2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
OCTOBER 14-15, 2025

Functional Twin: A Framework for Reusability of Virtual Realtime Systems

Sacha Loitz, Torsten Hermann, Martin Hruschka

AUMOVIO

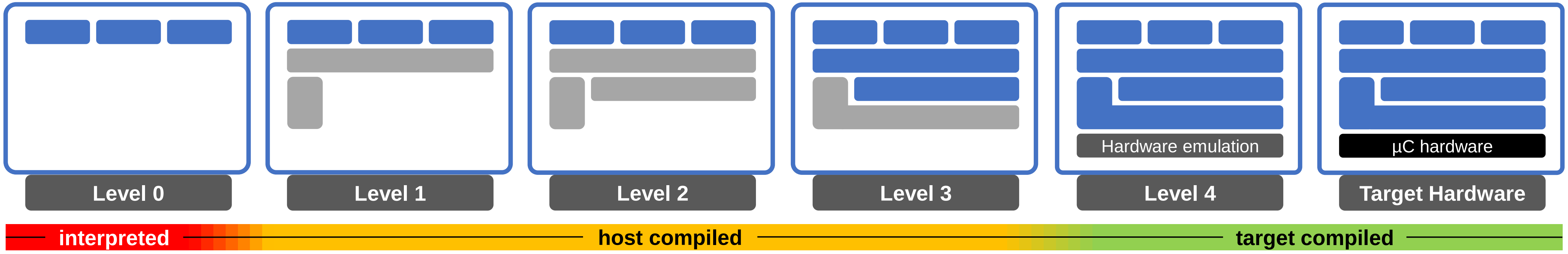
accellera SYSTEMS INITIATIVE

Functional Twin:
A Framework for Reusability of Virtual Realtime Systems

Agenda

- Introduction
- Comparison between L3 and L4
- Peripheral Wiring Adapter
- Proof of Concept
- Conclusion and Outlook

Introduction
Virtualization Levels



Level 0

Production code
- None

Simulated code
- None

- SWC are tested on model level only
- Focus on function development

Level 1

Production code
- Application

Simulated code
- RTE + some BSW services
- Environment for SWC / application-level development

Level 2

Production code
- Application

Simulated code
- RTE + BSW services
- Environment for SWC / application-level development

Level 3

Production code
- Application
- RTE + BSW services

Simulated code
- MCAL drivers

- No µC Hardware dependency
- Abstracted timing behavior

Level 4

Production code
- Full target binary (production SW)

Simulated code
- None

- Hardware is emulated
- (close to) real-time behavior

Target Hardware

Production code
- Full target binary (production SW)

Simulated code
- None

- Hardware is used

- full real-time behavior

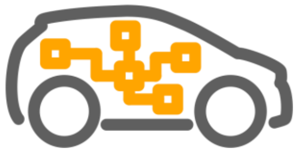
| Source | **ProSTEP iViP**, White Paper "Smart Systems Engineering; Requirements for the Standardization of Virtual Electronic Control Units (V-ECUs)" https://www.ps-ent-2023.de/fileadmin/prod-download/WhitePaper_V-ECU_2020_05_04-EN.pdf |

Introduction

Classic AUTOSAR within a virtual automotive setup

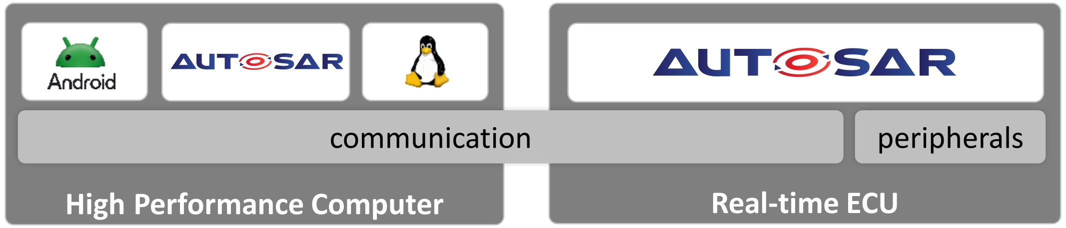
Virtual Car

All ECUs of a car are simulated to run **system and software functions** with their interactions including cloud connectivity



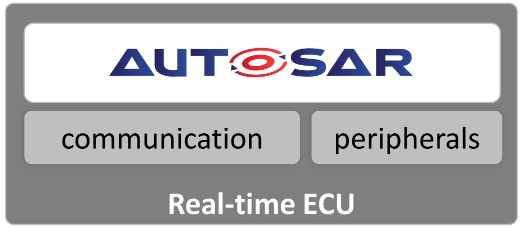
Virtual ECU-System

ECU-System simulation of **several virtual ECUs (e.g. HPC & Zone Control Units)** with distributed software functions and their interactions



Virtual ECU

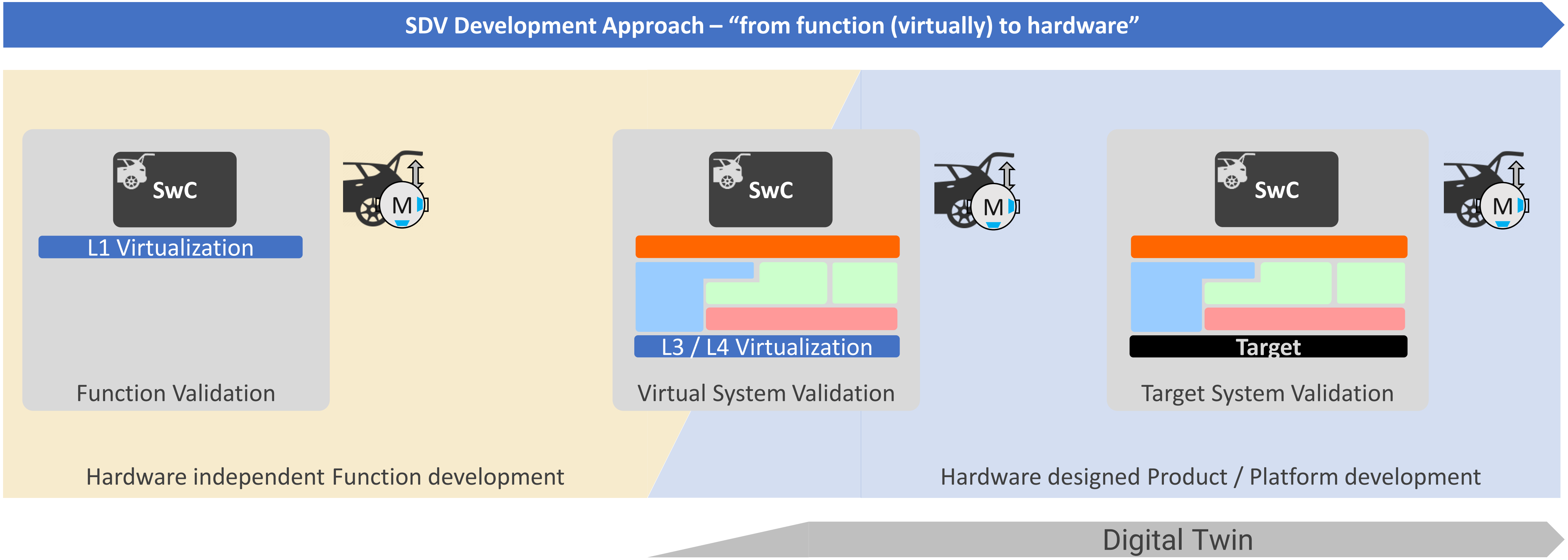
ECU simulation of **Classic AUTOSAR (e.g. Zone Control Units)** including application and middleware software with peripherals in real-time



*ECU = Electronic Control Unit, HPC = High-Performance Computer

Introduction

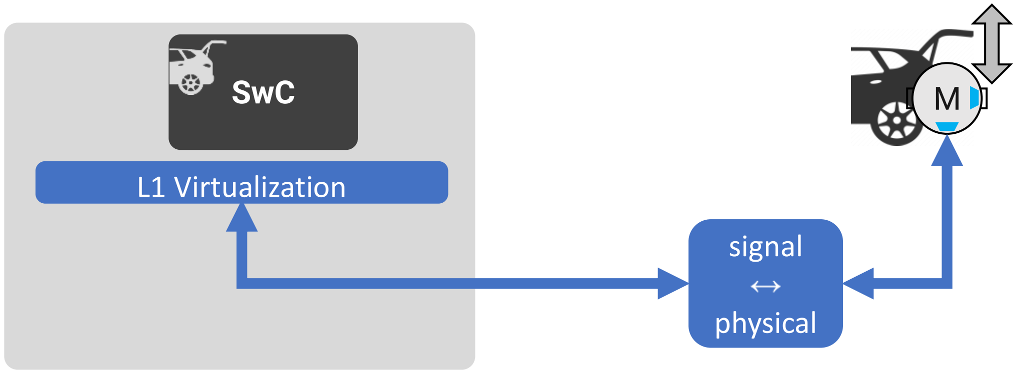
SDV Application implementation for I/O based ECUs



Simulation shall provide a functional representation of the HW for embedded SW integration testing

Introduction

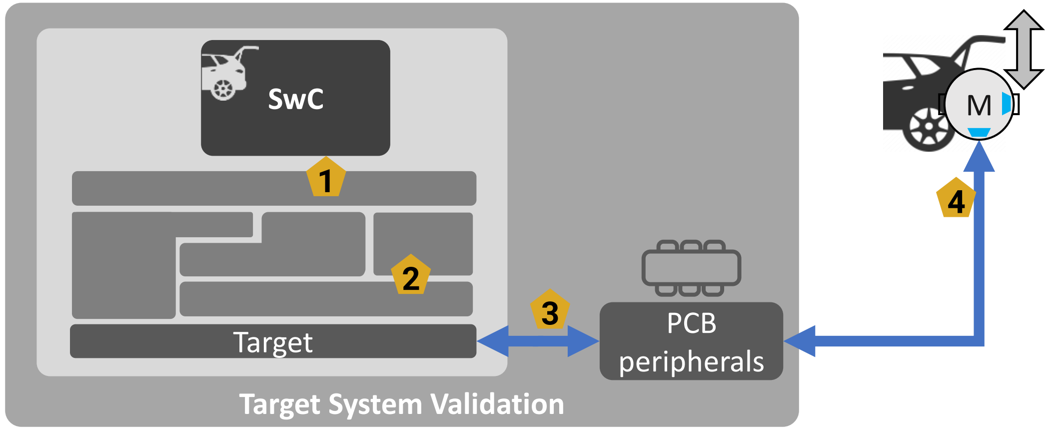
Need of virtual peripherals – Function Validation



Open with 50% speed	→	50% from 12V	→	Run motor with “6V“
250d (== 250mA)	←	0 – 15000 mA	←	Motor consumes 0,25A
500d (== 500mA)	←	0 – 15000 mA	←	Motor consumes 0,5A
1d (at upper end)	←	Signal from sensor	←	opened

Introduction

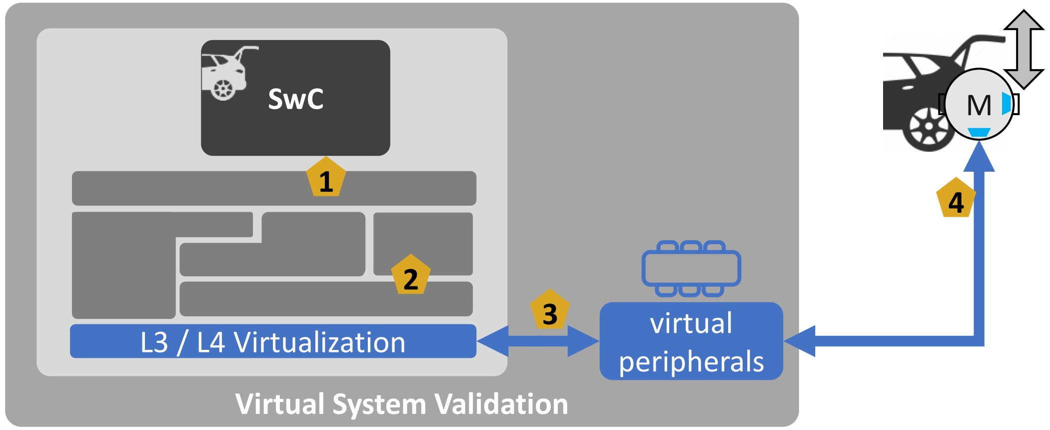
Need of virtual peripherals – Target System Validation



1 µController with BSW		2		3 peripheral		4 plant model
Open with 50% speed	→	PWM duty cycle to 50	→	PWM toggling	→	Run motor with “6V“
250d (== 250mA)	←	128d (== 2,5V)	←	Current sensing 2,5V	←	Motor consumes 0,25A
500d (== 500mA)	←	256d (== 5V)	←	Current sensing 5V	←	Motor consumes 0,5A
1d (at upper end)	←	1d (opened)	←	Dig In = 0 from sensor with pullup	←	opened

Introduction

Need of virtual peripherals – Virtual System Validation



1 µController with BSW	2	3 peripheral	4 plant model
Open with 50% speed	PWM duty cycle to 50	PWM toggling	Run motor with "6V"
250d (== 250mA)	128d (== 2,5V)	Current sensing 2,5V	Motor consumes 0,25A
500d (== 500mA)	256d (== 5V)	Current sensing 5V	Motor consumes 0,5A
1d (at upper end)	1d (opened)	Dig In = 0 from sensor with pullup	opened

With the implementation of virtual peripherals, a real "Digital Twin" can be provided

Comparison between L3 and L4
Characteristics

Aspect	Level 3 (L3)	Level 4 (L4)
Simulation fidelity	High fidelity	Very high fidelity
Software integration	Replacement of MCU hardware drivers	Binary compatible
Timing behavior	Timing simulation of RTOS	Precise timing & scheduling
Use cases	Early integration testing	Final validation & safety-critical testing
HW dependency	Excludes MCU hardware drivers	Simulates hardware interfaces
Complexity	Moderate to high	Very high
Toolchain requirements	RTOS and simulation environment	Full simulation stack
Validation scope	Functional validation	Full system validation

Comparison between L3 and L4

Challenges for Reuse of PCB peripherals

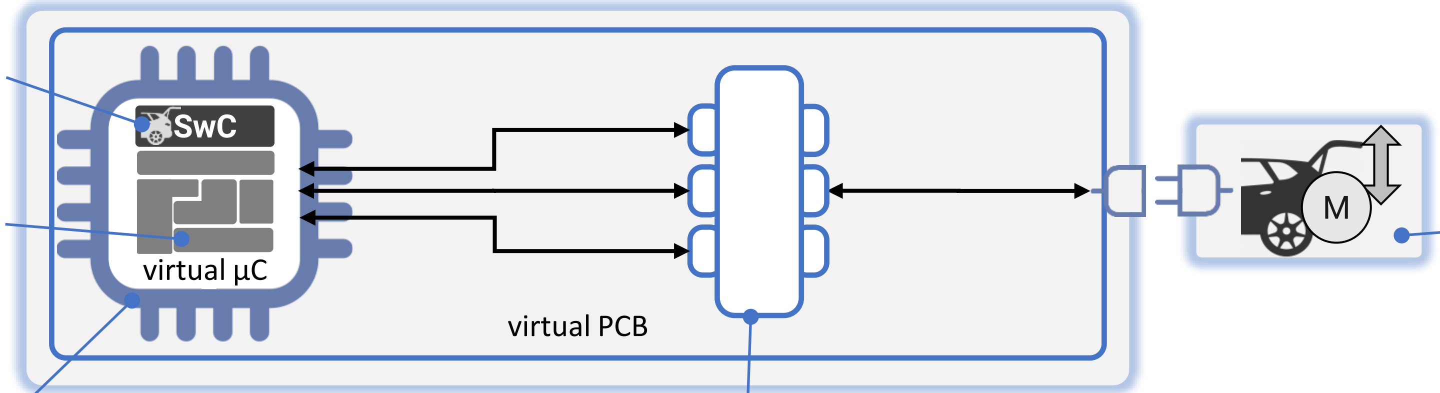
Aspect	Challenges
Tool incompatibilities	Different simulation tools and environments available → Direct reuse is technically difficult or almost impossible
Standardization	No standard for model interfaces or timing semantics → inconsistencies in how models are structured and executed
Modeling objectives	L3 – speed and functional correctness L4 – timing accuracy and hardware interaction → Significant differences in architecture and assumptions of the models
Performance tradeoffs	L4 often use TLM & loosely timed – brings overhead in L3 → Using L4 models in an L3 context would slow down simulation → Using L3 models in an L4 context would lack the required fidelity

Comparison between L3 and L4

Applying towards the use case

SwC under test
- L3 and L4 – equivalent

AUTOSAR and MCAL drivers
- L3 – abstracted
- L4 – target binary



Test tool
- plant models / test cases should be (almost) common for L3 and L4 with respect to efforts and acceptance

Microcontroller internal peripherals
- L3 – abstracted together with the drivers
- L4 –
  - register and register behavior accurately modelled
  - Interface to core and externals by TLM
  - Timing annotated but not accurate

PCB peripherals
- L3 and L4 – no accurate model required
- L3 and L4 – react on incoming transactions as expected by SwC
- **Timing** information is **subordinate**
  - peripheral is on lower speed than µC
  - exact observability **hidden by communication protocol** e.g. SPI

For embedded SW integration testing common peripheral modes are applicable

Comparison between L3 and L4

Requirements on the common interface for peripherals

- Virtual peripheral API
  - All peripheral devices communicate via pins
  - A pin has a unique name, a concrete signal type and its direction
  - The API shall provide functions for data exchange and to trigger the data processing
- Wiring
  - Use identic pins types for peripheral devices and microcontroller
  - Support all representation options of a wire, like logical, PWM, SPI, …
  - Connect compatible signal types in a directed, acyclic graph
  - Provide signal-operations as a part of the configuration, e.g. IN = AND(OUT1, NOT(OUT2))

Peripheral Wiring Adapter

Wiring and interfacing concept

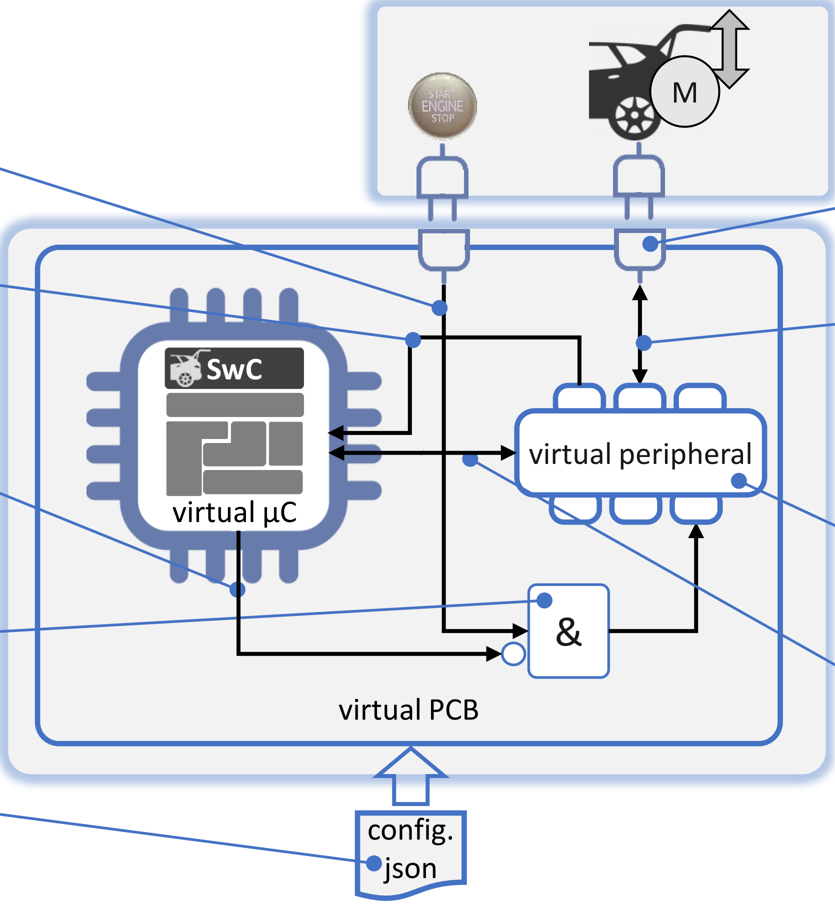
Volt / Ampere
represented as floating point values

Digital Inputs and Outputs
logical signals – LOW, HIGH and HIGH-Z

Pulse-Width-Modulation (PWM)
represented as duty-cycle

Logic gates
implements signal-operations which have a functional impact on embedded SW

JSON file configuration
All virtual PCB features and wiring connections is specified in a JSON file



FMU / FMI & SIL Kit
Interfacing to test tools uses (quasi) standardized interfaces

Voltage Divider / - Multiplier
implements HW circuit with functional impact on software under test

Virtual peripherals
Encapsulated in common API

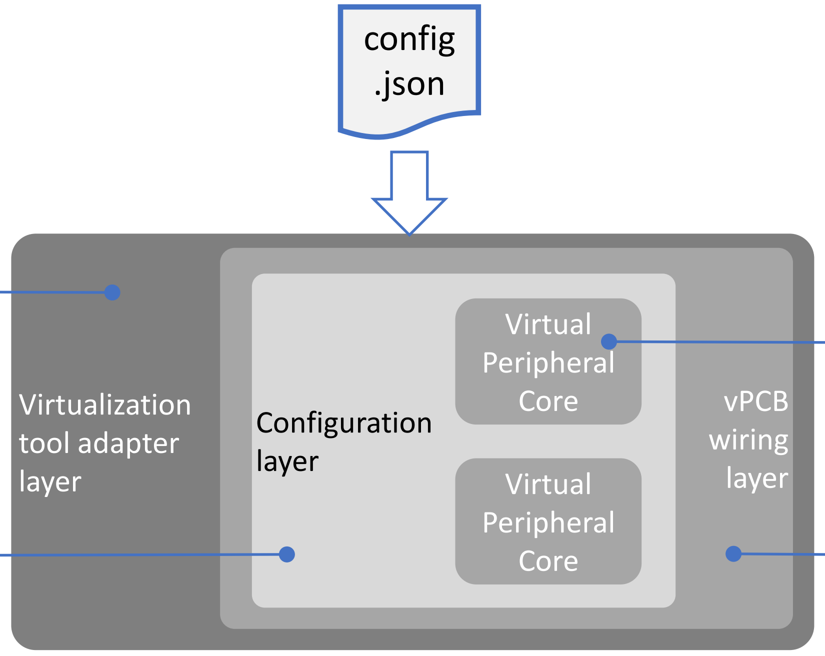
Serial Peripheral Interface (SPI)
Represented as message-based communication; supports Daisy Chain setups

Peripheral Wiring Adapter

Layered architecture and features

- Connects adapter with tool APIs
- Supports tool related programming languages
- Generic layer implemented once per tool and configured on use

- Virtual peripheral instantiation and configuration with default values

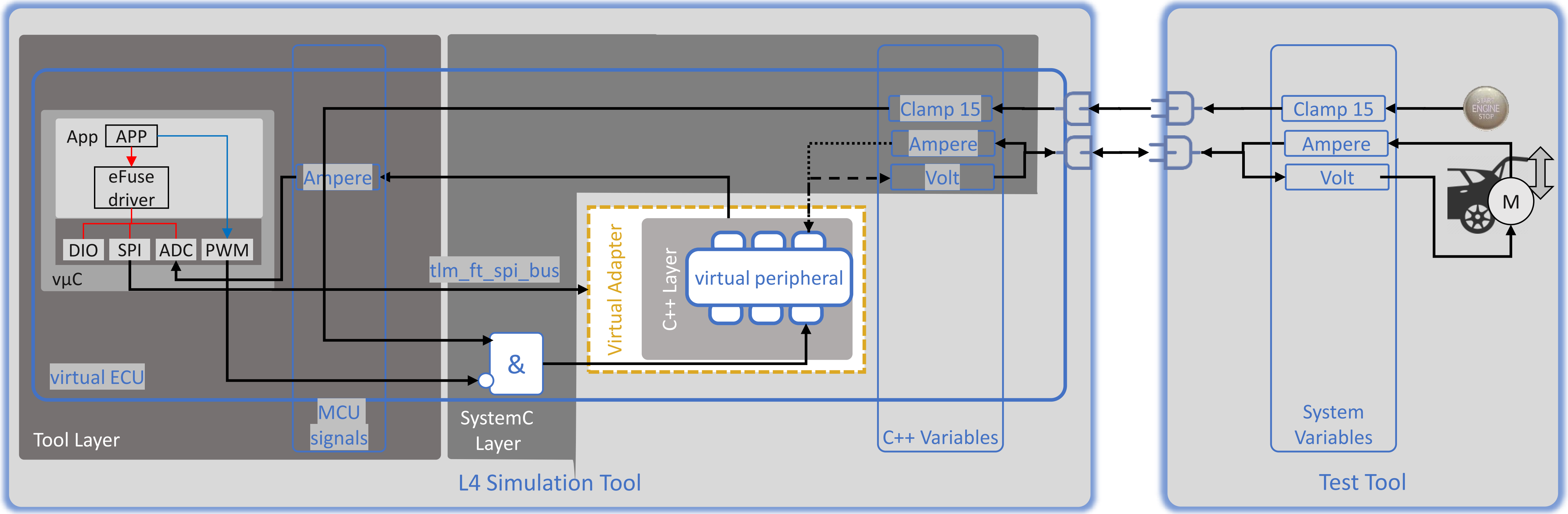


- One core implementation per virtual peripheral type, e.g. eFuse
- Implemented with unique API in C++

- Supports small HW circuits with functional impact to embedded SW, like AND, OR, Daisy Chain, …
- Configured at execution from file

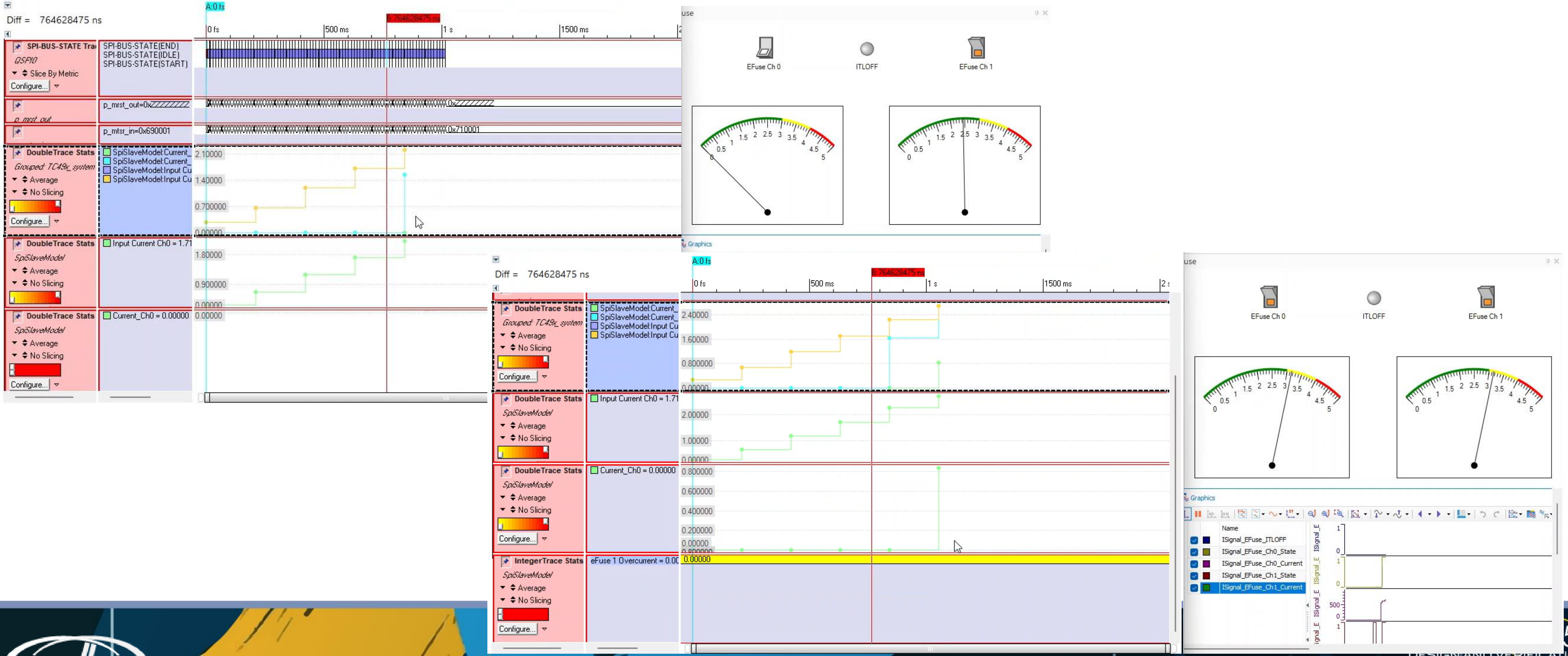
This adapter design enables the integration of one peripheral core implementation into all virtualization tools

Proof of Concept
Setup Level 4 simulation environment



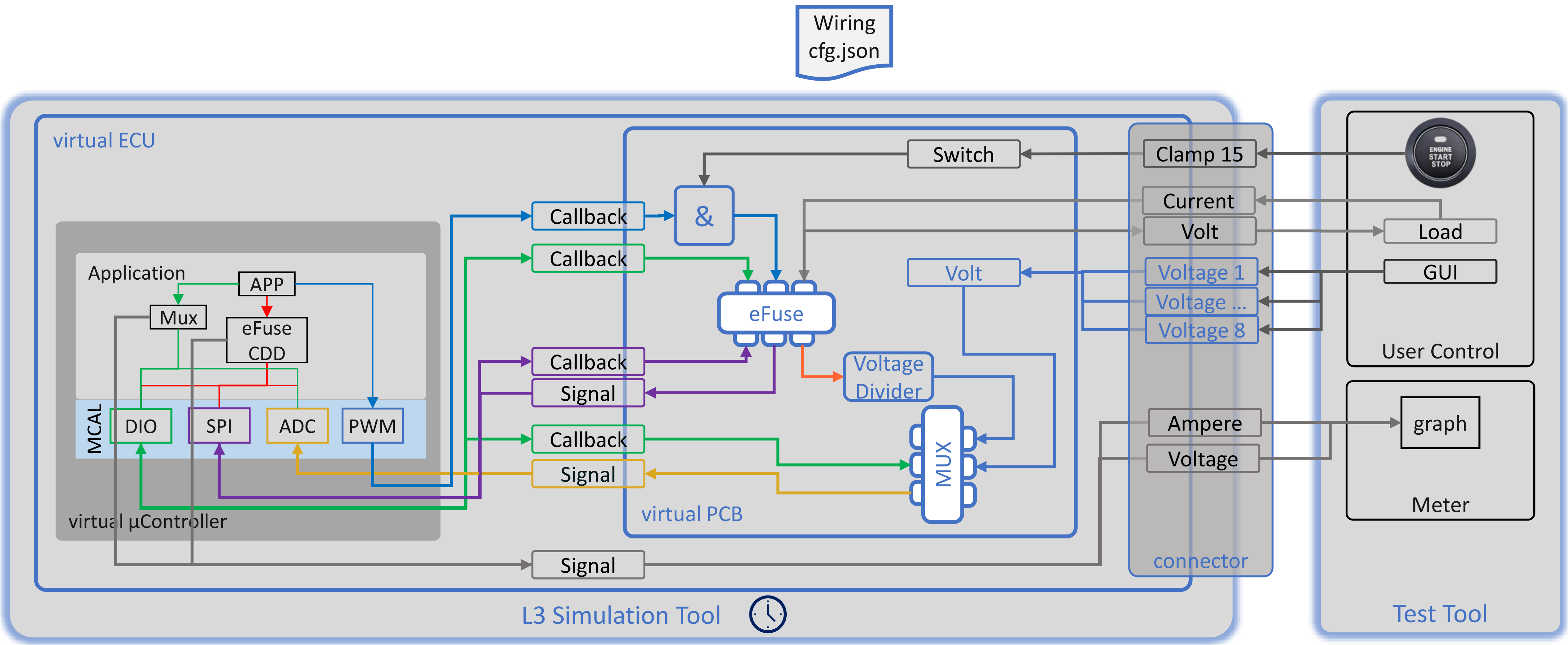
Proof of Concept

Setup Level 4 simulation environment



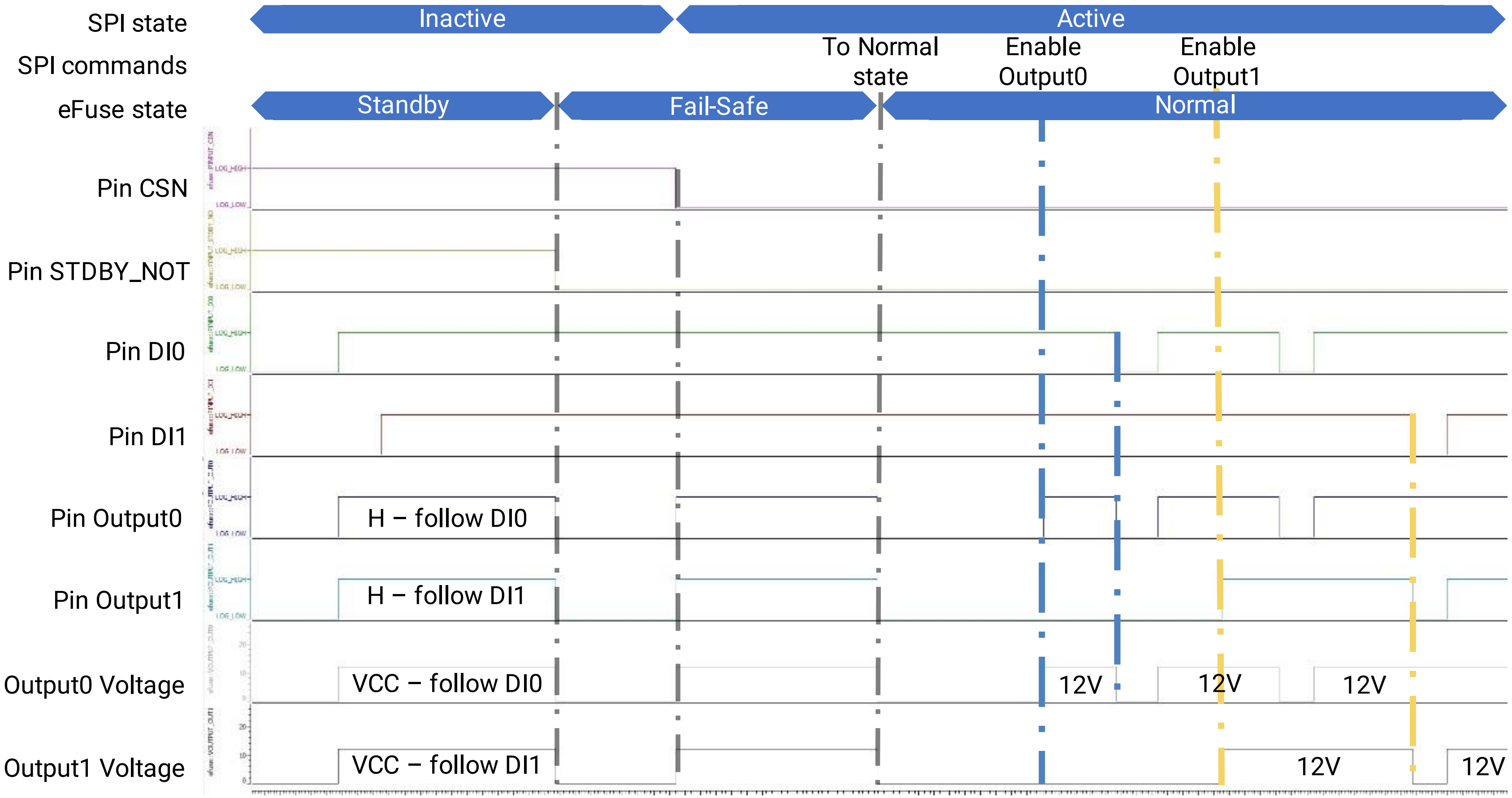
Proof of Concept

Setup Level 3 simulation environment



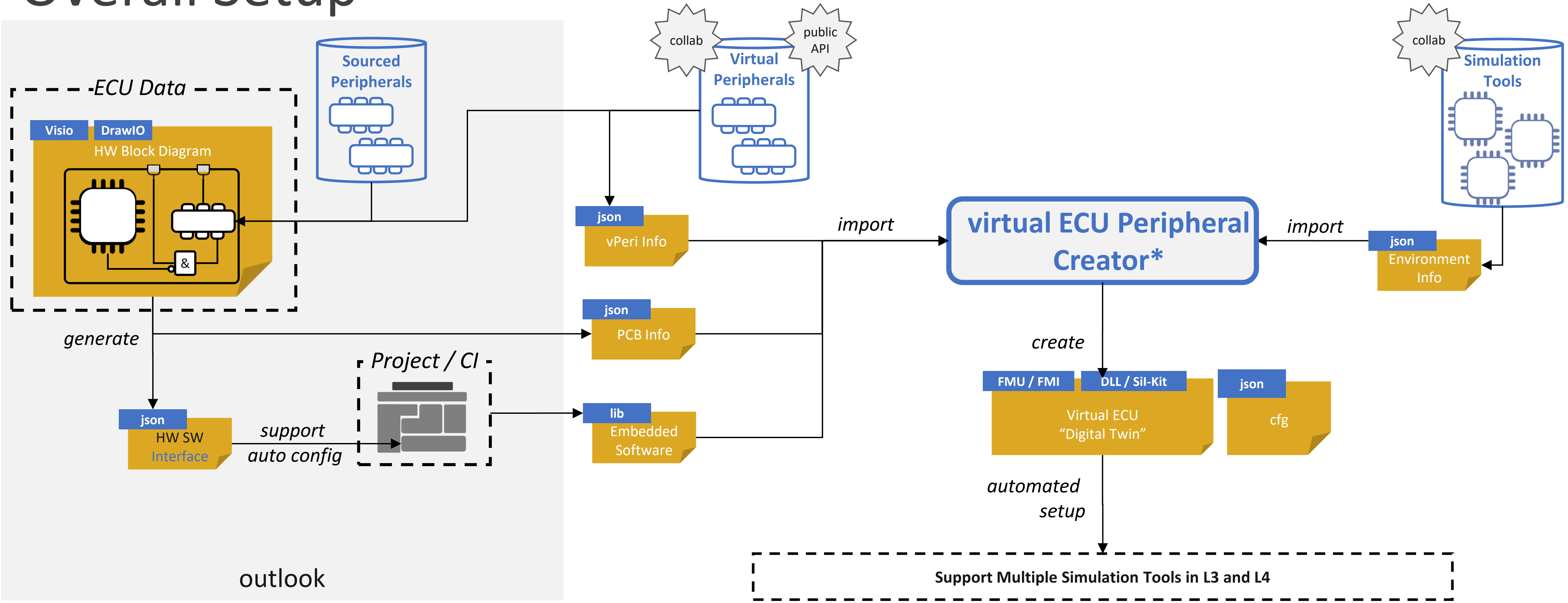
Proof of Concept

Setup Level 3 simulation environment



Conclusion and Outlook

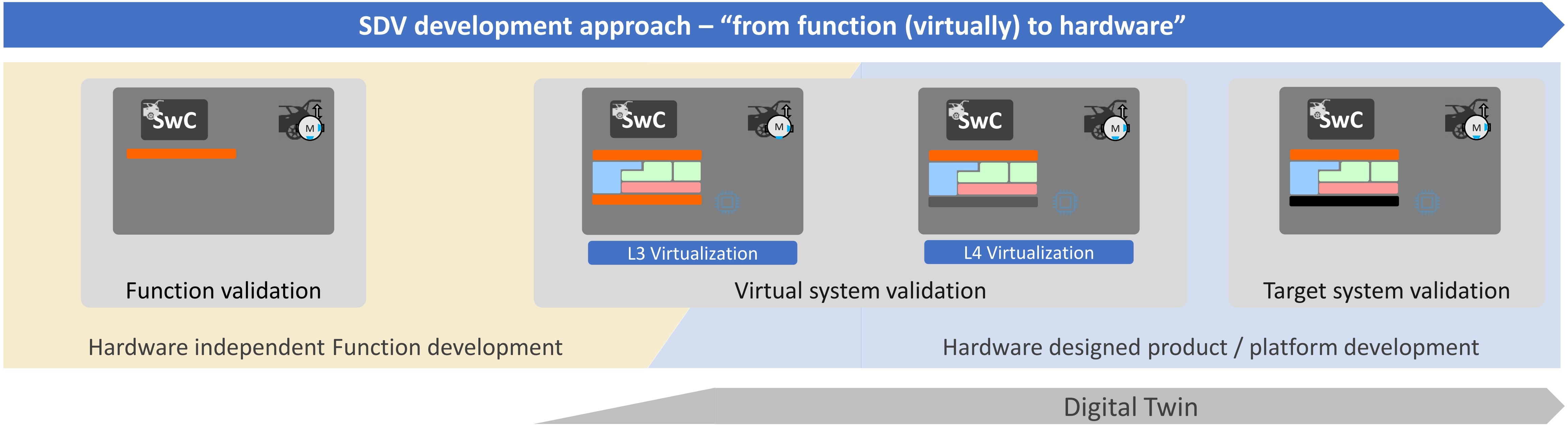
Overall Setup



* Part of Continental’s virtual SDV Composer

Conclusion and Outlook

Streamlined development with virtualization support



Virtual peripherals and virtual wiring are required to enable I/O based applications in virtual real-time ECUs.

Virtual ECU Peripheral Creator provides a “Digital Twin” for different virtualization levels and tools.

Support for streamlined development with benefits from all virtualization levels.

Questions

AUMOVIO

Thank You!

Sacha Loitz
Expert (IC Virtual Platforms)
Sacha.Loitz@aumovio.com

Torsten Hermann
Senior Expert Virtualization
Torsten.Hermann@aumovio.com

Martin Hruschka
Software Architekt
Martin.Hruschka@aumovio.com