LAUTERBACH
DEVELOPMENT TOOLS

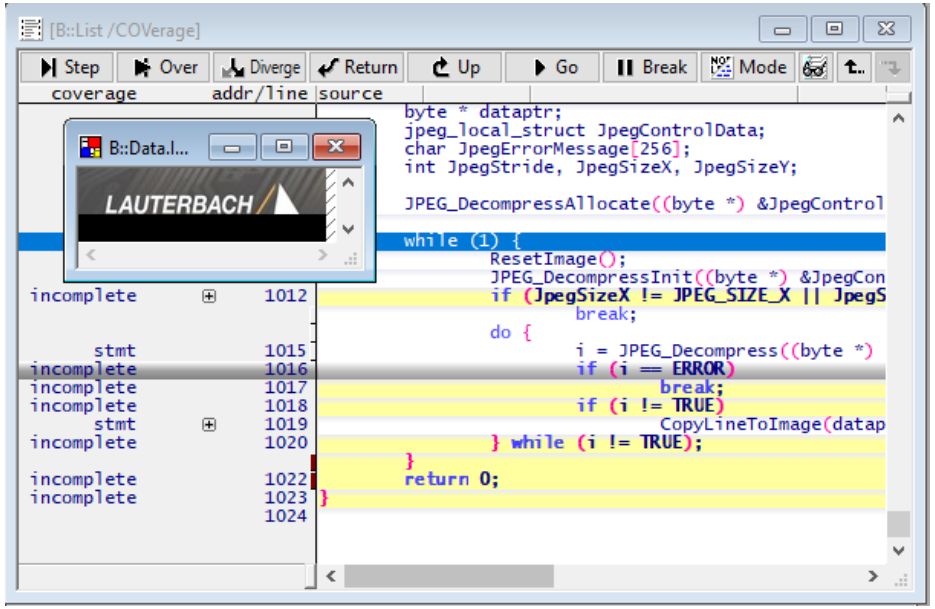
Speeding Up DO-178 Safety Certification with Trace-based Code Coverage

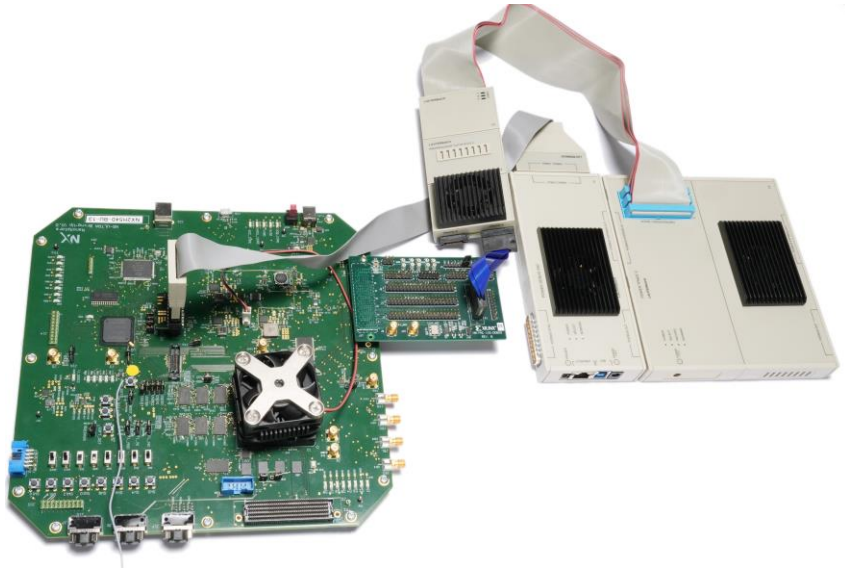
13th -15th June 2023

Ladislav Řehák ladislav.rehak@lauterbach.com

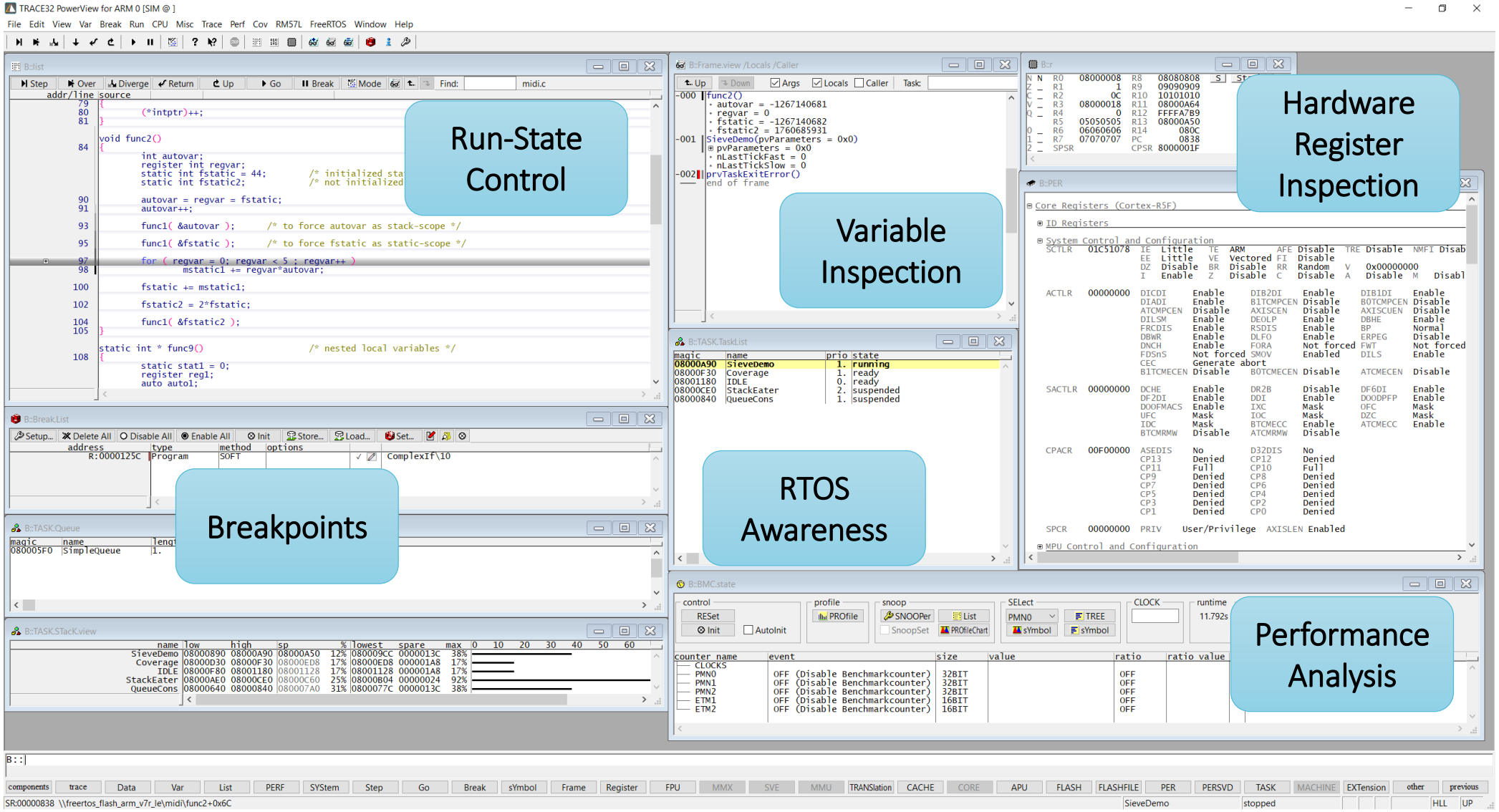
Introduction

- Debugging of NanoXplore SoCs
- Tracing of NanoXplore SoCs
- Code Coverage Measurement
- Lauterbach's
  Tool Qualification Support Kit (TQSK)



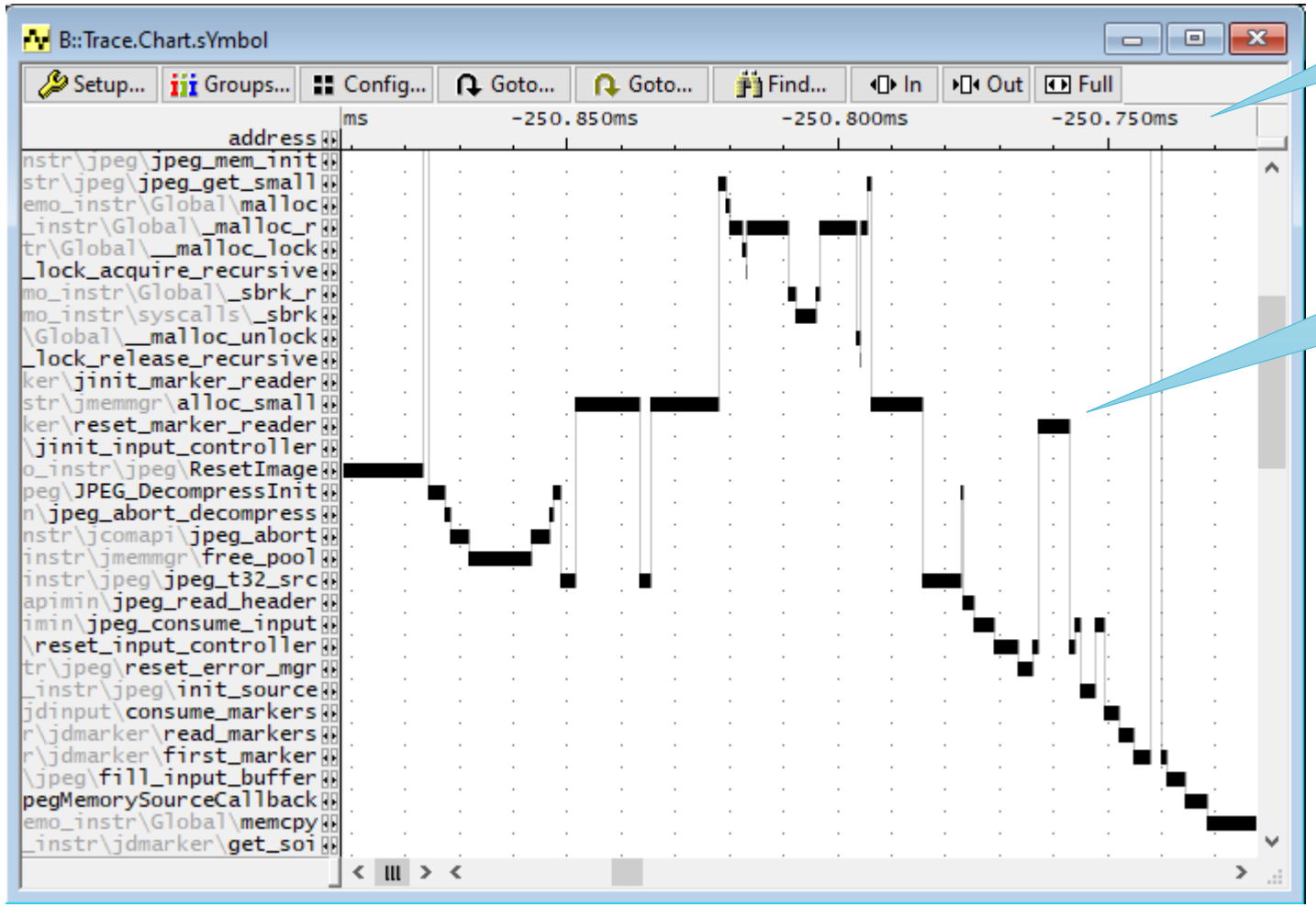


Debuggers are known for…



… but can actually do much more

Program Flow captured with Lauterbach TRACE32®



Run time until stop of trace recording

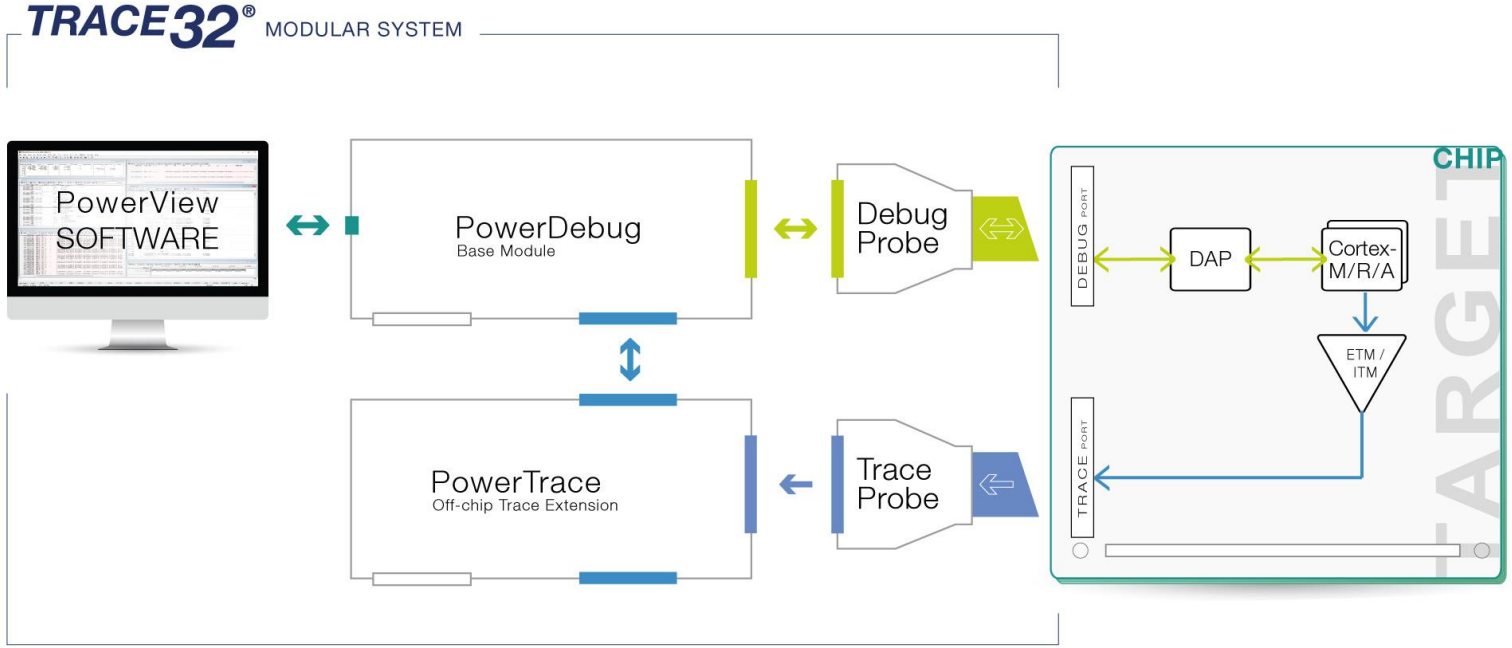
Time spent within specific program function, task

Allows capturing program flow, analyzing execution times, optimizing code

Can also provide data for code coverage analysis

A Little Bit of Theory: CPU Program Flow Trace

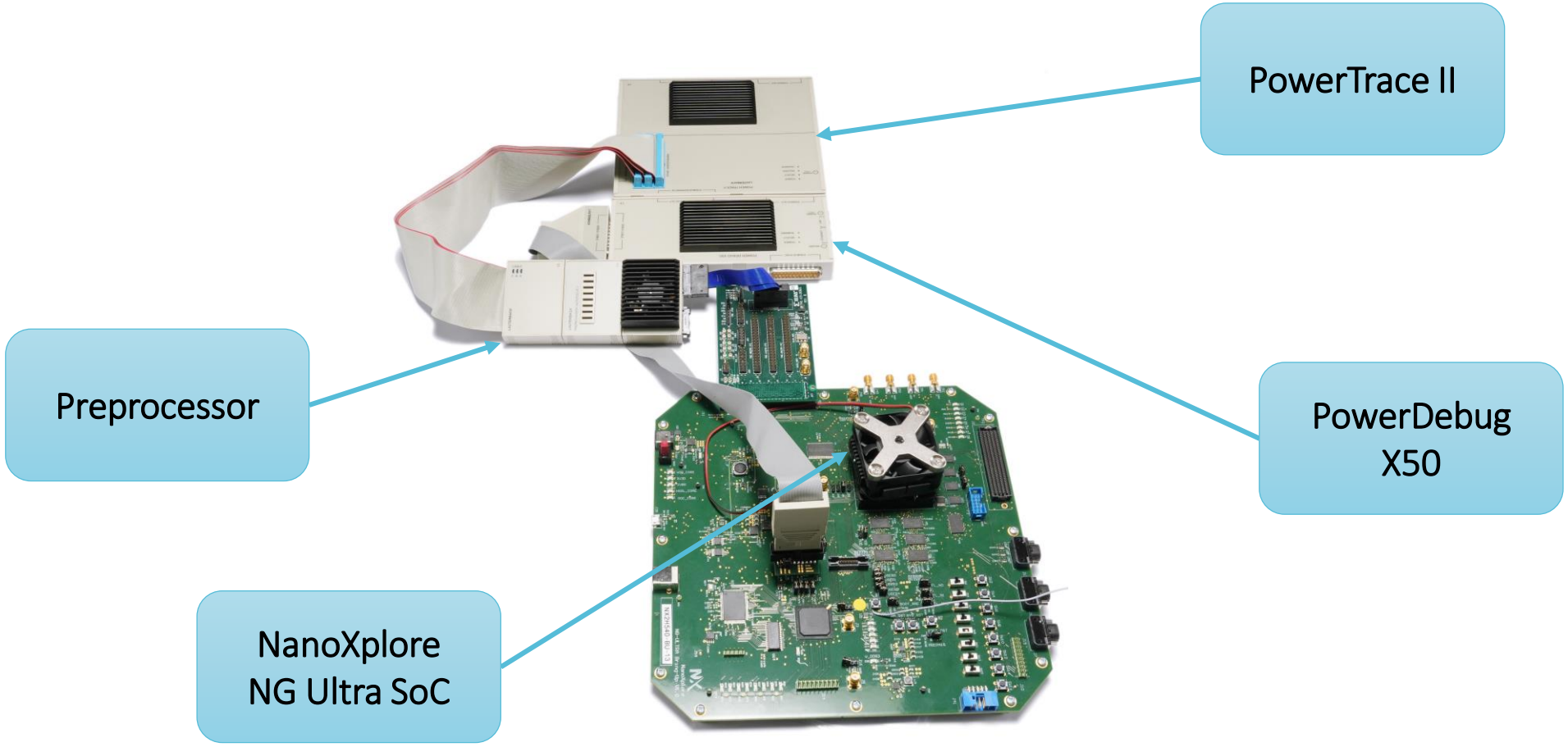
How Does it Work: Hardware Setup (I)



- Microprocessor emits program flow (trace) data via specialized on-chip blocks
- Data stream is saved in on-chip trace buffers or in trace hardware (e.g. TRACE32® PowerTrace Module)
- Later on, trace data is transferred to PC for analysis

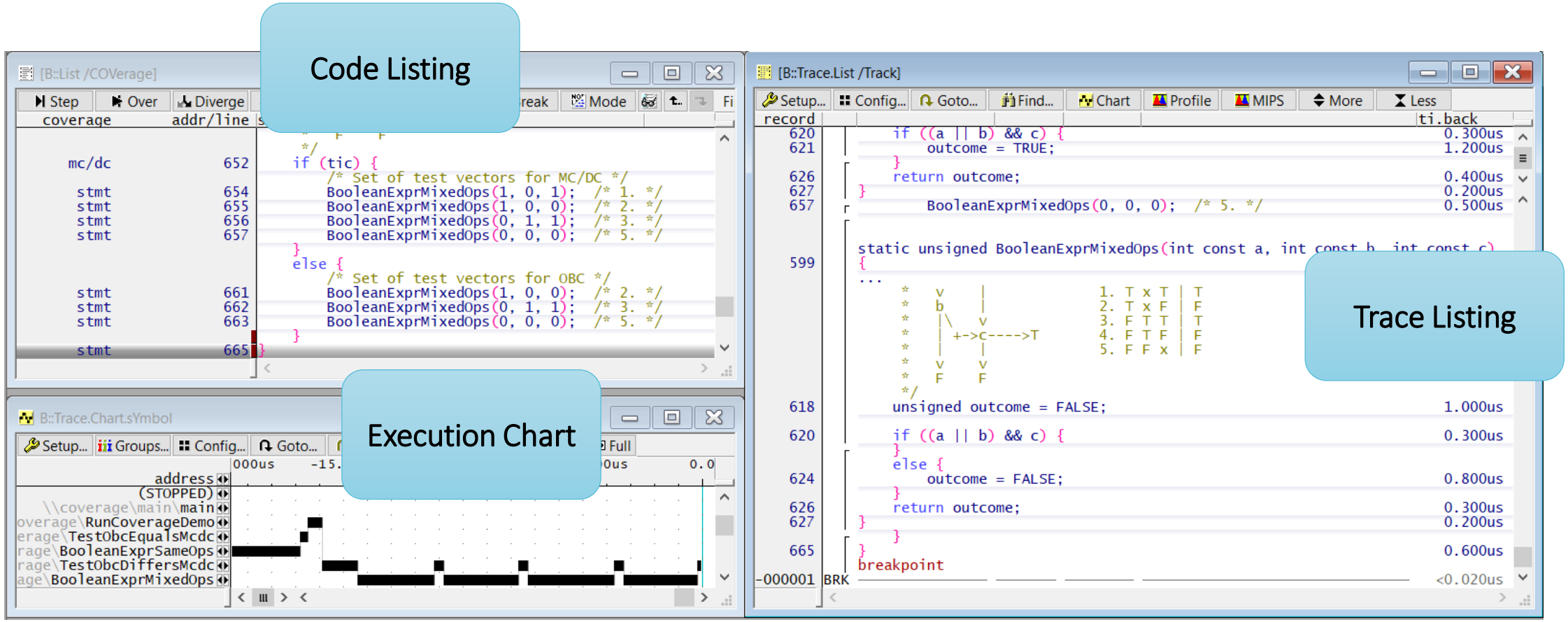
A Little Bit of Theory: CPU Program Flow Trace

How Does it Work: Hardware Setup (II)



A Little Bit of Theory: CPU Program Flow Trace

How Does it Work: TRACE32® Views (III)



How does this apply to Code Coverage?

Traditionally, Code Coverage Needs Instrumentation

A) Instrumentation of full source code* to capture program flow

B) Compile instrumented code and run code on target (or in simulator)

C) Acquire program flow data via functional interface

D) Analyze in Code Coverage Tool

Typical issues:

- Code size growth: Executable does not fit into target memory size
- RAM consumption
- Longer execution time due to overhead

*In case of object code instrumentation (OCI): similar process, but instrumentation after compilation

Trace Based Code Coverage
Results Calculated from Flow Trace

A) Targeted Instrumentation (covering trace gaps)
   - Instrumentation is required only for MC/DC and Decision coverage in cases like conditional instructions.

B) Compile instrumented code and run code on target (or in simulator)

C) Acquire program flow by TRACE32 via trace interface

D) Analyze in TRACE32 or external tool

Key Benefit

- Only small code size increase, no data memory occupied
- Less instrumentation overhead
- Execution possible on target HW possible as well as in simulator

Trace based Code Coverage

What are the challenges?

Map Decisions, Conditions and Calls

- Automatic mapping for most common architectures and compilers
- Heuristics

Compiler Optimization

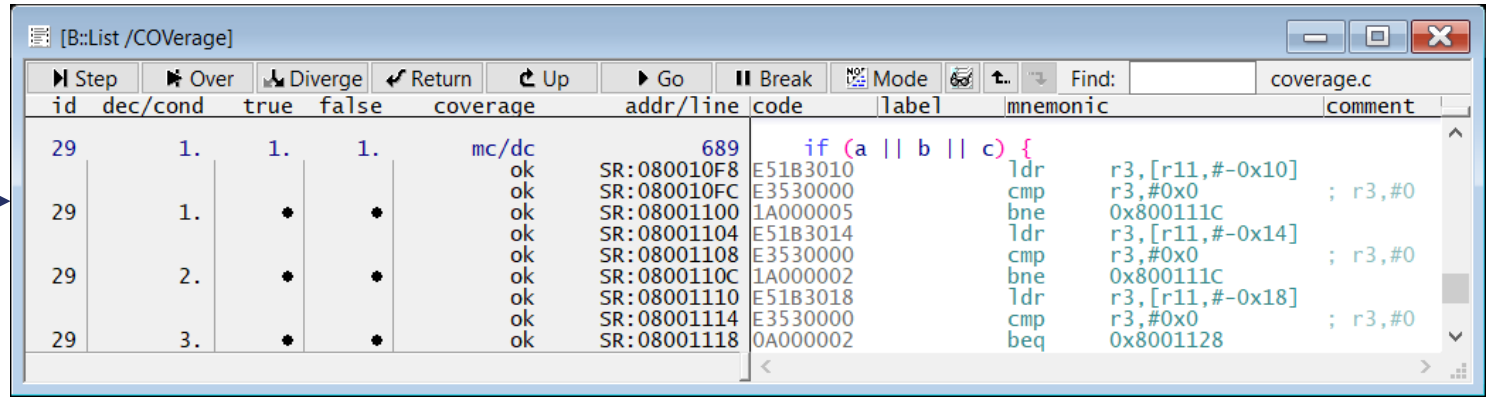
- Disable optimization or
- Use Targeted Instrumentation

Conditional Instructions not showing up in trace

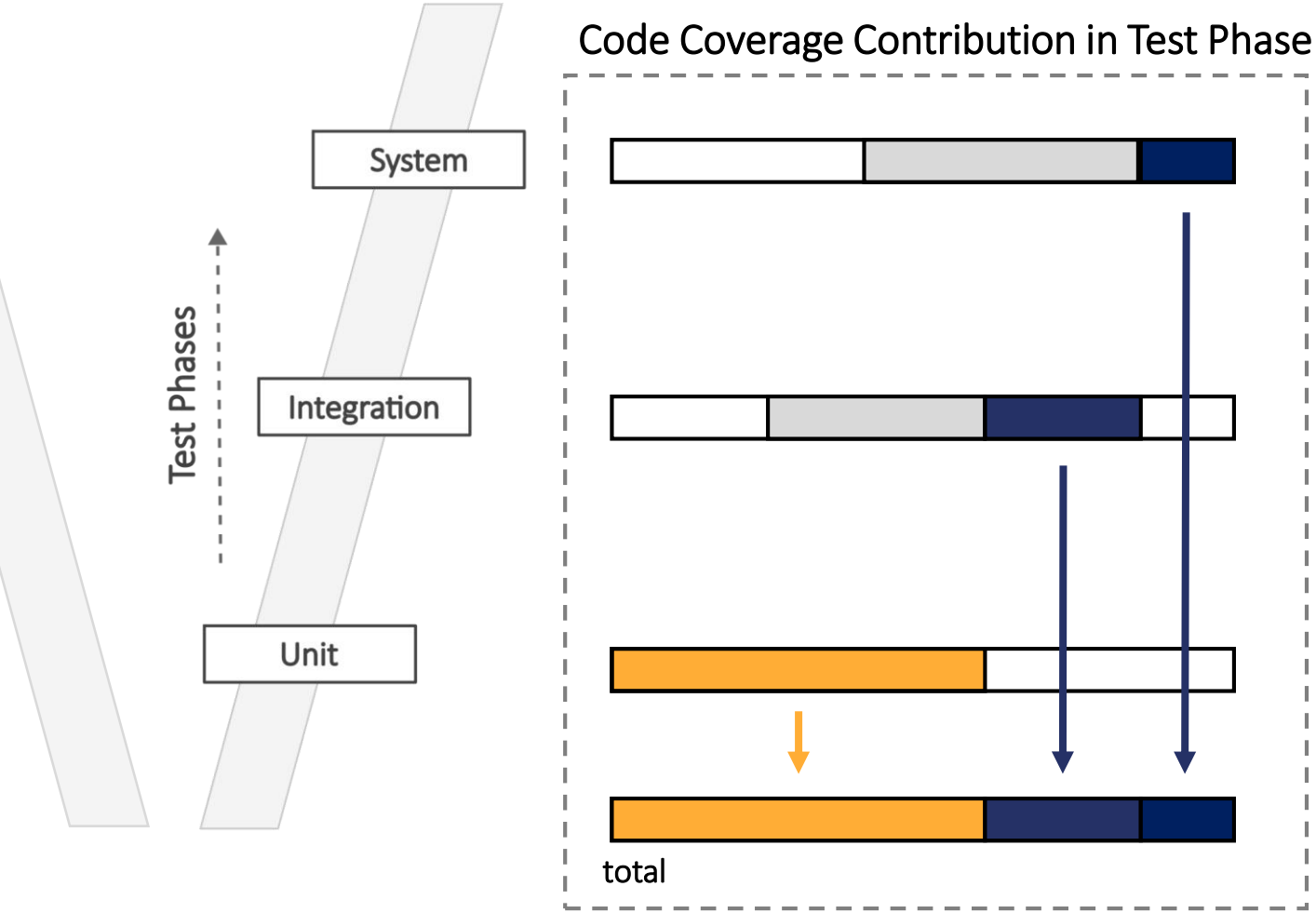
- Tune Compiler flag or
- Use Targeted Instrumentation

Many different Compiler and Architecture Combinations

- Support for combination can be added on demand
- Targeted Instrumentation as a fallback solution



Code Coverage Contribution in individual Test Phases



Code Coverage Gained Using Trace

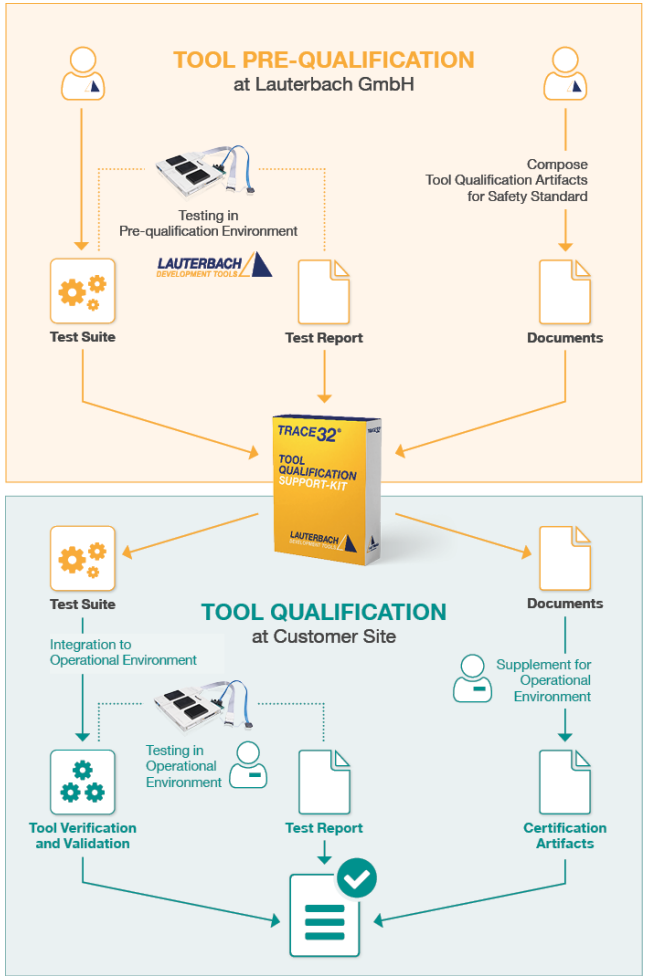
Code Coverage Gained Using Unit/Integration Testing

Code Coverage in Functional Safety Projects

- Measurement how thoroughly code has been stressed by testing
- Trace generation has no runtime overhead
- Scales well with application size
- Long-term tests can be monitored
- Code coverage can be efficiently measured throughout all testing phases
- Measurement of complex coverage criteria during later stages becomes feasible
- Less pressure to achieve high code coverage at unit test level
- Use of integration of system testing results can reduce the overall effort

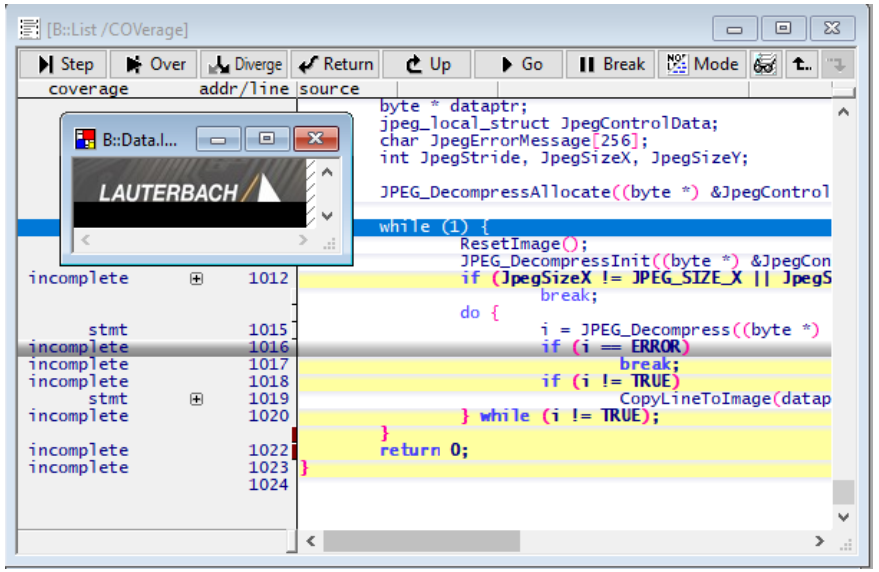
Tool Qualification Support Kit (TQSK):
Making Qualification Easy

- Domain-specific guidance for TRACE32®
- Covers safety standards for different industries
- Tool classification and application guidance for TRACE32® features
- User contributes knowledge of project context to adapt the generic guidance to project needs (use cases, process flow, operating environments, ...)
- Verification of TRACE32® in the user environment
- Test suites for requirements-based testing
- Designed for compatibility with diverse user environments



Invitation to our Booth Demo

- MC/DC coverage demo on Arm® Cortex®-R52 core in a NanoXplore NG-Ultra SoC
- Used hardware: PowerDebug X50
  - Fast USB3
  - Ethernet for remote usage in test labs
  - PCI Express interface for high bandwidth off-chip trace
- Used hardware: PowerTrace III
  - 8 GByte memory
  - Parallel trace with up to 36 signals
  - 600+ Mbit/s per signal for up to 17 signals
  - 400 MByte/s streaming speed
- Also works in instruction set simulator



Summary

- Regulatory environment of Avionics Industry leads to
  - Focus on early hardware-based testing
  - Object-code coverage helpful for projects with high criticality
  - Projects with high criticality use compiler optimizations very cautiously
- TRACE32® provides Long-term commitment for legacy platforms
- TRACE32® is your one-stop solution for all code coverage needs
- TRACE32® is more time/space efficient than other code coverage tools
- TRACE32® provides new opportunities for integration and system testing

- → TRACE32® speeds up DO-178 Safety Certification with Trace-based Code Coverage

LAUTERBACH
DEVELOPMENT TOOLS

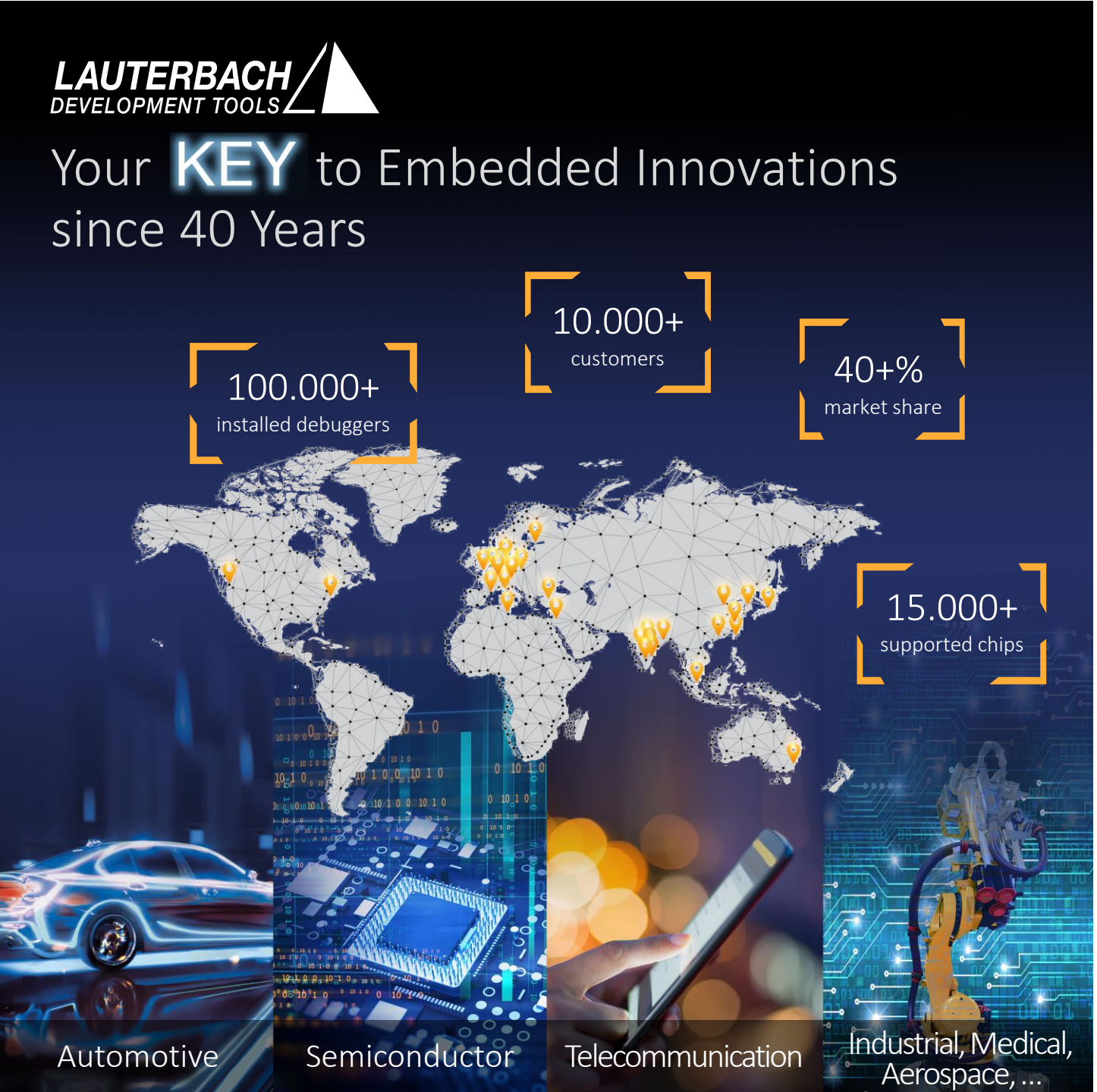
Your KEY to Embedded Innovations since 40 Years

100.000+
installed debuggers

10.000+
customers

40+%
market share

15.000+
supported chips



Questions?

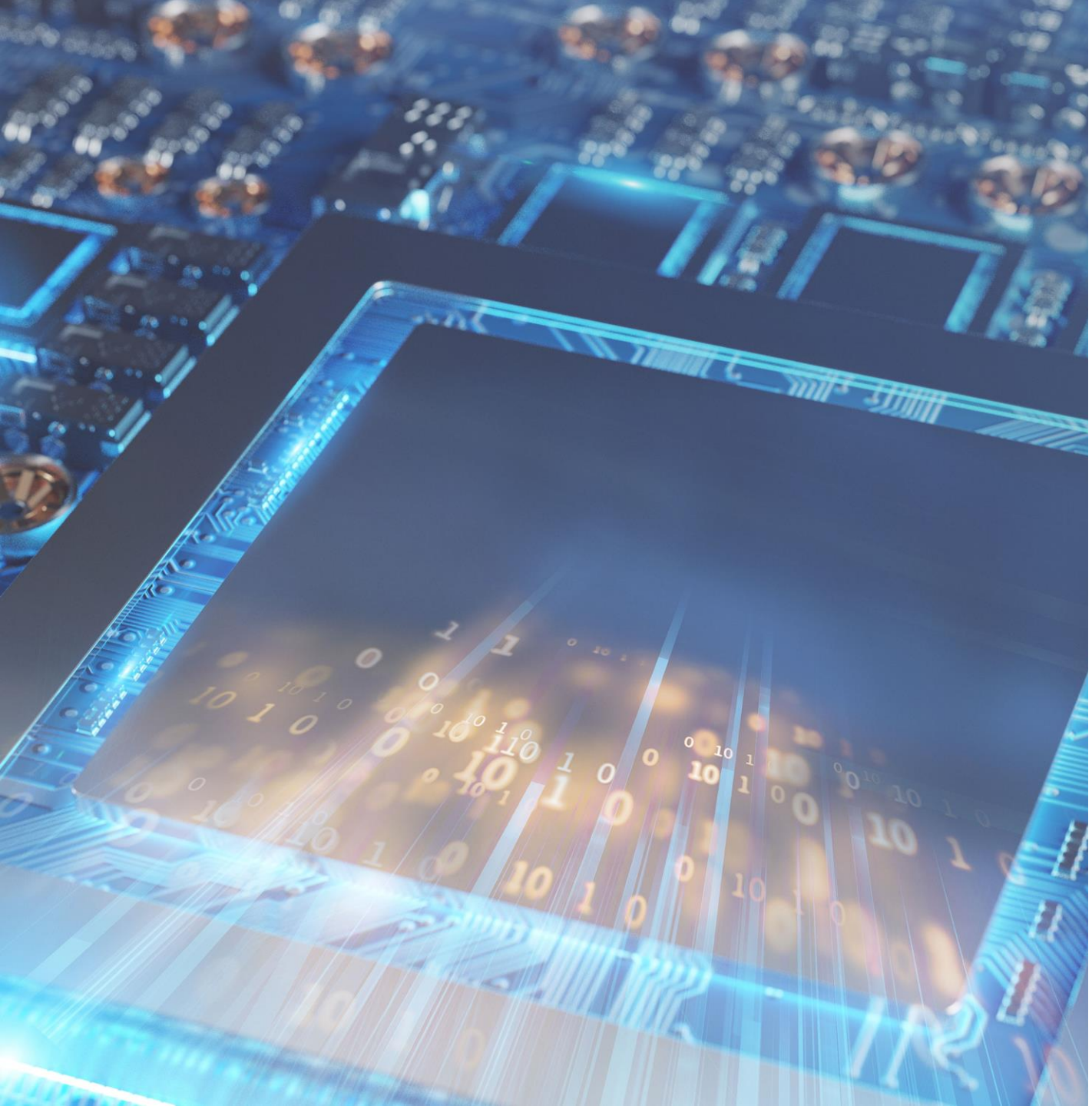
Speeding Up DO-178 Safety Certification with Trace-based Code Coverage

Live-Demo & Discussion at Lauterbach Booth

Ladislav Řehák
Systems Engineer

Lauterbach GmbH

Ladislav.Rehak@Lauterbach.com



THANK YOU!