Functional Mock-up Interface

FMI 3.0 and Layered Standards (FMI-LS-XCP & FMI-LS-BUS)

Christian Bertsch (Bosch)
Kahramon Jumayev (Akkodis)
Pierre Mai (PMSF)
Benedikt Menne (dSPACE)
Masoud Najafi (Altair)
Tim Pfitzer (Bosch)
Jan Ribbe (Synopsys)
Klaus Schuch (AVL)

Modelica Association

Overview

- FMI in General
  - The Functional Mock-up Interface 3.0
  - Layered Standard concept
- FMI in Automotive Context
  - Virtual ECU
    - Layered Standard for XCP
    - Layered Standard for Network Communication

  Demo from different SiL tool vendors

  - cross-manufacturer interoperability of FMI 3.0 and these layered standards.
- Summary and Outlook

fmi Functional Mock-up Interface

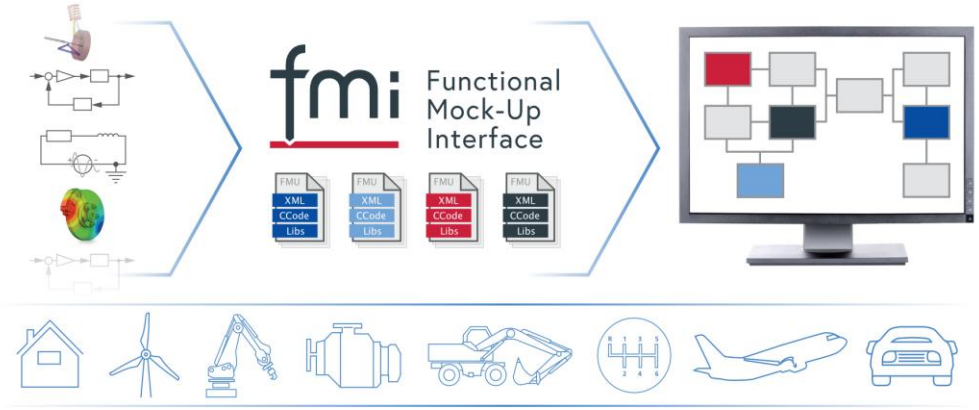
Get in touch

Web https://fmi-standard.org
Code https://github.com/modelica/fmi-standard
E-Mail contact@fmi-standard.org
News https://newsletter.modelica.org

The Functional Mock-Up Interface

The Functional Mock-up Interface is a free standard that defines a **container** and an **interface**

- to exchange **dynamic simulation models**
- using a combination of **XML files, binaries** and **C code**, distributed as a **ZIP file.**
- Current releases: **FMI 2.0.5** and **FMI 3.0.2**
- **260+ tools** and libraries support FMI



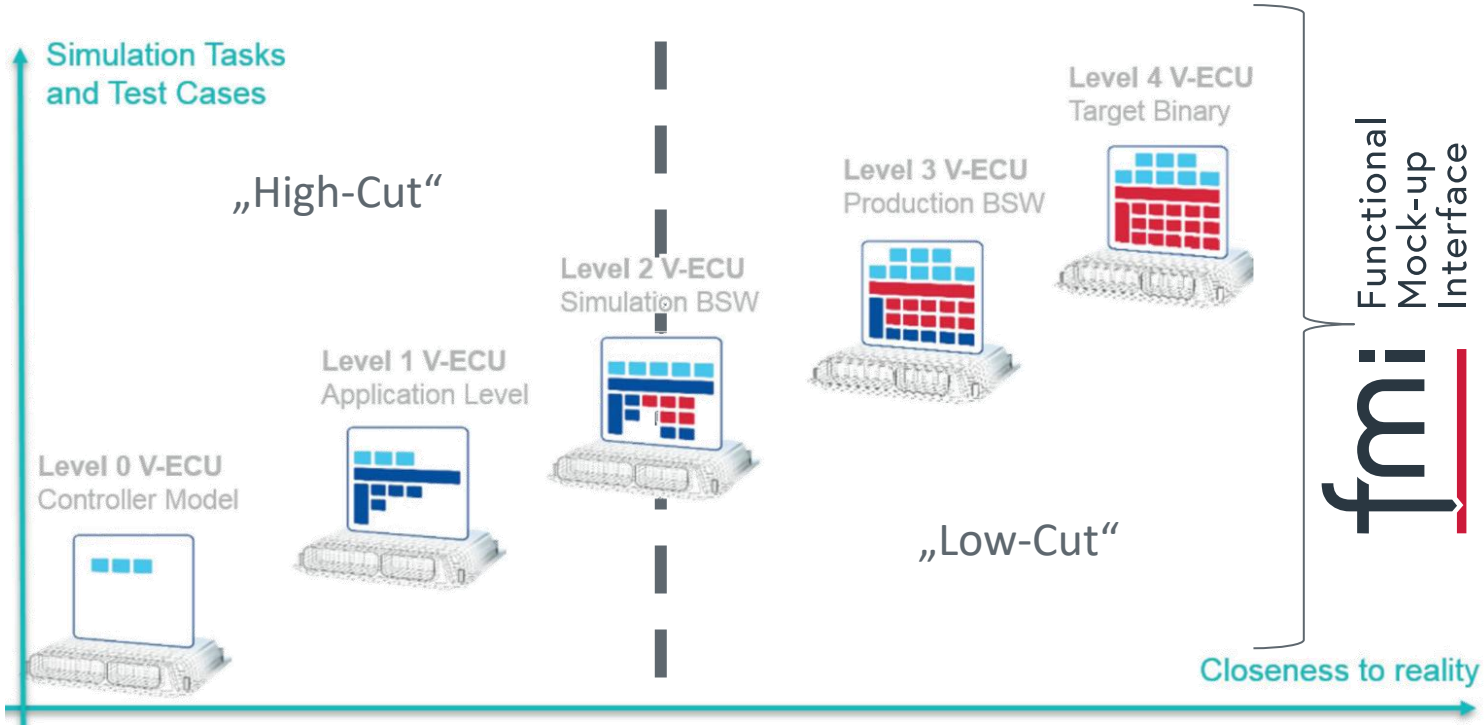
FMI Project

within the Modelica Association

Automotive Domain vECUs

- A virtual ECU (**vECU**) is an **abstraction** that contains all the SW parts needed to simulate specific aspects of a real ECU.

- **FMI** is an independent **Interface Standard** and therefore cover all abstraction Level (0-4)
  (well-established FMI **API** between FMU and importer)



Reference: prostep ivip White Paper Virtual Electronic Control Units (V-ECUs)

Layered Standards

- Layered standards **extend the FMI standard** for **new application domains**.
- They can be defined by different organizations

Layered Standards in development by the FMI Project:

Automotive Domain:

- **FMI-LS-XCP:** for XCP support
- **FMI-LS-BUS:** for network communication
- **FMI-LS-STRUCT**: for structuring of variables + lookup tables
- **FMI-LS-REF:** description of validation experiments and other files attached to an FMU

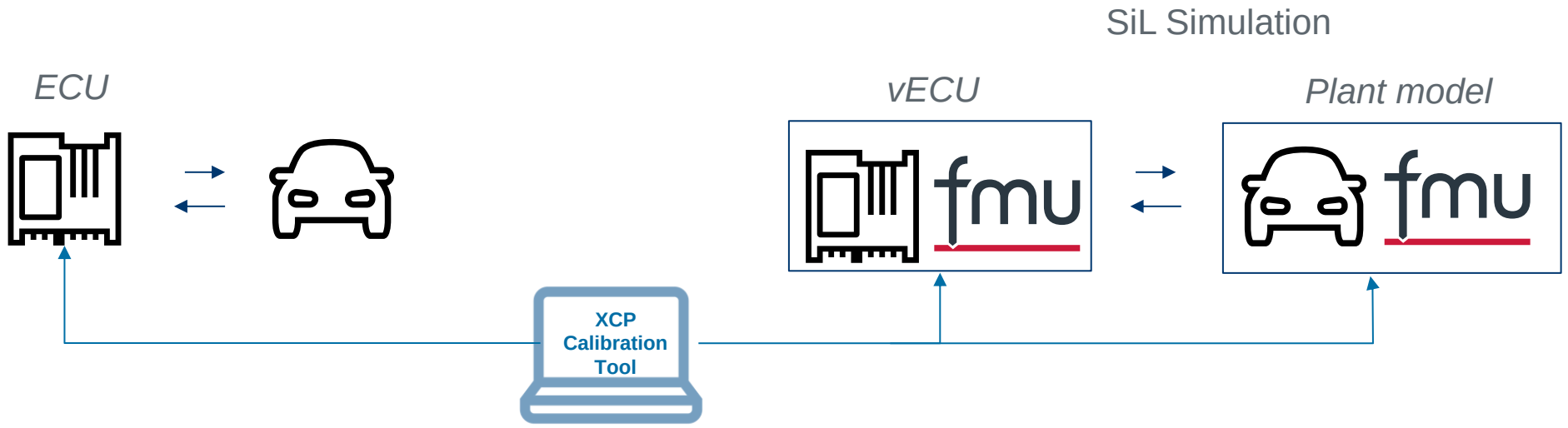
Not in focus of presentation

FMI-LS-XCP: Layered Standard for XCP support

XCP (Universal Measurement and Calibration Protocol) is a network protocol originating from ASAM for connecting calibration systems to electronic control units, ECU

Main idea:

- ship an **A2L file** according to the ASAM ASAP2 in standardized location inside the FMU
- **describe the capabilities** w.r.t. the XCP protocol



References: FMI Layered Standard for XCP

Released Version FMI-LS-XCP v1.0.0

- FMI Layered Standard for XCP 1.0 released and works with FMI 2.0 and FMI 3.0!
- **More information:**
  - FMI Layered Standard for XCP (FMI-LS-XCP) v1.0.0, (Github repository and Issue tracker)

Tool Support and Contribution:

- The adoption of the industry including implementations, prototypes and tool releases supporting the FMI Layered Standard for Network Communication is rapidly growing.

  Many thanks to all contributors!

Listed Tools for FMI-LS-XCP:

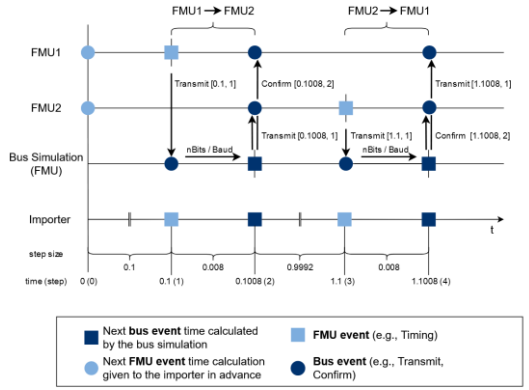
- AVL (Maestra)
- Akkodis (PROVEtech:RE, PROVEtech:TA)
- dSPACE (ConfigurationDesk, SystemDesk, VEOS)
- PMSF (FMI Bench)
- Synopsys (Silver)
- Tracetronic (ecu.test)

FMI-LS-BUS: Layered Standard Network Communication

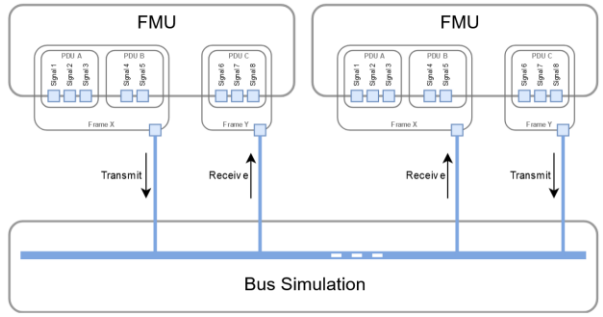
Simulation of automotive systems requires **network communication between vECUs.**

Idea:

- **Use FMI 3.0 core standard features** to specify a common bus interface, using **Co-Simulation**, **Binary Variables**, **Clocks** and **Terminals**
- Support **automotive networks protocols** and others from **different industries** (CAN, LIN, FlexRay, Ethernet, …)
- Using network description formats DBC, LDF, ASAM FIBEX, ARXML



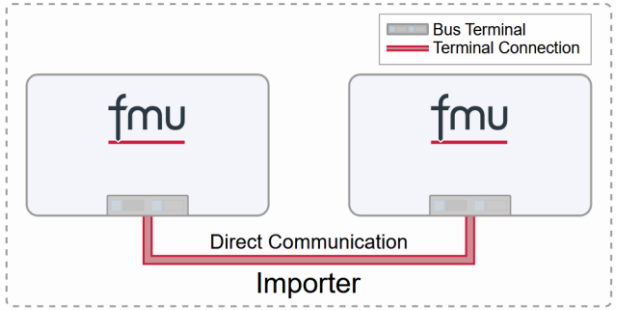
- **Two variants:**
  - **Physical abstraction layer ("High-Cut"):** Simply transport physical signal values between virtual ECUs by using unit-based and clocked variables
  - **Network abstraction ("Low-Cut"):** Detailed emulation of a specified bus system to realize virtualized bus driver implementations, including feedback from the physical drivers about transmission status or network node states



Bus Simulation Composition with FMI-LS-BUS

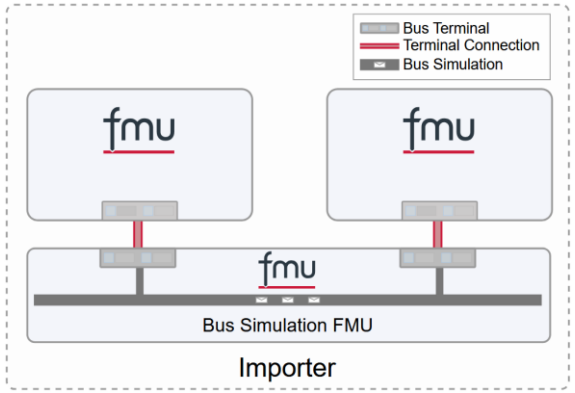
One Interface, Maximum Flexibility

- The same standardized FMU can be used across all three compositions—from simple direct connections to complex, realistic bus simulations - without requiring any changes to the FMU itself



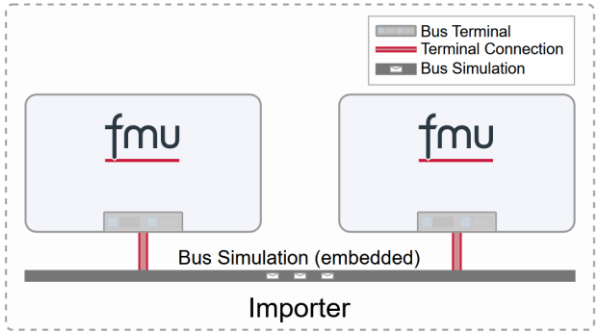
Bus simulation idealized

- 1:1 connection
- Idealized bus
  (only ideal timing, no bandwidth limits, …)
- Standard FMI Importer can be used



Bus simulation via Bus Simulation FMU

- n:m connection
- Bus behavior can be simulated
- Standard FMI Importer can be used
- Can be generated based on network descriptions
- Changes of on system level require regeneration



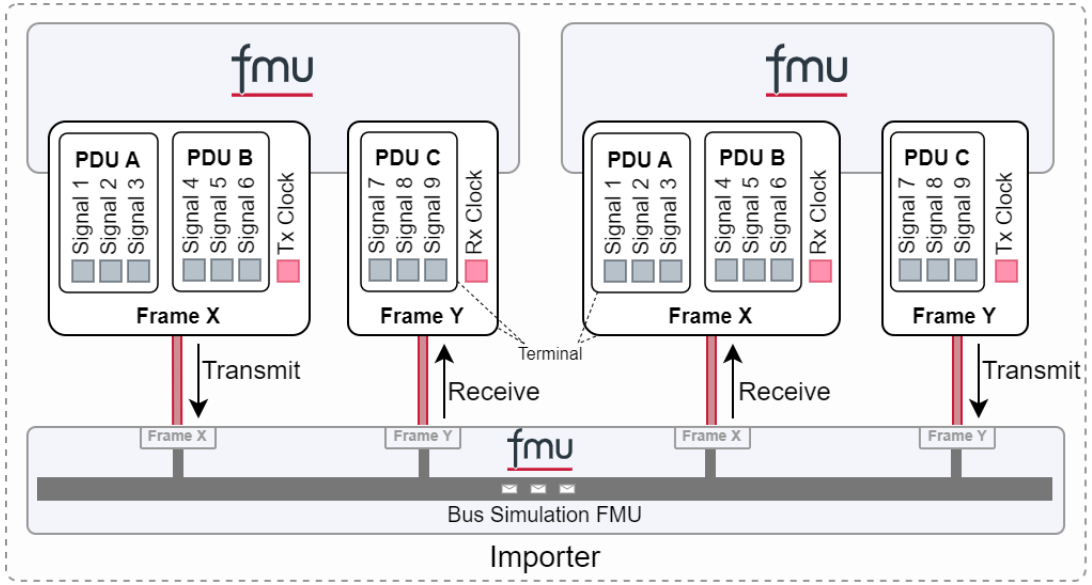
Bus simulation supported by the Importer

- n:m connections
- Bus behavior can be simulated
- Dynamic configuration
- No network descriptions needed

Physical Abstraction Layer (“High-Cut”)

Technical Breakdown:

- **FMI Variables:** Each bus signal is modeled as a standard fmi variable with its physical type
- **FMI Clocks:** A dedicated clock triggers the data exchange. A tick on a transmission clock (TxClock) indicates that the signal values are new and ready to be sent
- **Hierarchical Terminals:** Terminals are used to group the variables, mirroring the network database structure (Signals → PDUs → Frames)




```
<!-- modelDescription.xml -->
<Clock name="Clock" valueReference="1" causality="input" variability="discrete" intervalVariability="constant" intervalDecimal="0.020" shiftDecimal="0.010"/>
<Float64 name="Signal_A" valueReference="3" causality="output" variability="discrete" clocks="1"/>
<Float64 name="Signal_B" valueReference="4" causality="output" variability="discrete" clocks="1"/>

<!-- terminalsAndIcons.xml -->
<Terminal terminalKind="org.fmi-ls-bus.frame-terminal" name="Frame_A" matchingRule="bus">
    <TerminalMemberVariable variableKind="signal" variableName="Clock" memberName="Clock"/>
    <Terminal terminalKind="org.fmi-ls-bus.pdu-terminal" name="PDU_A" matchingRule="bus">
        <TerminalMemberVariable variableKind="signal" variableName="Signal_A" memberName="Signal_A"/>
        <TerminalMemberVariable variableKind="signal" variableName="Signal_B" memberName="Signal_B"/>
    </Terminal>
</Terminal>
```

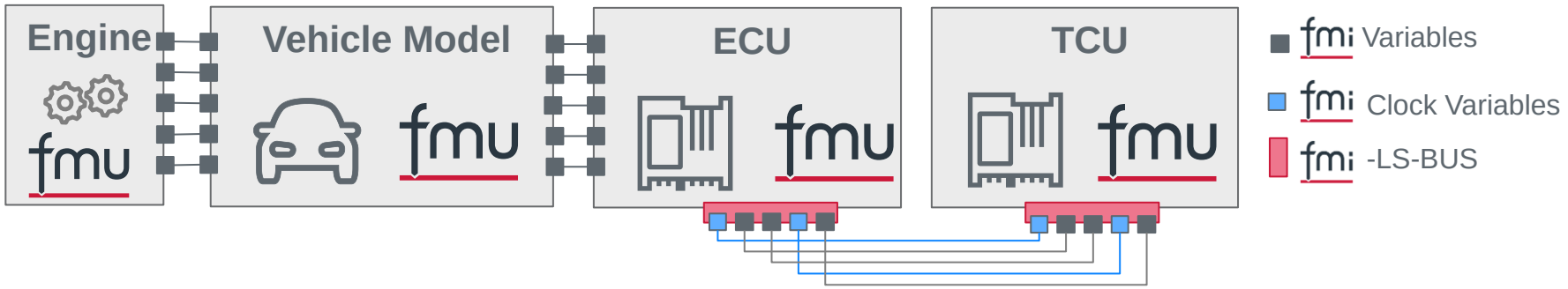
FMI-LS-BUS: High-Cut Demonstrator

Click to open Demo

Closed-loop fuel injection control system

Components:

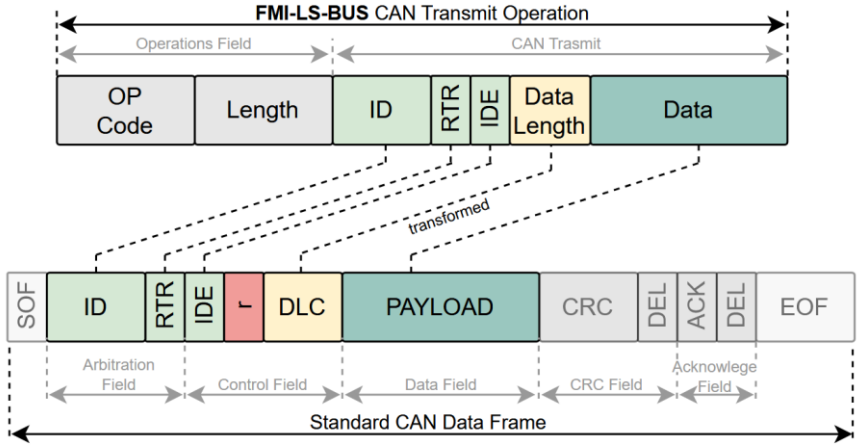
- **TCU:** Transmission Control Unit
- **ECU:** Engine Control Unit – Here only used as a bus adapter
- **Vehicle Model incl. Engine Model:** Simplified vehicle model



Network Abstraction Layer (“Low-Cut”)

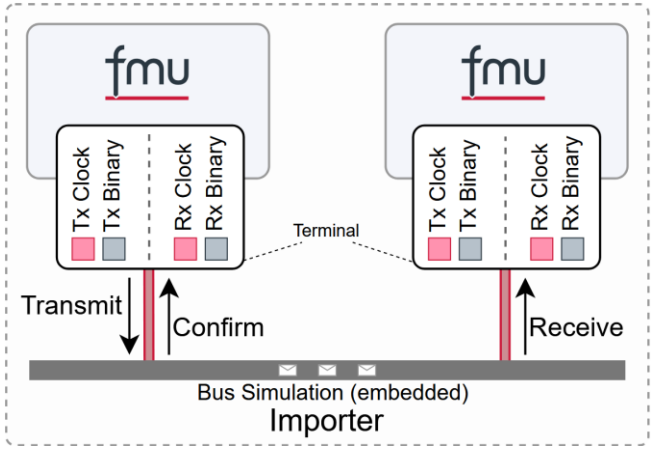
Technical Breakdown:

- **Binary Variables:** A single binary variable for each direction (Tx/Rx) acts as the data carrier. It holds a serialized payload a "bus operation" defined by the FMI-LS-BUS standard.
- **FMI Clocks:** A clock is tightly coupled with each binary variable. A tick on the TxClock is the event that signals "the data in TxBinary is valid now and should be processed by the bus simulation".
- **Terminals:** These elements are grouped into a single terminal per bus connection for simple, clean connections in the co-simulation environment.




```
<!-- modelDescription.xml -->
<Clock name="Tx_Clock" valueReference="1" causality="input" intervalVariability="countdown" />
<Clock name="Rx_Clock" valueReference="2" causality="input" intervalVariability="triggered" />
<Binary name="Tx_Data" valueReference="3" causality="output" variability="discrete" mimeType="application/org.fmi-standard.fmi-ls-bus.can; version=&quot;1.0.0&quot;" clocks="1" />
<Binary name="Rx_Data" valueReference="4" causality="input" variability="discrete" mimeType="application/org.fmi-standard.fmi-ls-bus.can; version=&quot;1.0.0&quot;" clocks="2" />
<Boolean name="BusNotification" valueReference="5" causality="parameter" variability="fixed" start="false"/>

<!-- terminalsAndIcons.xml -->
<Terminal terminalKind="org.fmi-ls-bus.network-terminal" name="CanController" matchingRule="org.fmi-ls-bus.transceiver">
    <TerminalMemberVariable variableKind="signal" variableName="Tx_Data" memberName="Tx_Data" />
    <TerminalMemberVariable variableKind="signal" variableName="Tx_Clock" memberName="Tx_Clock" />
    <TerminalMemberVariable variableKind="signal" variableName="Rx_Data" memberName="Rx_Data" />
    <TerminalMemberVariable variableKind="signal" variableName="Rx_Clock" memberName="Rx_Clock" />
    <TerminalMemberVariable variableKind="signal" variableName="BusNotification" memberName="BusNotification" />
</Terminal>
```



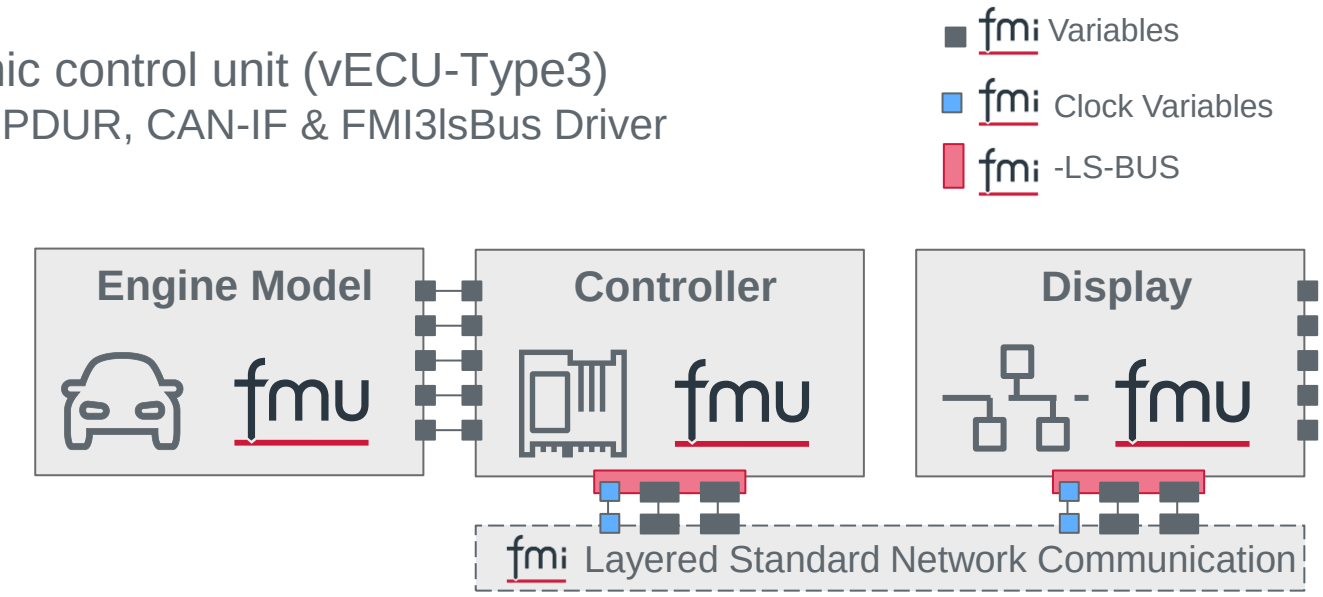
FMI-LS-BUS: Low-Cut Demonstrator

Click to open Demo

Closed-loop fuel injection control system

Components:

- **Controller:** AUTOSAR based virtual electronic control unit (vECU-Type3)
  including BSW configuration and components e.g. PDUR, CAN-IF & FMI3lsBus Driver
- **Engine Model:** Simplified vehicle model
  including relevant sensor and actuator functions
- **Display:** Restbus simulation model
  including simplified PDU scheduler
  (non-AUTOSAR) and signal to bus interface
  & FMI3lsBus Driver



Demonstration:

- **Seamless Tool Interoperability:** Demonstrating FMUs from diverse tools and vendors integrated into various simulation environment
- **Configurable Virtual Bus Setup:** Showcasing how the bus setup can adapt to various contexts without the need for re-exporting FMUs

FMI-LS-BUS: Roadmap

FlexRay

Specification and verification of FlexRay support for Network Abstraction

LIN

Specification and verification of LIN support for Network Abstraction

Physical Signal Abstraction and Network Abstraction: CAN support

Specification and verification of the complete Physical Signal Abstraction and CAN, CAN FD, CAN XL support for Network Abstraction

Ethernet

Specification and verification of Ethernet support for Network Abstraction

Future

Maintenance and identification phase for other bus systems to be supported

* The Alpha release of FMI-LS-BUS: Ethernet is released within the v1.1.0-beta release

References: FMI Layered Standard for Network Communication

Released Version FMI-LS-BUS v1.0.0

- FMI Layered Standard for Network Communication 1.0 released - enabling the simulation of virtual ECUs with FMI 3.0!
- **More information:**
  - FMI Layered Standard for Network Communication (FMI-LS-BUS) v1.0.0, (Github repository and Issue tracker)

Tool Support and Contribution:

- The adoption of the industry including implementations, prototypes and tool releases supporting the FMI Layered Standard for Network Communication is rapidly growing.

  Many thanks to all contributors!

Listed Tools for FMI-LS-BUS:

- Altair (Twin Activate)
- AVL (Model.CONNECT)
- Akkodis (PROVEtech:RE)
- Bosch (Power Solutions)
- dSPACE (SystemDesk, VEOS)
- Synopsys (Silver)
- Vector (SIL Kit FMU Importer, vVIRTUALtarget)

Take-Away: (R)evolution to SiL Simulation of Virtual ECUs

FMI-LS-BUS and FMI-LS-XCP are key enabler for modern virtual automotive development

- The FMI standard alone is available for exchanging simple virtual ECUs in early development phases
- FMI-LS-XCP standardizes usage of XCP as an extension to FMI 2.0 and FMI 3.0
- FMI-LS-BUS standardizes network simulation as an extension to FMI 3.0
  - Flexible High-Cut and Low-Cut layers align perfectly with the development lifecycle
- Enables early, continuous validation and improves collaboration and tool agnostics

The advantages for validation are obvious:

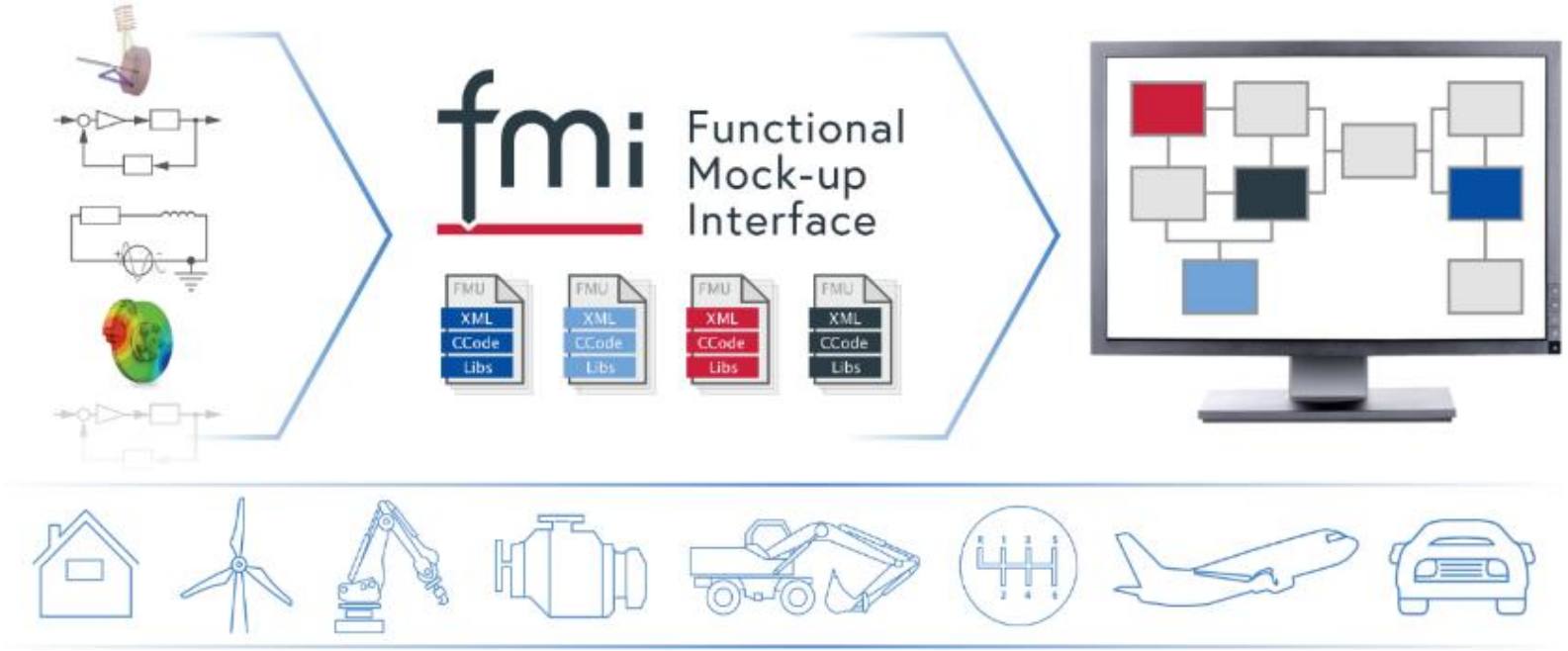
- **Lower integration costs** thanks to **better interoperability** of tools and test platforms from **different vendors** within virtual validation scenarios
- More **efficient collaboration** thanks to **significantly reduced coordination effort** between **OEMs and suppliers**

Questions?

Comments?

Ideas?

Feedback?



Join the FMI LinkedIn Group!