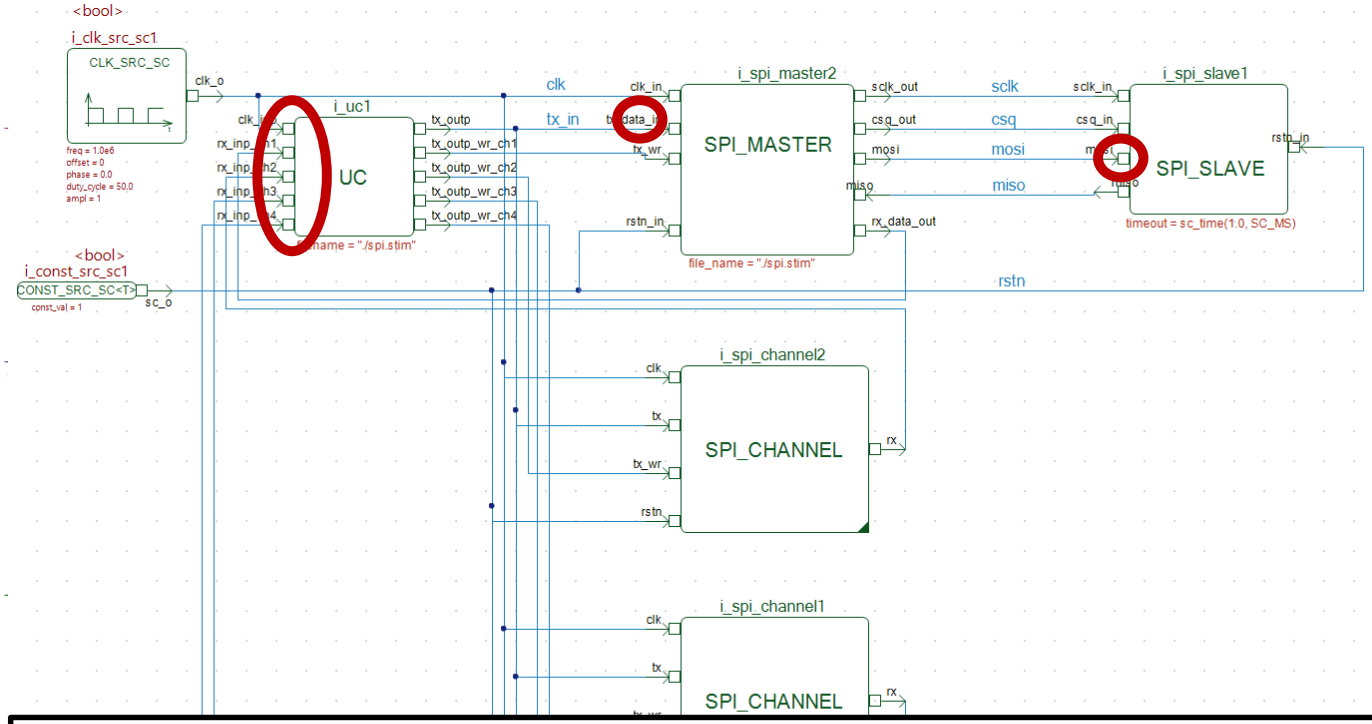
Cross-talk in PLL-Modell




```
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();
```

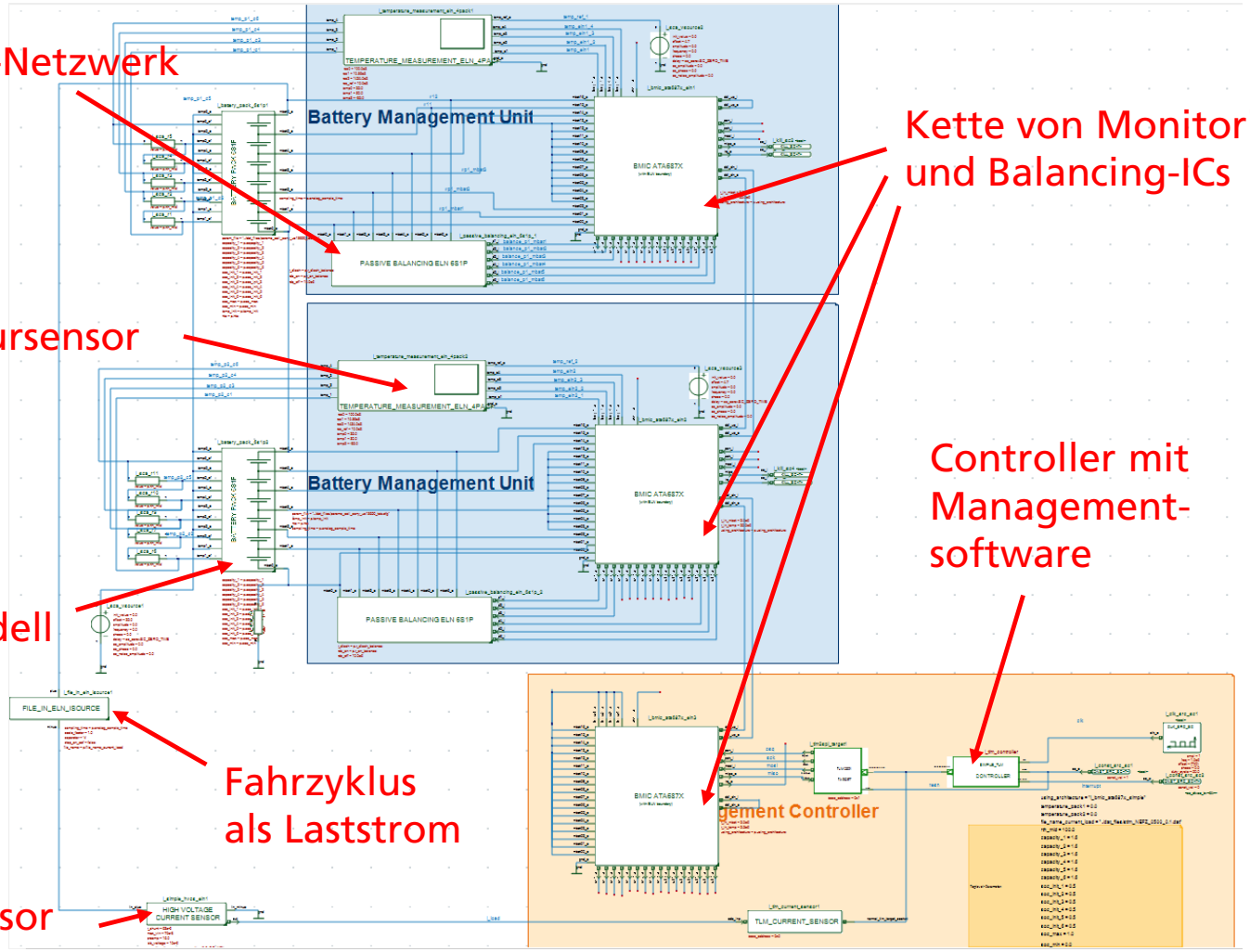
Fehlerinjektion für digitale Designs

Multiple Fehler in simplen SPI-Übertragungen (2)




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

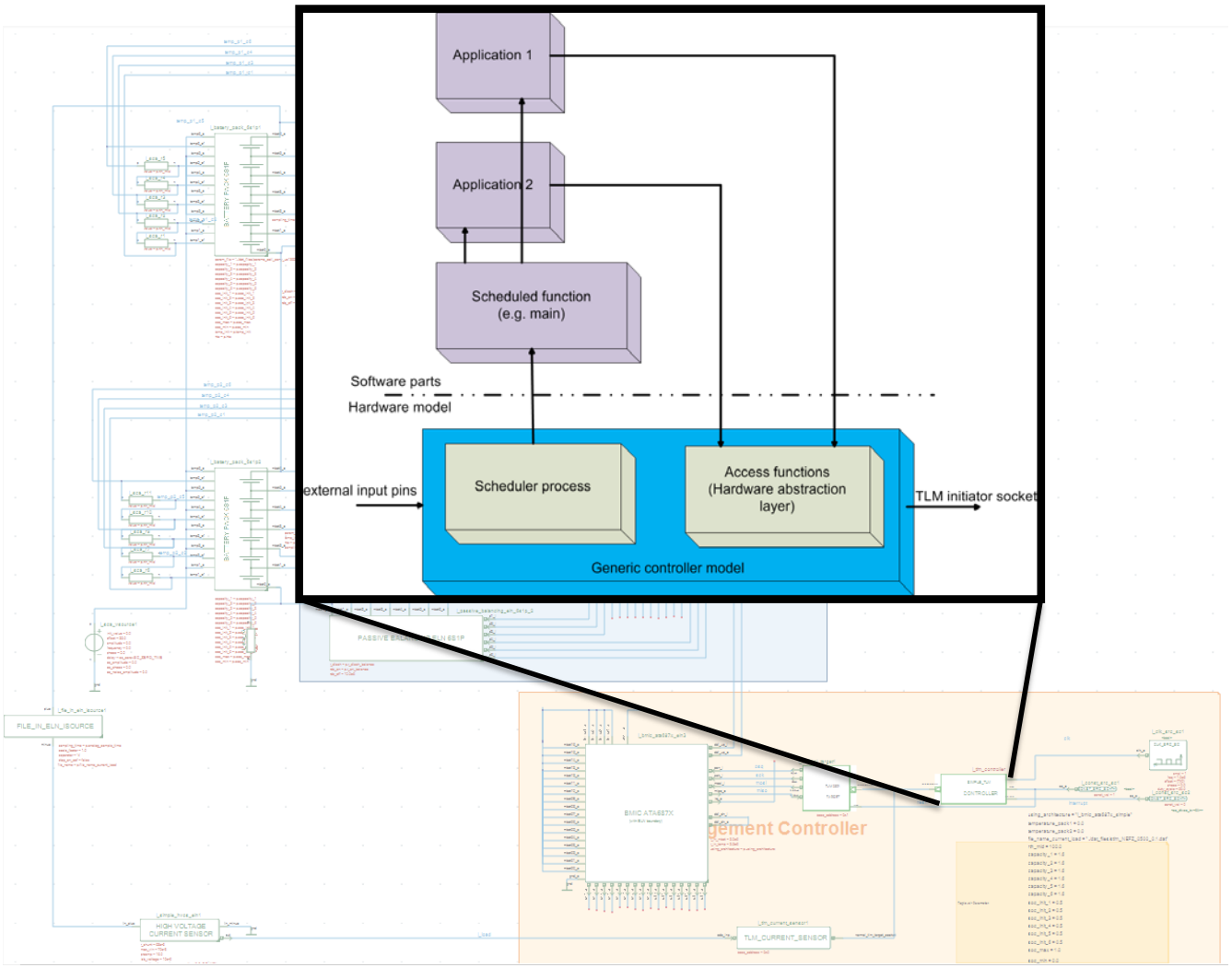
Projektapplikation (IKEBA): Batterie-Managementsystem
Toplevel



Fraunhofer IIS

Projektapplikation (IKEBA): Batterie-Managementsystem

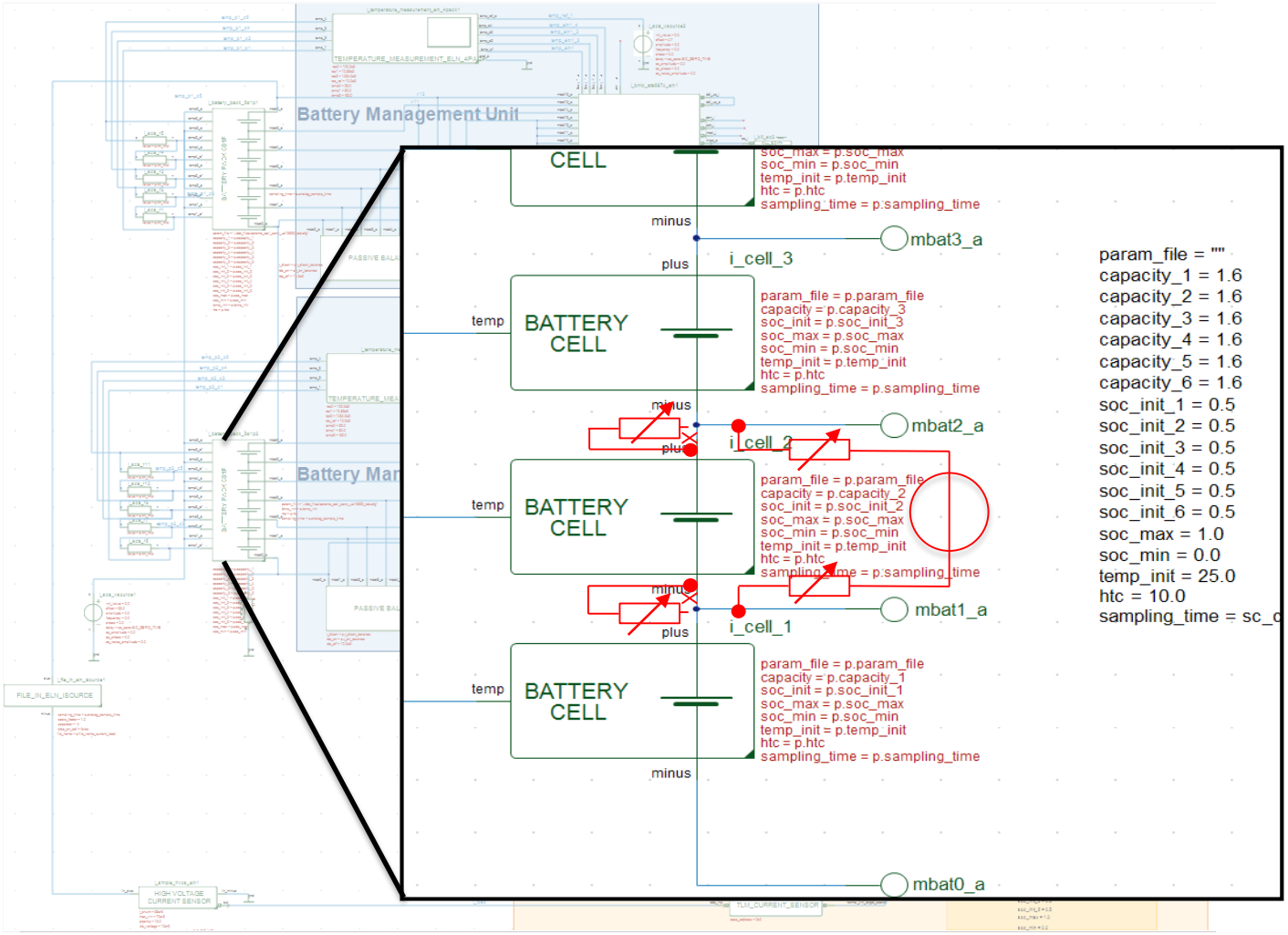
Einbindung Management-Software des Zielsystems



Fraunhofer IIS

Projektapplikation (IKEBA): Batterie-Management-System

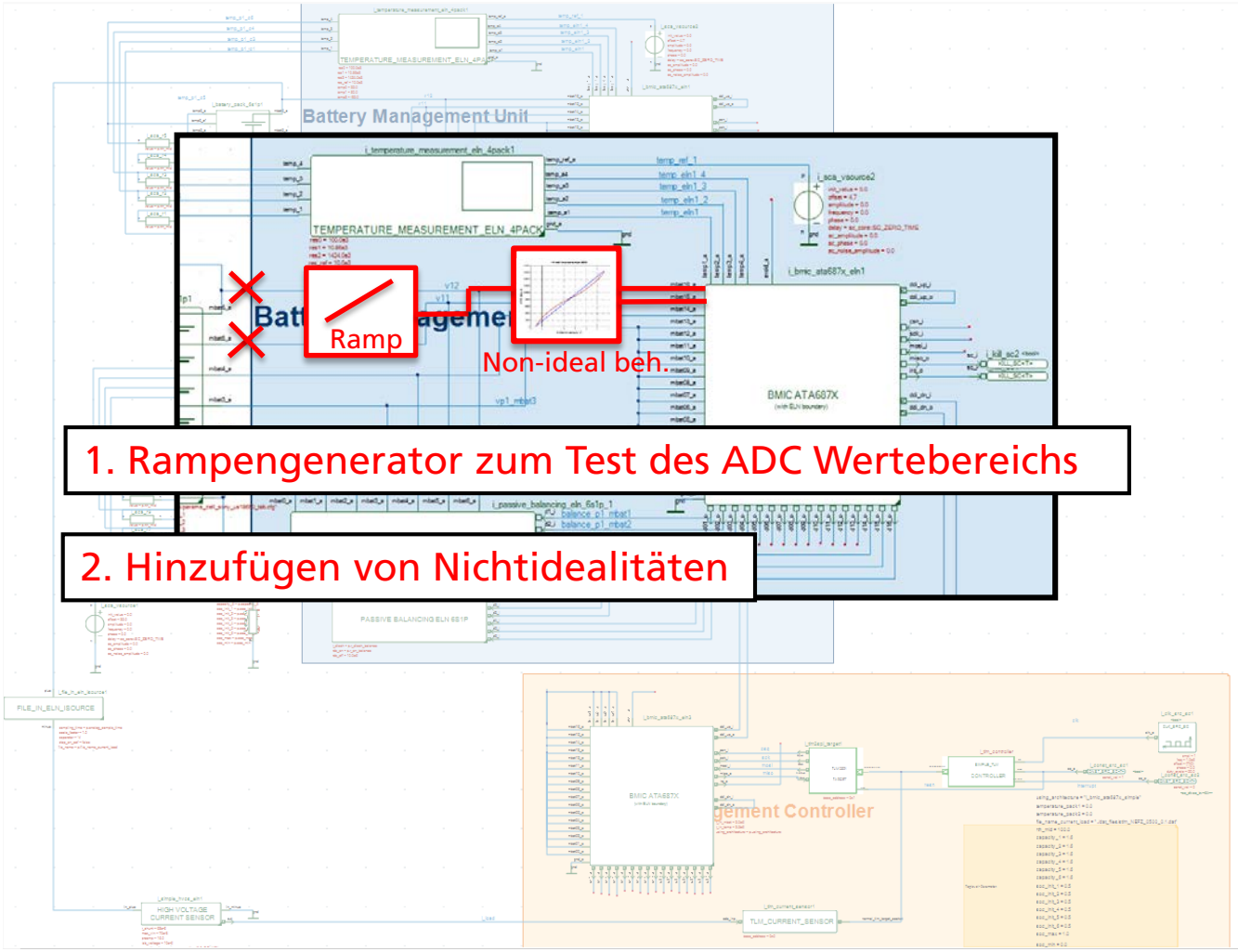
Handhabung alternde/fehlerhafte Batteriezellen



Fraunhofer IIS

Projektapplikation (IKEBA): Batterie-Managementsystem

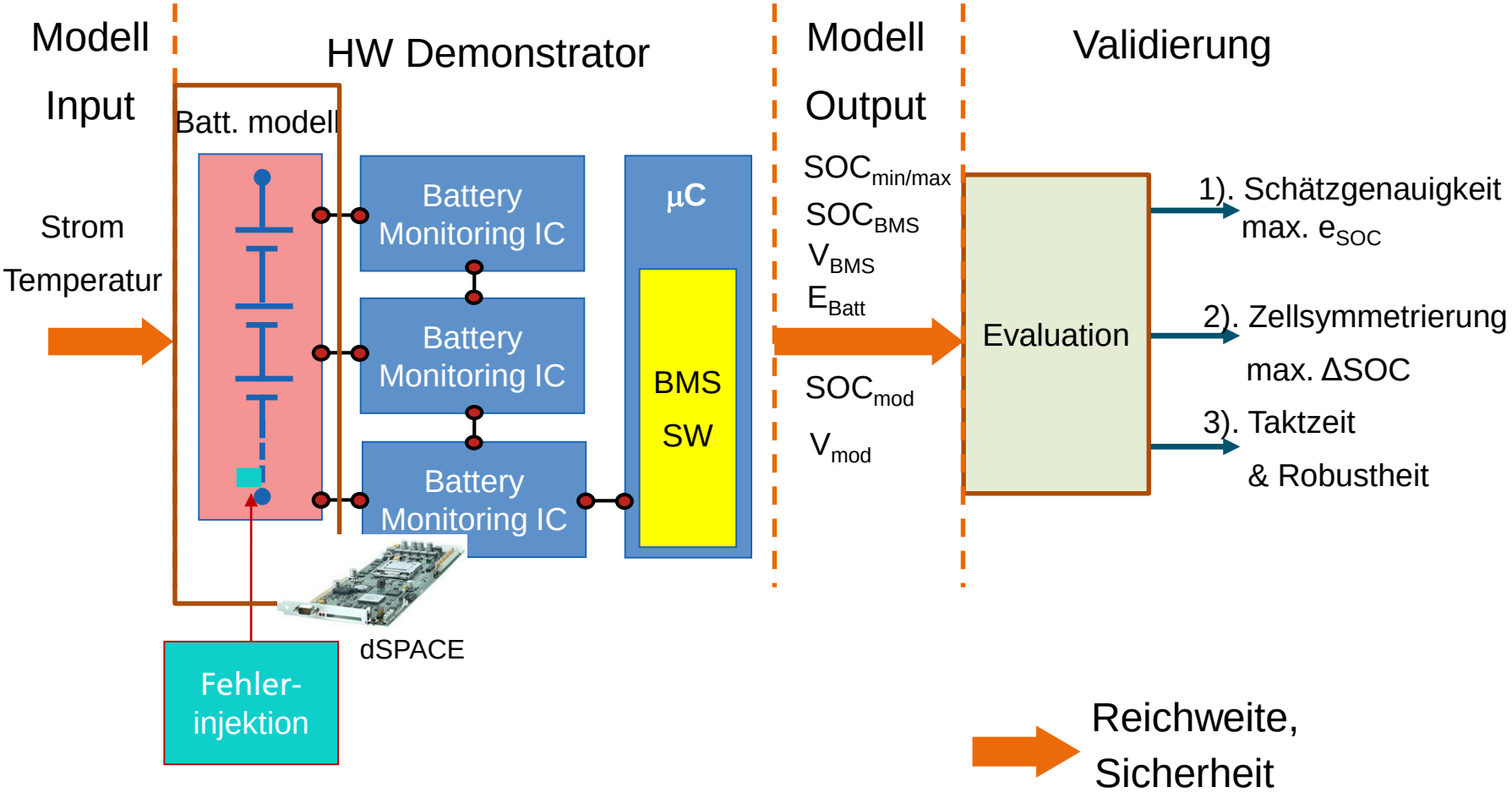
"In-depth" – Stimulusgenerator für BMIC-Tests



Fraunhofer IIS

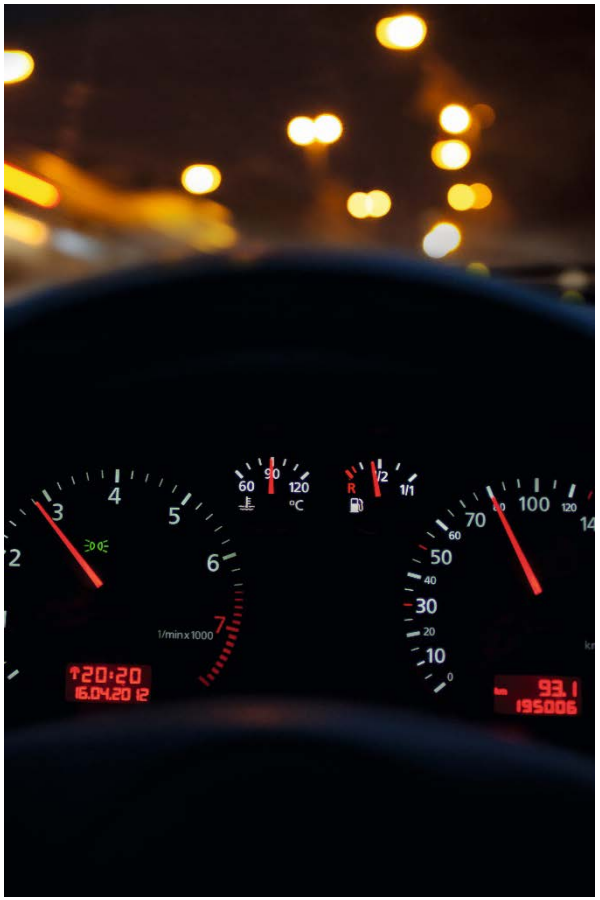
Projektapplikation (IKEBA): Batterie-Managementsystem

Funktionale Validation - HiL



Zusammenfassung

- Universeller Ansatz zur dynamischen Fehlerinjektion in SystemC / SystemC AMS
  - Keine DUT-Änderungen notwendig
  - Einfach anwendbar (Bibliothek)
  - Automatisierbar
  - Kombinierbar mit statist. Methoden
  - Anwendbar in HiL-Systemen
- Unterstützung ISO 26262
  - Low- & High-level Hardwarefehler
  - Software safety on unreliable Hardware
  - Verbesserung der Testabdeckung



Ausblick

- Erweiterung der Möglichkeiten für gleichzeitige und bedingte Fehler
  - Momentan noch einige Beschränkungen
- Automatisierte Fehlerinjektion aus externem Input
  - Per Skript oder aus Textdatei
  - Coside Plug-In
- Erstellung weiterer generischer Fehlermodelle

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT

IHRE KONTAKTE

Thomas Markwirth

Gruppe
Funktionale Modellierung und Verifikation

✉ Thomas.Markwirth@eas.iis.fraunhofer.de
☎ +49 351 4640-816

Stephan Gerth

Gruppenleiter
Funktionale Modellierung und Verifikation

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

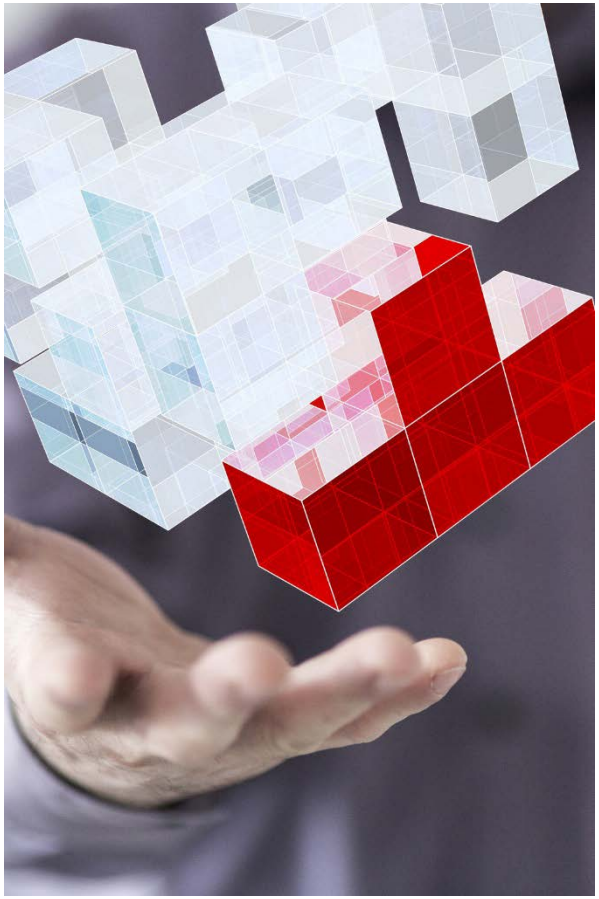
Fraunhofer-Institut für Integrierte Schaltungen IIS
Abteilung Entwurf von adaptiven Systemen EAS
Zeunerstraße 38
01069 Dresden

www.eas.iis.fraunhofer.de

Fraunhofer IIS

Fragen ?

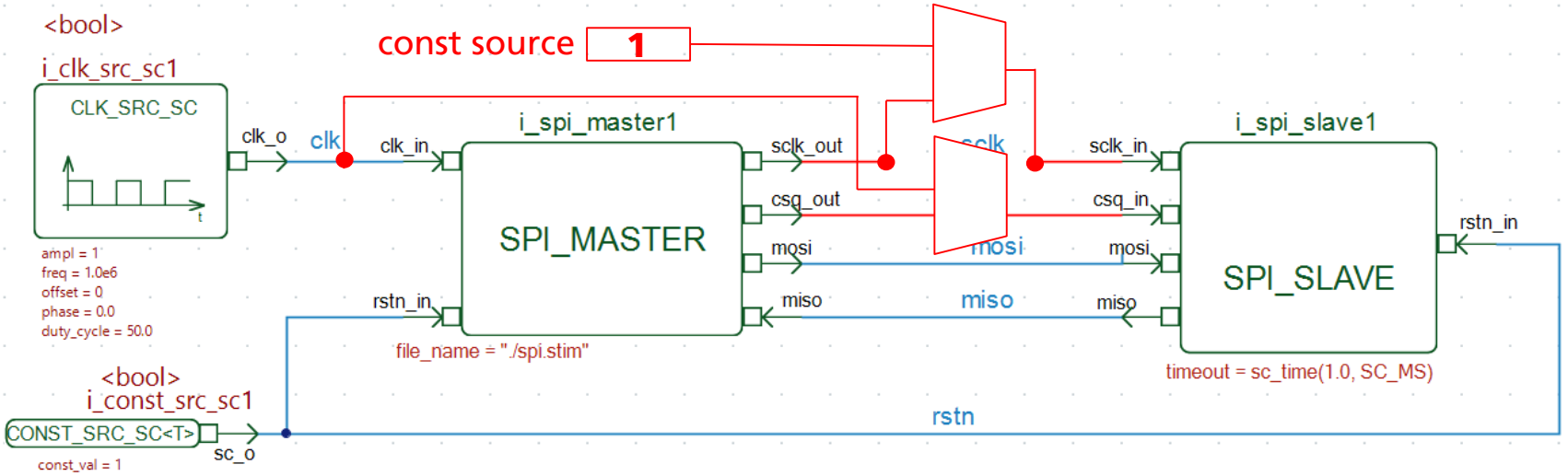
Lösung
Wie kann das implementiert werden?



- Bereitstellung einer Bibliothek mit
  - Fehlerinjektionsmechanismen
  - grundlegende Fehler-Modelle
  - Unterstützung unterschiedlicher MoCs
  - Statistische Konfiguration bezüglich Auftrittsrate und –region von Fehlern
- Ermöglichung von Fehler-Spezialisierungen bezüglich:
  - High-level Fehlern
  - anwenderspezifische Anforderungen
- Fehlerinjektion erfolgt dynamisch
  - Testbench getrieben
  - DUT bleibt unverändert

Fehlerinjektion für digitale Designs

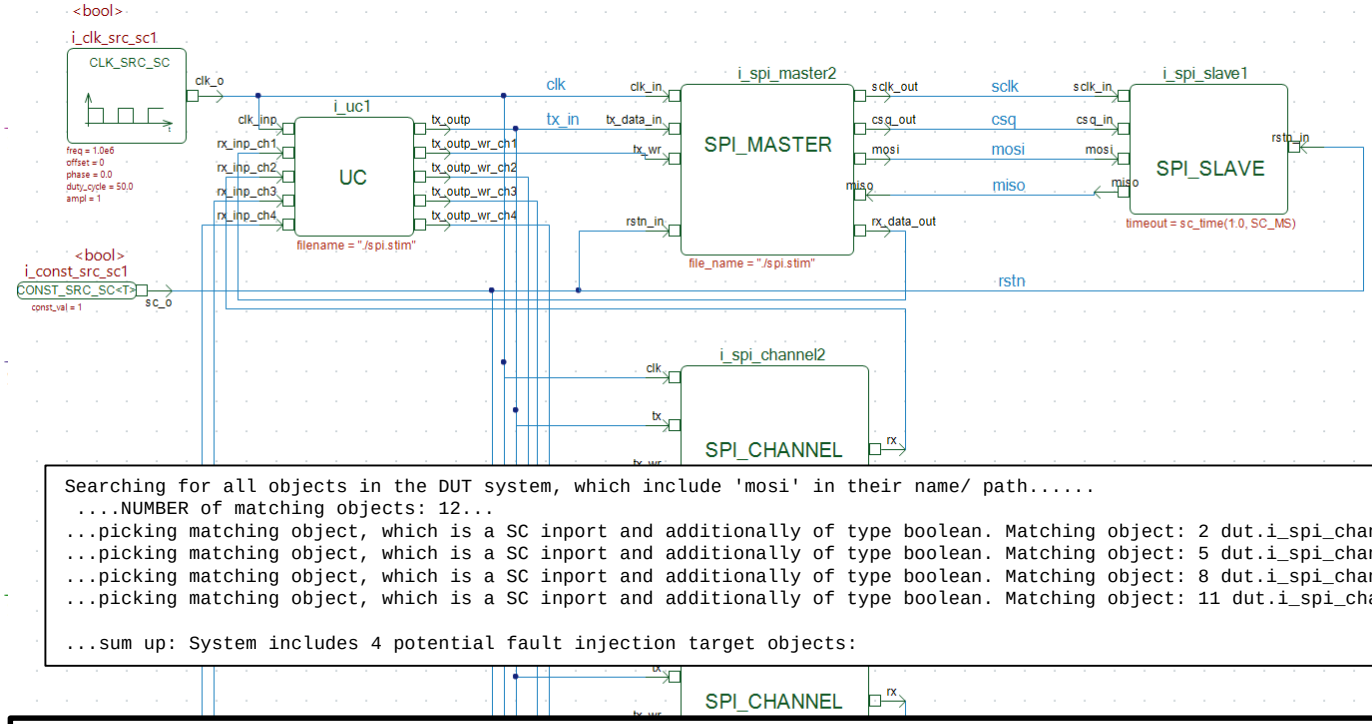
Multiple Fehler in simplen SPI-Übertragungen (1)




```
fault_scenario_template<bool>
    ({"*i_spi_slave1.csq_in","*i_clk_src_sc1.clk_o"}, 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::fault_location(fault_scenarios.size()) ]
    .activate(sc_time(3.0,SC_US)));
```

Fehlerinjektion für digitale Designs

Multiple Fehler in simplen SPI-Übertragungen (2)




```
Searching for all objects in the DUT system, which include 'mosi' in their name/ path......
 ....NUMBER of matching objects: 12...
...picking matching object, which is a SC inport and additionally of type boolean. Matching object: 2 dut.i_spi_channel1.i_spi_slave1.mosi 1
...picking matching object, which is a SC inport and additionally of type boolean. Matching object: 5 dut.i_spi_channel2.i_spi_slave1.mosi 2
...picking matching object, which is a SC inport and additionally of type boolean. Matching object: 8 dut.i_spi_channel3.i_spi_slave1.mosi 3
...picking matching object, which is a SC inport and additionally of type boolean. Matching object: 11 dut.i_spi_channel4.i_spi_slave1.mosi 4

...sum up: System includes 4 potential fault injection target objects:
```

```
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));
}
```

Status der Entwicklungsumgebung

- Vergleich der Simulationsergebnisse mit und ohne Fehlerinj.