Fraunhofer IIS

EffektiV

HiL Firmware Prototyping Using an COSIDE® Asynchronous Motor Model

Author: Paul Ehrlich

Fraunhofer Institute for Integrated Circuits

This work has been developed in the project EffektiV. EffektiV (reference number: 01IS13022) is funded by the German ministry of education and research (BMBF) within the research programme ICT 2020. The authors are responsible for the content of this publication.

SPONSORED BY THE Federal Ministry of Education and Research

Agenda

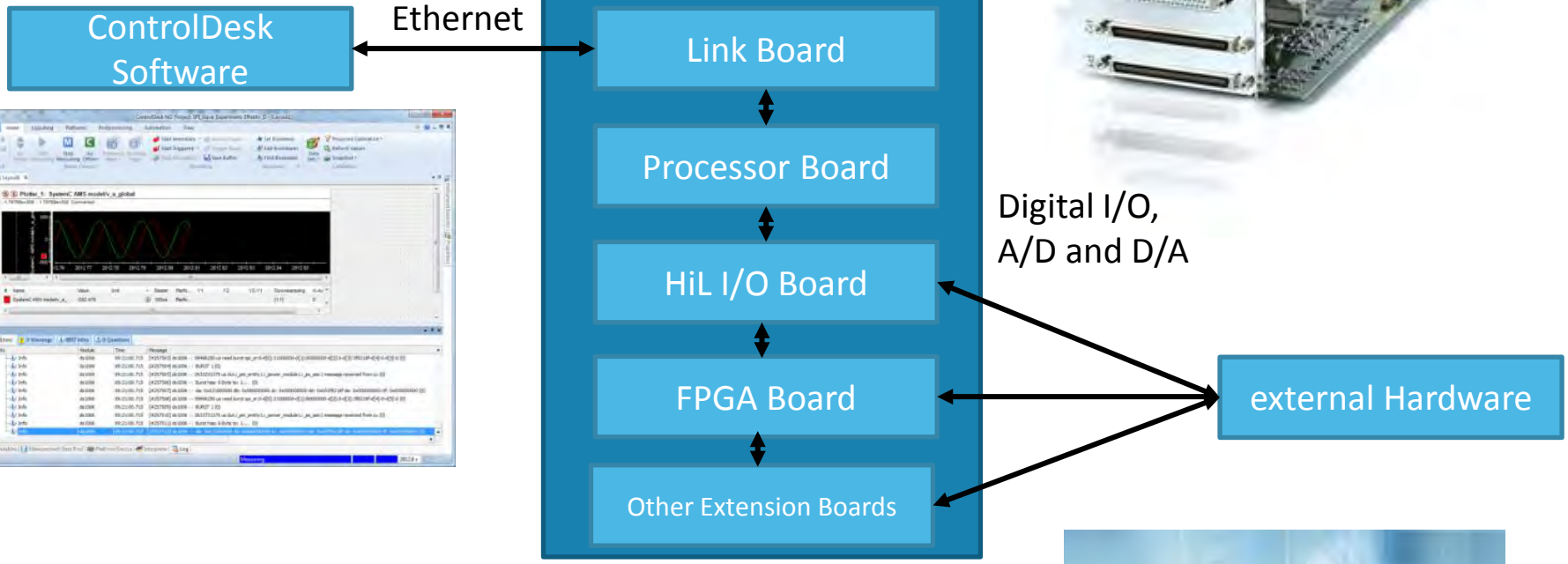
1. **Introduction**
   - EffektiV Project
   - HiL Intro
   - Example System
   - Motivation: Why HiL?
2. Tooling
3. Failure Injection
4. Demonstrator

Research project EffektiV

- Virtual stress tests for intelligent motion control systems
- German BMBF research project
- Fraunhofer as subcontractor of SIEMENS
- Consortium

- Our Focus: Hardware in the Loop Simulations with SystemC AMS

HiL Introduction



- **Other Extension Boards for specific needs (high speed or precision) or additional interfaces (LIN …) available**

Source: dSPACE

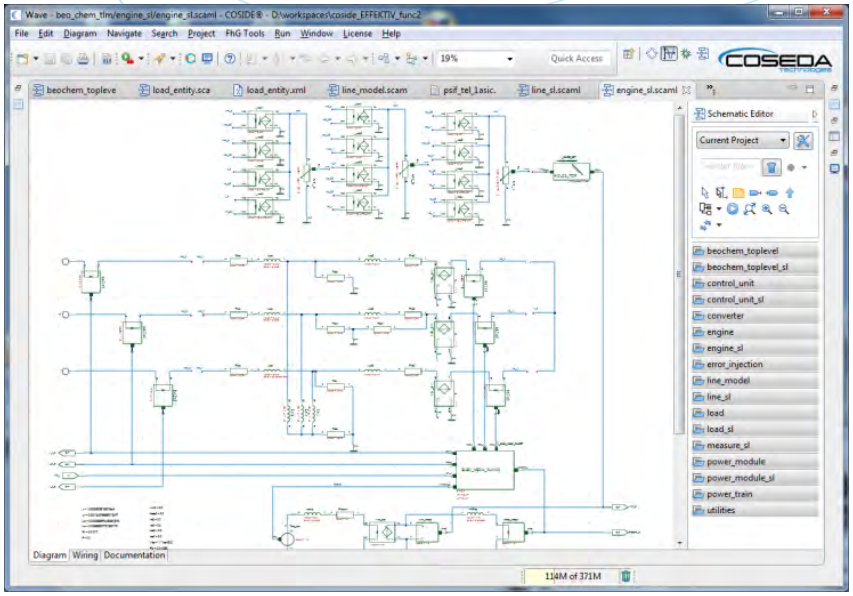
Why HiL?

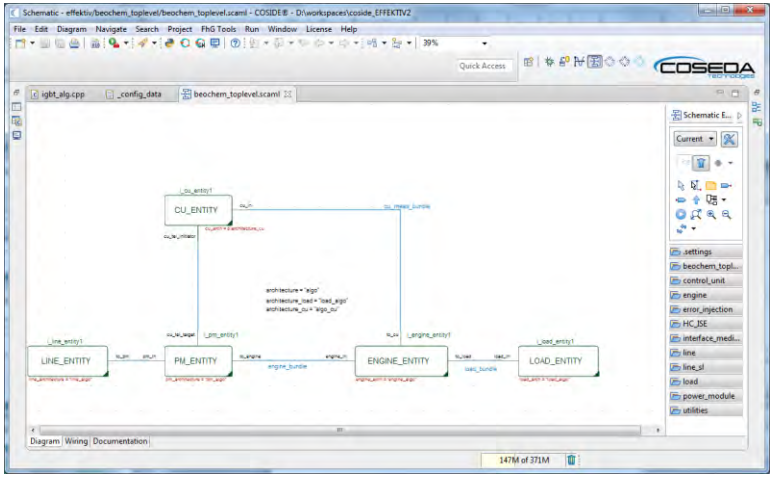
- **Debug –** Accessibility and Traceability from Signals for error root cause analysis
- **Speed/Detail –** Tradeoff between model speed vs detail level
- **Failure Injection –** Possibility to manipulate expected behavior to evaluate system robustness

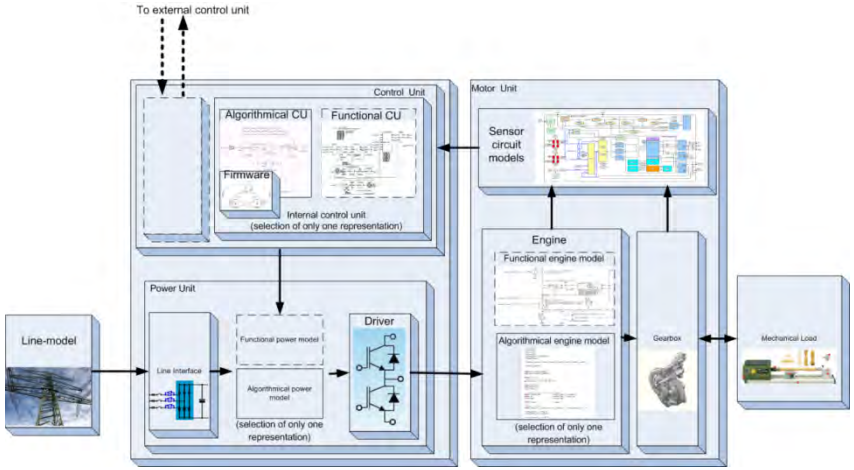
Pure Virtual	Real Hardware	Mixed (HiL)
• Debug ++ • Environment model (+) - Virtual Sensors • Speed/Detail -- • Failure Injection ++	• Debug -- • Environment model (+) - Hardware Sensor • Speed/Detail ++ • Failure Injection --	• Debug + • Environment model (++) • Both possible • Speed/Detail + • Failure Injection +

Example System

- **Motion-Control-System**
  - Control Unit (CU)
  - Power Module (PM)
  - Engine
  - Load
  - Line-Module







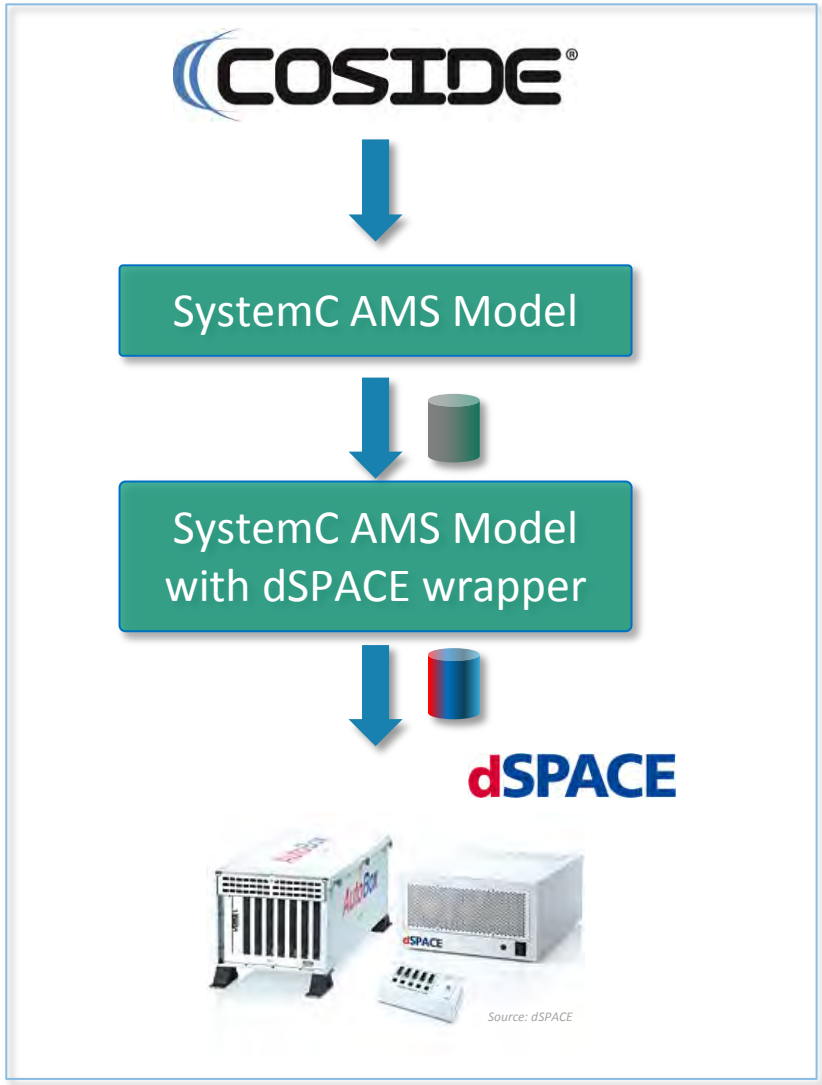
Why HiL? (2)

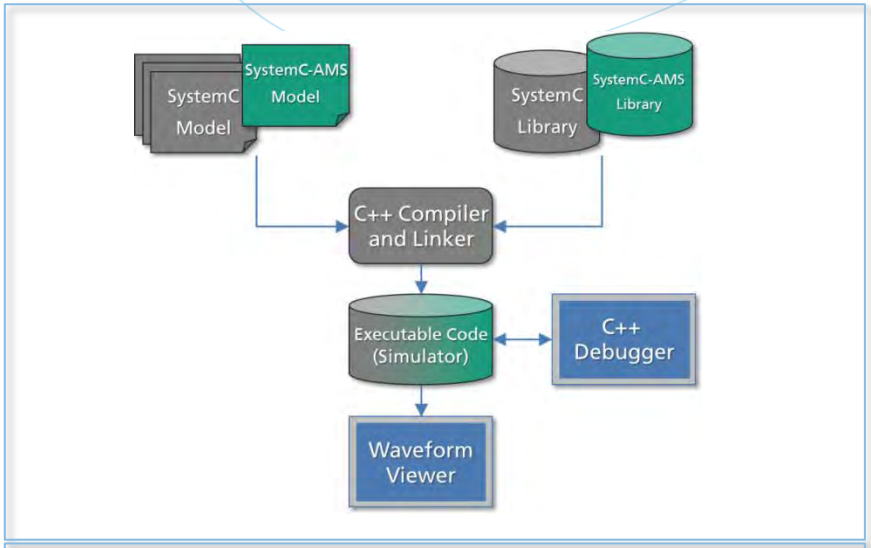
- **Task: Firmware development on the Control Unit (CU) for an asynchronous Motor**
- **Setup:**
  - CU as prototypes or hardware revision
  - SystemC AMS of PM, engine, load, line model running on dSPACE
- **Advantages:**
  - Realistic Firmware environment for CU through the use of real hardware
  - Use of proven failure injection models within SystemC AMS
    - Enables also the exploration of critical corner cases, which would destroy the real hardware

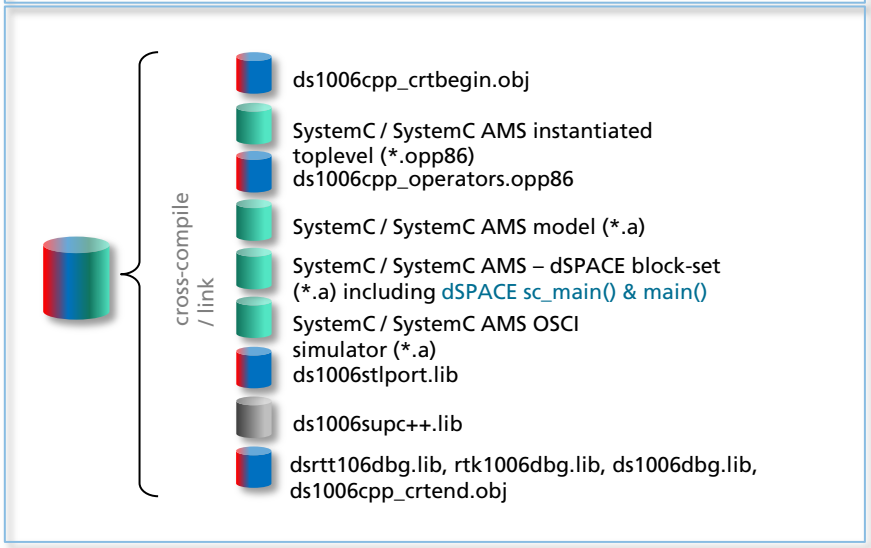
Agenda

1. Introduction
2. **Tooling**
   - SystemC AMS on dSPACE
   - Tooling Environment
3. Failure Injection
4. Demonstrator

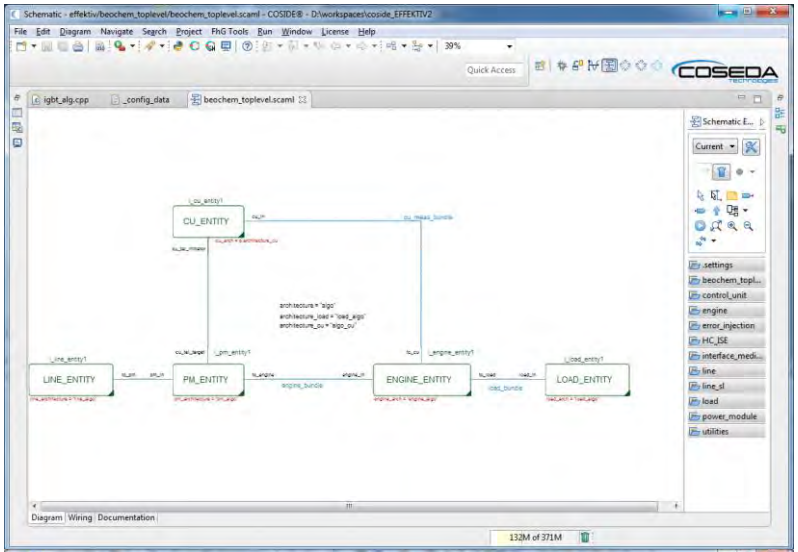
SystemC-AMS Executable

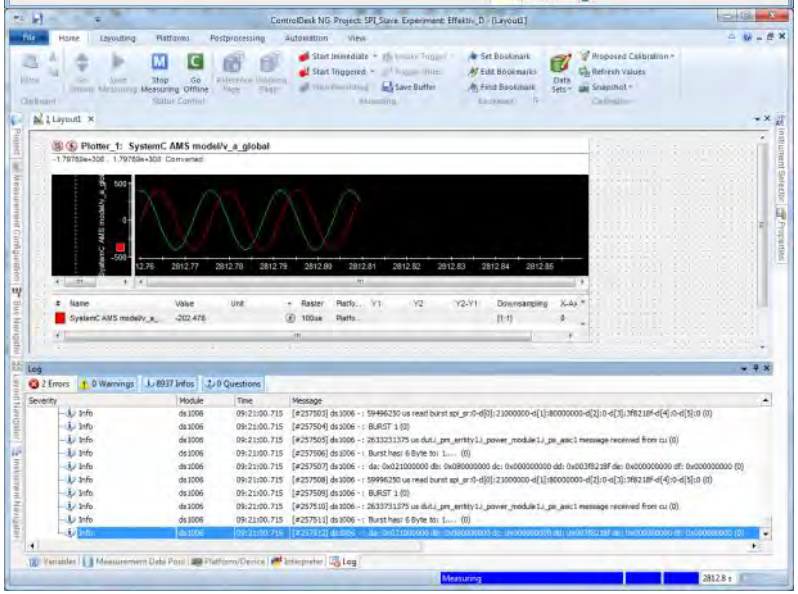


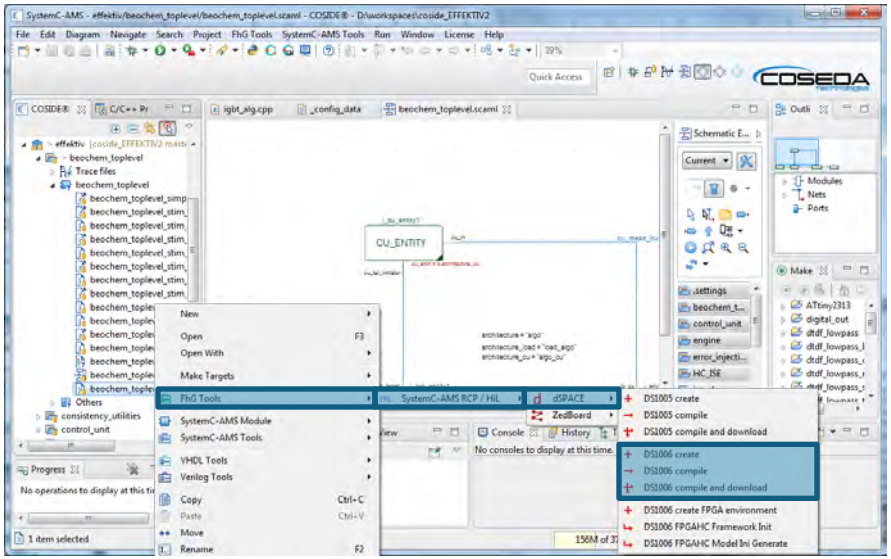




Development Environment

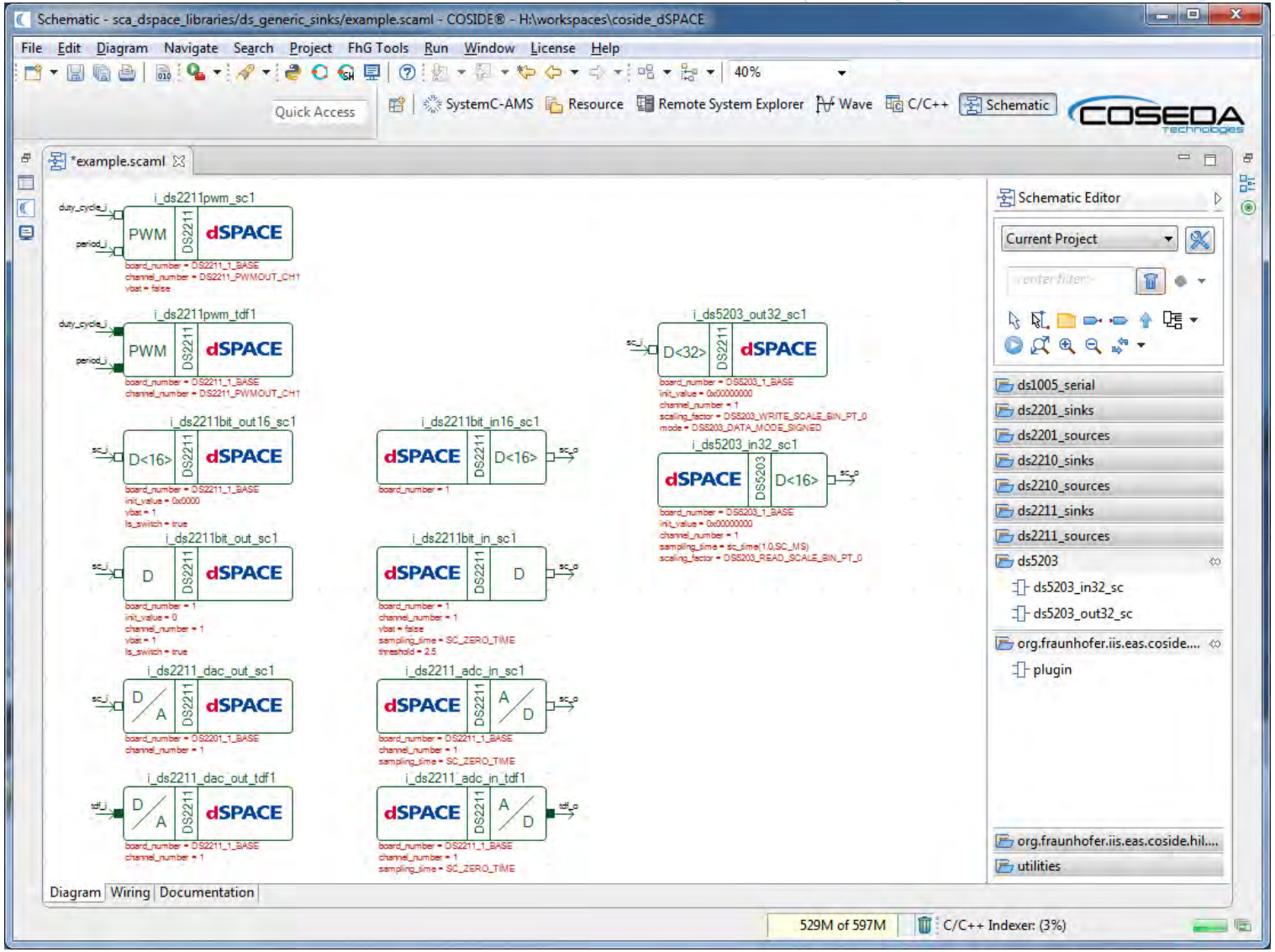






a) COSIDE® model entry
b) dSPACE specific Menus
c) ControlDesk Waveforms & Console

Interface Library



Agenda

1. Introduction
2. Tooling
3. **Failure Injection**
4. Demonstrator

Fault Injection

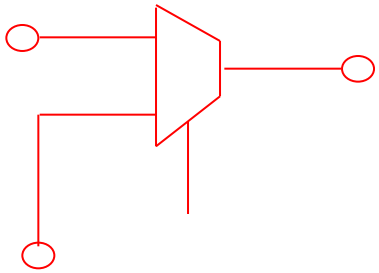
- **Properties**
  - Non intrusive failure injection, without model changes
    - Failure cases are connected into the netlist dynamically during runtime
    - Thread based for independent de/-activation
  - Failure scenario are build up hierarchically
    - Reusability (e.g. failure structures and scenarios)
    - Unified interfaces (for e.g. initialization, de/-activation)
  - Error location and time can be scattered statistically
    - Use of extensive statistic library

Fault Injection - hierarchical structure

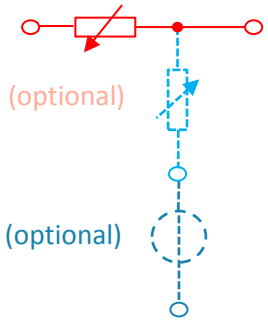
- **Hierarchical test structure**
  - Testbench
  - Testcase/Stimuli without failure injection ← includes additional fault stimuli
- **Extended hierarchical test structure with failure injection**
  - Failure stimuli (describes location and time behavior)
  - Fault scenario – combination of Fault models e.g. SPI
  - Fault models – e.g. Stuck at, Open-Short, Cross Talk …
  - Low-Level-Fault models (e.g. SC/TDF: MUX, ELN: resistor…)

COSIDE® Fehlerbibliothek

For sc/ tdf connections



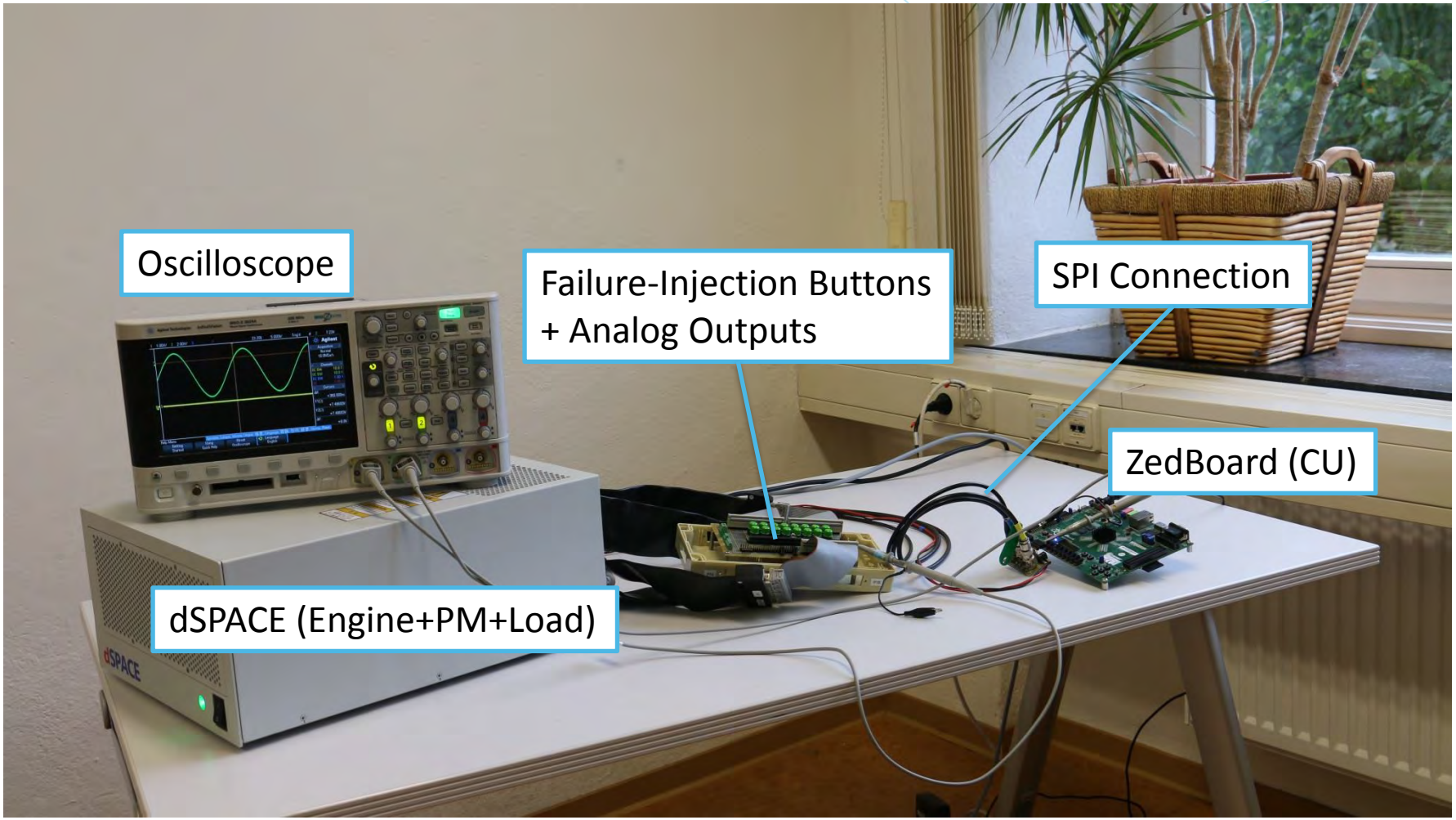
for ELN networks



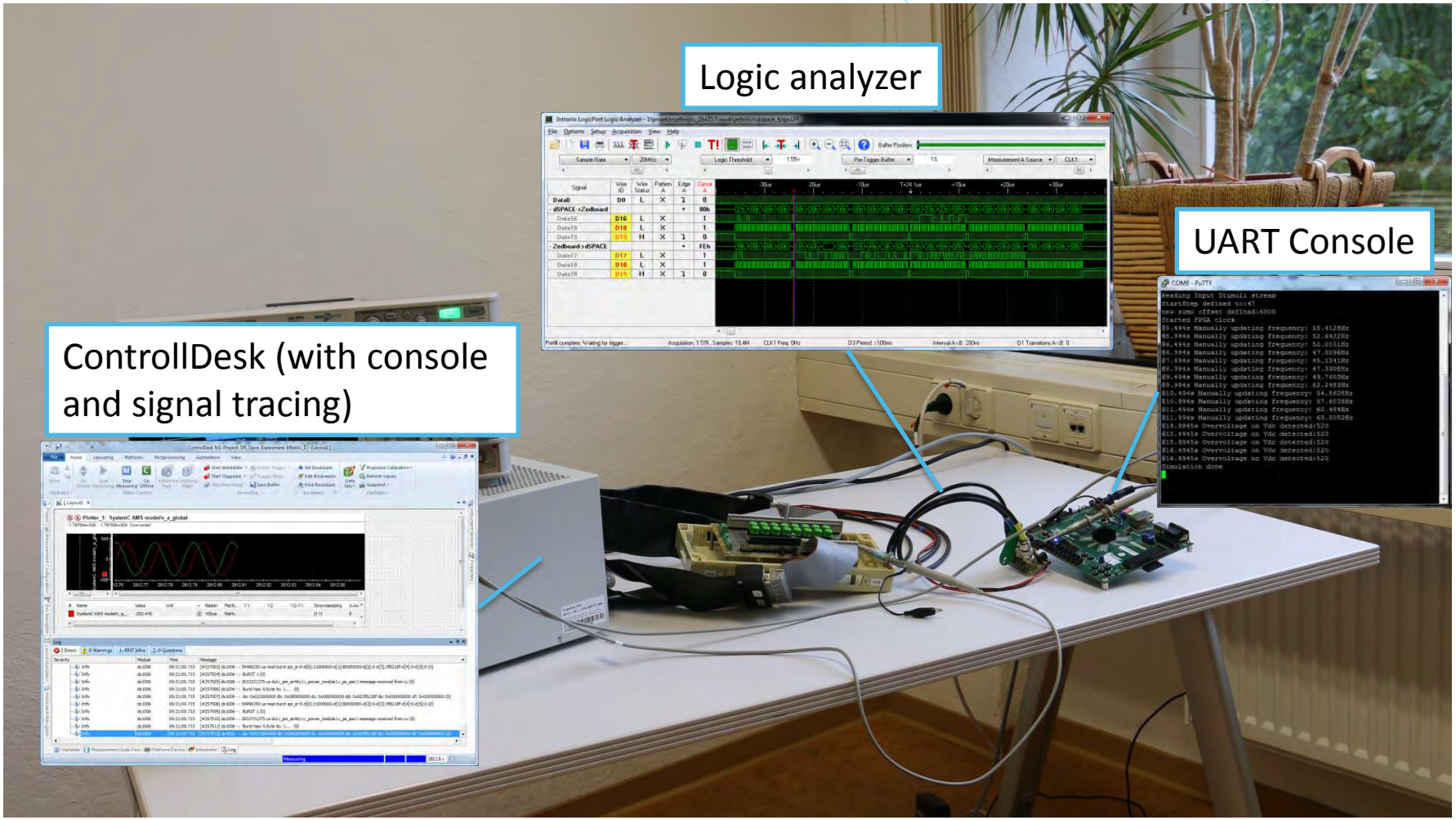
Agenda

1. Introduction
2. Tooling
3. Failure Injection
4. **Demonstrator**

Prototyp Demonstrator



Debug interfaces



Acknowledgement

The presented work has been partially supported by the German Federal Ministry of Education and Research (BMBF) under the grant 01IS13022

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Thank you for your attention!

Paul Ehrlich
- Heterogeneous System Specification
- Paul.Ehrlich@eas.iis.fraunhofer.de
- +49-351-4640-756