Highest availability needs in-field debugging

Albrecht Mayer, Distinguished Engineer
Infineon automotive microcontroller
MPSoC 2025, Megeve

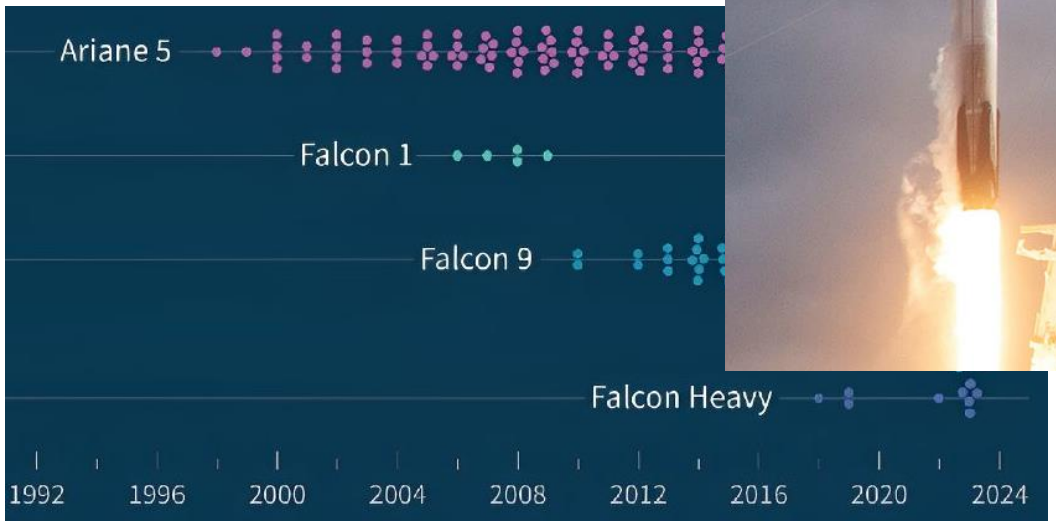
Supported by:
Bundesministerium für Bildung und Forschung
Mannheim EMDRIVE

In-field debugging

ESA: First time right
SpaceX: Let’s do it

Complex systems:
You can reduce the number of failures in the field by testing, but some effects will not be foreseeable

How to learn best from field failures?



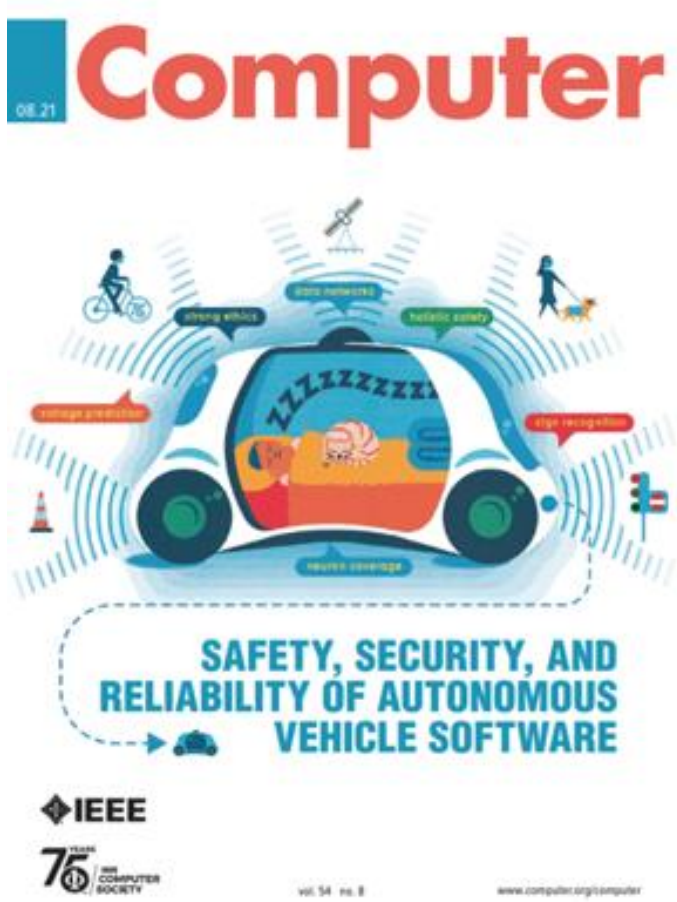
https://spacenews.com/nasa-investigation-linked-2015-falcon-9-failure-to-design-error/

Table of contents

Table of contents

Motivation for in-field debugging

- **Availability is affected more by software issues than by hardware faults: factor 10x**
- Understanding very rare issues
- Traditional diagnosis paradox:
  Writing error routines for unknown errors
- Non-intrusive hardware on-chip trace has no fundamental conflict with safety

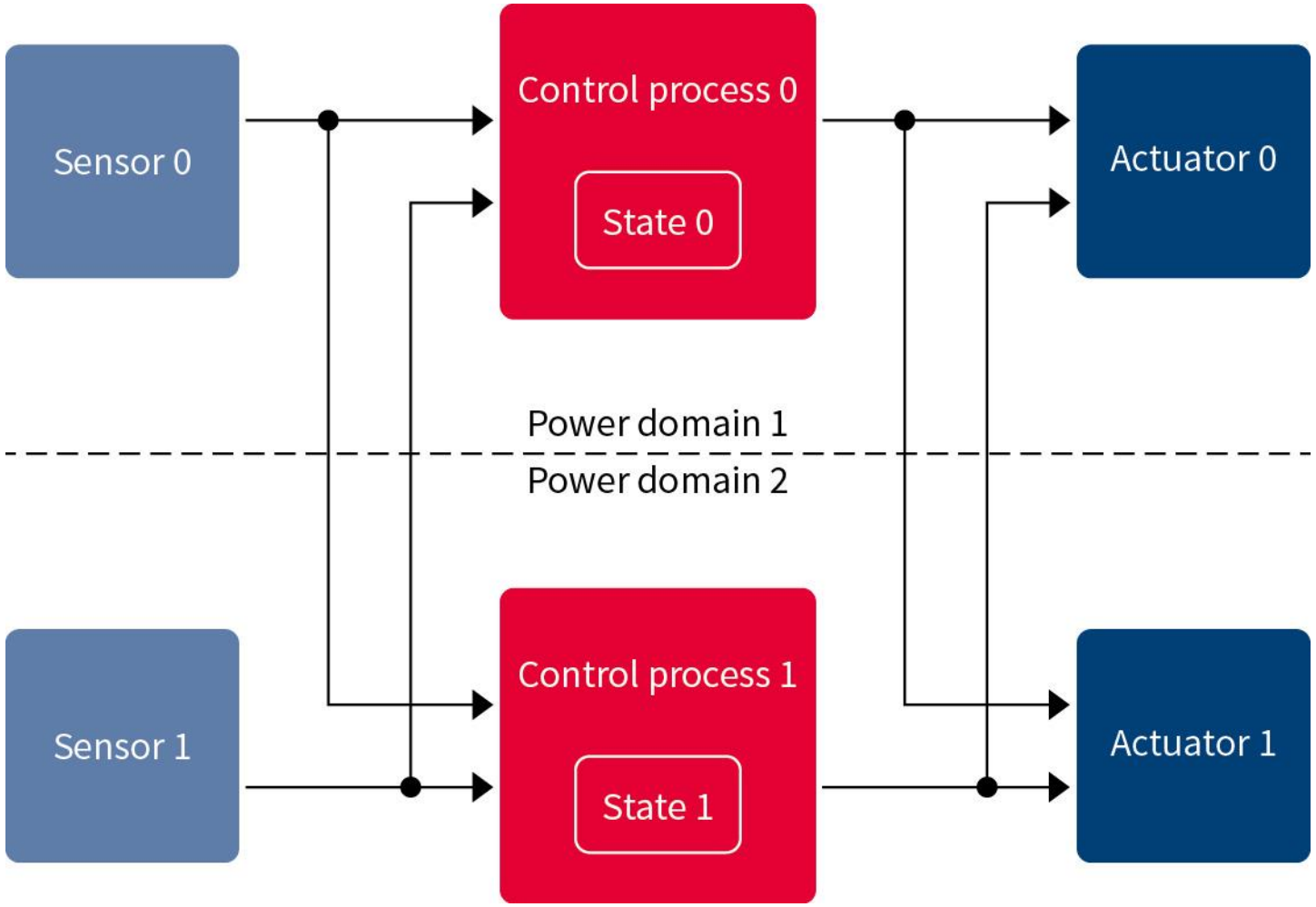


Source: Computer, vol. 54, no. 08, https://www.computer.org/csdl/magazine/co/2021/08

Detection of faults for HW

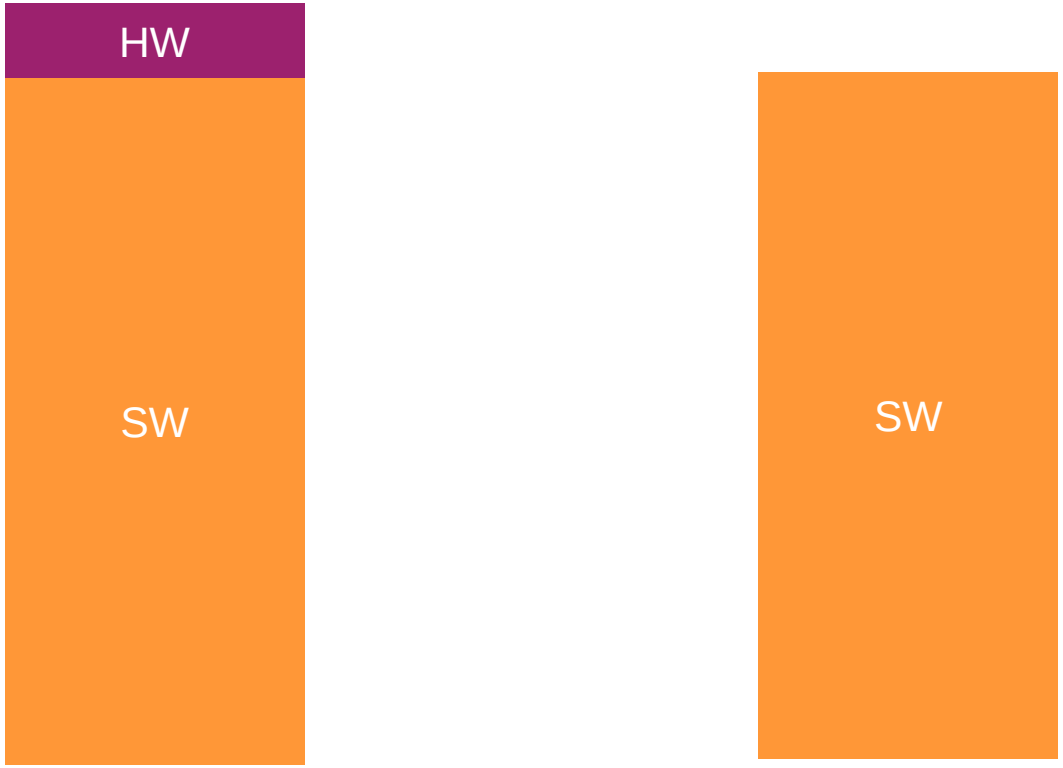
- FIT – Failures per 10E9 operating hours
- 1 FIT ~ 1 failure for 100000 cars with 10000h operation during lifetime
- Safety FIT rate != availability FIT rate
- Safety FIT := undetected dangerous faults
- Availability FIT := undetected + detected dangerous faults (safe state)
- AURIX™ class chip has
  - < 1 FIT safety FIT rate
  - **100x higher availability relevant FIT rate**

Fail-operational on one slide:
Random HW faults are no problem



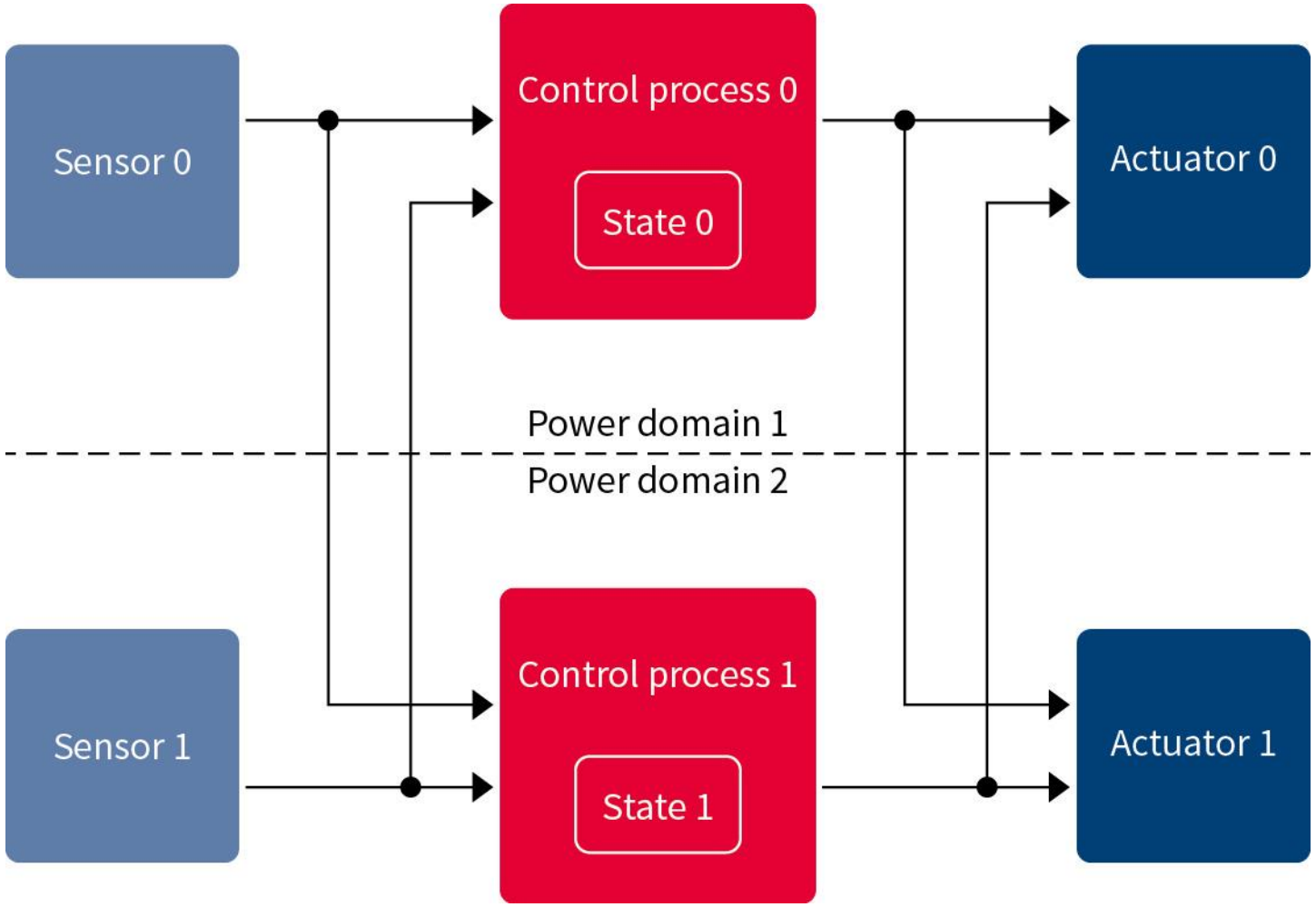
The dark zone of the availability FIT rate

Availability FIT rate: ~1000 FIT for HW + SW

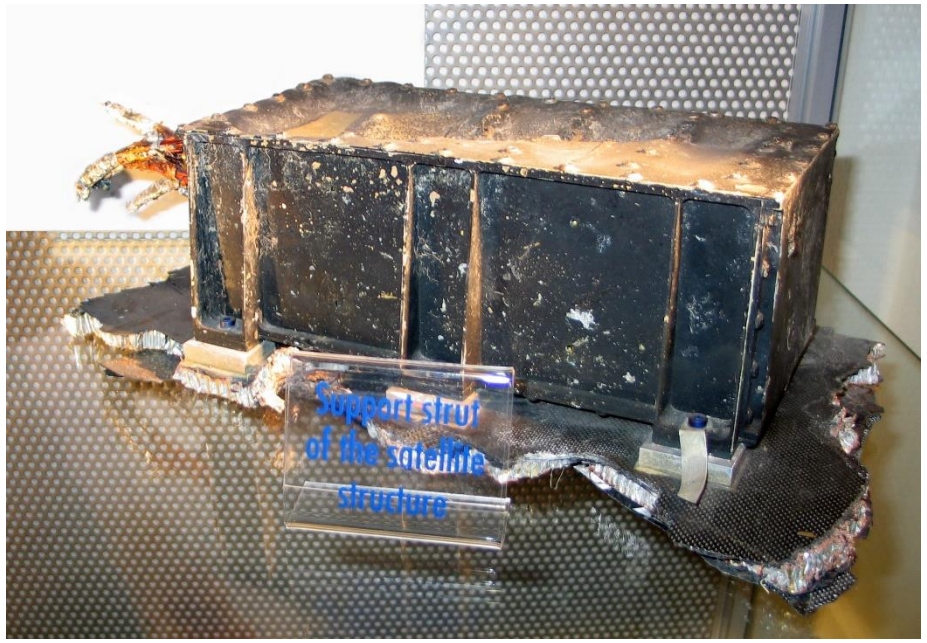


OEMs: > 90% of availability issues by software

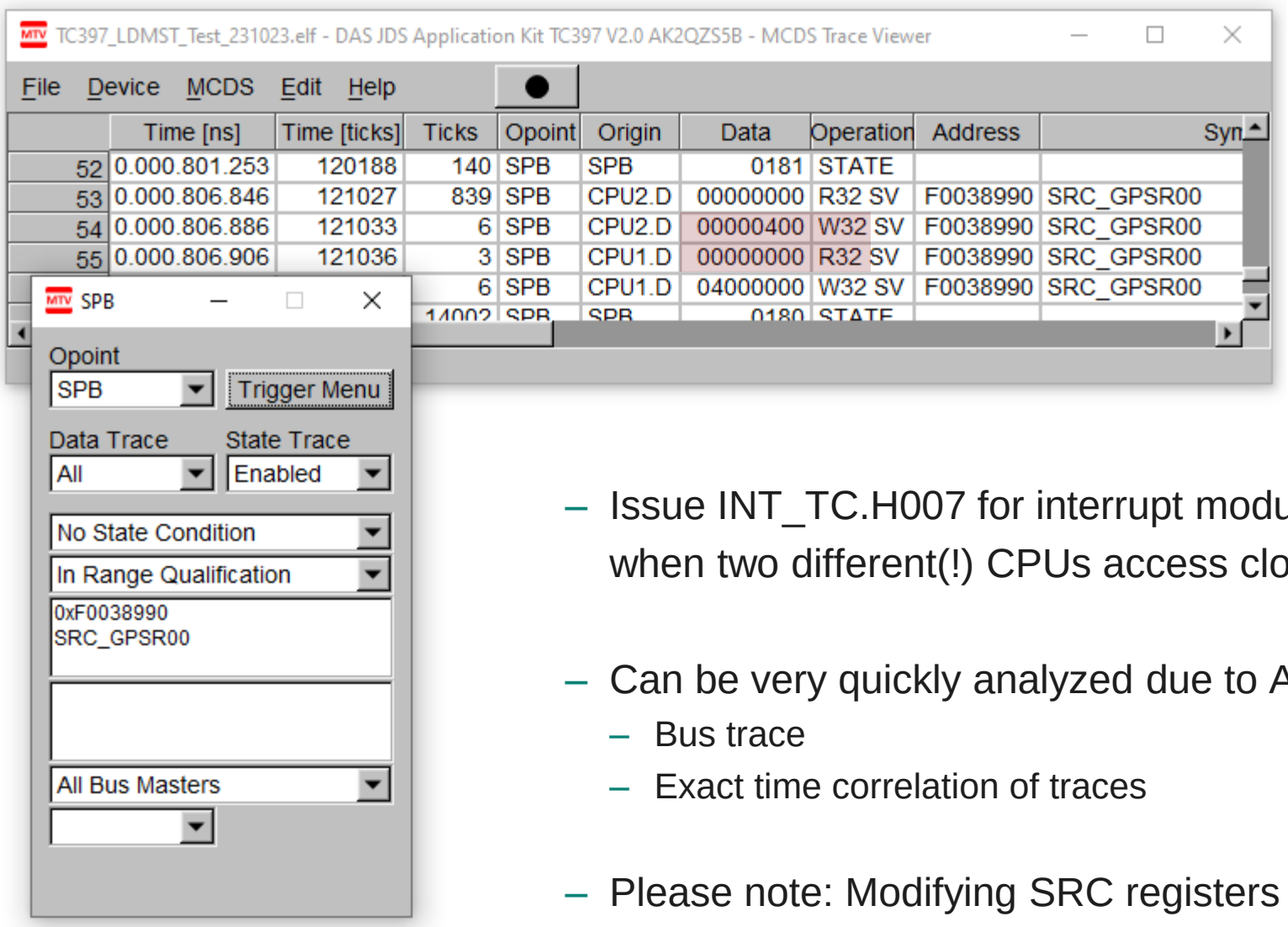
Fail-operational on one slide:
Systematic software faults are a big problem



Ariane 5 software bug
You can be sure they tried hard to avoid this



Old story and they didn't use AURIX …
… there can be always surprises



- Issue INT_TC.H007 for interrupt module: Write value of LDMST can get lost when two different(!) CPUs access close to each other
- Can be very quickly analyzed due to AURIX trace features:
  - Bus trace
  - Exact time correlation of traces
- Please note: Modifying SRC registers with different CPUs is strange

Table of contents

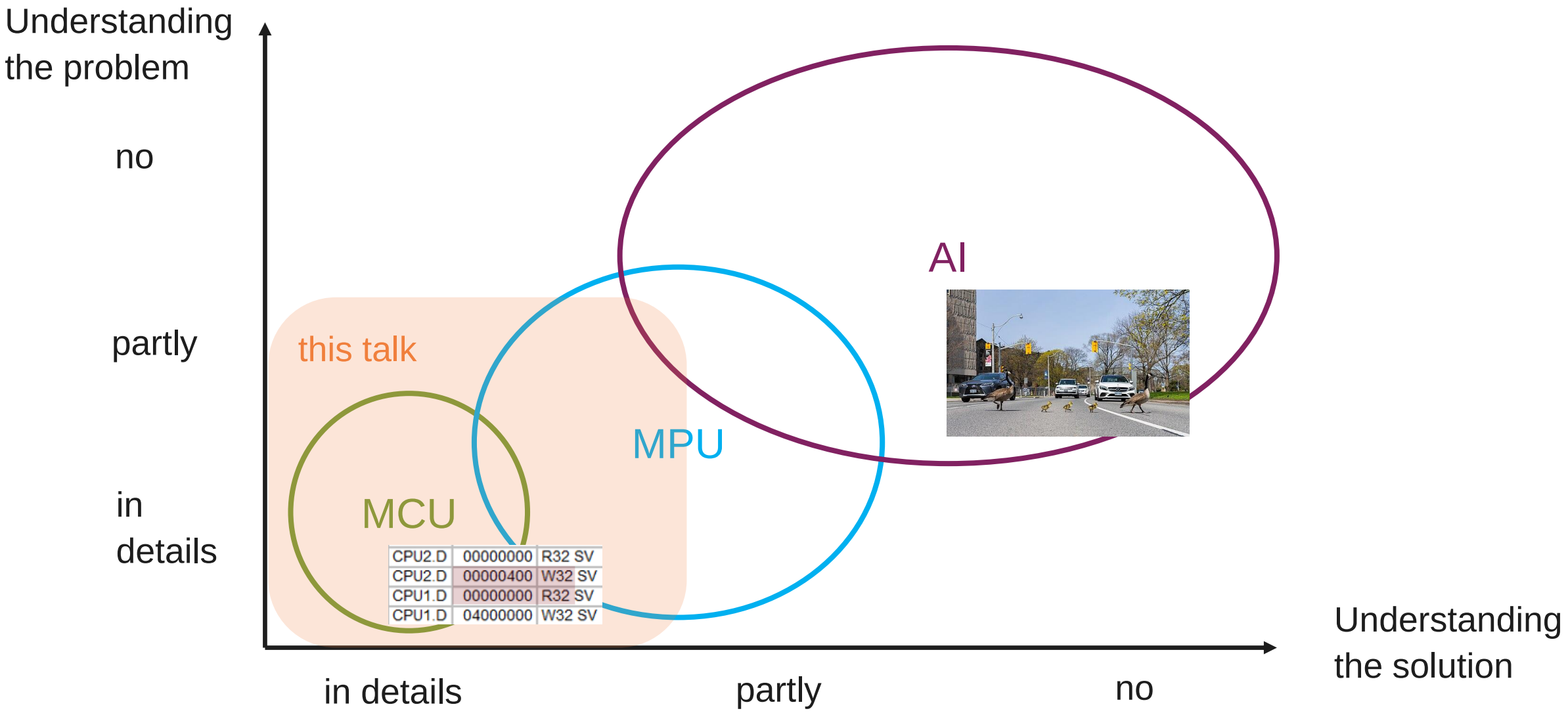
In-field debugging

Restrictions
- Safety → trace only
- Security → access protection
- Cost → limited trace resources

Opportunities
- “Test stand” with thousands of units
- “Text stand” with unexpected test vectors
- Find bugs before there is an accident

Strategy
→ Enable existing tools where possible
→ Create new methods and tools where needed

Problem and solution space from SW programmer perspective

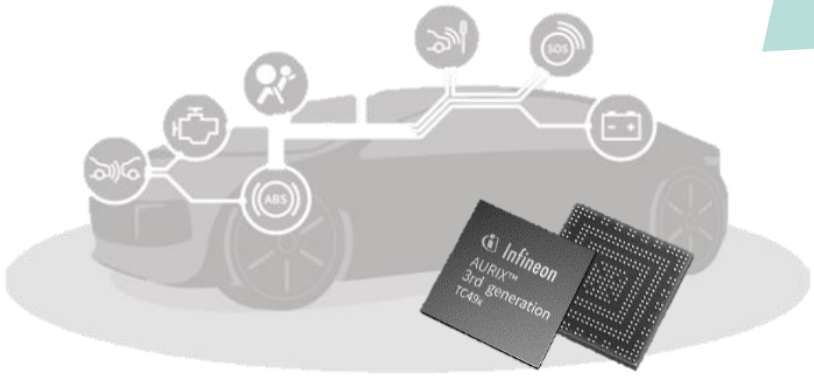


Example AURIX™ TC4x

Cloud
- Anomaly (SW bug, intrusion) analysis
- Big data AI analysis
- Service improvement

→ **software update**

Safe MCDS trace,
controlled by a **secured** environment



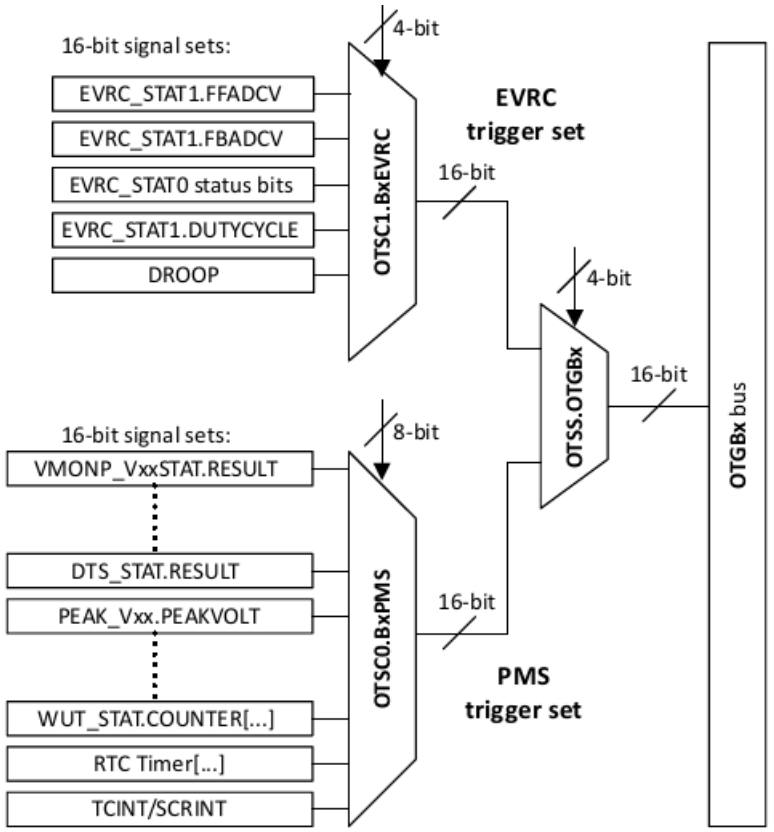
Advanced TC4x features:

- ✓ Parallel time aligned multi-core trace
- ✓ Tracing in a safe system
- ✓ Secured remote access
- ✓ Ethernet connection for tooling
- ✓ Seamless integration of all cores including security core
- ✓ Secured debug isolation for a virtualized system

Nothing has happened (yet)

- Debugging needs a symptom (car falls off cliff)
- Safe system design: Check if close to the cliff

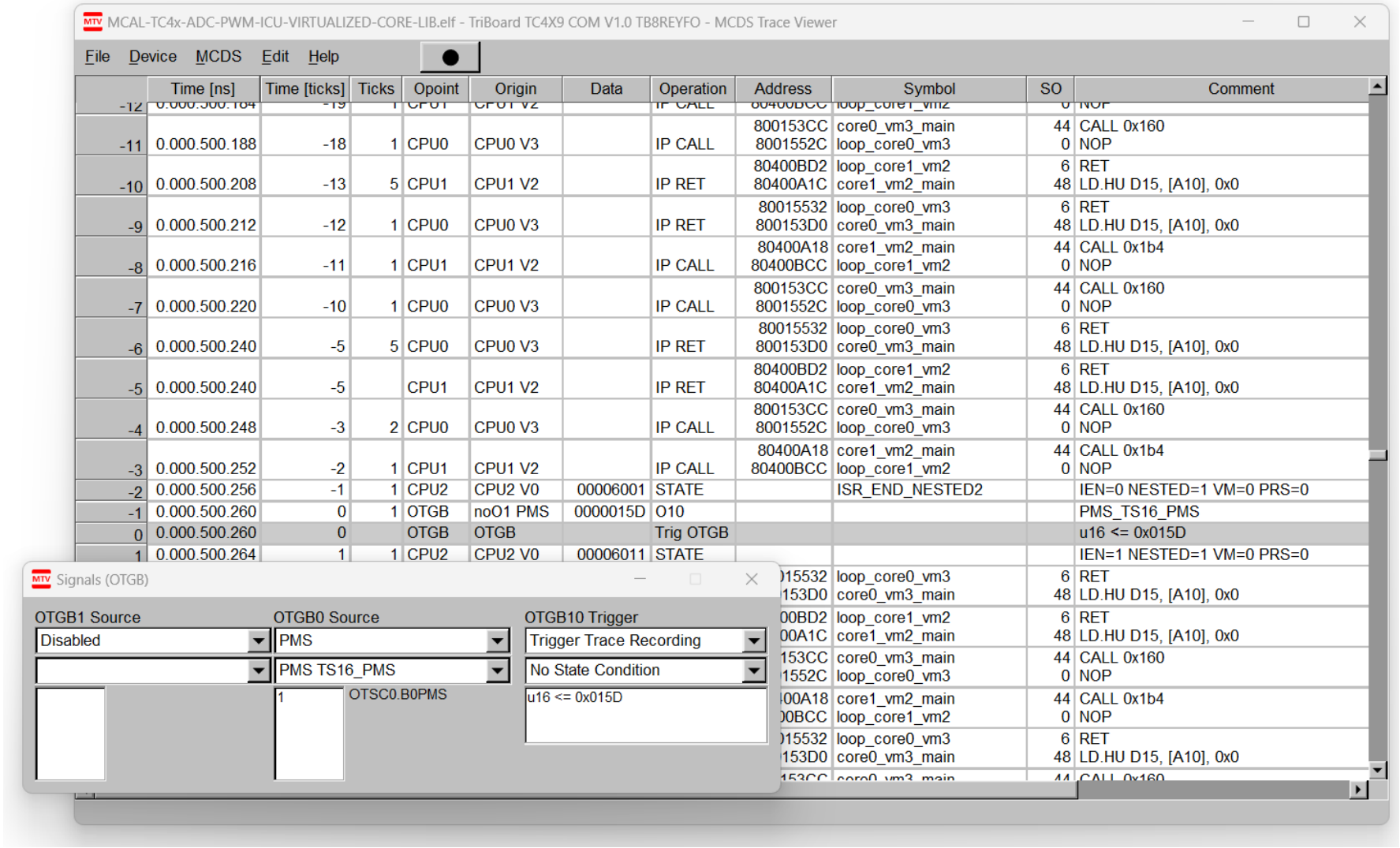
Demo example: Triggering on low supply voltage



AURIX™ OTGB trace source

- On-chip TriGger Bus
- 2x 16 bit logic analyzer
- Mux structure → any two
- Timer, ADC, PMS, etc.

Demo example: Triggering on low supply voltage



In-field debugging with on-chip trace and standard debugger

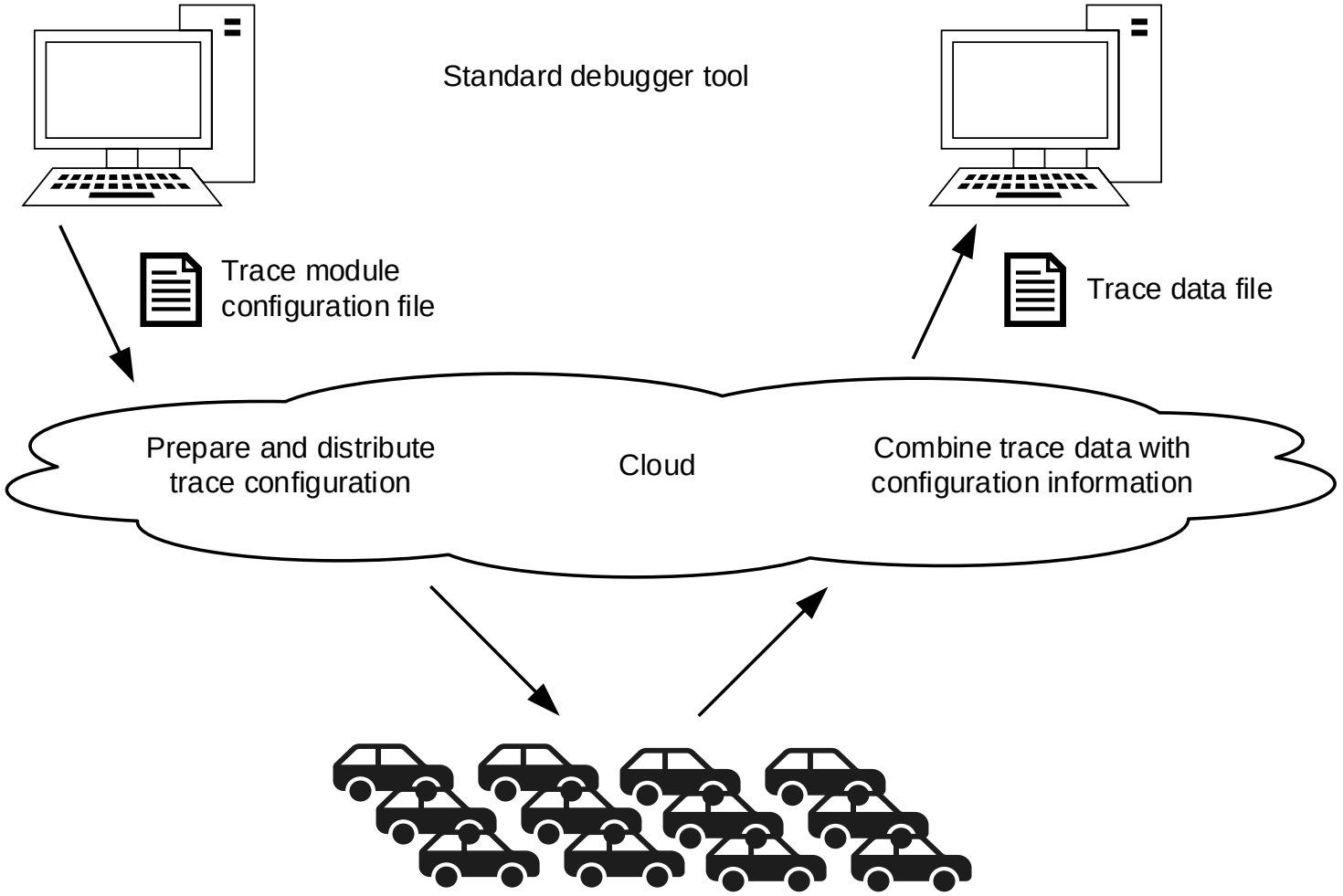


Table of contents

Summary

- In-field debugging is needed for very low failure rates
  … but can also “just” improves customer experience
- On-chip hardware trace is very flexible, safe and secured (for TC4x)
- Can reuse proven debug tooling
- New AI based debug tooling needed for in-field 100k units parallel “testing”