2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION JAPAN

Profiling and Optimization of Level 4 vECU Performance for faster ISO26262 Testing

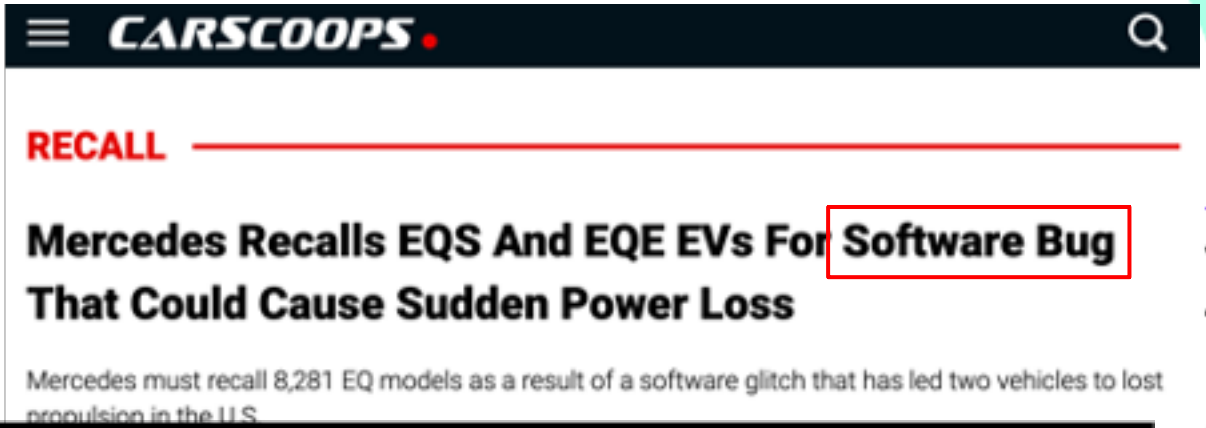
Lukas Jünger[1], Hitoshi Hamao[2], Megumi Yoshinaga[2] and Koichi Sato[2]

[1]MachineWare GmbH, Aachen, Germany

[2]Renesas Electronics Corporation, Tokyo, Japan

Agenda

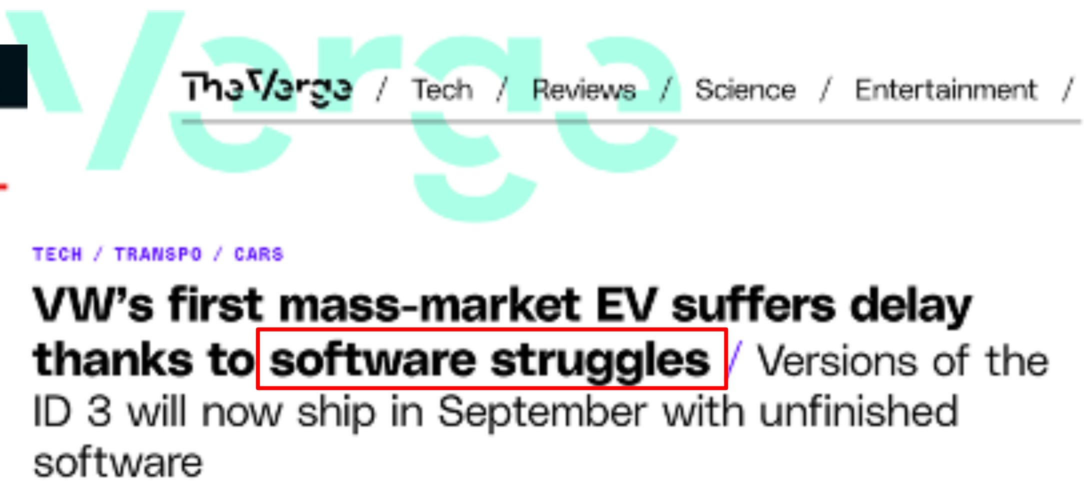
- Motivation
- Introduction of Virtual ECUs
- Level 4 Virtual ECUs and SystemC TLM-2.0
- MachineWare Level 4 Virtual ECU Architecture
- Performance Optimization with InSCight
- Case Study
- Summary



RECALL

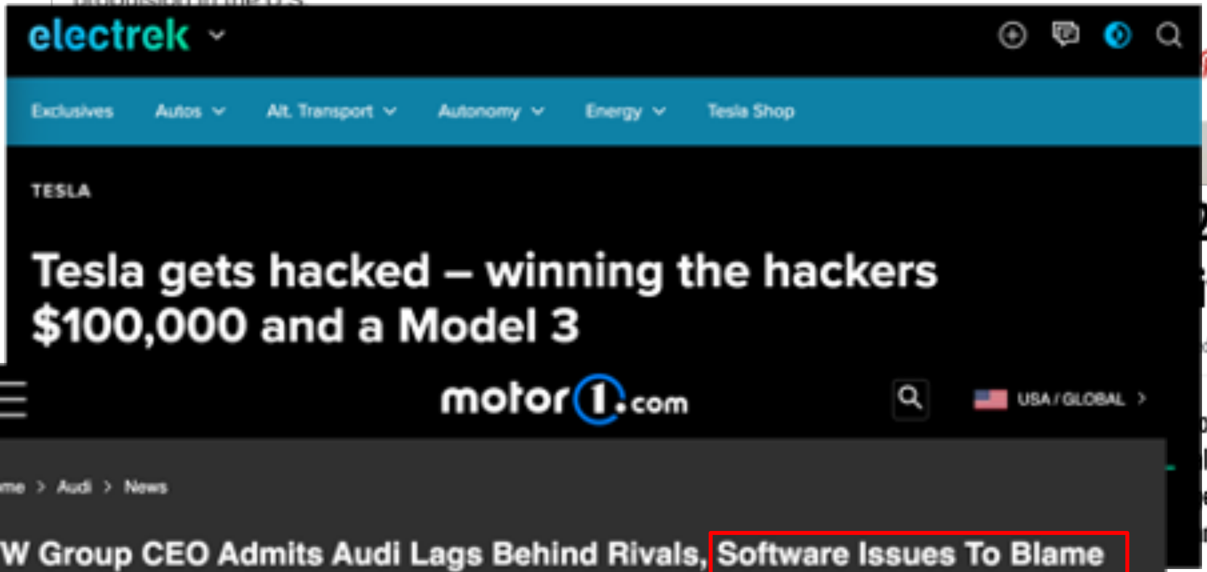
Mercedes Recalls EQS And EQE EVs For Software Bug That Could Cause Sudden Power Loss

Mercedes must recall 8,281 EQ models as a result of a software glitch that has led two vehicles to lost propulsion in the U.S.



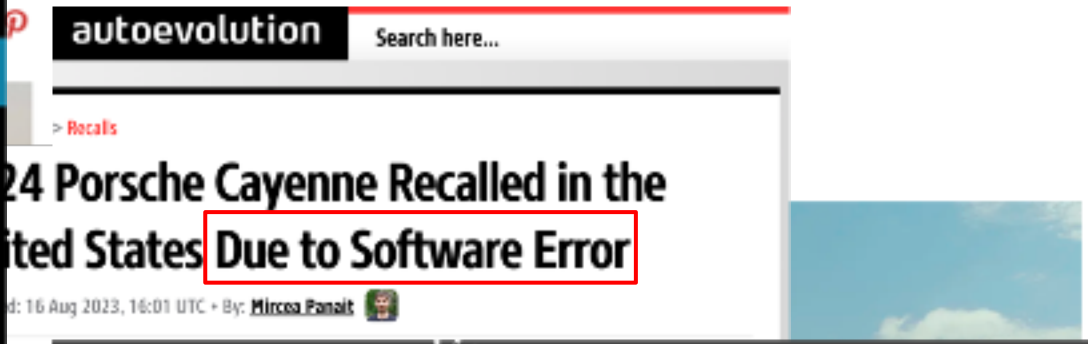
TECH / TRANSPO / CARS

VW's first mass-market EV suffers delay thanks to software struggles / Versions of the ID 3 will now ship in September with unfinished software



TESLA

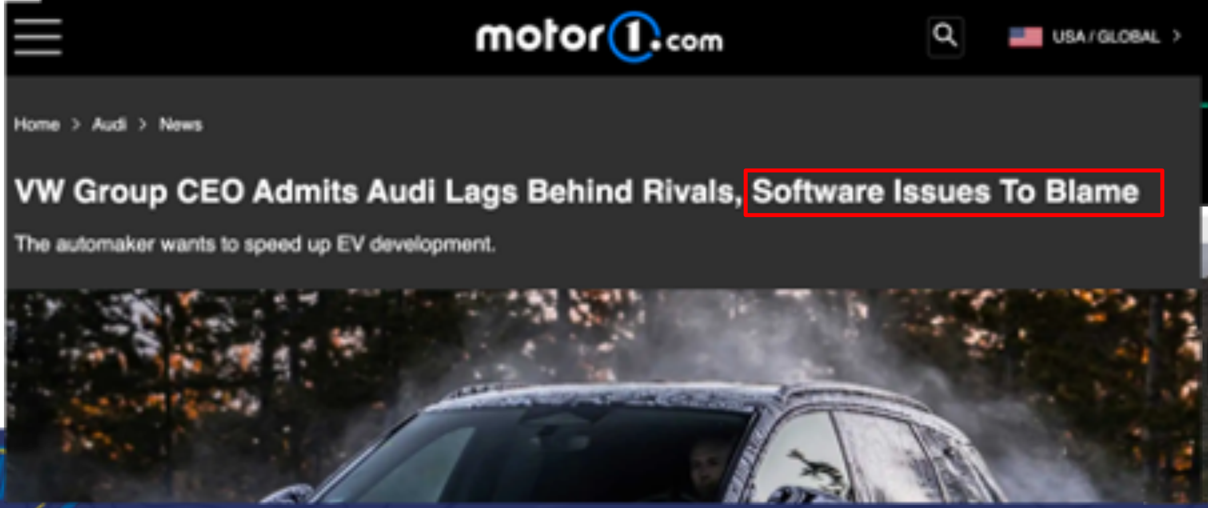
Tesla gets hacked – winning the hackers $100,000 and a Model 3



> Recalls

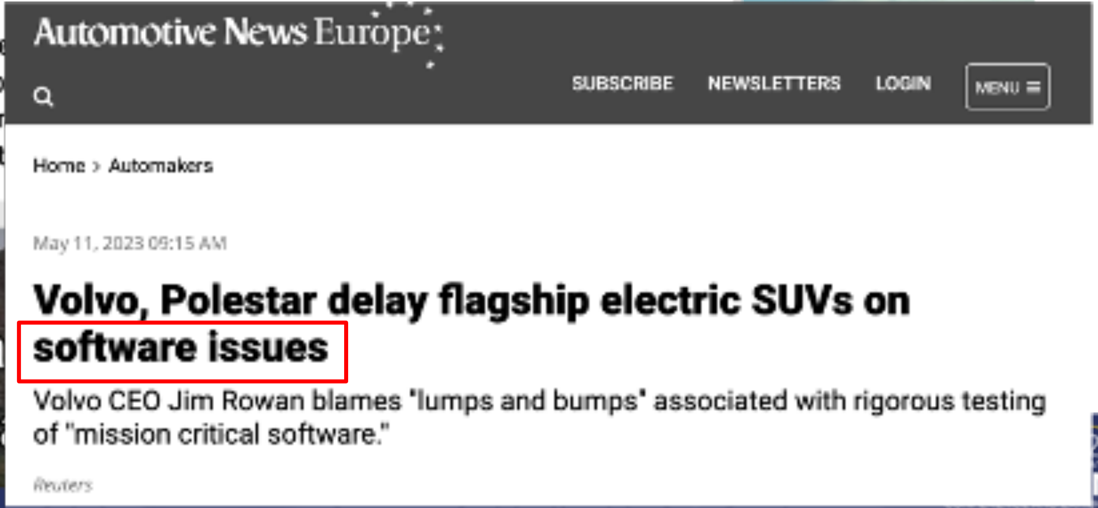
24 Porsche Cayenne Recalled in the United States Due to Software Error

d: 16 Aug 2023, 16:01 UTC • By: Mircea Panait



VW Group CEO Admits Audi Lags Behind Rivals, Software Issues To Blame

The automaker wants to speed up EV development.



May 11, 2023 09:15 AM

Volvo, Polestar delay flagship electric SUVs on software issues

Volvo CEO Jim Rowan blames "lumps and bumps" associated with rigorous testing of "mission critical software."

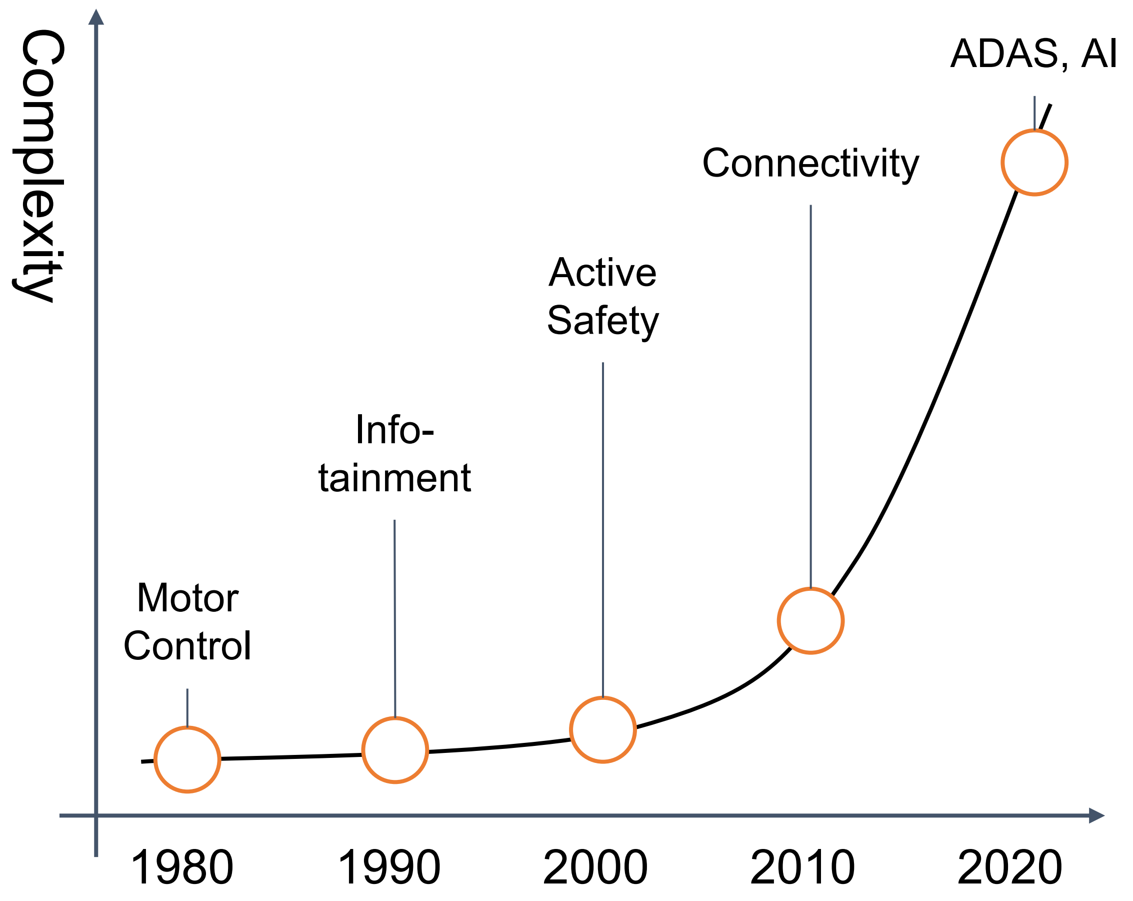
Reuters

Automotive Software Complexity

- Software-Defined Vehicle
    - Modern vehicle > 100. Mio LOC
    - SW becoming USP
- Bad software is expensive
- Managing complexity is key
- ISO26262/ASIL compliance

Problem:

- SW testing is hard to scale



ISO26262 Requirements

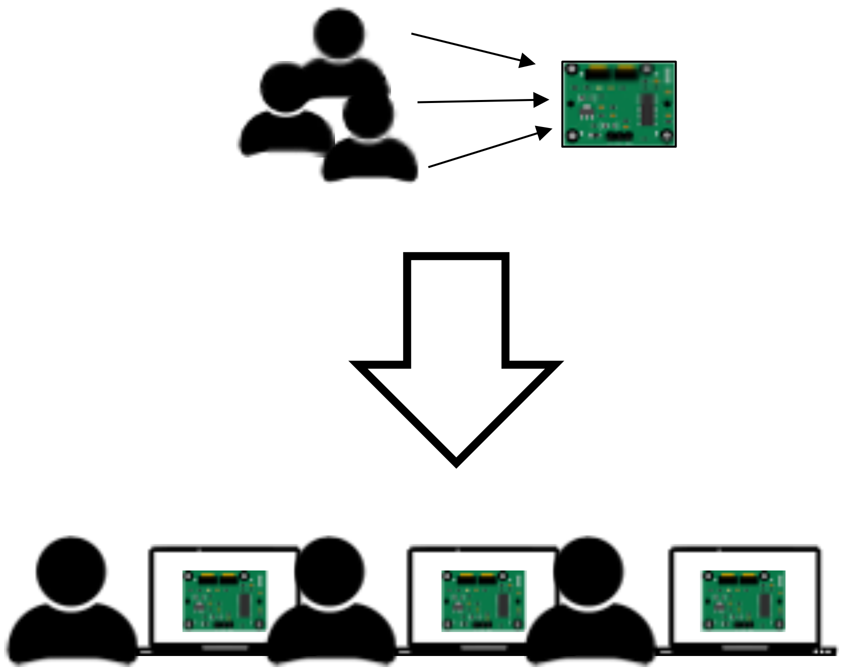
- Strong requirements towards hardware and software
  - Many recommended techniques for ASIL qualification

Examples:

- ISO26262-4-2018: Product development on System Level
  - Back-to-back tests: Comparison of hardware and simulation model
  - Fault injection tests
  - Test of interaction/communication, Test of internal/external interfaces
- ISO26262-6-2018: Product development on Software Level
  - Simulation of dynamic behavior of the design
  - Analysis of boundary values
  - Code Coverage Analysis
  - Fault injection test, interface test, back-to-back comparison, …

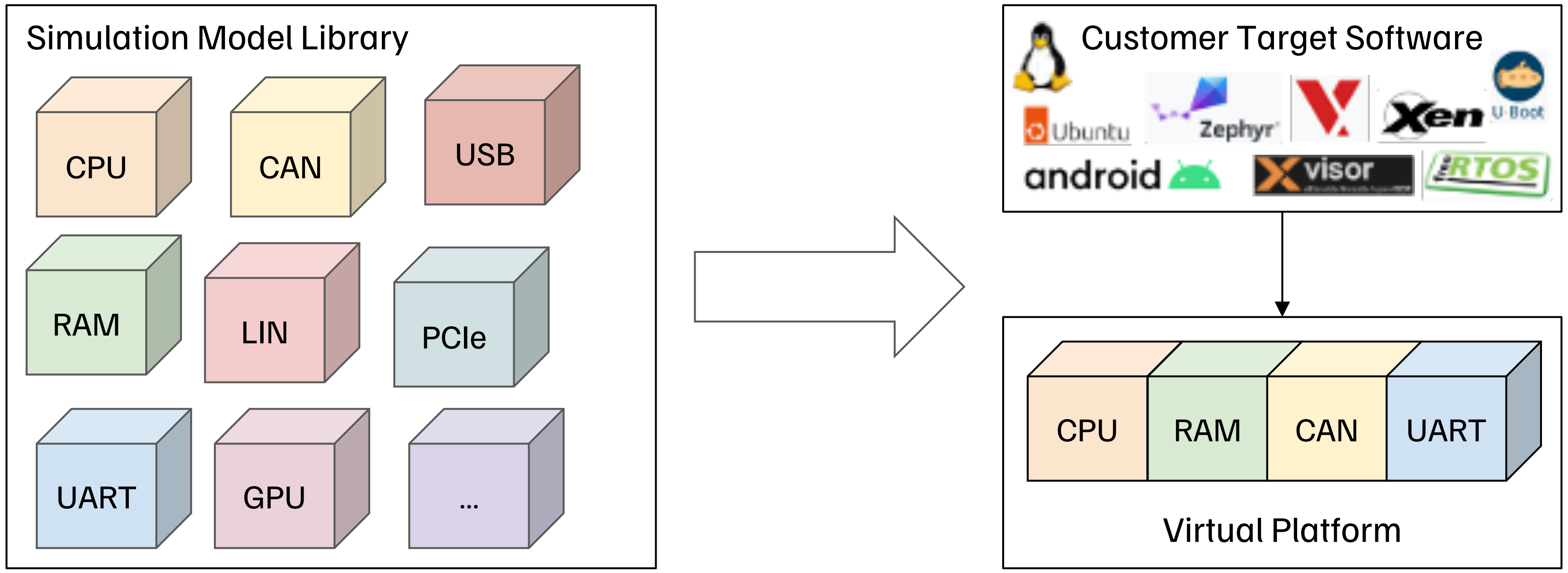
Virtual Prototyping

- Virtual Platform: Full System Simulation
- Indispensable in software development
  - Everising SW and HW complexity
- Advantages over physical prototypes
  - Available earlier (shift-left methodology)
  - Full flexibility, deep introspection
  - Non-intrusive debug
  - Scalable deployment

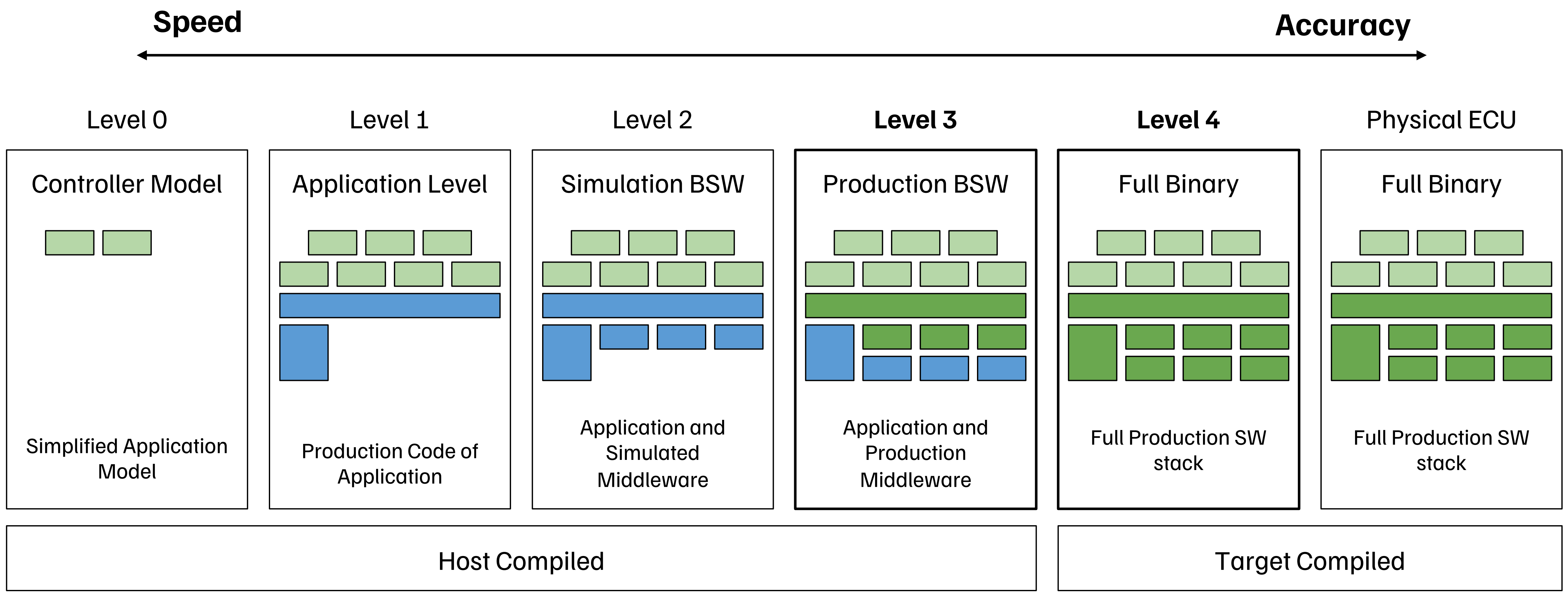


MachineWare Virtual Platform / vECU

- Virtual Platform: System simulator executing **unmodified software**
  - e.g. RISC-V, ARM, RH850, …
- Assemble Virtual Platform from “building blocks”



Virtual ECU Levels



L4 vECU Popular Use Cases

- Software development
  - Connect standard debuggers, IDEs
- Software test
  - Automate software test in CI, scaleable without hardware ECU
- Fault injection test
  - Inject faults via virtual bus, memory corruption, sensor value, …
- Code Coverage Analysis
  - Generate code coverage reports for every commit in CI
- Co-Simulation
  - Simulate several vECUs together connected
  - Test applications distributed over several ECUs

MachineWare Level 4 vECU Architecture

- Based on SystemC TLM-2.0 standard
  - Seamlessly integrate virtual HW models
  - Support QEMU models (QBox)
  - Create models in MW VCML
- Like physical hardware
  - Use debuggers, development tools, …
- Co-Simulate through common interfaces
  - e.g. FMI, SIL Kit, MW VSP
- Execute on-premise or in the cloud
- Flexible license model
  - Open-source and proprietary

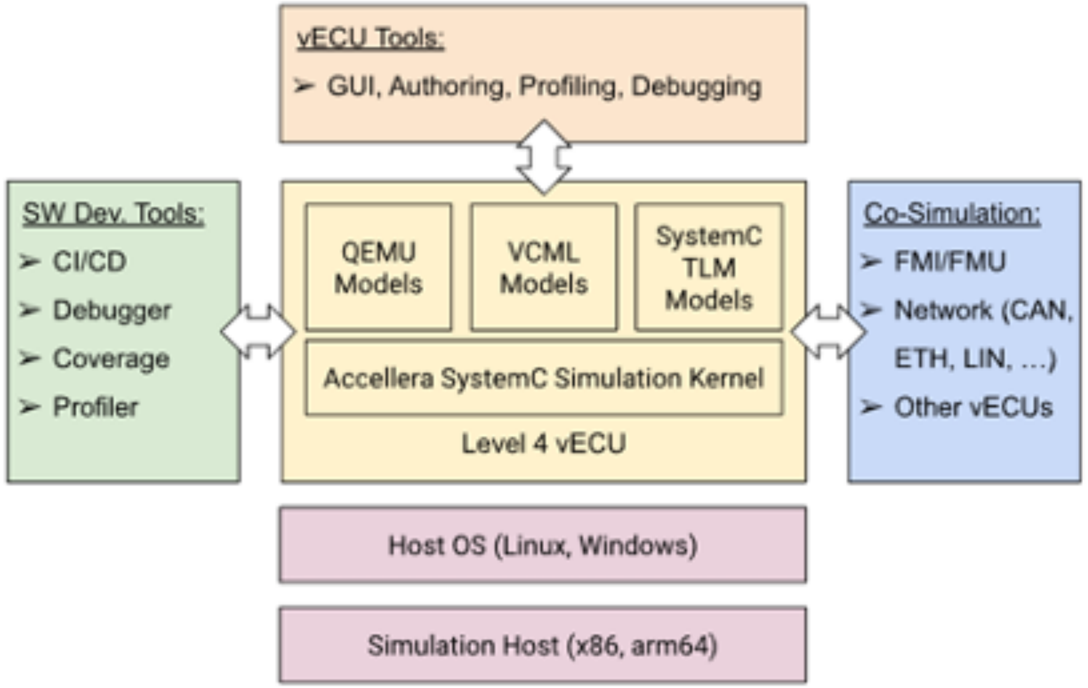


Figure 1: L4 vECU Overview.

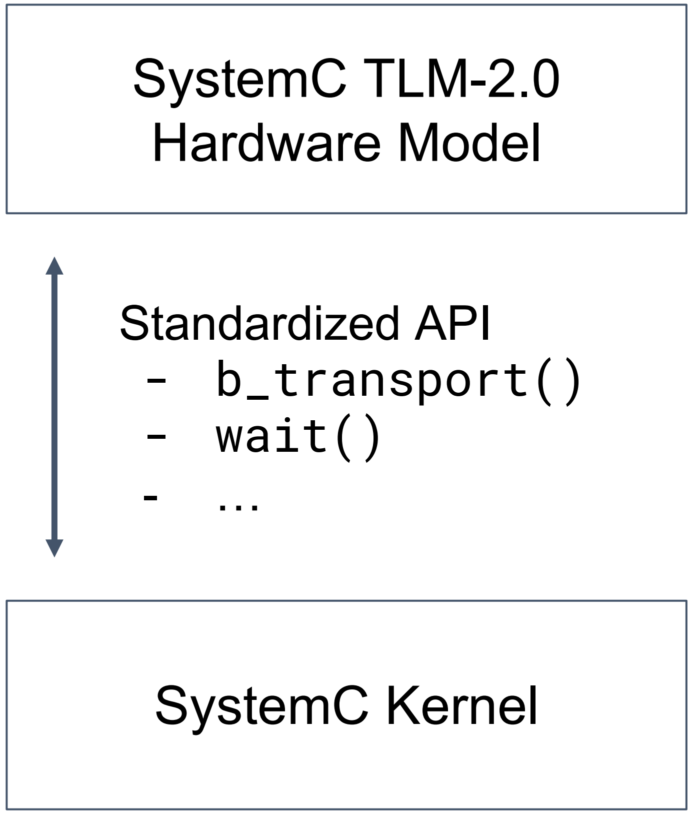
vECU Performance Problems

- Faster vECU means increased productivity less cost
  - Faster turnaround
  - Reduced compute cost
  - Reduced energy consumption
- Slow vECU can prevent successful deployment
  - Test runtimes prohibitively long
  - Bad developer experience, less adoption

- Problem: How to find simulation performance bottlenecks?
  - vECUs are extremely complex (100-1000 component)
  - vECU combines models of many teams, limited expertise during integration
  - Combination of target SW and simulated HW effects

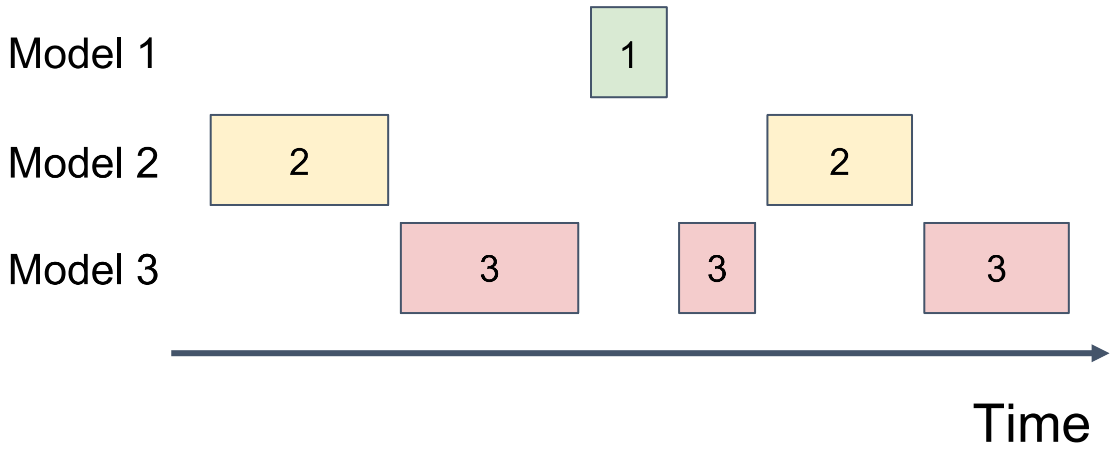
SystemC Compatibility

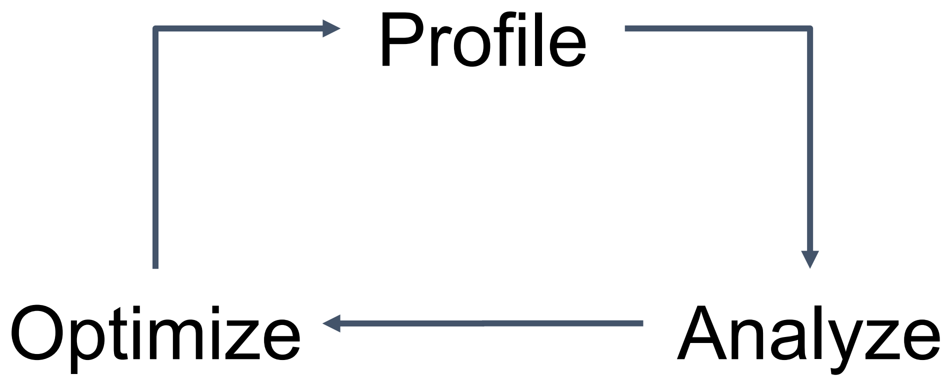
- Simulation is a compound of models
  - Models represent hardware blocks
  - SC_THREADS/METHODS for modeling hardware behavior
- Models communicate with kernel
  - wait() to yield time
  - b_transport() to access blocking transport interface
  - …
- Standardized API enables model interoperability
  - Binary model can link with binary kernel and communicate



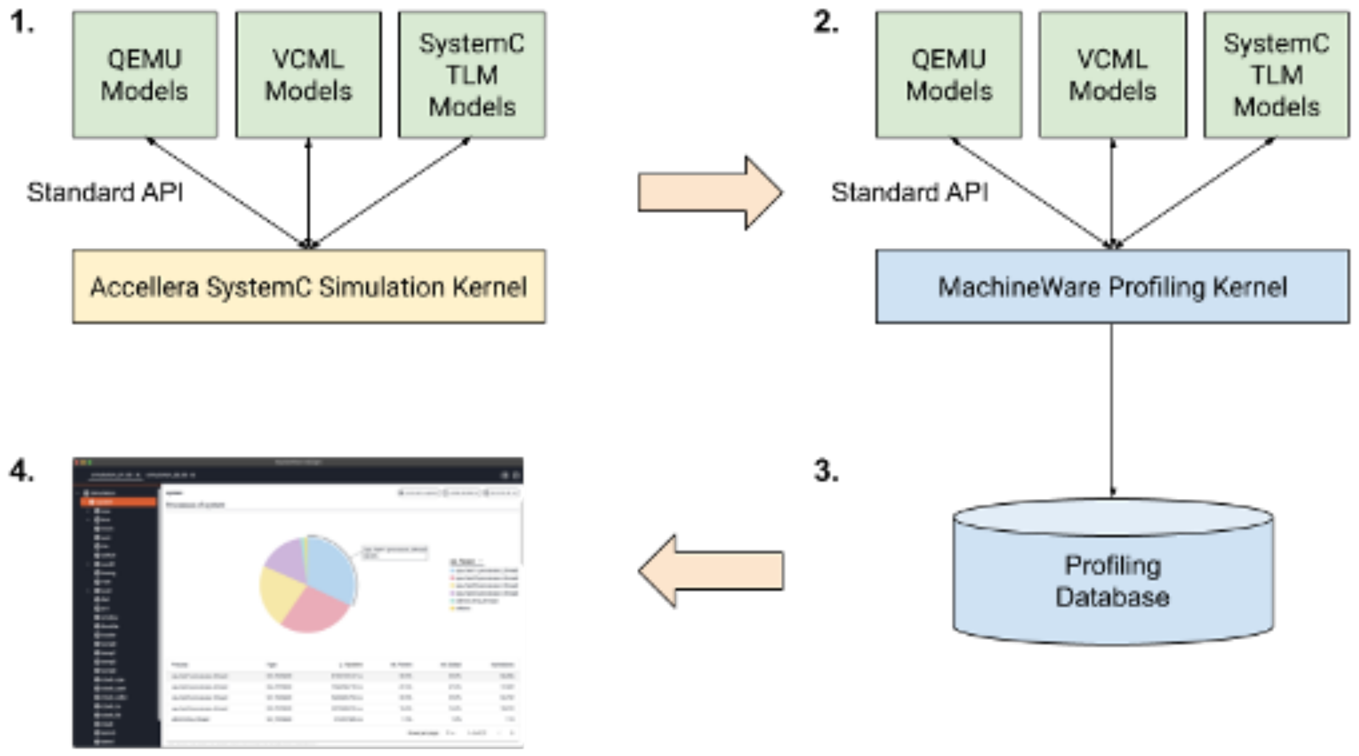
InSCight Architecture

- Goal: Identify slow models
  - Determine model compute overhead
- Solution: SystemC profiler
  - Event notifications
  - SC_THREAD/METHOD compute time
  - Kernel-internal state tracking
- Requirements
  - Minimum overhead
  - No change to the model code





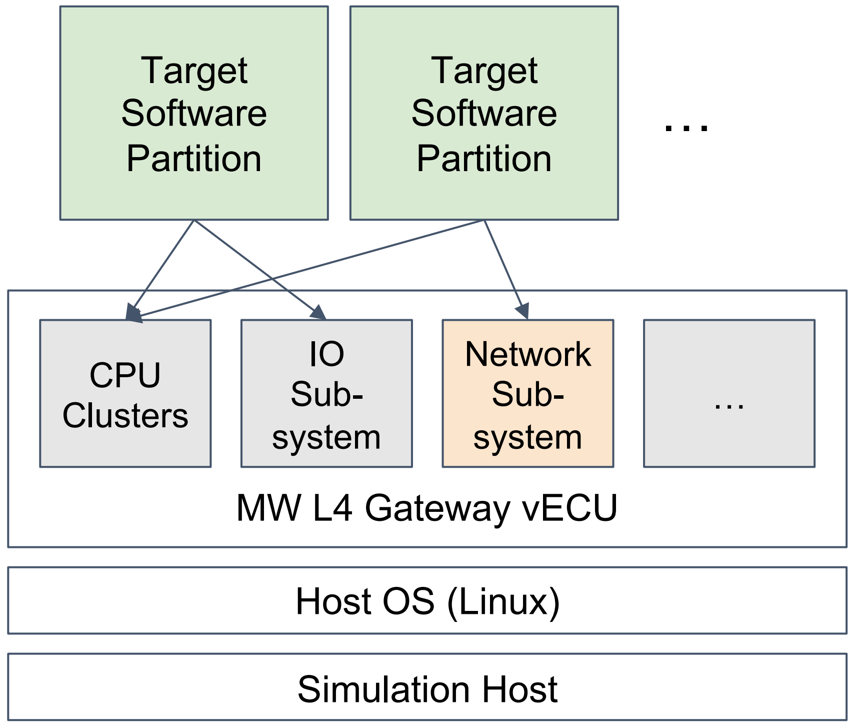
InSCight Flow



$$\begin{tikzcd}
& \text{Profile} \arrow[dr] & \\
\text{Optimize} \arrow[ur] & & \text{Analyze} \arrow[ll]
\end{tikzcd}$$

Case Study: Gateway L4 vECU

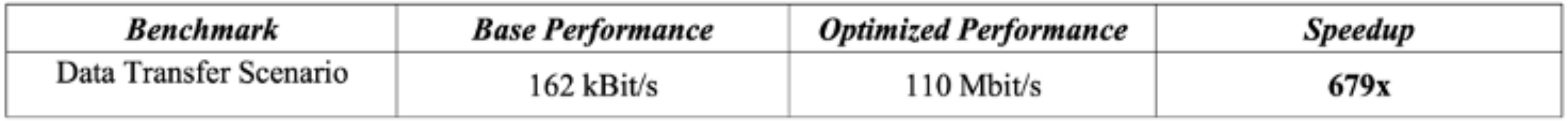
- Renesas ECU target: Gateway
  - Multi-processor architecture
    - Several compute domains
  - Runs complex software stack
    - Several OS/RTOS
    - Cross domain communication
  - Specialized hardware for networking function
- vECU built using MachineWare technologies
  - Executes Renesas SW stack
  - Near real-time performance



Case Study: Results

- Scenario: Data transfer using specialized HW
  - Target SW benchmark exercises driver code
  - Virtual HW models utilized for transfer function
- Goal: Identify performance bottlenecks leading to insufficient performance
- Technique: InSCight profile generation, analysis, model code optimization

Table I. Data Processing Performance measurements.



Benchmark	*Base Performance*	*Optimized Performance*	*Speedup*
Data Transfer Scenario	162 kBit/s	110 Mbit/s	**679x**

Case Study: Results

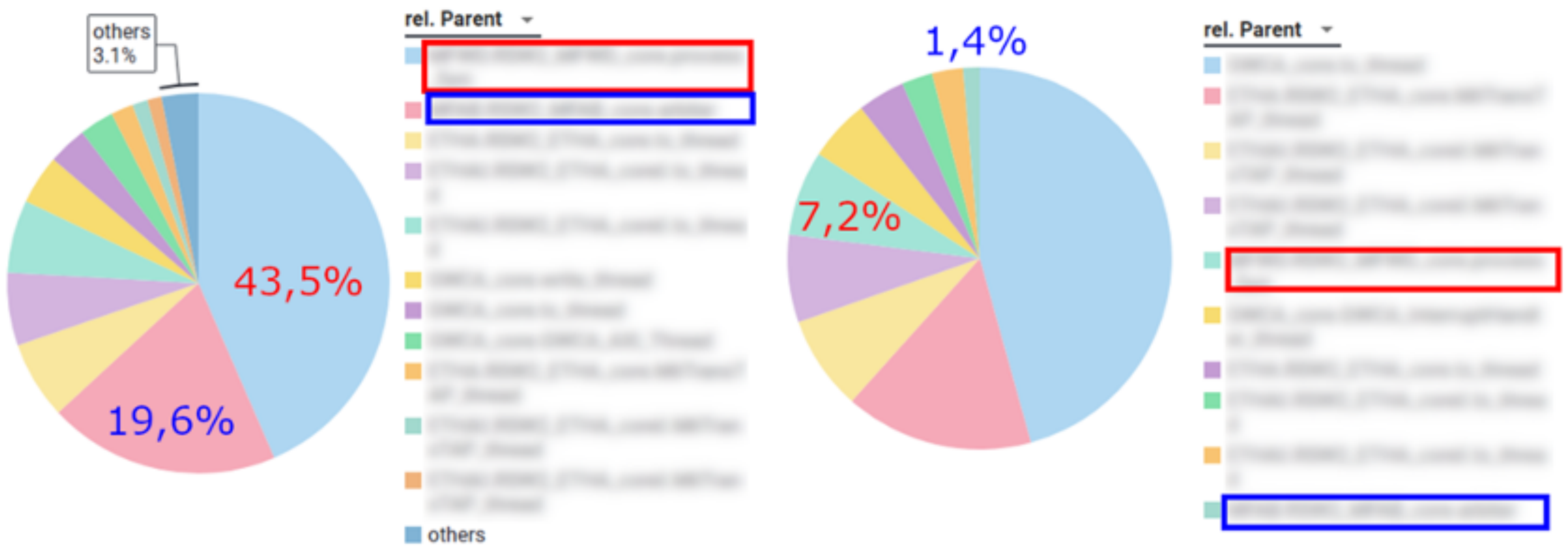


Figure 3: Profiling result comparison. Left side before, right side after optimization.

Summary

- Level 4 vECU
  - Run unmodified target software
  - “Like real hardware”
  - Accelerate ISO26262 testing
- SystemC TLM-2.0 L4 vECU
  - Standardized interfaces
  - Reuse of existing HW models
  - Enables profiling
- SystemC Profiler: InSCight
  - Tool to help handle platform complexity
  - Can unveil significant speed ups
  - Compatible with any SystemC TLM-2.0 simulator

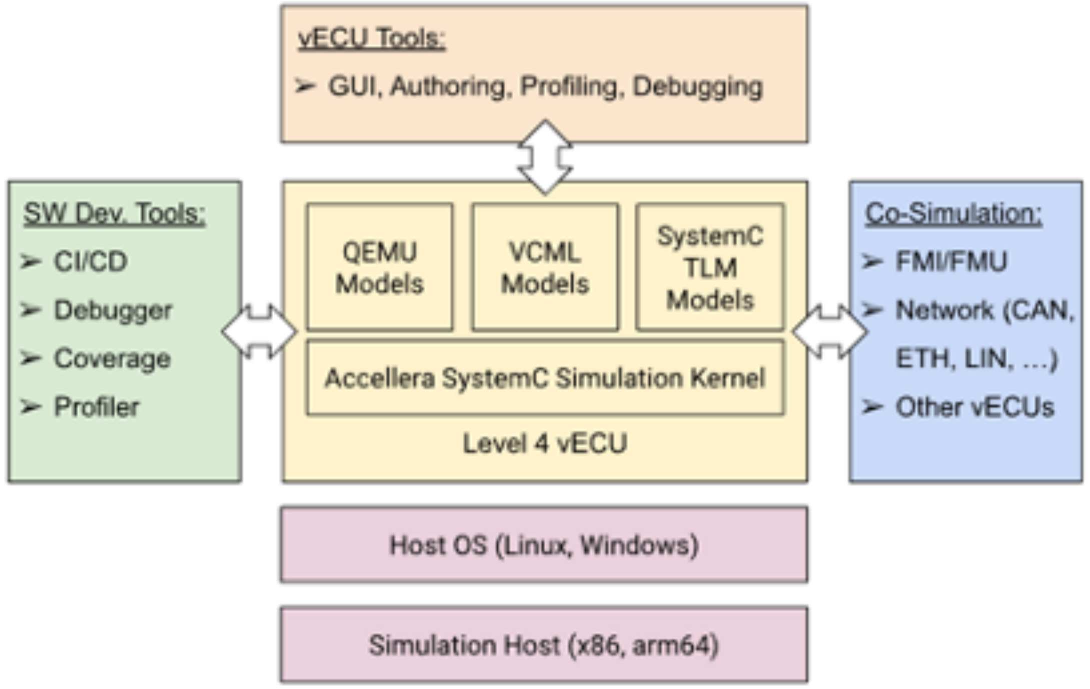


Figure 1: L4 vECU Overview.